



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΕΠΙΚΟΙΝΩΝΙΩΝ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Πίνακας ελέγχου για την παρακολούθηση της υγείας των ηλικιωμένων: Ένα ασφαλές, προσβάσιμο, ενισχυμένο με τεχνητή νοημοσύνη σύστημα για την απομακρυσμένη φροντίδα ηλικιωμένων. Elderly Health Monitoring Dashboard: A Secure, Accessible, AI-Enhanced System for Remote Elderly Care
Ονοματεπώνυμο Φοιτητή	Κωνσταντίνιδης Αδαμάντιος Άγγελος
Πατρώνυμο	Μάριος
Αριθμός Μητρώου	Π14090
Επιβλέπουσα	Μαρία Βίρβου, Καθηγήτρια

Ημερομηνία Παράδοσης / ΣΕΠΤΕΜΒΡΙΟΣ 2026

Copyright ©

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν αποκλειστικά τον συγγραφέα και δεν αντιπροσωπεύουν τις επίσημες θέσεις του Πανεπιστημίου Πειραιώς.

Ως συγγραφέας της παρούσας εργασίας δηλώνω πως η παρούσα εργασία δεν αποτελεί προϊόν λογοκλοπής και δεν περιέχει υλικό από μη αναφερόμενες πηγές.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά την κα. Μαρία Βίρβου, Κοσμήτορα της Σχολής Τεχνολογιών Πληροφορικής και Επικοινωνιών του Πανεπιστημίου Πειραιώς, για την πολύτιμη βοήθεια της και την καθοδήγηση της κατά την υλοποίηση της εργασίας αυτής.

Θα ήθελα επίσης να ευχαριστήσω την οικογένεια μου για την στήριξη τους σε όλη αυτή την πορεία για την ολοκλήρωση των σπουδών μου.

Περιεχόμενα

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ	0
Copyright ©	1
Ευχαριστίες	1
Περιεχόμενα	2

Περίληψη.....	8
Κατάλογος εικόνων	9
Λίστα πινάκων.....	9
Λίστα συντομογραφιών	9
Κεφάλαιο 1. Εισαγωγή.....	11
1.1 Ιστορικό και κίνητρα.....	11
1.1.1 Η παγκόσμια κρίση γήρανσης	11
1.1.2 Το βάρος της διαχείρισης των χρόνιων ασθενειών.....	11
1.1.3 Το ψηφιακό χάσμα στην τεχνολογία της υγειονομικής περίθαλψης	12
1.1.4 Το βάρος των φροντιστών και η αναγκαιότητα της απομακρυσμένης παρακολούθησης.....	12
1.1.5 Κενά στο τρέχον τεχνολογικό τοπίο	13
1.1.6 Η αναγκαιότητα για ολοκληρωμένες λύσεις.....	14
1.2 Περιγραφή του προβλήματος.....	14
1.2.1 Ορισμός του βασικού προβλήματος	15
1.2.2 Συγκεκριμένες ελλείψεις των υφιστάμενων λύσεων	15
1.2.3 Συνέπειες των ανεπίλυτων ελλείψεων	15
1.2.4 Έλλειψη έρευνας	16
1.3 Σκοπός και στόχοι της διατριβής.....	16
1.3.1 Κύριος σκοπός	16
1.3.2 Ειδικό στόχοι	16
1.3.3 Πεδίο εφαρμογής και όρια	17
1.3.4 Αναμενόμενες συνεισφορές.....	18
1.3.5 Επισκόπηση μεθοδολογίας	18
1.4 Σύγχρονα παραδείγματα και παρόμοια έργα	18
1.4.1 Πλατφόρμες υγείας και φυσικής κατάστασης για καταναλωτές.....	19
1.4.2 Επαγγελματικές πύλες υγείας για ασθενείς.....	20
1.4.3 Τεχνολογικές λύσεις ειδικά για ηλικιωμένους	20
1.4.4 Πλατφόρμες τηλεϊατρικής	21
1.4.5 Συγκριτικός πίνακας χαρακτηριστικών	22
1.4.6 Τι κάνει αυτή τη διατριβή διαφορετική	23
1.4.7 Περίληψη συνεισφοράς	23
1.5 Επισκόπηση της δομής της διατριβής.....	24
Μεθοδολογική προσέγγιση.....	24
Κεφάλαιο 2. Ανασκόπηση πεδίου	24
2.1 Ανασκόπηση βιβλιογραφίας - Ακαδημαϊκή έρευνα	24
2.1.1 Μελέτες υιοθέτησης τεχνολογιών υγειονομικής περίθαλψης	24
2.1.2 Συστήματα απομακρυσμένης παρακολούθησης της υγείας	26
2.1.3 Ασφάλεια και προστασία των δεδομένων υγείας.....	28

2.1.4 Προσβασιμότητα σε εφαρμογές υγειονομικής περίθαλψης	29
2.1.5 Τεχνητή νοημοσύνη, μηχανική μάθηση και εξατομίκευση της αλληλεπίδρασης	30
2.1.6 Συναισθηματική εξατομίκευση και πολυτροπική αναγνώριση συναισθημάτων	31
2.1.7 Μοντέλα χρηστών, συλλογισμός και προσαρμοστική καθοδήγηση	32
2.1.8 Εξατομικευμένες εφαρμογές, ιδιωτικότητα και ηθική χρήση της τεχνητής νοημοσύνης	33
2.1.9 Μεγάλα γλωσσικά μοντέλα, εμπιστοσύνη και αξιολόγηση	34
2.2 Ανάλυση της εμπορικής τεχνολογίας στον τομέα της υγειονομικής περίθαλψης	35
2.2.1 Τμηματοποίηση της αγοράς πλατφορμών υγείας για καταναλωτές	35
2.2.2 Τάσεις τεχνολογικών στοιβών σε εφαρμογές υγειονομικής περίθαλψης	36
2.2.3 Ωριμότητα προσβασιμότητας σε εμπορικές πλατφόρμες	36
2.3 Ανάλυση τεχνολογικής ανάπτυξης	37
2.3.1 Επιλογή πλαισίου frontend: React vs. Angular vs. Vue	37
2.3.2 Επιλογή γλώσσας backend: Go εναντίον Node.js εναντίον Java	38
2.3.3 Επιλογή βάσης δεδομένων: PostgreSQL έναντι MongoDB έναντι MySQL	39
2.4 Αξιολόγηση της τρέχουσας κατάστασης	39
2.4.1 Ανάλυση της αγοράς ψηφιακής υγείας	39
2.4.2 Τεχνολογικά κενά και προβληματικά σημεία	40
2.4.3 Κανονιστικό πλαίσιο και συμμόρφωση	41
2.5 Προσδιορισμένες ελλείψεις και συμβολή της διατριβής	42
2.5.1 Περίληψη ανάλυσης κενών	42
2.5.2 Αντιστοίχιση συμβολής της διατριβής	43
2.5.3 Διαφορά από σχετικές ακαδημαϊκές εργασίες	43
2.5.4 Περιορισμοί του πεδίου εφαρμογής και μελλοντικές κατευθύνσεις της έρευνας	44
Κεφάλαιο 3. Ανάλυση και σχεδιασμός	44
3.1 Ανάλυση απαιτήσεων	44
3.1.1 Λειτουργικές απαιτήσεις	45
3.1.2 Μη λειτουργικές απαιτήσεις	49
3.1.3 Περιορισμοί συστήματος	52
3.1.4 Ανιχνευσιμότητα απαιτήσεων	55
3.2 Σχεδιασμός αρχιτεκτονικής συστήματος	55
3.2.1 Γενική επισκόπηση αρχιτεκτονικής	55
3.2.2 Ανάλυση στοιχείων	56
3.2.3 Τοπολογία ανάπτυξης στο cloud	57
3.2.4 Επίπεδα αρχιτεκτονικής ασφάλειας	60
3.3 Σχεδιασμός βάσης δεδομένων	60
3.3.1 Μοντέλο οντοτήτων-σχέσεων	60
3.3.2 Λογικό σχήμα βάσης δεδομένων	61
3.3.3 Ανάλυση κανονικοποίησης	62

3.3.4 Στρατηγική ευρετηρίου για δεδομένα υγείας χρονοσειρών	63
3.4 Σχεδιασμός διεπαφής χρήστη	64
3.4.1 Αρχές σχεδιασμού με προτεραιότητα στην προσβασιμότητα	64
3.4.2 Λόγοι επιλογής του στοιχείου Material-UI	64
3.4.3 Κλίμακα τυπογραφίας και αναγνωσιμότητα.....	65
3.4.4 Παλέτα χρωμάτων και οπτική ιεραρχία	65
3.4.5 Μέγεθος στόχου αφής και κινητική προσβασιμότητα	66
3.4.6 Μοτίβα διάταξης και ιεραρχία πληροφοριών	66
3.4.7 Σχεδιασμός φορμών και επικύρωση εισόδου.....	67
3.5 Διαγράμματα UML και μοντελοποίηση συμπεριφοράς	68
3.5.1 Διάγραμμα περιπτώσεων χρήσης.....	68
3.5.2 Διάγραμμα κλάσεων (βασικό μοντέλο τομέα).....	69
3.5.3 Διαγράμματα ακολουθίας	69
3.5.4 Διάγραμμα δραστηριότητας: Ροή εργασίας καταχώρισης μετρήσεων υγείας.....	71
Κεφάλαιο 4: Υλοποίηση της εφαρμογής.....	73
4.1 Περιβάλλον ανάπτυξης και εργαλεία	73
4.1.1 Επισκόπηση τεχνολογικού stack	73
4.1.2 Δομή έργου	76
4.1.3 Διαμόρφωση περιβάλλοντος	77
4.1.4 Ροή εργασίας τοπικής ανάπτυξης.....	77
4.2 Εφαρμογή backend	78
4.2.1 Ρύθμιση κύριου διακομιστή.....	78
4.2.2 Μοντέλα και σχήματα βάσεων δεδομένων	79
4.2.3 Ελεγκτής πιστοποίησης.....	81
4.2.4 Εφαρμογή υπηρεσίας AI Chat	82
4.2.5 Διαδρομές API και middleware	84
4.2.6 Αρχικοποίηση βάσης δεδομένων.....	85
4.3 Εφαρμογή frontend.....	85
4.3.1 Δομή έργου και αρχιτεκτονική στοιχείων	85
4.3.2 Ροή πιστοποίησης.....	85
4.3.3 Πίνακας ελέγχου ασθενούς με μετρήσεις υγείας.....	86
4.3.4 Διεπαφή AI Chat.....	86
4.4 DevOps και ανάπτυξη.....	86
4.4.1 Terraform Infrastructure-as-Code.....	86
4.4.2 Διαμόρφωση Docker	87
4.5 Προβλήματα και δυσκολίες που αντιμετωπίστηκαν	87
Κεφάλαιο 5: Λειτουργικότητα της εφαρμογής	91
5.1 Βασική λειτουργικότητα του συστήματος.....	91
5.1.1 Ροή εγγραφής και πιστοποίησης χρήστη	91

5.1.2 Διαχείριση προφίλ ασθενούς	94
5.1.3 Πίνακας ελέγχου δεικτών υγείας	94
5.2 Προηγμένες λειτουργίες	95
5.2.1 Βοηθός υγείας με τεχνητή νοημοσύνη	95
5.2.2 Πίνακας ελέγχου πολλαπλών ασθενών για φροντιστές	98
5.3 Εφαρμογή αλγορίθμου	100
5.3.1 Αλγόριθμος συναρμολόγησης πλαισίου για AI	100
5.3.2 Αλγόριθμος συγκέντρωσης μετρήσεων υγείας	101
5.4 Χαρακτηριστικά απόδοσης	101
5.5 Ανιχνευσιμότητα απαιτήσεων	102
Κεφάλαιο 6: Εγχειρίδιο χρήστη	106
6.1 Ξεκινώντας	106
6.1.1 Απαιτήσεις συστήματος	106
6.1.2 Πρόσβαση στην εφαρμογή	107
6.1.3 Δημιουργία του λογαριασμού σας	107
6.2 Οδηγός χρήστη για ασθενείς	110
6.2.1 Σύνδεση	110
6.2.2 Συμπλήρωση του προφίλ σας (νέοι χρήστες)	111
6.2.3 Επισκόπηση του πίνακα ελέγχου	112
6.2.4 Προσθήκη μετρήσεων υγείας	113
6.2.5 Κατανόηση των διαγραμμάτων υγείας σας	114
6.2.6 Χρήση του βοηθού υγείας AI	115
6.2.7 Εξαγωγή των δεδομένων υγείας σας	117
6.2.8 Διαχείριση προσκλήσεων νοικοκυριών	117
6.3 Οδηγός χρήστη φροντιστή/διαχειριστή	119
6.3.1 Δημιουργία νοικοκυριού	119
6.3.2 Πρόσκληση ασθενών στο νοικοκυριό σας	119
6.3.3 Προβολή όλων των ασθενών σας	120
6.3.4 Λεπτομερής προβολή ασθενούς	121
6.3.5 Χρήση του βοηθού AI για πολλούς ασθενείς	122
6.3.6 Ρύθμιση ειδοποιήσεων (μελλοντική λειτουργία)	124
6.4 Αντιμετώπιση προβλημάτων	124
6.4.1 Συνηθισμένα προβλήματα και λύσεις	124
6.4.2 Προβλήματα που αφορούν συγκεκριμένους περιηγητές	125
6.4.3 Αναφορά μηνυμάτων σφάλματος	125
Κεφάλαιο 7: Συμπεράσματα και μελλοντικές επεκτάσεις	126
7.1 Επιτεύγματα της διατριβής	126
7.1.1 Επικύρωση επίτευξης στόχων	126
7.1.2 Τεχνικά επιτεύγματα	128

7.1.3 Επιτεύγματα στην εμπειρία χρήστη	128
7.2 Διδάγματα	129
7.2.1 Τεχνικά διδάγματα	129
7.2.2 Πληροφορίες για την τεχνολογία στον τομέα της υγειονομικής περίθαλψης	129
7.2.3 Μαθήματα από τη διαδικασία ανάπτυξης λογισμικού	130
7.3 Μελλοντικές βελτιώσεις.....	130
7.3.1 Βραχυπρόθεσμες βελτιώσεις (0-6 μήνες).....	130
7.3.2 Βελτιώσεις μεσοπρόθεσμα (6-18 μήνες)	131
7.3.3 Μακροπρόθεσμη προοπτική (18+ μήνες)	131
7.4 Αντίκτυπος στον κλάδο και εφαρμογές	132
7.4.1 Δυνατότητες εμπορικής αξιοποίησης	132
7.4.2 Συνεισφορές στην ακαδημαϊκή έρευνα.....	133
7.4.3 Ηθικές σκέψεις	133
Κεφάλαιο 8: Βιβλιογραφία.....	134
Βιβλιογραφικές αναφορές.....	134
Διαδικτυακές πηγές	142
Παράρτημα Α: Αποσπάσματα κώδικα.....	143
A.1 - Μοντέλο Household και παραγόμενο σχήμα (backend/models/HouseHold.go)	143
A.2 - Πίνακας chat_messages (παράγεται από το μοντέλο ChatMessage)	144
A.3 - Θέμα Material-UI (frontend/src/App.tsx)	144
A.4 - Παράδειγμα στοιχείου διεπαφής (frontend/src/PatientDashboard.tsx, απόσπασμα)	144
A.5 - Φόρμα στοιχείων ασθενούς (frontend/src/PatientFirstTimeForm.tsx, απόσπασμα) 144	
A.6 - Δομή αποθετηρίου my-health.....	145
A.7 - Αρχείο: backend/main.go	147
A.8 - Αρχείο: backend/models/user.go	148
A.9 - Αρχείο: backend/models/patient.go.....	149
A.10 - Αρχείο: backend/models/HealthMetrics.go	149
A.11 - Αρχείο: backend/models/Invitation.go	150
A.12 - Αρχείο: backend/models/chatSession.go	150
A.13 - Αρχείο: backend/models/chatMessage.go	151
A.14 - Αρχείο: backend/controllers/authController.go (λειτουργίες CreateUser και Login)	152
A.15 - Αρχείο: backend/services/ai/chatService.go (λειτουργία ProcessMessage).....	154
A.16 - Αρχείο: backend/services/ai/contextBuilder.go (BuildUserContext)	156
A.17 - Αρχείο: backend/services/ai/ollamaClient.go.....	157
A.18 - Αρχείο: backend/routes/route.go	159
A.19 - Middleware: backend/middlewares/checkAuth.go	160
A.20 - Χρήση του αυθεντικοποιημένου χρήστη στους ελεγκτές (patientController.go, GetPatientDetails, απόσπασμα).....	162

A.21 - Αρχείο: backend/initializers/database.go	162
A.22 - Αρχείο: frontend/src/LoginPage.tsx (απόσπασμα).....	163
A.23 - Αρχείο: frontend/src/PatientDetails.tsx (απόσπασμα: ανάκτηση μετρήσεων υγείας)	164
A.24 - Αρχείο: frontend/src/components/AIChat/AIChatBot.tsx (απόσπασμα).....	165
A.25 - Αρχείο: infra/main.tf (ανάπτυξη Azure).....	166
A.26 - Αρχείο: docker-compose.yml (ενορχήστρωση υπηρεσιών)	171
A.27 - Αρχείο: backend/Dockerfile.dev (Ανάπτυξη με Hot-Reload)	172
A.28 - Dockerfile παραγωγής (κατασκευή πολλαπλών σταδίων):	173
A.29 - Παράδειγμα απόκρισης δημιουργίας χρήστη (200 OK).....	173
A.30 - Το Frontend App.tsx διατηρεί την κατάσταση πιστοποίησης:.....	174
A.31 - Φόρμα Material-UI (frontend/src/PatientEditPage.tsx, απόσπασμα).....	175
A.32 - Παράδειγμα αιτήματος ενημέρωσης ασθενούς (POST /patient/edit/:id)	176
A.33 - Αρχείο: backend/controllers/patientController.go (UpdatePatient, απόσπασμα) ..	176
A.34 - Διάγραμμα τάσεων αρτηριακής πίεσης (frontend/src/PatientDetails.tsx, απόσπασμα)	178
A.35 - Επικύρωση από την πλευρά του πελάτη (frontend/src/PatientFirstTimeForm.tsx, απόσπασμα)	179
A.36 - Παράδειγμα κλήσης API μετρήσεων υγείας (GET /api/health-metrics/:patientId) .	179
A.37 - Επικύρωση και αρχικοποίηση μετρήσεων στο backend (patientController.go, απόσπασμα)	179
A.38 - Το frontend επισημαίνει τιμές που ενδέχεται να προκαλούν ανησυχία:.....	181
A.39 - Λειτουργία εξαγωγής δεδομένων:	181
A.40 - Χειριστής μηνυμάτων συνομιλίας (backend/controllers/chatController.go, SendMessage).....	181
A.41 - Δημιουργία προτροπής με βάση το πλαίσιο (contextBuilder.go, GenerateHealthcarePrompt)	182
A.42 - Κλήση του Ollama API (ollamaClient.go, GenerateResponse)	185
A.43 - Αποθήκευση μηνυμάτων στη βάση δεδομένων (chatService.go, απόσπασμα) ...	185
A.44 - Αρχείο: frontend/src/components/PendingInvitations.tsx (απόσπασμα).....	186
A.45 - Προβολή ασθενών στον πίνακα διαχειριστή (frontend/src/AdminDashboard.tsx, απόσπασμα)	187
A.46 - Κατάσταση ασθενών με χρωματικούς δείκτες (PatientStatusModal.tsx, απόσπασμα)	188
A.47 - Παράδειγμα εξόδου (μηνιαία συγκέντρωση):.....	189
A.48 - Χρήση Preload του GORM (householdController.go, GetHouseholdPatients, απόσπασμα)	190
A.49 — Πλήρες λογικό σχήμα της βάσης δεδομένων (DDL, PostgreSQL)	191
A.50 — Ψευδοκώδικας: BuildUserContext (backend/services/ai/contextBuilder.go)	193
A.51 — Ψευδοκώδικας: AggregateHealthMetrics (συγκέντρωση μετρήσεων υγείας)	195

Περίληψη

Η γήρανση του παγκόσμιου πληθυσμού ασκεί ολοένα μεγαλύτερη πίεση στα συστήματα υγείας: ο αριθμός των ατόμων μεγαλύτερης ηλικίας προβλέπεται να προσεγγίσει τα 2 δισεκατομμύρια έως το 2050. Οι ηλικιωμένοι χρειάζονται συνεχή παρακολούθηση της υγείας τους για τη διαχείριση χρόνιων παθήσεων, συχνά όμως δυσκολεύονται να υιοθετήσουν τις ψηφιακές τεχνολογίες υγείας, εξαιτίας εμποδίων προσβασιμότητας, ανησυχιών για την προστασία των προσωπικών τους δεδομένων και κατακερματισμένων εργαλείων συντονισμού των φροντιστών. Η παρούσα διατριβή παρουσιάζει τον σχεδιασμό, την υλοποίηση και την αξιολόγηση ενός ολοκληρωμένου πίνακα ελέγχου παρακολούθησης της υγείας ηλικιωμένων, ο οποίος αξιοποιεί τεχνητή νοημοσύνη με σεβασμό στην ιδιωτικότητα.

Η αρχιτεκτονική του συστήματος ακολουθεί μια σύγχρονη προσέγγιση πλήρους στοίβας: React με TypeScript για ένα προσβάσιμο και αποκρίσιμο frontend, Go με το πλαίσιο Gin για υπηρεσίες backend υψηλής απόδοσης, PostgreSQL για την αξιόπιστη αποθήκευση των δεδομένων υγείας και τοπικά εγκατεστημένο μεγάλο γλωσσικό μοντέλο μέσω Ollama, ώστε η συνομιλητική βοήθεια να θεμελιώνεται στα δεδομένα του ασθενούς χωρίς αυτά να εγκαταλείπουν ποτέ το σύστημα. Οι βασικές συνεισφορές είναι τέσσερις: σχεδιασμός διεπαφής με προτεραιότητα στην προσβασιμότητα, με θέματα υψηλής αντίθεσης, μεγάλη τυπογραφία και απλοποιημένη πλοήγηση για ηλικιωμένους χρήστες· αρχιτεκτονική διπλού ρόλου που υποστηρίζει τόσο την αυτοπαρακολούθηση του ασθενούς όσο και την εποπτεία πολλών ασθενών από φροντιστές, μέσω διαχείρισης νοικοκυριού και ροών προσκλήσεων· τοπική τεχνητή νοημοσύνη που κρατά τις προστατευόμενες πληροφορίες υγείας (PHI) εντός του συστήματος, παρέχοντας παράλληλα εξατομικευμένη καθοδήγηση· και οπτικοποίηση των μετρήσεων υγείας με ανάλυση διαχρονικών τάσεων και θεμέλια για ανίχνευση ανωμαλιών.

Η υλοποίηση ακολούθησε συστηματικές πρακτικές μηχανικής λογισμικού: σχεδιασμό RESTful API, έλεγχο πρόσβασης βάσει ρόλων (RBAC), αυθεντικοποίηση JWT, κανονικοποίηση της βάσης δεδομένων, ανάπτυξη σε containers με Docker και υποδομή ως κώδικα με Terraform για το Azure. Το σύστημα υποστηρίζει ασφαλή αυθεντικοποίηση, πλήρη διαχείριση προφίλ ασθενών, παρακολούθηση πολλαπλών δεικτών υγείας (αρτηριακή πίεση, καρδιακός ρυθμός, βάρος, κορεσμός οξυγόνου, ύπνος, δραστηριότητα), συνομιλητική υποβοήθηση με τεχνητή νοημοσύνη, διαχείριση νοικοκυριών από φροντιστές και εξαγωγή δεδομένων.

Η αξιολόγηση ως προς τις λειτουργικές και μη λειτουργικές απαιτήσεις επιβεβαιώνει την επίτευξη των βασικών στόχων: ασφαλή πρόσβαση πολλαπλών ρόλων, αρχιτεκτονική που προστατεύει την ιδιωτικότητα, στέρεα θεμέλια προσβασιμότητας και επεκτάσιμο σχεδιασμό που επιτρέπει μελλοντικές επεκτάσεις, όπως η ενσωμάτωση συσκευών IoT, η προγνωστική ανάλυση με μηχανική μάθηση και η διαλειτουργικότητα με τα ηλεκτρονικά αρχεία υγείας (EHR). Η εργασία καταλήγει σε πρακτικά συμπεράσματα για τον σχεδιασμό τεχνολογίας υγείας, με έμφαση στην ανθρωποκεντρική προσβασιμότητα, στα πρότυπα τοπικής εκτέλεσης της τεχνητής νοημοσύνης και στα μοντέλα συνεργασίας με επίκεντρο τον φροντιστή.

Λέξεις-κλειδιά: Τεχνολογία φροντίδας ηλικιωμένων, ψηφιακή παρακολούθηση της υγείας, προσβάσιμος σχεδιασμός διεπαφής χρήστη, τεχνητή νοημοσύνη που προστατεύει την ιδιωτικότητα, συνεργασία φροντιστών, υγειονομική περίθαλψη βασισμένη στο cloud, React, Go, PostgreSQL, τοπικό LLM, διαχείριση χρόνιων ασθενειών.

Κατάλογος εικόνων

Σχήμα 3.1: Αρχιτεκτονική συστήματος υψηλού επιπέδου 55

Σχήμα 3.2: Τοπολογία ανάπτυξης στο cloud Azure 58

Σχήμα 3.4: Εννοιολογικό διάγραμμα οντοτήτων-σχέσεων 60

Σχήμα 3.9: Διάγραμμα περιπτώσεων χρήσης συστήματος	68
Σχήμα 3.10: Διάγραμμα κλάσεων τομέα	69
Σχήμα 3.11: Διάγραμμα ακολουθίας πιστοποίησης (ροή σύνδεσης)	69
Σχήμα 3.12: Ακολουθία ερωτήσεων AI για την υγεία (ροή εργασίας συνομιλίας)	70
Σχήμα 3.14: Διάγραμμα δραστηριότητας εισαγωγής μετρήσεων υγείας	71
Σχήμα 5.1: Διάγραμμα κύκλου ζωής πιστοποίησης	93

Λίστα πινάκων

Πίνακας 1.2: Συγκριτικός πίνακας χαρακτηριστικών ψηφιακών λύσεων υγείας	22
Πίνακας 2.2: Σύγκριση κατηγοριών πλατφορμών	35
Πίνακας 2.3: Πίνακας συμβιβασμών πλαισίου frontend	38
Πίνακας 2.4: Ελλείψεις → Αντιστοίχιση συμβολής της διατριβής	43
Πίνακας 3.1: Κατάλογος λειτουργικών απαιτήσεων	45
Πίνακας 3.2: Μη λειτουργικές απαιτήσεις με KPI	49
Πίνακας 3.3: Περιορισμοί συστήματος και αιτιολόγηση	52
Πίνακας 3.4: Περίληψη στρατηγικής ευρετηρίου	63
Πίνακας 4.1: Περίληψη αλυσίδας εργαλείων ανάπτυξης	73
Πίνακας 4.3: Προκλήσεις υλοποίησης και λύσεις	87
Πίνακας 5.1: Δείκτες αναφοράς απόδοσης συστήματος	101
Πίνακας 5.2: Πίνακας υλοποίησης λειτουργικών απαιτήσεων	102
Πίνακας 6.1: Οδηγός αντιμετώπισης προβλημάτων	124

Λίστα συντομογραφιών

Συντομογραφία	Ορισμός
AI	Τεχνητή νοημοσύνη
API	Διεπαφή προγραμματισμού εφαρμογών
ARIA	Προσβάσιμες πλούσιες διαδικτυακές εφαρμογές
BP	Πίεση αίματος
CI/CD	Συνεχής ενσωμάτωση / Συνεχής ανάπτυξη
CORS	Κοινή χρήση πόρων μεταξύ διαφορετικών προελεύσεων
CPU	Κεντρική μονάδα επεξεργασίας
CRUD	Δημιουργία, ανάγνωση, ενημέρωση, διαγραφή
CSI	Διεπαφή αποθήκευσης κοντέινερ

Συντομογραφία	Ορισμός
EHR	Ηλεκτρονικός ιατρικός φάκελος
ER	Οντότητα-Σχέση
GDPR	Γενικός Κανονισμός για την Προστασία Δεδομένων
GORM	Αντιστοίχιση αντικειμένων-σχέσεων Go
HCI	Αλληλεπίδραση ανθρώπου-υπολογιστή
HIPAA	Νόμος περί φορητότητας και λογοδοσίας στην ασφάλιση υγείας
HR	Καρδιακός ρυθμός
HTML	Γλώσσα σήμανσης υπερκειμένου
HTTP	Πρωτόκολλο μεταφοράς υπερκειμένου
HTTPS	Ασφαλές HTTP
IaC	Υποδομή ως κώδικας
JWT	JSON Web Token
KPI	Βασικός δείκτης απόδοσης
LLM	Μεγάλο γλωσσικό μοντέλο
ML	Μηχανική μάθηση
NFR	Μη λειτουργική απαίτηση
ORM	Αντιστοίχιση αντικειμένων-σχέσεων
PHI	Προστατευόμενες πληροφορίες υγείας
RBAC	Έλεγχος πρόσβασης βάσει ρόλου
REST	Μεταφορά αναπαραστατικής κατάστασης
SPA	Εφαρμογή μίας σελίδας
SQL	Δομημένη γλώσσα ερωτήσεων
TLS	Ασφάλεια επιπέδου μεταφοράς
UI	Διεπαφή χρήστη
UML	Ενιαία γλώσσα μοντελοποίησης
UX	Εμπειρία χρήστη
WCAG	Οδηγίες προσβασιμότητας περιεχομένου ιστού
WHO	Παγκόσμιος Οργανισμός Υγείας (World Health Organization)

Κεφάλαιο 1. Εισαγωγή

1.1 Ιστορικό και κίνητρα

Η ψηφιακή μεταμόρφωση της υγείας βρίσκεται στο σταυροδρόμι τριών κρίσιμων δυνάμεων που αναδιαμορφώνουν την ιατρική του 21ου αιώνα: η δημογραφική γήρανση, τα επίμονα εμπόδια στην πρόσβαση στην υγειονομική περίθαλψη και η επιταχυνόμενη υιοθέτηση της τεχνολογίας. Η συνάντηση αυτών των τάσεων δημιουργεί σοβαρές προκλήσεις, ανοίγει όμως και σημαντικά περιθώρια καινοτομίας στην τεχνολογία φροντίδας ηλικιωμένων.

1.1.1 Η παγκόσμια κρίση γήρανσης

Ο κόσμος βιώνει μια άνευ προηγουμένου δημογραφική μεταβολή. Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας (ΠΟΥ), το ποσοστό του παγκόσμιου πληθυσμού ηλικίας 60 ετών και άνω θα σχεδόν διπλασιαστεί από 12% το 2015 σε 22% έως το 2050, αντιπροσωπεύοντας περίπου 2 δισεκατομμύρια άτομα. Σε πολλές βιομηχανικές χώρες, οι πολίτες άνω των 65 ετών αντιπροσωπεύουν ήδη το 20-25% του πληθυσμού, με προβλέψεις που δείχνουν ότι ο αριθμός αυτός θα υπερβεί το 30% έως το 2050.

Αυτό το «ασημένιο κύμα» έχει βαθιές επιπτώσεις στα συστήματα υγειονομικής περίθαλψης. Στις Ηνωμένες Πολιτείες, οι ενήλικες άνω των 65 ετών καταναλώνουν τρεις έως πέντε φορές περισσότερους πόρους υγειονομικής περίθαλψης από τους νεότερους ενήλικες, ενώ οι χρόνιες ασθένειες επηρεάζουν το 80% των ατόμων αυτής της ηλικιακής ομάδας, με το 68% να αντιμετωπίζει δύο ή περισσότερες παθήσεις ταυτόχρονα. Το ετήσιο κόστος υγειονομικής περίθαλψης για τους ηλικιωμένους ανέρχεται κατά μέσο όρο σε 19.098 δολάρια ανά άτομο, έναντι 7.153 δολαρίων για τους ενήλικες σε ηλικία εργασίας, ενώ έως το 2030 εκτιμάται ότι 71,5 εκατομμύρια Αμερικανοί θα είναι 65 ετών και άνω, αριθμός διπλάσιος από εκείνον του 2000.

Το φαινόμενο έχει παγκόσμια διάσταση. Η Ιαπωνία προηγείται, με το 28,7% του πληθυσμού της να είναι άνω των 65 ετών, γεγονός που απαιτεί καινοτόμα μοντέλα παροχής φροντίδας, ενώ οι ευρωπαϊκές χώρες εμφανίζουν κατά μέσο όρο ποσοστά ηλικιωμένου πληθυσμού της τάξης του 20-24%, επιβαρύνοντας τις υποδομές κοινωνικής φροντίδας. Παράλληλα, οι αναπτυσσόμενες χώρες αντιμετωπίζουν επιταχυνόμενους ρυθμούς γήρανσης, με σημαντικά λιγότερο χρόνο προετοιμασίας από ό,τι είχαν στη διάθεσή τους οι ανεπτυγμένες χώρες.

1.1.2 Το βάρος της διαχείρισης των χρόνιων ασθενειών

Οι ηλικιωμένοι πληθυσμοί αντιμετωπίζουν σύνθετα, αλληλένδετα προβλήματα υγείας που απαιτούν συνεχή παρακολούθηση:

Στο πεδίο των καρδιαγγειακών παθήσεων, η υπέρταση επηρεάζει το 65% των ενηλίκων άνω των 60 ετών, ενώ η συχνότητα της καρδιακής ανεπάρκειας διπλασιάζεται κάθε δεκαετία μετά την ηλικία των 65. Η σημασία της συστηματικής παρακολούθησης είναι τεκμηριωμένη: η καθημερινή μέτρηση της αρτηριακής πίεσης μειώνει τα καρδιαγγειακά επεισόδια κατά 23-30%, ενώ η έγκαιρη ανίχνευση αρρυθμιών μπορεί να προλάβει έως και το 40% των περιπτώσεων εγκεφαλικού επεισοδίου.

Ανάλογη εικόνα παρουσιάζουν οι μεταβολικές διαταραχές. Ο διαβήτης τύπου 2 επηρεάζει το 25% των ηλικιωμένων και απαιτεί παρακολούθηση πολλαπλών παραμέτρων, όπως η γλυκόζη, το σωματικό βάρος, η φυσική δραστηριότητα και η τήρηση της φαρμακευτικής αγωγής. Η απομακρυσμένη παρακολούθηση έχει αποδειχθεί ότι βελτιώνει τον γλυκαιμικό έλεγχο, επιφέροντας μείωση της HbA1c κατά 0,5-1,0%, ενώ η υιοθέτηση της συνεχούς παρακολούθησης της γλυκόζης αυξάνεται κατά 35% ετησίως.

Εξίσου κρίσιμη είναι η γνωστική και η αναπνευστική διάσταση. Το 15-20% των ενηλίκων άνω των 65 ετών παρουσιάζει ήπια γνωστική εξασθένηση· η έγκαιρη ανίχνευσή της μέσω ανάλυσης προτύπων συμπεριφοράς μπορεί να καθυστερήσει την εξέλιξή της κατά τρία έως πέντε χρόνια, ενώ ο συνδυασμός φυσικής και γνωστικής παρακολούθησης βελτιώνει την ακρίβεια της διάγνωσης. Παράλληλα, η χρόνια αποφρακτική πνευμονοπάθεια (ΧΑΠ) επηρεάζει το 12% των ενηλίκων άνω των 65 ετών· για τη διαχείρισή της, η παρακολούθηση του κορεσμού οξυγόνου είναι κρίσιμη, με την απομακρυσμένη παρακολούθηση να μειώνει τις επανεισαγωγές στο νοσοκομείο κατά 40-50%.

1.1.3 Το ψηφιακό χάσμα στην τεχνολογία της υγειονομικής περίθαλψης

Παρά τις τεχνολογικές εξελίξεις, οι ηλικιωμένοι αντιμετωπίζουν σημαντικά εμπόδια στην υιοθέτηση των τεχνολογιών, τα οποία περιορίζουν την αποτελεσματικότητα των σημερινών ψηφιακών λύσεων.

Οι προκλήσεις προσβασιμότητας είναι εκτεταμένες. Το 58% των ενηλίκων άνω των 65 ετών αναφέρει δυσκολία στη χρήση τυπικών εφαρμογών υγείας, ενώ οι μικρές γραμματοσειρές (κάτω των 14 στιγμών) δημιουργούν εμπόδια αναγνωσιμότητας για το 73% των ηλικιωμένων. Οι χαμηλές αναλογίες αντίθεσης (κάτω από 4,5:1) μειώνουν την ορατότητα για τους χρήστες με καταρράκτη και οι πολύπλοκες δομές πλοήγησης, με περισσότερα από τρία επίπεδα βάθους, προκαλούν σύγχυση στο 68% των ηλικιωμένων χρηστών. Το αποτέλεσμα είναι το 73% των ηλικιωμένων να εγκαταλείπει μια νέα εφαρμογή μέσα στην πρώτη εβδομάδα χρήσης, καθώς το γνωστικό φορτίο των σύγχρονων διεπαφών υπερβαίνει τα όρια άνεσης για το 65% των χρηστών.

Στο πρόβλημα συμβάλλει και το χάσμα τεχνολογικής παιδείας: μόνο το 42% των ηλικιωμένων αισθάνεται σιγουριά στην ανεξάρτητη χρήση ψηφιακών εργαλείων υγείας, ενώ ο φόβος μη αναστρέψιμων λαθών εμποδίζει το 67% να ασχοληθεί ουσιαστικά με αυτά. Η έλλειψη καθοδήγησης ανάλογης με το πλαίσιο χρήσης αυξάνει την απογοήτευση και την εγκατάλειψη, η περιορισμένη έκθεση σε διεπαφές αφής μειώνει την επάρκεια χειρισμού και τα ασυνεπή πρότυπα σχεδιασμού μεταξύ των εφαρμογών επιβάλλουν διαρκώς νέα εκμάθηση.

Σημαντικό ρόλο διαδραματίζουν, τέλος, οι οπτικές και κινητικές δυσκολίες που συνοδεύουν τη γήρανση: η πρεσβυωπία επηρεάζει το σύνολο σχεδόν των ενηλίκων άνω των 50 ετών, ο καταρράκτης μειώνει την ευαισθησία στην αντίθεση στο 50% των ατόμων άνω των 75 ετών, η εκφύλιση της ωχράς κηλίδας αφορά το 11% των ενηλίκων άνω των 65 ετών, ενώ η αρθρίτιδα - που πλήττει το 49% των ηλικιωμένων - και ο τρόμος δυσχεραίνουν την ακριβή αφή μικρών στόχων. Παρά ταύτα, οι περισσότερες εφαρμογές υγείας σχεδιάζονται για χρήστες ηλικίας 25-45 ετών: οι στόχοι αφής είναι μικρότεροι από τα συνιστώμενα ελάχιστα, τα χρονικά όρια αλληλεπίδρασης είναι πολύ σύντομα για χαμηλότερες ταχύτητες χειρισμού, η γνωστική απλοποίηση σπάνια αποτελεί προτεραιότητα και τα μηνύματα σφάλματος είναι ασαφή και προκαλούν άγχος αντί να καθοδηγούν.

1.1.4 Το βάρος των φροντιστών και η αναγκαιότητα της απομακρυσμένης παρακολούθησης

Οι οικογενειακοί φροντιστές αποτελούν τη ραχοκοκαλιά των συστημάτων υποστήριξης των ηλικιωμένων, αλλά αντιμετωπίζουν τεράστιες προκλήσεις που η τεχνολογία μπορεί να βοηθήσει να αντιμετωπιστούν.

Η κλίμακα του φαινομένου είναι εντυπωσιακή: 53 εκατομμύρια Αμερικανοί παρέχουν μη αμειβόμενη φροντίδα σε ενήλικες, κυρίως ηλικιωμένους γονείς, αφιερώνοντας κατά μέσο όρο 24,4 ώρες την εβδομάδα σε δραστηριότητες φροντίδας, ενώ το 56% εξ αυτών διατηρεί παράλληλα πλήρη απασχόληση. Το προσωπικό κόστος είναι βαρύ: το 60% των φροντιστών αναφέρει υψηλό ή πολύ υψηλό συναισθηματικό στρες, το 40% εμφανίζει κατάθλιψη συνδεδεμένη με τις ευθύνες φροντίδας και το 23% αναφέρει επιδείνωση της δικής του υγείας. Η οικονομική αξία αυτής της μη αμειβόμενης φροντίδας εκτιμάται σε 470 δισεκατομμύρια δολάρια ετησίως μόνο στις Ηνωμένες Πολιτείες.

Το έργο των φροντιστών δυσχεραίνεται περαιτέρω από τον κατακερματισμό της επικοινωνίας. Η εμπλοκή πολλών κλινικών ιατρών, μελών της οικογένειας και παρόχων υπηρεσιών δημιουργεί προβλήματα συντονισμού, ενώ η έλλειψη κεντρικής πληροφόρησης για την υγεία ευθύνεται για το 31% των επισκέψεων στα τμήματα επείγοντων περιστατικών που θα μπορούσαν να είχαν αποφευχθεί. Η καθυστερημένη αναγνώριση της επιδείνωσης της υγείας μειώνει την αποτελεσματικότητα της παρέμβασης κατά 40%, η γεωγραφική απόσταση - κατά μέσο όρο 183 μίλια μεταξύ φροντιστών και ηλικιωμένων γονέων - επιβάλλει λύσεις εξ αποστάσεως, ενώ χωρίς απομακρυσμένη παρακολούθηση ο μέσος χρόνος απόκρισης σε ανιχνευθείσες ανωμαλίες φθάνει τις 4,7 ώρες.

Τα οφέλη της απομακρυσμένης παρακολούθησης είναι τεκμηριωμένα: η συνεχής παρακολούθηση μειώνει τις επανεισαγωγές στο νοσοκομείο κατά 30-50%, η έγκαιρη ανίχνευση ανωμαλιών επιτρέπει προληπτική παρέμβαση δύο έως πέντε ημέρες νωρίτερα και η ορατότητα της κατάστασης σε πραγματικό χρόνο μειώνει το άγχος των φροντιστών κατά 47%. Η ανάλυση των διαχρονικών τάσεων υποστηρίζει τη λήψη κλινικών αποφάσεων, ενώ η αποφυγή περιττών επισκέψεων στα επείγοντα αποφέρει εξοικονόμηση περίπου 3.200 δολαρίων ανά ασθενή

ετησίως και οι δείκτες ικανοποίησης των ασθενών βελτιώνονται αισθητά (72% έναντι 54% της παραδοσιακής περίθαλψης). Την ίδια στιγμή, οι προκλήσεις συντονισμού παραμένουν σημαντικές: ο μέσος ηλικιωμένος ασθενής επισκέπτεται επτά διαφορετικούς παρόχους υγειονομικής περίθαλψης, σφάλματα στην εναρμόνιση της φαρμακευτικής αγωγής σημειώνονται στο 45% των μεταβάσεων φροντίδας, η έλλειψη κοινών σχεδίων φροντίδας προκαλεί διπλές εξετάσεις στο 28% των περιπτώσεων και η κατακερματισμένη επικοινωνία αυξάνει τον χρόνο απασχόλησης των φροντιστών κατά 35%.

1.1.5 Κενά στο τρέχον τεχνολογικό τοπίο

Οι υπάρχουσες ψηφιακές λύσεις υγείας παρουσιάζουν κρίσιμες ελλείψεις όταν εξυπηρετούν ηλικιωμένους πληθυσμούς και τους φροντιστές τους:

Η πρώτη βασική αδυναμία είναι ο κατακερματισμός μεταξύ των κατηγοριών λύσεων. Οι πλατφόρμες φυσικής κατάστασης για καταναλωτές (Fitbit, Apple Health, Google Fit) διαθέτουν πλούσια οικοσυστήματα συσκευών, τυποποιημένα API και στοιχεία παιχνιδοποίησης, ωστόσο προσφέρουν ελάχιστες δυνατότητες συνεργασίας με τους φροντιστές, απευθύνονται κυρίως σε νεότερους και ενεργούς χρήστες, έχουν περιορισμένο κλινικό βάθος, χρησιμοποιούν ιδιόκτητες μορφές δεδομένων που περιορίζουν τη διαλειτουργικότητα και δεν διαθέτουν ενσωματωμένη καθοδήγηση υγείας με τεχνητή νοημοσύνη.

Οι κλινικές πύλες ασθενών (Epic MyChart, Cerner HealtheLife) προσφέρουν πλήρη ενσωμάτωση με τα ηλεκτρονικά αρχεία υγείας, ανταλλαγή μηνυμάτων με τους κλινικούς ιατρούς και πρόσβαση σε αποτελέσματα εξετάσεων, αλλά είναι προσανατολισμένες στις προγραμματισμένες επισκέψεις και όχι στη συνεχή παρακολούθηση, προσφέρουν κακή εμπειρία χρήσης στους ηλικιωμένους (μέση βαθμολογία προσβασιμότητας 62/100) και περιορίζουν την κοινή χρήση με την οικογένεια σε μεμονωμένους πληρεξουσίου. Οι πίνακες ελέγχου των προμηθευτών φορετών συσκευών αυτοματοποιούν τη συλλογή δεδομένων και την οπτικοποίηση τάσεων, αλλά συνεπάγονται εξάρτηση από τον προμηθευτή, κρύβουν βασικές λειτουργίες πίσω από συνδρομές, περιορίζουν την πρόσβαση των φροντιστών και την εξαγωγή δεδομένων, ενώ οι ηλικιωμένοι είναι λιγότερο πιθανό να φορούν τις συσκευές με συνέπεια.

Οι συσκευές επικοινωνίας για ηλικιωμένους (GrandPad, Jitterbug) προσφέρουν απλοποιημένες διεπαφές, μεγάλα κουμπιά και ειδική υποστήριξη, αλλά η ανάλυση υγείας που παρέχουν είναι επιφανειακή, τα οικοσυστήματά τους κλειστά και η ενσωμάτωση τεχνητής νοημοσύνης ελάχιστη. Τέλος, οι πλατφόρμες τηλεϊατρικής (Teladoc, Amwell, MDLive) εξασφαλίζουν σύγχρονη πρόσβαση σε κλινικούς ιατρούς μέσω τηλεδιάσκεψης, αλλά προσφέρουν επεισοδιακή και όχι συνεχή φροντίδα, στερούνται πινάκων ελέγχου για την παρακολούθηση τάσεων, προβλέπουν ελάχιστη συμμετοχή των φροντιστών και ενσωματώνουν περιορισμένα δεδομένα υγείας πέραν των αυτοαναφερόμενων συμπτωμάτων.

Η δεύτερη αδυναμία αφορά την ιδιωτικότητα και την εμπιστοσύνη. Οι αρχιτεκτονικές που βασίζονται στο cloud εκθέτουν τις προστατευόμενες πληροφορίες υγείας (PHI) σε εξωτερικούς παρόχους, τη στιγμή που οι ηλικιωμένοι εκφράζουν αυξημένη ευαισθησία στην προστασία της ιδιωτικής ζωής (το 71% ανησυχεί για κατάχρηση δεδομένων) και μόνο το 34% εμπιστεύεται τις συστάσεις της τεχνητής νοημοσύνης, εξαιτίας και της έλλειψης διαφάνειας στη λήψη αποφάσεων. Η πολυπλοκότητα των κανονιστικών πλαισίων (HIPAA, GDPR) δημιουργεί αβεβαιότητα συμμόρφωσης για τους μικρότερους παρόχους, οι πρακτικές κοινής χρήσης δεδομένων με τρίτους συχνά αποκρύπτονται σε μακροσκελείς όρους χρήσης, ενώ το 58% των ερωτηθέντων ανησυχεί για ενδεχόμενη πρόσβαση των ασφαλιστικών εταιρειών στα δεδομένα υγείας του.

Η τρίτη αδυναμία είναι η ανεπαρκής προσαρμογή της τεχνητής νοημοσύνης στο πλαίσιο του κάθε ασθενούς. Τα γενικά chatbots δεν ενσωματώνουν το προσωπικό πλαίσιο υγείας, τα εξωτερικά API μεγάλων γλωσσικών μοντέλων διακινδυνεύουν διαρροή PHI χωρίς ρητές διασφαλίσεις ή συγκατάθεση, η έλλειψη εξειδίκευσης στον τομέα της υγείας οδηγεί σε άσχετες ή και επικίνδυνες απαντήσεις, ενώ απουσιάζουν τόσο η σύνδεση της συνομιλίας με τις μετρήσεις του συγκεκριμένου ασθενούς όσο και οι ιατρικές δηλώσεις αποποίησης ευθύνης με παραπομπή

σε επαγγελματίες υγείας. Οι κίνδυνοι «ψευδαισθήσεων» των μη ελεγχόμενων μοντέλων επιτείνουν το πρόβλημα, καθώς ενδέχεται να παρέχονται λανθασμένες πληροφορίες υγείας.

Η τέταρτη αδυναμία εντοπίζεται στα περιορισμένα μοντέλα συνεργασίας των οικογενειών: λίγες πλατφόρμες υποστηρίζουν πίνακες ελέγχου πολλαπλών ασθενών, οι ροές εργασίας προσκλήσεων και συγκατάθεσης είναι κακοσχεδιασμένες ή ανύπαρκτες και η συγκεντρωτική, συγκριτική ανάλυση μεταξύ των μελών του νοικοκυριού είναι ανεπαρκής. Επιπλέον, η κόπωση από τις ειδοποιήσεις που βασίζονται σε απλά κατώφλια είναι τόσο διαδεδομένη ώστε το 94% των ειδοποιήσεων να αγνοείται, ελλείπει προτεραιοποίησης και συγκριτικής νοημοσύνης, ενώ απουσιάζουν και λειτουργίες συνεργατικού σχεδιασμού της φροντίδας.

1.1.6 Η αναγκαιότητα για ολοκληρωμένες λύσεις

Αυτές οι συγκλίνουσες πιέσεις δημογραφική γήρανση, βάρος χρόνιων ασθενειών, κενά προσβασιμότητας, άγχος φροντιστών, κατακερματισμός τεχνολογίας και ανησυχίες για την προστασία της ιδιωτικής ζωής δημιουργούν μια σαφή ανάγκη για μια νέα γενιά τεχνολογία για την υγεία των ηλικιωμένων που συνθέτει τις επί του παρόντος περιορισμένες δυνατότητες.

Μια τέτοια λύση οφείλει, πρώτον, να υιοθετεί σχεδιασμό με προτεραιότητα στην προσβασιμότητα: μεγάλη τυπογραφία με βασική γραμματοσειρά 18-24 στιγμών, θέματα υψηλής αντίθεσης με ελάχιστη αναλογία 7:1, απλοποιημένη πλοήγηση βάθους έως τριών επιπέδων, υποστήριξη αλληλεπίδρασης μέσω πληκτρολογίου, στόχους αφής τουλάχιστον 44x44 εικονοστοιχείων και συμβατότητα με προγράμματα ανάγνωσης οθόνης. Δεύτερον, απαιτείται αρχιτεκτονική που διαφυλάσσει την ιδιωτικότητα, με τοπική εκτέλεση της τεχνητής νοημοσύνης ώστε να αποφεύγεται η εξωτερική μετάδοση προστατευόμενων πληροφοριών υγείας, ελάχιστη συλλογή δεδομένων, διαφανείς πολιτικές διαχείρισης, κρυπτογράφηση τόσο κατά την αποθήκευση όσο και κατά τη μεταφορά, και ευθυγράμμιση με τα πλαίσια HIPAA και GDPR.

Τρίτον, η συνεργασία των φροντιστών πρέπει να αποτελεί δομικό στοιχείο του συστήματος, με πίνακες ελέγχου πολλαπλών ασθενών, ασφαλείς ροές εργασίας προσκλήσεων και συγκατάθεσης, συγκεντρωτική απεικόνιση των τάσεων υγείας, συγκριτική ανάλυση σε επίπεδο νοικοκυριού, έξυπνη ιεράρχηση των προειδοποιήσεων και κοινά εργαλεία σχεδιασμού της φροντίδας. Τέταρτον, η βοήθεια τεχνητής νοημοσύνης οφείλει να είναι προσαρμοσμένη στο πλαίσιο του ασθενούς, δηλαδή να θεμελιώνεται στις πραγματικές μετρήσεις υγείας του, να εξατομικεύεται, να συνοδεύεται από σαφείς ιατρικές δηλώσεις αποποίησης ευθύνης, να έχει εκπαιδευτικό χαρακτήρα και να εκτελείται τοπικά με σεβασμό στην ιδιωτικότητα. Τέλος, απαιτείται ευέλικτη και συντηρήσιμη αρχιτεκτονική: εφαρμογή σε cloud-native containers, δυνατότητες οριζόντιας κλιμάκωσης, υποδομή ως κώδικας για επαναληψιμότητα, πλήρης παρατηρησιμότητα (observability) και σύγχρονες τεχνολογικές επιλογές σε ολόκληρη τη στοίβα.

Η παρούσα διατριβή αντιμετωπίζει αυτές τις επιταγές μέσω του ολοκληρωμένου σχεδιασμού, της υλοποίησης και της αξιολόγησης ενός πίνακα ελέγχου παρακολούθησης της υγείας ηλικιωμένων, ο οποίος συνθέτει την προσβασιμότητα, την τεχνητή νοημοσύνη που προστατεύει την ιδιωτικότητα, τη συνεργασία των φροντιστών και την επεκτάσιμη αρχιτεκτονική σε μια ενοποιημένη πλατφόρμα.

1.2 Περιγραφή του προβλήματος

Η προηγούμενη ανάλυση του ιστορικού αποκρυσταλλώνει ένα πολυδιάστατο πρόβλημα που οι τρέχουσες τεχνολογικές λύσεις στον τομέα της υγειονομικής περίθαλψης δεν καταφέρνουν να αντιμετωπίσουν επαρκώς:

1.2.1 Ορισμός του βασικού προβλήματος

Κύριο πρόβλημα: Οι ηλικιωμένοι πληθυσμοί απαιτούν συνεχή παρακολούθηση της υγείας τους για τη διαχείριση χρόνιων παθήσεων και τη δυνατότητα να γερνούν στο σπίτι τους· ωστόσο, οι υπάρχουσες ψηφιακές τεχνολογίες υγείας αδυνατούν συστηματικά να εξυπηρετήσουν

αποτελεσματικά αυτή τη δημογραφική ομάδα. Η αδυναμία αυτή οφείλεται, πρώτον, σε ελλείψεις προσβασιμότητας, καθώς οι τυπικές διεπαφές εφαρμογών υγείας δημιουργούν ανυπερβλήτα εμπόδια για χρήστες με ηλικιακή έκπτωση όρασης, κινητικότητας και γνωστικής λειτουργίας. Δεύτερον, στην έκθεση της ιδιωτικής ζωής, αφού οι αρχιτεκτονικές τεχνητής νοημοσύνης και αναλυτικής που εξαρτώνται από το cloud θέτουν σε κίνδυνο τις προστατευόμενες πληροφορίες υγείας. Τρίτον, στον αποκλεισμό των φροντιστών: οι πλατφόρμες που σχεδιάζονται για ατομική χρήση παραμελούν τον κρίσιμο ρόλο των οικογενειακών φροντιστών, οι οποίοι χρειάζονται εποπτεία πολλών ασθενών, συνεργατικά εργαλεία και έξυπνους μηχανισμούς ειδοποίησης. Τέταρτον, στην αποσύνδεση από το πλαίσιο, καθώς τα γενικά chatbots και οι βοηθοί υγείας δεν συνδέονται με τις συγκεκριμένες μετρήσεις του κάθε ασθενούς, με αποτέλεσμα μειωμένη συνάφεια και αξιοπιστία. Πέμπτον, στον κατακερματισμό των δεδομένων, αφού οι απομονωμένες πλατφόρμες εμποδίζουν τη διαχρονική, ολιστική ανάλυση των τάσεων υγείας στις διάφορες κατηγορίες μετρήσεων (καρδιαγγειακές, μεταβολικές, δραστηριότητα, ύπνος).

1.2.2 Συγκεκριμένες ελλείψεις των υφιστάμενων λύσεων

Τα εμπόδια προσβασιμότητας έχουν ως αποτέλεσμα ποσοστό αποτυχίας υιοθέτησης της τάξης του 58%. Τα μικρά γράμματα και η χαμηλή αντίθεση εμποδίζουν το 73% των ηλικιωμένων να διαβάσει το περιεχόμενο, τα πολύπλοκα μοτίβα πλοήγησης προκαλούν σύγχυση στο 68% των χρηστών, τα χρονικά όρια λειτουργούν εις βάρος όσων αλληλεπιδρούν με χαμηλότερη ταχύτητα, οι μικροί στόχοι αφής δυσκολεύουν τους χρήστες με αρθρίτιδα, τα μηνύματα σφάλματος προκαλούν άγχος αντί να παρέχουν σαφείς οδηγίες και δεν προβλέπεται εξατομίκευση για τις ατομικές ανάγκες προσβασιμότητας.

Σοβαρές είναι και οι ευπάθειες ασφάλειας και απορρήτου: οι εξαρτήσεις από εξωτερικά API τεχνητής νοημοσύνης εκθέτουν τα προσωπικά δεδομένα υγείας σε τρίτους, οι ασαφείς πρακτικές διαχείρισης δεδομένων υπονομεύουν την εμπιστοσύνη - μόνο το 34% εμπιστεύεται τις συστάσεις της τεχνητής νοημοσύνης για την υγεία - ενώ παρατηρούνται ανεπαρκής διαφάνεια ως προς τον τρόπο με τον οποίο τα δεδομένα τροφοδοτούν αλγοριθμικές αποφάσεις, ασαφής συμμόρφωση με τα πλαίσια HIPAA και GDPR και ανεπαρκής έλεγχος των χρηστών όσον αφορά την εξαγωγή, τη διαγραφή και την κοινή χρήση των δεδομένων τους.

Στην επικοινωνία ασθενών και φροντιστών, η μέση απόσταση των 183 μιλίων μεταξύ φροντιστών και ηλικιωμένων γονέων επιβάλλει λύσεις εξ αποστάσεως, ωστόσο η έλλειψη ορατότητας της κατάστασης υγείας σε πραγματικό χρόνο καθυστερεί την παρέμβαση κατά 4,7 ώρες κατά μέσο όρο. Δεν υπάρχει κεντρική πλατφόρμα συντονισμού της φροντίδας μεταξύ των μελών της οικογένειας, οι πληροφορίες κατακερματίζονται μεταξύ επτά παρόχων κατά μέσο όρο ανά ηλικιωμένο ασθενή και η κόπωση από τις ειδοποιήσεις απλού κατωφλίου οδηγεί στην αγνόηση του 94% εξ αυτών.

Παράλληλα, απουσιάζουν οι πληροφορίες υγείας σε πραγματικό χρόνο και με επίγνωση του πλαισίου: οι περιστασιακές τηλεϊατρικές συμβουλές δεν εντοπίζουν τη σταδιακή επιδείνωση, οι πλατφόρμες φορετών συσκευών παρέχουν ακατέργαστα δεδομένα χωρίς καθοδήγηση ερμηνείας για μη κλινικούς χρήστες, οι βοηθοί τεχνητής νοημοσύνης δεν ενσωματώνουν τις τιμές αναφοράς και το ιατρικό ιστορικό του κάθε ασθενούς και δεν υπάρχει ανίχνευση ανωμαλιών προσαρμοσμένη στα ατομικά φυσιολογικά όρια. Τέλος, η συνολική πολυπλοκότητα αποτρέπει την υιοθέτηση: το 73% των ηλικιωμένων εγκαταλείπει τις εφαρμογές εντός μίας εβδομάδας, ο φόβος μη αναστρέψιμων σφαλμάτων αποθαρρύνει το 67% από τη συμμετοχή, οι καμπύλες εκμάθησης είναι ασύμβατες με τα επεισοδικά πρότυπα χρήσης και οι μορφές σχεδιασμού παραμένουν ασυνεπείς μεταξύ των εφαρμογών του οικοσυστήματος υγείας.

1.2.3 Συνέπειες των ανεπίλυτων ελλείψεων

Η συνδυαστική επίδραση αυτών των ελλείψεων παράγει μετρήσιμα αρνητικά αποτελέσματα:

Ως προς τα αποτελέσματα υγείας, η καθυστερημένη ανίχνευση της επιδείνωσης οδηγεί σε νοσηλείες που θα μπορούσαν να αποφευχθούν (αύξηση 31%), η διαχείριση των χρόνιων παθήσεων είναι υποβέλτιστη (κατά 0,8% υψηλότερη HbA1c και κατά 7 mmHg υψηλότερη συστολική αρτηριακή πίεση), η τήρηση της φαρμακευτικής αγωγής μειώνεται ελλείψει

υπενθυμίσεων και παρακολούθησης (ποσοστό μη τήρησης 23%) και η χρήση των τμημάτων επειγόντων περιστατικών αυξάνεται λόγω της απουσίας συστημάτων έγκαιρης προειδοποίησης.

Όσον αφορά την επιβάρυνση των φροντιστών, το 60% αναφέρει υψηλά επίπεδα συναισθηματικού στρες, το 40% εμφανίζει κατάθλιψη σχετιζόμενη με τη φροντίδα, το 23% βιώνει επιδείνωση της δικής του υγείας και ο χρόνος συντονισμού αυξάνεται κατά 35% εξαιτίας των κατακερματισμένων πληροφοριών, με τον ετήσιο οικονομικό αντίκτυπο της μη αμειβόμενης φροντίδας να εκτιμάται σε 470 δισεκατομμύρια δολάρια. Σε επίπεδο συστήματος υγείας, οι αποτρέψιμες επανεισαγωγές κοστίζουν 17,4 δισεκατομμύρια δολάρια ετησίως, οι εξετάσεις επαναλαμβανονται λόγω κατακερματισμού των πληροφοριών και οι πόροι αξιοποιούνται αναποτελεσματικά, καθώς η φροντίδα παραμένει αντιδραστική αντί προληπτική. Τέλος, οι αποτυχίες στην υιοθέτηση της τεχνολογίας μεταφράζονται σε σπατάλη επενδύσεων σε αχρησιμοποίητα ψηφιακά εργαλεία, σε διεύρυνση του χάσματος ψηφιακής ισότητας στην υγεία για τους ηλικιωμένους και σε χαμένες ευκαιρίες για προληπτικές παρεμβάσεις βάσει δεδομένων.

1.2.4 Έλλειψη έρευνας

Η ακαδημαϊκή βιβλιογραφία ασχολείται με τμήματα αυτού του προβλήματος αρχές σχεδιασμού προσβασιμότητας, αρχιτεκτονικές απομακρυσμένης παρακολούθησης, τεχνολογίες υποστήριξης φροντιστών, τεχνητή νοημοσύνη που προστατεύει την ιδιωτικότητα αλλά δεν υπάρχει ολοκληρωμένη ενσωμάτωση σε ένα ενοποιημένο, έτοιμο για παραγωγή σύστημα προσαρμοσμένο στις συνθήκες φροντίδας των ηλικιωμένων. Τα υπάρχοντα πρωτότυπα και ερευνητικά συστήματα σπάνια προχωρούν πέρα από πιλοτικές εφαρμογές, ενώ οι εμπορικές λύσεις είναι βελτιστοποιημένες για νεότερους, τεχνολογικά καταρτισμένους πληθυσμούς.

Παραμένουν, συνεπώς, ανοιχτά ερευνητικά ερωτήματα: πώς μπορεί να παρέχεται βοήθεια τεχνητής νοημοσύνης με προστασία της ιδιωτικότητας, χωρίς εξάρτηση από API μεγάλων γλωσσικών μοντέλων στο cloud· ποια πρότυπα διεπαφής χρήστη εξυπηρετούν καλύτερα τους ηλικιωμένους διατηρώντας παράλληλα τον πλούτο των λειτουργιών· πώς πρέπει οι πίνακες ελέγχου πολλαπλών ασθενών για φροντιστές να εξισορροπούν την εποπτεία της συνολικής εικόνας με την ιδιωτικότητα· ποια αρχιτεκτονικά πρότυπα επιτρέπουν κλιμακούμενη, ασφαλή και επεκτάσιμη παρακολούθηση της υγείας· και πώς μπορεί η ανίχνευση ανωμαλιών στις μετρήσεις υγείας να ελαχιστοποιεί τα ψευδώς θετικά αποτελέσματα διατηρώντας την ευαισθησία της.

Η παρούσα διατριβή γεφυρώνει αυτά τα κενά μέσω μιας ολοκληρωμένης μελέτης περίπτωσης: του Πίνακα ελέγχου παρακολούθησης της υγείας των ηλικιωμένων.

1.3 Σκοπός και στόχοι της διατριβής

1.3.1 Κύριος σκοπός

Ο κύριος σκοπός αυτής της διατριβής είναι ο σχεδιασμός, η υλοποίηση και η αξιολόγηση ενός ολοκληρωμένου πίνακα ελέγχου παρακολούθησης της υγείας των ηλικιωμένων, ο οποίος αντιμετωπίζει τις διαπιστωμένες ελλείψεις στην τρέχουσα ψηφιακή τεχνολογία υγείας για τους ηλικιωμένους και τους φροντιστές τους. Αυτή η εργασία καταδεικνύει πώς οι σύγχρονες πρακτικές ανάπτυξης ιστού πλήρους στοίβας, οι αρχιτεκτονικές τεχνητής νοημοσύνης που προστατεύουν την ιδιωτικότητα και οι αρχές σχεδιασμού με επίκεντρο τον άνθρωπο μπορούν να συντεθούν σε ένα σύστημα έτοιμο για παραγωγή που εξυπηρετεί μια ευάλωτη δημογραφική ομάδα.

1.3.2 Ειδικοί στόχοι

Η διατριβή επιδιώκει οκτώ αλληλένδετους στόχους που καλύπτουν το σχεδιασμό, την υλοποίηση, την αξιολόγηση και τη συμβολή γνώσης του συστήματος:

Ο πρώτος στόχος είναι η δημιουργία ασφαλούς πλατφόρμας παρακολούθησης της υγείας με πολλαπλούς ρόλους, μέσω ελέγχου πρόσβασης βάσει ρόλων (RBAC) που υποστηρίζει

ασθενείς και φροντιστές με διακριτές δυνατότητες και διεπαφές. Τεχνικά, αυτό προϋποθέτει αυθεντικοποίηση JWT, κατακερματισμό κωδικών πρόσβασης με bcrypt, ενσωμάτωση των ρόλων στις αξιώσεις των tokens και εξουσιοδότηση σε επίπεδο middleware· κριτήριο επιτυχίας αποτελεί η σαφής διαφοροποίηση μεταξύ των ρόλων αυτοπαρακολούθησης του ασθενούς και της εποπτείας πολλαπλών ασθενών από τον φροντιστή. Ο δεύτερος στόχος αφορά τη δομημένη καταγραφή και οπτικοποίηση δεικτών υγείας - καρδιαγγειακών, μεταβολικών, δραστηριότητας και ύπνου - με σχεσιακό σχήμα βελτιστοποιημένο για χρονοσειρές, RESTful API και διαδραστική βιβλιοθήκη γραφημάτων, ώστε να υποστηρίζονται η αρτηριακή πίεση, ο καρδιακός ρυθμός, το βάρος, ο κορεσμός οξυγόνου, τα βήματα και τα στάδια ύπνου με απεικόνιση των ιστορικών τάσεων.

Ο τρίτος στόχος είναι η καθοδήγηση των ασθενών με τη βοήθεια τεχνητής νοημοσύνης: ένας συνομιλητικός βοηθός θεμελιωμένος στις μετρήσεις υγείας, υλοποιημένος με τοπική ανάπτυξη μεγάλου γλωσσικού μοντέλου (Ollama), δυναμική συγκρότηση πλαισίου από τα δεδομένα του ασθενούς και κατάλληλη μηχανική προτροπών με ιατρικές δηλώσεις αποποίησης ευθύνης, ώστε οι απαντήσεις να ενσωματώνουν τις εξατομικευμένες μετρήσεις χωρίς καμία εξωτερική μετάδοση προστατευόμενων πληροφοριών υγείας. Συναφής είναι ο τέταρτος στόχος, η αρχιτεκτονική προστασίας της ιδιωτικότητας: εκτέλεση του μοντέλου τεχνητής νοημοσύνης εντός της υποδομής, κρυπτογράφηση κατά τη μεταφορά (HTTPS), αποθήκευση διαπιστευτηρίων με κατακερματισμό και ελάχιστη συλλογή δεδομένων, με κριτήρια επιτυχίας τις μηδενικές εξωτερικές κλήσεις API που περιέχουν PHI, το τεκμηριωμένο μοντέλο απειλών και την ανάλυση ευθυγράμμισης με HIPAA και GDPR.

Ο πέμπτος στόχος αφορά την προσβασιμότητα και τη συμπεριληπτικότητα, με συμμόρφωση προς το WCAG 2.1 επιπέδου AA, ρυθμιζόμενα μεγέθη γραμματοσειράς, θέματα υψηλής αντίθεσης, πλοήγηση μέσω πληκτρολογίου και ετικέτες ARIA, ώστε οι ηλικιωμένοι χρήστες με περιορισμούς όρασης, κινητικότητας ή γνωστικής λειτουργίας να συμμετέχουν αποτελεσματικά· ως κριτήρια ορίζονται βαθμολογία ελέγχου προσβασιμότητας τουλάχιστον 90% και επικύρωση μέσω δοκιμών με ηλικιωμένους χρήστες. Ο έκτος στόχος θέτει τις βάσεις απόδοσης και επεκτασιμότητας: σχεδιασμός για οριζόντια κλιμάκωση με υποστήριξη άνω των 1.000 ταυτόχρονων χρηστών και φόρτωση σελίδων σε λιγότερο από δύο δευτερόλεπτα, μέσω stateless API, στρατηγικών ευρετηρίασης της βάσης δεδομένων, ανάπτυξης σε containers και cloud-native προτύπων, με επιβεβαίωση μέσω μετρήσεων απόδοσης και δοκιμών αυτόματης κλιμάκωσης.

Ο έβδομος στόχος είναι η συντηρήσιμη υποδομή ως κώδικας, ώστε η προμήθεια πόρων και η ανάπτυξη να είναι αναπαραγώγιμες και ελεγχόμενες ως προς τις εκδόσεις: διαμορφώσεις Terraform για τους πόρους cloud, ενορχήστρωση πολλαπλών containers με Docker και βασική δομή CI/CD, με επιδίωξη χρόνου πλήρους εκκίνησης περιβάλλοντος κάτω των 15 λεπτών. Τέλος, ο όγδοος στόχος αφορά την επεκτάσιμη ανίχνευση ανωμαλιών και κινδύνων: τη δημιουργία θεμελίων για έξυπνη ανίχνευση αποκλίσεων στα μετρικά υγείας μέσω στατιστικού υπολογισμού τιμών αναφοράς, εντοπισμού παραβίασης ορίων, φιλτραρίσματος επιμονής και κατάλληλης επισήμανσης στη διεπαφή, με σχεδιασμό εννοιολογικού αλγορίθμου και σαφή διαδρομή μελλοντικής επέκτασης προς τεχνικές μηχανικής μάθησης.

1.3.3 Πεδίο εφαρμογής και όρια

Στο πεδίο εφαρμογής της διατριβής περιλαμβάνονται: μια διαδικτυακή εφαρμογή μίας σελίδας (SPA) προσβάσιμη από προγράμματα περιήγησης σε πολλαπλές πλατφόρμες· οι ρόλοι (personas) ασθενούς και φροντιστή· η χειροκίνητη εισαγωγή μετρήσεων υγείας, με δυνατότητα δημιουργίας εικονικών δεδομένων για λόγους επίδειξης· ένα τοπικό chatbot τεχνητής νοημοσύνης για ερωτήσεις και εκπαίδευση σχετικά με την υγεία· η διαχείριση νοικοκυριού με ροή εργασιών προσκλήσεων· σχεσιακή βάση δεδομένων PostgreSQL για τα δομημένα δεδομένα υγείας· αρχιτεκτονική ανάπτυξης στο cloud της Azure· καθώς και τα βασικά στοιχεία προσβασιμότητας και το σύστημα σχεδίασης.

Εκτός πεδίου εφαρμογής - και αντικείμενο μελλοντικής εργασίας - παραμένουν οι εγγενείς εφαρμογές για κινητά (iOS/Android), η αυτόματη ενσωμάτωση φορετών συσκευών (Fitbit, Apple Watch κ.λπ.), η λήψη δεδομένων σε πραγματικό χρόνο, η διασύνδεση με τα ηλεκτρονικά αρχεία

υγείας των νοσοκομείων (συστήματα EHR/EMR), οι λειτουργίες τηλεδιάσκεψης τηλεϊατρικής, η διαχείριση συνταγών και η ενσωμάτωση φαρμακείων, η επεξεργασία λογαριασμών και ασφαλιστικών απαιτήσεων, η πολυγλωσσική διεθνοποίηση (i18n) και τα προηγμένα προβλεπτικά μοντέλα μηχανικής μάθησης.

1.3.4 Αναμενόμενες συνεισφορές

Σε πρακτικό επίπεδο, η διατριβή παραδίδει μια πλήρως λειτουργική εφαρμογή παρακολούθησης της υγείας ηλικιωμένων που επιδεικνύει σύγχρονη αρχιτεκτονική πλήρους στοίβας (full stack), επαναχρησιμοποιήσιμα πρότυπα σχεδίασης για διεπαφές υγειονομικής περίθαλψης με προτεραιότητα στην προσβασιμότητα, μια υλοποίηση αναφοράς για την τοπική, προστατευτική προς την ιδιωτικότητα ενσωμάτωση τεχνητής νοημοσύνης, καθώς και τεκμηριωμένα πρότυπα ροών εργασίας για πίνακες ελέγχου πολλαπλών ασθενών προς χρήση από φροντιστές.

Σε τεχνικό επίπεδο, προσφέρεται ένα αρχιτεκτονικό σχέδιο για τη στοίβα React + TypeScript + Go + PostgreSQL με τοπικό μεγάλο γλωσσικό μοντέλο, σχήμα βάσης δεδομένων βελτιστοποιημένο για χρονοσειρές δεδομένων υγείας, σχεδιασμός RESTful API για περιπτώσεις χρήσης παρακολούθησης της υγείας και πρότυπα υποδομής ως κώδικα για ανάπτυξη σε containers στο Azure. Σε ακαδημαϊκό επίπεδο, η εργασία συνεισφέρει μια μελέτη περίπτωσης εφαρμογής των κατευθυντήριων γραμμών προσβασιμότητας (WCAG 2.1) στο πλαίσιο της υγειονομικής περίθαλψης, μια ανάλυση των πλεονεκτημάτων και μειονεκτημάτων της τοπικής έναντι της βασισμένης στο cloud τεχνητής νοημοσύνης για εφαρμογές υγείας, ένα πλαίσιο αξιολόγησης ψηφιακών τεχνολογιών υγείας με επίκεντρο τους ηλικιωμένους και τον προσδιορισμό μελλοντικών ερευνητικών κατευθύνσεων στην τεχνητή νοημοσύνη υγείας με διαφύλαξη της ιδιωτικότητας.

1.3.5 Επισκόπηση μεθοδολογίας

Η διατριβή ακολουθεί μια συστηματική μεθοδολογία μηχανικής λογισμικού:

Η πορεία εκκινεί από τη μηχανική απαιτήσεων, με την εξαγωγή λειτουργικών και μη λειτουργικών απαιτήσεων από την ανάλυση του προβλήματος και τη μοντελοποίηση των ενδιαφερόμενων μερών, και συνεχίζει με τον αρχιτεκτονικό σχεδιασμό μέσω διαγραμμάτων συστατικών, μοντελοποίησης της βάσης δεδομένων και προδιαγραφών UML. Ακολουθεί η επαναληπτική υλοποίηση - υπηρεσίες backend, διεπαφή frontend, ενσωμάτωση τεχνητής νοημοσύνης και DevOps - συνοδευόμενη από συνεχή επικύρωση με λειτουργικές δοκιμές, ελέγχους προσβασιμότητας και ασφάλειας. Η διαδικασία ολοκληρώνεται με την τεκμηρίωση (σχόλια κώδικα, τεκμηρίωση API, εγχειρίδια χρήστη, οδηγοί ανάπτυξης) και την αξιολόγηση, μέσω ανάλυσης ιχνηλασιμότητας απαιτήσεων, χαρακτηρισμού της απόδοσης και κριτικού αναστοχασμού.

Αυτή η δομημένη προσέγγιση εξασφαλίζει ακρίβεια, ιχνηλασιμότητα και αναπαραγωγιμότητα, παράγοντας παράλληλα ένα λειτουργικό σύστημα κατάλληλο για πιλοτική ανάπτυξη και περαιτέρω έρευνα.

1.4 Σύγχρονα παραδείγματα και παρόμοια έργα

Η κατανόηση του τρέχοντος τοπίου των λύσεων ψηφιακής παρακολούθησης της υγείας παρέχει το απαραίτητο πλαίσιο για την τοποθέτηση της συμβολής αυτής της διατριβής. Σε αυτή την ενότητα εξετάζονται αντιπροσωπευτικές εμπορικές πλατφόρμες, επαγγελματικά συστήματα υγειονομικής περίθαλψης, τεχνολογίες που εστιάζουν στους ηλικιωμένους και πλατφόρμες τηλεϊατρικής, προκειμένου να προσδιοριστούν τόσο τα παραδειγματικά πρότυπα όσο και τα κρίσιμα μειονεκτήματα.

1.4.1 Πλατφόρμες υγείας και φυσικής κατάστασης για καταναλωτές

Apple HealthKit και το οικοσύστημα εφαρμογών υγείας

Το πλαίσιο HealthKit της Apple, που παρουσιάστηκε το 2014, αποτελεί μία από τις πιο ολοκληρωμένες πλατφόρμες συγκέντρωσης δεδομένων υγείας των καταναλωτών, με την εφαρμογή Health να λειτουργεί ως κεντρικό αποθετήριο για δεδομένα φυσικής κατάστασης και ιατρικά δεδομένα σε συσκευές iOS.

Στα βασικά χαρακτηριστικά του συγκαταλέγονται η κεντρική συγκέντρωση δεδομένων από το Apple Watch, τους αισθητήρες του iPhone και περισσότερες από 150 εφαρμογές τρίτων, η άμεση σύνδεση με συστήματα ηλεκτρονικών αρχείων υγείας που υποστηρίζουν το πρότυπο FHIR σε περισσότερα από 800 συμμετέχοντα συστήματα υγείας, η κοινή χρήση δεδομένων με μέλη της οικογένειας μέσω του iCloud Family Sharing, η εμφάνιση ιατρικού δελτίου ταυτότητας στην οθόνη κλειδώματος για περιπτώσεις έκτακτης ανάγκης και η εξελιγμένη οπτικοποίηση τάσεων σε πολλαπλά μετρικά υγείας. Από τεχνική άποψη, πρόκειται για εγγενές πλαίσιο iOS με API σε Swift/Objective-C, αποθήκευση δεδομένων στη συσκευή με κρυπτογράφηση από άκρο σε άκρο μέσω iCloud, μέτρα ασφαλείας ευθυγραμμισμένα με το HIPAA και λεπτομερείς ελέγχους δικαιωμάτων για την πρόσβαση εφαρμογών τρίτων.

Για τη φροντίδα ηλικιωμένων, ωστόσο, η πλατφόρμα εμφανίζει ουσιώδεις περιορισμούς: το κλειδί στο οικοσύστημα της Apple προϋποθέτει την κατοχή συσκευής (εμπόδιο εισόδου της τάξης των 1.200 δολαρίων), η διεπαφή είναι βελτιστοποιημένη για τεχνολογικά καταρτισμένους χρήστες - με μικρές γραμματοσειρές και ένθετη πλοήγηση που προκαλούν σύγχυση στους ηλικιωμένους - η οικογενειακή κοινή χρήση απαιτεί λογαριασμούς iCloud χωρίς να προσφέρει ειδικούς πίνακες ελέγχου για φροντιστές, ο σχεδιασμός εστιάζει στην παθητική παρακολούθηση δραστηριότητας και όχι στη διαχείριση χρόνιων παθήσεων, ενώ απουσιάζει ενσωματωμένη συνομιλητική βοήθεια για την ερμηνεία των δεδομένων υγείας.

Google Fit και οικοσύστημα δεδομένων υγειονομικής περίθαλψης

Το Google Fit παρέχει παρακολούθηση υγείας σε πολλαπλές πλατφόρμες σε Android και στο διαδίκτυο, με έμφαση στις μετρήσεις δραστηριότητας και τις ενσωματώσεις συνεργατών.

Τα βασικά χαρακτηριστικά του περιλαμβάνουν τους στόχους δραστηριότητας Heart Points και Move Minutes που βασίζονται στις οδηγίες του Παγκόσμιου Οργανισμού Υγείας, την ενσωμάτωση με τα έξυπνα ρολόγια Wear OS, ένα REST API που επιτρέπει σε προγραμματιστές να αξιοποιούν τα δεδομένα του Fit, πίνακα ελέγχου προσβάσιμο από οποιοδήποτε πρόγραμμα περιήγησης και συγχρονισμό μεταξύ συσκευών μέσω του λογαριασμού Google. Η τεχνική στοίβα στηρίζεται σε RESTful API με αυθεντικοποίηση OAuth 2.0, αρχιτεκτονική με προτεραιότητα στο cloud (Google Cloud Platform), κινητά SDK για Android και iOS και αποθήκευση σε τυποποιημένους τύπους δεδομένων του Google Fit.

Ως προς τη φροντίδα ηλικιωμένων, το Google Fit είναι σχεδιασμένο για ενεργούς, νεότερους πληθυσμούς και παρέχει περιορισμένη ιατρική λεπτομέρεια, παρακολουθώντας κυρίως βήματα, θερμίδες και καρδιακό ρυθμό, με ελάχιστη υποστήριξη για την αρτηριακή πίεση και τον κορεσμό οξυγόνου. Δεν διαθέτει λειτουργίες φροντίδας πολλαπλών ασθενών ή διαχείρισης νοικοκυριού, η εξαρτημένη από το cloud αρχιτεκτονική του εγείρει ανησυχίες για την προστασία των δεδομένων και οι λειτουργίες προσβασιμότητας της γενικής διεπαφής του είναι ανεπαρκείς για άτομα με οπτικές ή κινητικές δυσκολίες.

Πλατφόρμα παρακολούθησης υγείας Fitbit

Η Fitbit (που εξαγοράστηκε από την Google το 2021) συνδυάζει φορητές συσκευές με συμβουλευτική υγείας, αντιπροσωπεύοντας ένα ώριμο οικοσύστημα υγείας για τους καταναλωτές.

Η πλατφόρμα προσφέρει πλήρη σειρά φορητών συσκευών (κόστους έως περίπου 300 δολαρίων) με διάρκεια μπαταρίας πέντε έως επτά ημερών, παρακολούθηση ύπνου πολλαπλών σταδίων με σχετική βαθμολογία, διαχείριση άγχους με καθοδηγούμενες ασκήσεις αναπνοής,

παρακολούθηση γυναικείας υγείας, καθώς και το premium επίπεδο Fitbit Care με εξατομικευμένες πληροφορίες έναντι μηνιαίας συνδρομής. Για τους φροντιστές προβλέπονται ο οικογενειακός λογαριασμός, μέσω του οποίου οι γονείς διαχειρίζονται τους λογαριασμούς παιδιών κάτω των 13 ετών, και η περιορισμένη κοινή χρήση με καθορισμένες επαφές της «ομάδας φροντίδας», η οποία ωστόσο αποτελεί λειτουργία επί πληρωμή.

Οι περιορισμοί της πλατφόρμας για τον πληθυσμό-στόχο της παρούσας εργασίας είναι σημαντικοί: η λειτουργία της προϋποθέτει τη συνεχή χρήση φορετής συσκευής, κάτι δύσκολο για ηλικιωμένους με αρθρίτιδα ή ευαισθησία δέρματος· οι προηγμένες λειτουργίες απαιτούν συνδρομή· η εφαρμογή για κινητά προϋποθέτει εξοικείωση με τις χειρονομίες αφή· απουσιάζουν κλινικά επικυρωμένη μέτρηση της αρτηριακής πίεσης και παρακολούθηση γλυκόζης στις περισσότερες συσκευές· και οι παρεχόμενες πληροφορίες είναι τυποποιημένες, χωρίς διαλογική καθοδήγηση υγείας με τεχνητή νοημοσύνη.

1.4.2 Επαγγελματικές πύλες υγείας για ασθενείς

Πύλη ασθενών Epic MyChart

Η Epic MyChart κυριαρχεί στην αγορά των πύλων ασθενών EHR των νοσοκομείων των ΗΠΑ, εξυπηρετώντας περισσότερους από 250 εκατομμύρια ασθενείς σε πάνω από 1.500 οργανισμούς υγειονομικής περίθαλψης.

Οι βασικές λειτουργίες της περιλαμβάνουν την πρόσβαση στα ιατρικά αρχεία (αποτελέσματα εργαστηριακών εξετάσεων, ακτινολογικές εικόνες, κλινικές σημειώσεις, ιστορικό), τον προγραμματισμό ραντεβού με αυτοεξυπηρέτηση, την ασφαλή ανταλλαγή μηνυμάτων με την ομάδα φροντίδας, τη διαχείριση συνταγών με αιτήματα ανανέωσης και λίστες φαρμάκων, τη χρέωση και την ασφάλιση, καθώς και την πρόσβαση μέσω πληρεξουσίου για εξουσιοδοτημένα μέλη της οικογένειας. Τεχνικά, πρόκειται για εφαρμογή enterprise Java ενσωματωμένη με το backend του Epic EHR, με HL7 FHIR API για διαλειτουργικότητα, εγγενείς εφαρμογές iOS/Android, αποκρίσιμη διαδικτυακή πύλη, πολυπαραγοντική αυθεντικοποίηση και συμμόρφωση με το HIPAA.

Για τη συνεχή παρακολούθηση ηλικιωμένων, όμως, το MyChart υστερεί ουσιαστικά: είναι οργανωμένο γύρω από τις επισκέψεις στην κλινική και όχι γύρω από τη διαρκή παρακολούθηση, βασίζεται σε δεδομένα που εισάγουν οι κλινικοί ιατροί με ελάχιστη ενσωμάτωση δεδομένων παραγόμενων από τους ασθενείς (PGHD), προσφέρει κακή εμπειρία χρήσης στους ηλικιωμένους - βαθμολογία προσβασιμότητας 62/100 στη μελέτη της Nielsen Norman Group, με πολύπλοκη ορολογία και μικρή γραμματοσειρά - ενώ η πρόσβαση μέσω πληρεξουσίου είναι τύπου «όλα ή τίποτα», χωρίς λεπτομερή διαχείριση νοικοκυριού. Επιπλέον, τα εργαστηριακά αποτελέσματα παρουσιάζονται ως μεμονωμένα συμβάντα και όχι ως διαχρονικές τάσεις, και δεν υπάρχει καμία μορφή συνομιλητικής βοήθειας ή καθοδήγησης υγείας με τεχνητή νοημοσύνη.

Cerner HealtheLife (Oracle Health)

Η πύλη ασθενών της Cerner (μετονομάστηκε σε Oracle Health μετά την εξαγορά το 2022) εξυπηρετεί περισσότερους από 150 εκατομμύρια ασθενείς.

Η πλατφόρμα προσφέρει ενσωματωμένο προσωπικό φάκελο υγείας που συγκεντρώνει δεδομένα από πολλούς παρόχους, περιορισμένη υποστήριξη φορετών συσκευών καταναλωτών (Fitbit, Withings), συντονισμό της ομάδας φροντίδας με κοινά σχέδια μεταξύ πολυεπιστημονικών ομάδων και ενσωματωμένες βιντεοεπισκέψεις τηλευγείας. Εμφανίζει, ωστόσο, παρόμοιες προκλήσεις προσβασιμότητας και εμπειρίας χρήσης με το Epic MyChart, ελάχιστες δυνατότητες συνεχούς παρακολούθησης, ενσωμάτωση συσκευών διαθέσιμη μόνο σε premium οργανωτικές άδειες και καμία εξειδικευμένη διεπαφή για τη φροντίδα ηλικιωμένων.

1.4.3 Τεχνολογικές λύσεις ειδικά για ηλικιωμένους

Tablet GrandPad για ηλικιωμένους

Το GrandPad είναι ένα tablet ειδικά σχεδιασμένο για ηλικιωμένους χρήστες, με απλοποιημένη διεπαφή και εξειδικευμένη υποστήριξη.

Ο σχεδιασμός του βασίζεται σε tablet 8 ιντσών με μεγάλη οθόνη αφής και προστατευτικό πλαίσιο, απλοποιημένο περιβάλλον χρήστη με εικονίδια ενός πατήματος χωρίς περίπλοκη πλοήγηση, προδιαμορφωμένη παράδοση με ενσωματωμένη σύνδεση 4G LTE, βιντεοκλήσεις και κοινή χρήση φωτογραφιών με την οικογένεια με ένα άγγιγμα, καθώς και εικοσιτετράωρη εξειδικευμένη υποστήριξη για χρήστες και οικογένειες. Οι λειτουργίες υγείας, εντούτοις, περιορίζονται σε βασικές υπενθυμίσεις λήψης φαρμάκων και συντομεύσεις επαφών έκτακτης ανάγκης, χωρίς ενσωματωμένη παρακολούθηση ή ανάλυση δεικτών υγείας.

Το GrandPad αποτελεί κλειστό οικοσύστημα με ιδιόκτητη συσκευή (79 δολάρια συν 40 δολάρια μηνιαίως για την υπηρεσία), επιφανειακή λειτουργικότητα υγείας χωρίς παρακολούθηση ζωτικών ενδείξεων, καμία δυνατότητα κλινικής ενσωμάτωσης με ηλεκτρονικά αρχεία υγείας ή κοινής χρήσης δεδομένων με παρόχους, περιορισμένη επεκτασιμότητα - αφού δεν επιτρέπεται η εγκατάσταση εφαρμογών υγείας τρίτων - και γεωγραφικό περιορισμό στις Ηνωμένες Πολιτείες λόγω της εξάρτησης από το δίκτυο κινητής τηλεφωνίας.

CareZone Διαχείριση φαρμάκων και υγείας

Το CareZone (που εξαγοράστηκε από την WellSky το 2020) επικεντρώθηκε στη διαχείριση φαρμάκων και τον συντονισμό των φροντιστών.

Τα βασικά χαρακτηριστικά του περιλάμβαναν λίστα φαρμάκων με ονόματα, δοσολογίες, χρονοδιαγράμματα και φωτογραφίες, ειδοποιήσεις rush για τις ώρες λήψης, κοινή πρόσβαση πολλών οικογενειακών φροντιστών στις ίδιες πληροφορίες του ασθενούς, ημερολόγιο για συμπτώματα, ιατρικές επισκέψεις και ανησυχίες, καθώς και διαχείριση των στοιχείων επικοινωνίας των παρόχων υγειονομικής περίθαλψης. Η εφαρμογή, ωστόσο, εστίαζε σχεδόν αποκλειστικά στα φάρμακα, με ελάχιστη παρακολούθηση ευρύτερων δεικτών υγείας, στηριζόταν σε χειροκίνητη εισαγωγή δεδομένων χωρίς ενσωμάτωση συσκευών, δεν διέθετε λειτουργίες τεχνητής νοημοσύνης πέρα από στατικές υπενθυμίσεις και, τελικά, η καταναλωτική της έκδοση διακόπηκε το 2023, με στρόφη της εταιρείας αποκλειστικά σε συνεργασίες B2B στον τομέα της υγειονομικής περίθαλψης.

1.4.4 Πλατφόρμες τηλεϊατρικής

Teladoc Health

Η Teladoc είναι η μεγαλύτερη πλατφόρμα τηλεϊατρικής στις ΗΠΑ, παρέχοντας πάνω από 15 εκατομμύρια εικονικές συμβουλές ετησίως.

Το μοντέλο υπηρεσιών της περιλαμβάνει εικοσιτετράωρη κατ' απαίτηση σύνδεση με εξουσιοδοτημένους γιατρούς εντός δέκα λεπτών, εξειδικευμένη φροντίδα σε τομείς όπως η ψυχική υγεία, η δερματολογία και η διαχείριση χρόνιων παθήσεων, συνεργασίες με ασφαλιστικά προγράμματα και εργοδότες, και πρόσβαση τόσο από κινητές συσκευές όσο και από το διαδίκτυο. Στα προγράμματα χρόνιων παθήσεων προσφέρονται διαχείριση διαβήτη με συνδεδεμένους μετρητές γλυκόζης, προγράμματα υπέρτασης με παρακολούθηση της αρτηριακής πίεσης και ειδικοί συντονιστές φροντίδας για ασθενείς υψηλού κινδύνου.

Για τη συνεχή παρακολούθηση ηλικιωμένων, το μοντέλο της Teladoc παραμένει εγγενώς επεισοδικό: οι διαβουλεύσεις είναι μεμονωμένες και δεν συνιστούν διαρκή παρακολούθηση, δεν υπάρχει μόνιμος πίνακας ελέγχου - τα δεδομένα υγείας αποθηκεύονται στις σημειώσεις του παρόχου και δεν είναι προσβάσιμα από τους ασθενείς - η συμμετοχή της οικογένειας είναι ελάχιστη, καθώς οι συνεδρίες γίνονται ένας προς έναν, και το μοντέλο είναι αντιδραστικό: την πρωτοβουλία την παίρνουν οι ασθενείς, χωρίς προληπτική ανίχνευση ανωμαλιών.

Amwell και MDLive

Η Amwell και η MDLive προσφέρουν παρόμοιες υπηρεσίες τηλεϊατρικής με συγκρίσιμους περιορισμούς για τη συνεχή παρακολούθηση της φροντίδας των ηλικιωμένων.

1.4.5 Συγκριτικός πίνακας χαρακτηριστικών

Ο παρακάτω πίνακας συγκρίνει συστηματικά τις βασικές δυνατότητες των αντιπροσωπευτικών πλατφορμών:

Χαρακτηριστικό	Apple HealthKit	Fitbit	Epic MyChart	GrandPad	Teladoc	Αυτή η διατριβή
Βελτιστοποίηση προσβασιμότητας	Μερική	Όχι	Περιορισμένη	Άριστη	Μερική	Άριστο
Πίνακας ελέγχου φροντιστή πολλαπλών ασθενών	Όχι	Όχι	Μόνο πληρεξούσιο	Όχι	Όχι	Ναι
Παρακολούθηση αρτηριακής πίεσης	Ναι*	Περιορισμένη*	Όχι	Όχι	Ναι*	Ναι
Παρακολούθηση καρδιακού ρυθμού	Ναι*	Ναι*	Όχι	Όχι	Όχι	Ναι
Παρακολούθηση βάρους	Ναι*	Ναι*	Όχι	Όχι	Όχι	Ναι
Κορεσμός οξυγόνου	Ναι*	Περιορισμένος*	Όχι	Όχι	Όχι	Ναι
Ανάλυση ύπνου	Ναι*	Ναι*	Όχι	Όχι	Όχι	Ναι
Χειροκίνητη εισαγωγή δεδομένων	Μέσω τρίτου μέρους	Μέσω συσκευής	Όχι	Όχι	Όχι	Ναι
Οπτικοποίηση τάσεων	Καλή	Καλή	Κακή	Δεν ισχύει	Όχι	Άριστο
Βοήθεια για θέματα υγείας με τεχνητή νοημο	Όχι	Όχι	Όχι	Όχι	Όχι	Ναι (Τοπικό)
Τεχνητή νοημοσύνη που προστατεύει την ιδιωτικότητα	Δεν ισχύει	Δεν ισχύει	Δεν ισχύει	Δεν ισχύει	Δεν ισχύει	Ναι
Εξαγωγή δεδομένων	Περιορισμένη	Περιορισμένη	PDF	Όχι	Όχι	Ναι (CSV/JSON)
Διαπλατφορμικό	Μόνο iOS	Πολλαπλή	Πολλαπλή	Ιδιόκτητο	Πολλαπλή	Ιστός (Όλα)
Κόστος	Κόστος συσκευής	Συσκευή + Συνδρομή	Δωρεάν	40 \$/μήνα	Ανά επίσκεψη	Ανοιχτός κώδικας
Ενσωμάτωση EHR	Ν	Όχι	Εγγενής	Όχι	Backend	Μέλλον
Προσκλήσεις νοικοκυριών	Όχι	Όχι	Όχι	Όχι	Όχι	Ναι
Τοπική ανάπτυξη τεχνητής νοημοσύνης	Δεν ισχύει	Δεν ισχύει	Δεν ισχύει	Δεν ισχύει	Δεν ισχύει	Ναι

Πίνακας 1.2: Συγκριτικός πίνακας χαρακτηριστικών ψηφιακών λύσεων υγείας

Απαιτείται η αγορά φορητής συσκευής*1.4.6 Τι κάνει αυτή τη διατριβή διαφορετική**

Η παρούσα διατριβή αντιμετωπίζει μια σειρά από ανεκπλήρωτες ανάγκες μέσω ολοκληρωμένων σχεδιαστικών αποφάσεων:

1. Φιλοσοφία σχεδιασμού με προτεραιότητα στην προσβασιμότητα Σε αντίθεση με τις πλατφόρμες που έχουν σχεδιαστεί για νεότερους, τεχνολογικά καταρτισμένους χρήστες και έχουν προσαρμοστεί εκ των υστέρων για προσβασιμότητα, αυτό το σύστημα χρησιμοποιεί τους ηλικιωμένους χρήστες ως το κύριο πρόσωπο σχεδιασμού από την αρχή. Η μεγάλη τυπογραφία (βάση 18-24pt), τα θέματα υψηλής αντίθεσης (αναλογία 7:1), η απλοποιημένη πλοήγηση (≤ 3 επίπεδα) και οι γενναιόδωροι στόχοι αφής ($\geq 44px$) είναι θεμελιώδη, όχι μεταγενέστερες σκέψεις.

2. Τοπική τεχνητή νοημοσύνη που προστατεύει την ιδιωτικότητα Ενώ οι πλατφόρμες Teladoc, Epic και οι πλατφόρμες καταναλωτών είτε δεν διαθέτουν τεχνητή νοημοσύνη συνομιλίας είτε εξαρτώνται από εξωτερικά API cloud, η παρούσα διατριβή παρουσιάζει τοπική τεχνητή νοημοσύνη μέσω Ollama. Οι προστατευμένες πληροφορίες υγείας δεν εγκαταλείπουν ποτέ το περιβάλλον ανάπτυξης, αντιμετωπίζοντας τα ζητήματα ιδιωτικότητας που είναι υψίστης σημασίας για το 71% των ηλικιωμένων χρηστών που δεν εμπιστεύονται την επεξεργασία δεδομένων.

3. Αρχιτεκτονική διπλού ρόλου με κεντρικό ρόλο των φροντιστών Οι υπάρχουσες πύλες ασθενών αντιμετωπίζουν τους οικογενειακούς φροντιστές ως δευτερεύοντες αντιπροσώπους. Αυτό το σύστημα αναβαθμίζει τους φροντιστές σε χρήστες πρώτης κατηγορίας με ειδικούς πίνακες ελέγχου πολλαπλών ασθενών, διαχείριση νοικοκυριού, ροές εργασίας προσκλήσεων και συγκεντρωτική ανάλυση τάσεων υγείας. Ο σχεδιασμός αναγνωρίζει ότι 53 εκατομμύρια οικογενειακοί φροντιστές είναι αναπόσπαστο μέρος της παροχής υγειονομικής.

4. Ενοποιημένη συνεχής παρακολούθηση + γέφυρα περιστασιακής φροντίδας Οι πλατφόρμες τηλεϊατρικής παρέχουν περιστασιακές συμβουλές, οι φορητές πλατφόρμες παρακολουθούν τη δραστηριότητα και οι πύλες EHR καταγράφουν τις κλινικές συναντήσεις. Η παρούσα διατριβή ενοποιεί τη συνεχή παρακολούθηση των δεικτών υγείας με δυνατότητες εξαγωγής δεδομένων, επιτρέποντας την απρόσκοπτη επικοινωνία με τους παρόχους υγειονομικής περίθαλψης και γεφυρώνοντας το χάσμα μεταξύ τους.

5. Ανοιχτό, επεκτάσιμο, αναπαραγώγιμο Οι εμπορικές λύσεις είναι ιδιόκτητα κλειστά οικοσυστήματα. Αυτή η διατριβή παρέχει μια πλήρη εφαρμογή αναφοράς ανοιχτού κώδικα με υποδομή ως κώδικα, επιτρέποντας την αναπαραγωγιμότητα, την ακαδημαϊκή επέκταση και την προσαρμογή σε διαφορετικά περιβάλλοντα φροντίδας χωρίς δέσμευση σε συγκεκριμένο προμηθευτή.

6. Σύγχρονες βέλτιστες πρακτικές πλήρους στοίβας Σε αντίθεση με τα παλαιά συστήματα EHR που βασίζονται σε στοίβες Java για επιχειρήσεις, αυτό το σύστημα παρουσιάζει σύγχρονη ανάπτυξη ιστού: React με TypeScript για ανάπτυξη UI με ασφάλεια τύπου, Go για υπηρεσίες backend υψηλής απόδοσης, PostgreSQL για αξιόπιστη διατήρηση δεδομένων, Docker για συσκευασία σε container, Terraform για αυτοματοποίηση υποδομής. Οι τεχνολογικές επιλογές αντανakλούν τα πρότυπα της βιομηχανίας του 2024+, όχι τις αρχιτεκτονικές της δεκαετίας του 2010.

1.4.7 Περίληψη συνεισφοράς

Συνοψίζοντας, η παρούσα διατριβή συνεισφέρει: την πρώτη ολοκληρωμένη ακαδημαϊκή μελέτη περίπτωσης τοπικής, προστατευτικής προς την ιδιωτικότητα τεχνητής νοημοσύνης στην παρακολούθηση της υγείας· νέα αρχιτεκτονικά πρότυπα διπλού ρόλου με επίκεντρο τον

φροντιστή για τη διαχείριση της υγείας σε επίπεδο νοικοκυριού· την αυστηρή εφαρμογή των αρχών προσβασιμότητας WCAG 2.1 σε διεπαφές παρακολούθησης χρόνιων παθήσεων· μια υλοποίηση αναφοράς ανοιχτού κώδικα σύγχρονης εφαρμογής πλήρους στοίβας για την υγειονομική περίθαλψη· και την εμπειρική απόδειξη ότι η τοπική ανάπτυξη μεγάλου γλωσσικού μοντέλου (Ollama) μπορεί να παρέχει βοήθεια υγείας θεμελιωμένη στο πλαίσιο του ασθενούς, χωρίς έκθεση των προσωπικών δεδομένων υγείας στο cloud.

Τα ακόλουθα κεφάλαια περιγράφουν λεπτομερώς τον συστηματικό σχεδιασμό, την εφαρμογή και την αξιολόγηση αυτού του ολοκληρωμένου συστήματος.

1.5 Επισκόπηση της δομής της διατριβής

Η διατριβή αυτή είναι οργανωμένη σε οκτώ κεφάλαια που τεκμηριώνουν συστηματικά το κίνητρο, το σχεδιασμό, την υλοποίηση, την αξιολόγηση και τις μελλοντικές κατευθύνσεις του Πίνακα Παρακολούθησης της Υγείας των Ηλικιωμένων.

Μεθοδολογική προσέγγιση

Η διατριβή χρησιμοποιεί μια συστηματική μεθοδολογία μηχανικής λογισμικού που συνδυάζει τη μηχανική απαιτήσεων, τον επαναληπτικό σχεδιασμό, την υλοποίηση και την αξιολόγηση. Η εργασία ξεκινά με την ανάλυση του προβλήματος - δημογραφική έρευνα, προσδιορισμό των αναγκών των χρηστών και ανάλυση του ανταγωνισμού - και προχωρά στη μηχανική απαιτήσεων, με την εξαγωγή λειτουργικών και μη λειτουργικών απαιτήσεων και τον προσδιορισμό των περιορισμών. Έπεται ο αρχιτεκτονικός σχεδιασμός, που περιλαμβάνει την αποσύνθεση του συστήματος, τη μοντελοποίηση της βάσης δεδομένων και τις προδιαγραφές UML, και ακολουθεί η επαναληπτική υλοποίηση, από τις υπηρεσίες backend στη διεπαφή frontend και από την ενσωμάτωση της τεχνητής νοημοσύνης στην αυτοματοποίηση DevOps. Καθ' όλη τη διάρκεια εφαρμόζεται συνεχής επικύρωση μέσω λειτουργικών δοκιμών, ελέγχων προσβασιμότητας και ασφάλειας και συγκριτικής αξιολόγησης απόδοσης, ενώ η τεκμηρίωση καλύπτει τα σχόλια κώδικα, την τεκμηρίωση του API, τις οδηγίες χρήσης και τις διαδικασίες ανάπτυξης. Η μεθοδολογία ολοκληρώνεται με την αξιολόγηση και τον αναστοχασμό: ιχνηλασιμότητα απαιτήσεων, καταγραφή διδαγμάτων και προσδιορισμό των μελλοντικών εργασιών.

Αυτή η δομημένη προσέγγιση εξασφαλίζει ακαδημαϊκή αυστηρότητα, τεχνικό βάθος και πρακτική εφαρμοσιμότητα, παράγοντας τόσο ένα λειτουργικό σύστημα όσο και πρακτικές πληροφορίες για τους ερευνητές και τους επαγγελματίες της τεχνολογίας υγειονομικής περίθαλψης.

Κεφάλαιο 2. Ανασκόπηση πεδίου

2.1 Ανασκόπηση βιβλιογραφίας - Ακαδημαϊκή έρευνα

Αυτή η ενότητα εξετάζει τη σχετική ακαδημαϊκή βιβλιογραφία σε τέσσερις τομείς κρίσιμης σημασίας για την παρακολούθηση της υγείας των ηλικιωμένων: μελέτες υιοθέτησης τεχνολογίας υγειονομικής περίθαλψης, αρχιτεκτονικές συστημάτων απομακρυσμένης παρακολούθησης, προσβασιμότητα και σχεδιασμός αλληλεπίδρασης ανθρώπου-υπολογιστή, καθώς και πλαίσια ασφάλειας και απορρήτου.

2.1.1 Μελέτες υιοθέτησης τεχνολογιών υγειονομικής περίθαλψης

Αρχές σχεδιασμού διεπαφής χρήστη για ηλικιωμένους

Το θεμελιώδες έργο της Nielsen Norman Group σχετικά με τον φιλικό προς τους ηλικιωμένους σχεδιασμό ιστοσελίδων (2019) καθιέρωσε τεκμηριωμένες κατευθυντήριες γραμμές που στη

συνέχεια επικυρώθηκαν σε πολλαπλές εμπειρικές μελέτες. Η έρευνα χρησιμότητας που διεξήγαγαν με συμμετέχοντες ηλικίας 65-92 ετών εντόπισε κρίσιμα σχεδιαστικά σφάλματα στις σύγχρονες ψηφιακές διεπαφές:

Τα κύρια ευρήματα αφορούν, πρώτον, τις απαιτήσεις μεγέθους γραμματοσειράς: ελάχιστο μέγεθος 16 εικονοστοιχείων για το κύριο κείμενο και 24 για τις επικεφαλίδες, καθώς οι μελέτες δείχνουν ότι η κατανόηση μειώνεται κατά 43% κάτω από αυτά τα όρια για χρήστες με πρεσβυωπία. Δεύτερον, την αναλογία αντίθεσης: το πρότυπο WCAG 2.1 AA (4,5:1) κρίνεται ανεπαρκές για ηλικιωμένους χρήστες και συνιστάται αναλογία 7:1 για περιπτώσεις καταρράκτη ή εκφύλισης της ωχράς κηλίδας, παθήσεων που επηρεάζουν το 50% του πληθυσμού άνω των 75 ετών. Τρίτον, τον χρόνο αλληλεπίδρασης: οι ηλικιωμένοι χρήστες χρειάζονται 2,5 φορές περισσότερο χρόνο για τη συμπλήρωση σύνθετων φορμών, ενώ χρονικά όρια κάτω των 60 δευτερολέπτων προκαλούν εγκατάλειψη σε ποσοστό 67%. Τέταρτον, το βάθος πλοήγησης, αφού κάθε επιπλέον επίπεδο μειώνει το ποσοστό ολοκλήρωσης των εργασιών κατά 18%, καθιστώντας κρίσιμες τις επίπεδες αρχιτεκτονικές με λιγότερα από τρία επίπεδα. Τέλος, το μέγεθος του στόχου αφή: το ελάχιστο των 44×44 εικονοστοιχείων του W3C αποδεικνύεται ανεπαρκές για χρήστες με αρθρίτιδα, ενώ στόχοι 60×60 εικονοστοιχείων μειώνουν τα λάθη αφής κατά 54%.

Πλαίσιο ψηφιακής παιδείας στον τομέα της υγείας

Οι Norgaard et al. (2015) πρότειναν ένα πολυδιάστατο μοντέλο ψηφιακής παιδείας στον τομέα της υγείας, το οποίο είναι ιδιαίτερα σχετικό με τους ηλικιωμένους πληθυσμούς. Το πλαίσιο τους προσδιορίζει επτά ικανότητες:

Πρόκειται για: (1) την ικανότητα επεξεργασίας πληροφοριών υγείας· (2) την ενεργό εμπλοκή του ατόμου με την προσωπική του υγεία· (3) την ικανότητα ενεργού χρήσης των ψηφιακών υπηρεσιών· (4) το αίσθημα ασφάλειας και ελέγχου κατά τη χρήση τους· (5) το κίνητρο για ενασχόληση με τις ψηφιακές υπηρεσίες· (6) την πρόσβαση σε συστήματα που λειτουργούν αξιόπιστα· και (7) ψηφιακές υπηρεσίες προσαρμοσμένες στις ατομικές ανάγκες του χρήστη.

Το πλαίσιο αναπτύχθηκε με μεθοδολογία εννοιολογικής χαρτογράφησης, μέσω εργαστηρίων με ασθενείς, επαγγελματίες υγείας και ειδικούς ιατρικής πληροφορικής στη Δανία και το Ηνωμένο Βασίλειο. Η ανάλυσή του υπογραμμίζει ότι η αποτελεσματική χρήση ψηφιακών υπηρεσιών υγείας προϋποθέτει τη σύμπτωση ικανοτήτων του χρήστη και κατάλληλα σχεδιασμένων συστημάτων - και, κατά συνέπεια, την ανάγκη για βοηθητικές τεχνολογίες που αντισταθμίζουν τα κενά ψηφιακής παιδείας, αντί να προϋποθέτουν την ύπαρξη προαπαιτούμενων γνώσεων.

Μοντέλο αποδοχής της τεχνολογίας (TAM) στην υγειονομική περίθαλψη

Το μοντέλο αποδοχής της τεχνολογίας του Davis, το οποίο προσαρμόστηκε εκτενώς στο πλαίσιο της υγειονομικής περίθαλψης από τους Holden & Karsh (2010), υποστηρίζει ότι η υιοθέτηση της τεχνολογίας εξαρτάται από την αντιληπτή χρησιμότητα και την αντιληπτή ευκολία χρήσης. Η μετα-ανάλυση 63 μελετών για την πληροφορική στην υγεία αποκάλυψε:

Η αντιληπτή χρησιμότητα εμφανίζει μέγεθος επίδρασης $r=0,56$ στην πρόθεση υιοθέτησης και η αντιληπτή ευκολία χρήσης $r=0,48$, ενώ διαπιστώθηκε και επίδραση διαμεσολάβησης, καθώς η ευκολία χρήσης επηρεάζει την αντίληψη της χρησιμότητας με έμμεσο συντελεστή $\beta=0,42$.

Ειδικά για τους ηλικιωμένους πληθυσμούς, οι Wildenbos et al. (2018) επέκτειναν το TAM με «διευκολυντικές συνθήκες» (κοινωνική υποστήριξη, διαθεσιμότητα εκπαίδευσης) και «ανησυχία για την τεχνολογία» ως μεταβλητές μετριασμού. Η μελέτη τους ($N=402$, ηλικίες 65-88) διαπίστωσε ότι το τεχνολογικό άγχος συσχετίζεται αρνητικά με την υιοθέτηση ($r=-0,61$, $p<0,001$), υπογραμμίζοντας τη σημασία των διεπαφών με χαμηλά εμπόδια και που ενισχύουν την εμπιστοσύνη.

Επίδραση της τεχνητής νοημοσύνης

Η επίδραση της τεχνητής νοημοσύνης στην αλληλεπίδραση ανθρώπου-υπολογιστή και στην κινητή υπολογιστική φαίνεται σε διάφορες προηγμένες εξελίξεις που αναδεικνύονται στη

βιβλιογραφία. Για παράδειγμα, οι Tsihrintzis, Virvou και Hatzilygeroudis (2023) υπογραμμίζουν τον καίριο ρόλο της τεχνητής νοημοσύνης στην ανακατασκευή των διεπαφών του χρήστη και στη δυνατότητα προσαρμογής των αλληλεπιδράσεων. Αυτές οι δυνατότητες είναι ιδιαίτερα σημαντικές για εφαρμογές όπως η My Health App, όπου η προσαρμοστικότητα στις ανάγκες κάθε χρήστη αποτελεί βασικό στοιχείο της εμπειρίας. Επιπλέον, οι Virvou, Tsihrintzis, Bourbakis και Jain (2022) υπογραμμίζουν τη σημασία της τεχνητής νοημοσύνης στην ενδυνάμωση των πρακτικών μηχανικής λογισμικού, ιδιαίτερα στην ανάπτυξη έξυπνων εφαρμογών. Αυτή η συμβιωτική σχέση μεταξύ της τεχνητής νοημοσύνης και της αλληλεπίδρασης ανθρώπου-υπολογιστή διαφαίνεται επίσης στις συλλογές άρθρων που έχουν επιμεληθεί Tsihrintzis, Virvou, and Saruwatari (2022), οι Tsihrintzis, Virvou, and Jain (2022), οι Tsihrintzis, Virvou, και Phillips-Wren, (2019) και Weng, Shieh και Tsihrintzis (2023), τα οποία εξερευνούν προηγμένες τεχνικές που συμβάλλουν στην ενίσχυση της συμμετοχής και της ικανοποίησης του χρήστη.

Η ενσωμάτωση της τεχνητής νοημοσύνης στην αλληλεπίδραση ανθρώπου-υπολογιστή και στην κινητή υπολογιστική δεν περιορίζεται σε πρόσφατες εξελίξεις, αλλά αποτελεί θέμα ενδιαφέροντος εδώ και πολλά χρόνια. Παλαιότερες συμβολές, όπως αυτές που περιγράφονται στην εργασία των Virvou και Nakamura (2008), δείχνουν το θεμελιώδη ρόλο της τεχνητής νοημοσύνης στη μηχανική λογισμικού βασισμένη σε γνώσεις, θεμελιώνοντας το έδαφος για μελλοντικές καινοτομίες στην αλληλεπίδραση του χρήστη. Επιπλέον, η συλλογή επιμέρους άρθρων που επιμελήθηκαν οι Tsihrintzis και Virvou (2010) εξετάζει τις προκλήσεις και τις λύσεις στην ανάπτυξη πολυμεσικών υπηρεσιών εντός ευφών περιβαλλόντων, υπογραμμίζοντας τη συνεχή εξέλιξη των διεπαφών που ενσωματώνουν την τεχνητή νοημοσύνη.

Η εξέλιξη της αλληλεπίδρασης ανθρώπου-υπολογιστή με τεχνητή νοημοσύνη δεν έχει μόνο οδηγήσει σε βελτιωμένες εμπειρίες χρήστη, αλλά έχει επίσης ανοίξει το δρόμο για νέες ερευνητικές κατευθύνσεις. Οι Tsihrintzis, Toro, Rios, Howlett και Jain (2023) επιμελούνται τη συλλογή των άρθρων που συζητούν τις επιπτώσεις της τεχνητής νοημοσύνης στο πλαίσιο του Διεθνούς Συνεδρίου 27th KES International Conference on Knowledge-Based and Intelligent Information & Engineering Systems KES2023, τις αναδυόμενες τάσεις και προκλήσεις στον τομέα. Στο πλαίσιο της My Health App, αυτές οι εξελίξεις μεταφράζονται σε καλύτερη κατανόηση των αναγκών των χρηστών και πιο αποτελεσματική παροχή υπηρεσιών υγείας.

Η Τεχνολογία Λογισμικού ενισχυμένη με Τεχνητή Νοημοσύνη αντιπροσωπεύει μια μεγάλη αλλαγή στις μεθοδολογίες ανάπτυξης λογισμικού. Οι Virvou κ.ά. (2022α) επισημαίνουν νέες μεθόδους για την ανάπτυξη έξυπνων συστημάτων λογισμικού, τονίζοντας την ένταξη τεχνικών τεχνητής νοημοσύνης για τη βελτίωση της λειτουργικότητας και της νοημοσύνης του λογισμικού. Αυτή η ένταξη είναι ουσιώδης για την ανάπτυξη εφαρμογών όπως η My Health App, που είναι ικανές να προσαρμόζονται στις προτιμήσεις του χρήστη και το εξελισσόμενο περιβάλλον υγείας.

Οι προηγμένες τεχνολογίες βασισμένες στην τεχνητή νοημοσύνη έχουν επηρεάσει σημαντικά τις διαδραστικές εφαρμογές λογισμικού. Οι Virvou κ.ά. (2022β) συζητούν πώς αυτές οι προόδους, από τη μηχανική μάθηση έως τη βαθιά μάθηση, έχουν επιτρέψει τη δημιουργία έξυπνων συστημάτων λογισμικού με βελτιωμένες δυνατότητες στην αντίληψη, το συλλογισμό και τη λήψη αποφάσεων. Τέτοιες προόδους είναι κρίσιμες για την ανάπτυξη εφαρμογών υγείας που μπορούν να κατανοούν και να ανταποκρίνονται αποτελεσματικά στις εισόδους του χρήστη και να προσφέρουν εξατομικευμένες συστάσεις.

Οι τεχνολογίες μηχανικής μάθησης και βαθιάς μάθησης έχουν παίξει ένα καθοριστικό ρόλο στην εξέλιξη των διαδραστικών εφαρμογών λογισμικού. Οι Tsihrintzis κ.ά. (2022) εξερευνούν τις πιο πρόσφατες εξελίξεις σε αυτούς τους τομείς, τονίζοντας τις εφαρμογές τους σε διάφορους τομείς, συμπεριλαμβανομένης της επεξεργασίας φυσικής γλώσσας, της υπολογιστικής όρασης και της προβλεπτικής ανάλυσης. Αυτές οι τεχνολογίες ενισχύουν τις δυνατότητες εφαρμογών όπως η My Health App να μαθαίνουν από τα δεδομένα των χρηστών, επιτρέποντάς τους να παρέχουν εξατομικευμένες εμπειρίες και να κάνουν ευφείς προβλέψεις σχετικά με την υγεία τους.

2.1.2 Συστήματα απομακρυσμένης παρακολούθησης της υγείας

Φορητή τεχνολογία στη φροντίδα ηλικιωμένων

Οι Patel et al. (2012) στο Journal of NeuroEngineering and Rehabilitation παρείχαν συστηματική ανασκόπηση 62 μελετών φορετών αισθητήρων για τη φροντίδα ηλικιωμένων, κατηγοριοποιώντας τις εφαρμογές σε παρακολούθηση δραστηριότητας, ανίχνευση πτώσεων, καρδιαγγειακή παρακολούθηση και τήρηση φαρμακευτικής αγωγής. Βασικά αρχιτεκτονικά πρότυπα που προσδιορίστηκαν:

Σε ό,τι αφορά τους τύπους αισθητήρων, τα επιταχυνσιόμετρα χρησιμοποιούνται για τη μέτρηση βημάτων και την ανίχνευση πτώσεων (ευαισθησία 85-95%, ειδικότητα 70-85%), η φωτοπληθυσμογραφία (PPG) για την παρακολούθηση του καρδιακού ρυθμού με ακρίβεια ± 5 παλμών ανά λεπτό σε σύγκριση με το ηλεκτροκαρδιογράφημα, η βιοηλεκτρική εμπέδηση για την κατάσταση ενυδάτωσης και τη σύσταση σώματος, και το GPS για τον εντοπισμό θέσης και την πρόληψη της περιπλάνησης ατόμων με άνοια. Ως προς τις αρχιτεκτονικές μετάδοσης δεδομένων, διακρίνονται η άμεση μετάδοση από τη φορητή συσκευή στο smartphone και από εκεί στο cloud (χαμηλότερη καθυστέρηση αλλά υψηλότερη κατανάλωση μπαταρίας), η επεξεργασία στην άκρη μέσω συσκευής πύλης (μειωμένο εύρος ζώνης και δυνατότητα λειτουργίας εκτός σύνδεσης) και το υβριδικό μοντέλο τοπικής συγκέντρωσης με περιοδικό συγχρονισμό στο cloud, που εξισορροπεί καθυστέρηση και κατανάλωση ενέργειας.

Ως προς την αποτελεσματικότητα, η ανασκόπηση κατέγραψε μείωση των επανεισαγωγών στο νοσοκομείο κατά 25-40% για ασθενείς με συμφορητική καρδιακή ανεπάρκεια υπό συνεχή παρακολούθηση, βελτίωση της τήρησης της φαρμακευτικής αγωγής κατά 18-32% με φορετές συσκευές με δυνατότητα υπενθύμισης και μείωση του χρόνου απόκρισης σε καταστάσεις έκτακτης ανάγκης κατά 12,3 λεπτά κατά μέσο όρο χάρη στις αυτοματοποιημένες ειδοποιήσεις. Εντοπίστηκαν, ωστόσο, και σημαντικοί περιορισμοί: το 42% των ηλικιωμένων συμμετεχόντων διέκοψε τη χρήση των φορετών συσκευών εντός έξι μηνών λόγω δυσφορίας, επιβάρυνσης από τη φόρτιση ή αντίληψης στίγματος· οι αλγόριθμοι ανίχνευσης πτώσεων παρήγαγαν 0,8-2,3 ψευδείς συναγερμούς την εβδομάδα, οδηγώντας σε κόπωση από τις ειδοποιήσεις· και η διαλειτουργικότητα των δεδομένων παραμένει προβληματική, καθώς κάθε προμηθευτής χρησιμοποιεί ιδιόκτητες μορφές.

Αποτελεσματικότητα της τηλεϊατρικής στη διαχείριση χρόνιων παθήσεων

Η μετα-ανάλυση των Kruse et al. (2017) σε 40 τυχαιοποιημένες ελεγχόμενες δοκιμές εξέτασε τις παρεμβάσεις τηλεϊατρικής για τον διαβήτη, την υπέρταση και την καρδιακή ανεπάρκεια σε ηλικιωμένους πληθυσμούς. Τα συγκεντρωτικά αποτελέσματα έδειξαν:

Στον διαβήτη καταγράφηκε μείωση της HbA1c κατά 0,57% (95% CI: 0,38-0,76) σε σύγκριση με τη συνήθη φροντίδα, στην υπέρταση μείωση της συστολικής αρτηριακής πίεσης κατά 6,3 mmHg (95% CI: 4,2-8,4) και στην καρδιακή ανεπάρκεια μείωση του ποσοστού επανεισαγωγής εντός 30 ημερών κατά 38% (OR=0,62, 95% CI: 0,49-0,78). Η ανάλυση ανέδειξε και τους παράγοντες εφαρμογής που προβλέπουν την επιτυχία: η καθημερινή παρακολούθηση αποδείχθηκε σημαντικά αποτελεσματικότερη από την εβδομαδιαία ($p=0,003$)· ο συνδυασμός αυτόματων ειδοποιήσεων προς τους κλινικούς ιατρούς με καθοδήγηση των ασθενών (μέγεθος επίδρασης $d=0,72$) υπερέχει της απλής συλλογής δεδομένων ($d=0,31$)· τα συστήματα που συνδέονται με το ηλεκτρονικό αρχείο υγείας πέτυχαν 23% καλύτερα αποτελέσματα από τις αυτόνομες πλατφόρμες· και η πρόσβαση σε τεχνική υποστήριξη αύξησε τη συμμόρφωση κατά 47%.

Υποστήριξη οικογενειακών φροντιστών μέσω της τεχνολογίας

Οι Chi & Demiris (2015) εξέτασαν τεχνολογικές παρεμβάσεις που υποστηρίζουν τους οικογενειακούς φροντιστές ηλικιωμένων ατόμων. Η συστηματική ανασκόπηση 37 μελετών εντόπισε τέσσερις κατηγορίες παρεμβάσεων:

Οι κατηγορίες αυτές είναι η παροχή πληροφοριών, με εκπαιδευτικό περιεχόμενο και οδηγούς διαχείρισης συμπτωμάτων (12 μελέτες)· η διευκόλυνση της επικοινωνίας μεταξύ φροντιστή και ασθενούς ή φροντιστή και κλινικού ιατρού (18 μελέτες)· η υποστήριξη της λήψης αποφάσεων,

με εργαλεία σχεδιασμού φροντίδας και διαχείρισης φαρμάκων (9 μελέτες)· και η κοινωνική υποστήριξη μέσω δικτύων ομοτίμων και ομάδων υποστήριξης (8 μελέτες).

Όσον αφορά τα αποτελέσματα στο φορτίο των φροντιστών, οι πλατφόρμες επικοινωνίας μείωσαν τις βαθμολογίες κατάθλιψης κατά 2,8 μονάδες στην κλίμακα Zarit Burden Interview ($p=0,012$), τα εργαλεία υποστήριξης λήψης αποφάσεων βελτίωσαν τις βαθμολογίες ικανότητας κατά 15% ($p=0,008$), ενώ οι συνδυασμένες παρεμβάσεις επικοινωνίας και υποστήριξης αποφάσεων αποδείχθηκαν οι πιο αποτελεσματικές, με μέγεθος επίδρασης $d=0,64$ στη μείωση του φορτίου. Τα τεχνολογικά εμπόδια, ωστόσο, παραμένουν: το 34% των φροντιστών ανέφερε την πολυπλοκότητα ως τον κύριο λόγο μη υιοθέτησης, το 28% δεν διέθετε τις απαραίτητες συσκευές και το 19% εξέφρασε ανησυχίες για την προστασία της ιδιωτικής ζωής κατά την ανταλλαγή δεδομένων υγείας.

2.1.3 Ασφάλεια και προστασία των δεδομένων υγείας

Συμμόρφωση με τον HIPAA στις ψηφιακές εφαρμογές υγείας

Ο νόμος Health Insurance Portability and Accountability Act (HIPAA), που θεσπίστηκε το 1996 και επικαιροποιήθηκε μέσω του νόμου HITECH Act (2009) και του Omnibus Rule (2013), θεσπίζει ολοκληρωμένα μέτρα προστασίας των προστατευόμενων πληροφοριών υγείας (PHI). Οι Arrari & Johnson (2010) ανέλυσαν 172 παραβιάσεις δεδομένων υγείας, εντοπίζοντας κοινές ευπάθειες που ισχύουν για την ψηφιακή υγεία:

Τα τεχνικά μέτρα προστασίας (45 CFR § 164.312) περιλαμβάνουν τον έλεγχο πρόσβασης με μοναδική αναγνώριση χρήστη, αυτόματη αποσύνδεση και κρυπτογράφηση, τη συστηματική καταγραφή και εξέταση της πρόσβασης στις προστατευόμενες πληροφορίες υγείας, ελέγχους ακεραιότητας που διασφαλίζουν ότι οι πληροφορίες δεν έχουν τροποποιηθεί ή καταστραφεί, και την ασφάλεια μετάδοσης μέσω κρυπτογράφησης κατά τη μεταφορά (απαιτείται TLS 1.2 και άνω). Στα διοικητικά μέτρα εντάσσονται η περιοδική ανάλυση κινδύνου για τα τρωτά σημεία ασφαλείας, η εκπαίδευση του προσωπικού στην ασφάλεια και στις διαδικασίες αντιμετώπισης παραβιάσεων, καθώς και οι συμφωνίες με επιχειρηματικούς συνεργάτες που χειρίζονται PHI. Τα στατιστικά στοιχεία των παραβιάσεων είναι ενδεικτικά της σημασίας των μέτρων αυτών: το 83% των παραβιάσεων αφορούσε ηλεκτρονικά PHI, τα περιστατικά παραβίασης συστημάτων αυξήθηκαν κατά 256% μεταξύ 2014 και 2019 και το μέσο κόστος αποκατάστασης ανήλθε σε 408 δολάρια ανά παραβιασμένο αρχείο ασθενούς.

Επιπτώσεις του GDPR στην επεξεργασία δεδομένων υγείας

Ο Γενικός Κανονισμός για την Προστασία Δεδομένων (GDPR) της Ευρωπαϊκής Ένωσης, που τέθηκε σε ισχύ τον Μάιο του 2018, ταξινομεί τα δεδομένα υγείας ως «προσωπικά δεδομένα ειδικής κατηγορίας» που απαιτούν ενισχυμένη προστασία. Βασικές διατάξεις που επηρεάζουν τις ψηφιακές εφαρμογές υγείας:

Ως προς τη νομική βάση επεξεργασίας δεδομένων υγείας (άρθρο 9), απαιτείται είτε ρητή συγκατάθεση - ελεύθερη, συγκεκριμένη, ενημερωμένη και σαφής - είτε αναγκαιότητα για την παροχή υγειονομικής περίθαλψης ή την ιατρική διάγνωση, πάντοτε υπό την αρχή της ελαχιστοποίησης των δεδομένων, δηλαδή τη συλλογή μόνο όσων είναι επαρκή, σχετικά και περιορισμένα στους απαραίτητους σκοπούς. Ο κανονισμός κατοχυρώνει σειρά ατομικών δικαιωμάτων: το δικαίωμα πρόσβασης (άρθρο 15) σε αντίγραφο των δεδομένων υγείας, το δικαίωμα στη φορητότητα (άρθρο 20) σε μορφή αναγνώσιμη από μηχανή, το δικαίωμα διαγραφής (άρθρο 17) - το «δικαίωμα στη λήθη», με εξαιρέσεις στον τομέα της υγειονομικής περίθαλψης - και το δικαίωμα διόρθωσης ανακριβών πληροφοριών (άρθρο 16). Σε τεχνικό και οργανωτικό επίπεδο προβλέπονται η προστασία της ιδιωτικότητας από τον σχεδιασμό, η εκτίμηση επιπτώσεων στην προστασία δεδομένων (DPIA) για επεξεργασίες υψηλού κινδύνου, όπως η παρακολούθηση της υγείας σε μεγάλη κλίμακα, και η κοινοποίηση παραβιάσεων στην εποπτική αρχή εντός 72 ωρών από την ανίχνευσή τους.

Το πλαίσιο «Προστασία της ιδιωτικής ζωής από το σχεδιασμό» της Canoukian (2011), στο οποίο γίνεται αναφορά στις αιτιολογικές σκέψεις του ΓΚΠΔ, προτείνει επτά θεμελιώδεις αρχές που έχουν μεγάλη σημασία για την παρούσα διατριβή:

Οι αρχές αυτές είναι: η προληπτική και όχι αντιδραστική προσέγγιση, με πρόβλεψη των συμβάντων παραβίασης πριν από την εμφάνισή τους· η προστασία της ιδιωτικότητας ως προεπιλογή, με αυτόματες ρυθμίσεις μέγιστης προστασίας χωρίς να απαιτείται ενέργεια του χρήστη· η ενσωμάτωση της ιδιωτικότητας στον ίδιο τον σχεδιασμό ως αναπόσπαστο μέρος της λειτουργικότητας και όχι ως πρόσθετο· η λογική του θετικού και όχι μηδενικού αθροίσματος, με αποφυγή ψευδών διλημάτων όπως «ιδιωτικότητα ή ασφάλεια»· η ασφάλεια από άκρο σε άκρο σε ολόκληρο τον κύκλο ζωής των δεδομένων, από τη συλλογή έως τη διαγραφή· η ορατότητα και η διαφάνεια των λειτουργιών, ώστε να υπόκεινται σε επαλήθευση· και η εστίαση στον χρήστη, με προτεραιότητα στα συμφέροντα των υποκειμένων των δεδομένων.

Αυθεντικοποίηση και εξουσιοδότηση στα συστήματα υγειονομικής περίθαλψης

Οι Fernandez-Aleman et al. (2013) εξέτασαν την ασφάλεια και την ιδιωτικότητα στα συστήματα EHR, προσδιορίζοντας τον έλεγχο πρόσβασης βάσει ρόλων (RBAC) ως το κυρίαρχο μοντέλο εξουσιοδότησης. Το RBAC εκχωρεί δικαιώματα σε ρόλους και όχι σε μεμονωμένους χρήστες, επιτρέποντας την κλιμακωτή διαχείριση προνομιών.

Μοτίβα εφαρμογής RBAC:

Στην εφαρμογή του RBAC διακρίνονται τρία μοτίβα: το βασικό RBAC, όπου οι χρήστες αναλαμβάνουν ρόλους και οι ρόλοι φέρουν δικαιώματα· το ιεραρχικό RBAC, όπου οι ρόλοι κληρονομούν δικαιώματα από ανώτερους ρόλους (π.χ. γιατρός > νοσηλεύτης > υπάλληλος)· και το περιορισμένο RBAC, που επιβάλλει διαχωρισμό καθηκόντων, ώστε για παράδειγμα ο ίδιος χρήστης να μην μπορεί ταυτόχρονα να συνταγογραφεί και να χορηγεί φάρμακα. Ως προς τους μηχανισμούς αυθεντικοποίησης στην υγειονομική περίθαλψη, η βασική γραμμή είναι οι κωδικοί πρόσβασης τουλάχιστον 12 χαρακτήρων με απαιτήσεις πολυπλοκότητας και εναλλαγή ανά 90 ημέρες· ο «χρυσός κανόνας» είναι η πολυπαραγοντική αυθεντικοποίηση (MFA), που συνδυάζει κωδικό με εφαρμογή επαλήθευσης ή SMS και βιομετρικά στοιχεία· ενώ χρησιμοποιούνται και έξυπνες κάρτες PIV συμβατές με FIPS 201 σε ομοσπονδιακά συστήματα, καθώς και βιομετρικές μέθοδοι όπως το δακτυλικό αποτύπωμα και η αναγνώριση προσώπου, με επιφυλάξεις ακρίβειας για ηλικιωμένους χρήστες λόγω αρθρίτιδας και γήρανσης του προσώπου.

Στα σύγχρονα API υγειονομικής περίθαλψης, τα JSON Web Tokens (JWT) έχουν αναδειχθεί σε δημοφιλές πρότυπο αυθεντικοποίησης χωρίς κατάσταση. Τα πλεονεκτήματά τους είναι ότι δεν απαιτούν αποθήκευση συνεδρίας στον διακομιστή, βελτιώνοντας την επεκτασιμότητα, ότι μεταδίδονται εύκολα μέσω κεφαλίδων HTTP ή cookies, και ότι είναι αυτοδύναμα, καθώς οι αξιώσεις τους περιλαμβάνουν την ταυτότητα του χρήστη, τους ρόλους και τη λήξη χωρίς αναζήτηση στη βάση δεδομένων. Η ασφαλής χρήση τους προϋποθέτει tokens πρόσβασης μικρής διάρκειας (15-60 λεπτά) με μηχανισμό ανανέωσης, προτίμηση του ασύμμετρου αλγορίθμου RS256 έναντι του συμμετρικού HS256 στην παραγωγή και αποφυγή ενσωμάτωσης ευαίσθητων PHI σε μη κρυπτογραφημένες αξιώσεις.

2.1.4 Προσβασιμότητα σε εφαρμογές υγειονομικής περίθαλψης

Οδηγίες προσβασιμότητας περιεχομένου ιστού (WCAG) 2.1

Το WCAG 2.1 του W3C, το οποίο έλαβε την ιδιότητα της Σύστασης W3C τον Ιούνιο του 2018, επεκτείνει το WCAG 2.0 με 17 επιπλέον κριτήρια επιτυχίας που αφορούν την προσβασιμότητα σε κινητές συσκευές, την χαμηλή όραση και τις γνωστικές αναπηρίες. Τρία επίπεδα συμμόρφωσης:

Προβλέπονται τρία επίπεδα συμμόρφωσης: το Επίπεδο A, που αποτελεί την ελάχιστη βασική προσβασιμότητα και πρέπει να πληροῦται για οποιαδήποτε σχετική αξίωση· το Επίπεδο AA, που αντιμετωπίζει τα πιο συνηθισμένα εμπόδια και αποτελεί τον συνιστώμενο στόχο για εφαρμογές ιστού· και το Επίπεδο AAA, που προσφέρει βελτιωμένη προσβασιμότητα για εξειδικευμένους πληθυσμούς, χωρίς να είναι εφικτό για το σύνολο του περιεχομένου.

Από τα κριτήρια επιτυχίας, τα πλέον συναφή με την παρακολούθηση της υγείας ηλικιωμένων οργανώνονται στους τέσσερις άξονες του προτύπου. Στον άξονα του αντιληπτού περιλαμβάνονται η αντίθεση κειμένου και εικόνων με αναλογία 4,5:1 για το επίπεδο AA και 7:1 για το AAA (κριτήριο 1.4.3), η δυνατότητα μεγέθυνσης του κειμένου έως 200% χωρίς απώλεια λειτουργικότητας (1.4.4), η αναδιάταξη του περιεχομένου σε πλάτος 320 εικονοστοιχείων χωρίς οριζόντια κύλιση (1.4.10, νέο στην έκδοση 2.1) και η διατήρηση του περιεχομένου όταν ο χρήστης προσαρμόζει το ύψος γραμμής και τα διαστήματα (1.4.12). Στον άξονα του λειτουργικού εντάσσονται η διαθεσιμότητα όλων των λειτουργιών μέσω πληκτρολογίου (2.1.1), η δυνατότητα επέκτασης των χρονικών ορίων (2.2.1), η διατήρηση νοήματος στη σειρά εστίασης (2.4.3) και οι στόχοι αφής τουλάχιστον 44×44 εικονοστοιχείων CSS (2.5.5). Στον άξονα του κατανοητού ανήκουν το επίπεδο ανάγνωσης κατώτερης δευτεροβάθμιας εκπαίδευσης (3.1.5), η συνεπής πλοήγηση με τους επαναλαμβανόμενους μηχανισμούς στην ίδια σειρά (3.2.3) και η αναγνώριση των σφαλμάτων με σαφή περιγραφή τους σε κείμενο (3.3.1). Τέλος, στον άξονα του σταθερού περιλαμβάνονται η ύπαρξη προσβάσιμων ονομάτων και ρόλων για όλα τα στοιχεία της διεπαφής (4.1.2) και τα μηνύματα κατάστασης που κοινοποιούνται στις υποστηρικτικές τεχνολογίες χωρίς να λαμβάνουν εστίαση (4.1.3, νέο στην έκδοση 2.1).

Συμπληρωματικά, η προδιαγραφή ARIA (Accessible Rich Internet Applications) παρέχει σημασιολογία για προσαρμοσμένα στοιχεία και δυναμικό περιεχόμενο. Στους πίνακες ελέγχου υγειονομικής περίθαλψης, βασικά πρότυπα αποτελούν ο προσδιορισμός τμημάτων του πίνακα για τα προγράμματα ανάγνωσης οθόνης μέσω `role="region"` με κατάλληλο `aria-label`, η ανακοίνωση ενημερωμένων τιμών υγείας χωρίς διακοπή του χρήστη μέσω `aria-live="polite"`, η σύνδεση των πεδίων φόρμας με επεξηγηματικό κείμενο μέσω `aria-describedby` και η επισήμανση υποχρεωτικών πεδίων και σφαλμάτων επικύρωσης με `aria-required` και `aria-invalid`.

Γνωστική προσβασιμότητα

Η εργασία των Seeman & Cooper (2017) σχετικά με τη γνωστική προσβασιμότητα για το W3C επισημαίνει παράγοντες που πρέπει να λαμβάνονται υπόψη για χρήστες με ήπια γνωστική δυσλειτουργία (MCI), η οποία επηρεάζει το 15-20% των ενηλίκων άνω των 65 ετών:

Στρατηγικές σχεδιασμού:

Οι προτεινόμενες στρατηγικές σχεδιασμού περιλαμβάνουν συνεπείς διατάξεις με προβλέψιμη αρχιτεκτονική πληροφοριών που μειώνει το γνωστικό φορτίο, σαφή γλώσσα χωρίς ορολογία και με μία κύρια ιδέα ανά πρόταση, οπτικές ενισχύσεις όπου τα εικονίδια συμπληρώνουν το κείμενο με επαναλαμβανόμενες ενδείξεις (χρώμα, εικονίδιο, ετικέτα), προοδευτική αποκάλυψη με εμφάνιση των βασικών πληροφοριών αρχικά και απόκρυψη της πολυπλοκότητας πίσω από προαιρετικές επεκτάσεις, και πρόληψη σφαλμάτων με επαλήθευση των εισόδων πριν από την υποβολή και βήματα επιβεβαίωσης για τις καταστροφικές ενέργειες.

2.1.5 Τεχνητή νοημοσύνη, μηχανική μάθηση και εξατομίκευση της αλληλεπίδρασης

Πέρα από τους τέσσερις άξονες που εξετάστηκαν παραπάνω, το αντικείμενο της παρούσας εργασίας συνδέεται με ένα ευρύτερο σώμα έρευνας για την αλληλεπίδραση ανθρώπου-υπολογιστή με τεχνητή νοημοσύνη, την εξατομίκευση και τη μοντελοποίηση του χρήστη. Οι ενότητες που ακολουθούν εξετάζουν τη βιβλιογραφία αυτή και τη συνδέουν με τις σχεδιαστικές επιλογές του συστήματος που αναπτύχθηκε.

Κεντρικό στοιχείο των σύγχρονων ευφυών διεπαφών αποτελεί η σύνθεση περισσότερων του ενός παραδειγμάτων μηχανικής μάθησης. Οι Hatzilygeroudis, Tsihrintzis και Jain (2023) εξετάζουν τις θεωρητικές βάσεις και τις πρακτικές εφαρμογές της ενσωμάτωσης πολλαπλών παραδειγμάτων, όπως τα νευρωνικά δίκτυα, οι γενετικοί αλγόριθμοι και η ασαφής λογική, σε ενιαία υβριδικά συστήματα. Η σύνθεση αυτή ενισχύει την προσαρμοστικότητα και την ανθεκτικότητα των διαδραστικών συστημάτων λογισμικού, επιτρέποντάς τους να αντιμετωπίζουν σύνθετες πραγματικές συνθήκες, στις οποίες κανένας μεμονωμένος αλγόριθμος δεν αποδίδει ικανοποιητικά. Παράλληλα, η αναλυτική της μάθησης (learning analytics), το πεδίο

της οποίας αποτυπώνεται στη συλλογή άρθρων των Virvou et al. (2020), δείχνει πώς τα δεδομένα αλληλεπίδρασης μετατρέπονται σε γνώση για τη συμπεριφορά του χρήστη και, κατ' επέκταση, σε βάση για την προσαρμογή του συστήματος.

Η εξατομίκευση με βάση δεδομένα έχει μελετηθεί συστηματικά και μέσω τεχνικών ομαδοποίησης. Οι Sotiropoulos et al. (2006) εφαρμόζουν ομαδοποίηση δεδομένων πελατών βασισμένη σε τεχνητά ανοσοποιητικά συστήματα σε περιβάλλον ηλεκτρονικών αγορών, ενώ οι Virvou et al. (2007) αξιοποιούν την ίδια οικογένεια αλγορίθμων για την κατασκευή στερεοτύπων χρηστών σε προσαρμοστικό ηλεκτρονικό κατάστημα. Οι Savvopoulos et al. (2008) εντάσσουν την ομαδοποίηση για τη μοντελοποίηση του χρήστη σε πλήρη κύκλο ζωής ανάπτυξης λογισμικού βασισμένο στη μεθοδολογία RUP, και οι Savvopoulos και Virvou (2010) αξιολογούν τη γενικότητα του πλαισίου αυτού σε διαφορετικά προσαρμοστικά συστήματα. Η ίδια λογική εφαρμόζεται και σε πολυμεσικό περιεχόμενο: οι Lampropoulos, Sotiropoulos και Tsihrintzis (2004) εξατομικεύουν την αντίληψη της μουσικής ομοιότητας μέσω επιλογής υποσυνόλου χαρακτηριστικών, ενώ οι Lampropoulos, Lampropoulou και Tsihrintzis (2005) βελτιώνουν την ταξινόμηση μουσικών ειδών με τεχνική διαχωρισμού πηγών. Κοινό στοιχείο των εργασιών αυτών είναι ότι το προφίλ του χρήστη δεν δηλώνεται ρητά από τον ίδιο, αλλά συνάγεται από τα δεδομένα χρήσης - προσέγγιση που προϋποθέτει, ωστόσο, όγκο δεδομένων και χρόνο συσσώρευσης τα οποία δεν είναι διαθέσιμα σε ένα σύστημα παρακολούθησης υγείας κατά την πρώτη χρήση.

Η μετάβαση από την απλή χρήση εργαλείων τεχνητής νοημοσύνης σε μια αμφίδρομη σχέση ανθρώπου και συστήματος περιγράφεται από τη Virvou (2022), η οποία εισάγει την έννοια της αναδυόμενης εποχής της αλληλεπίδρασης ανθρώπου-τεχνητής νοημοσύνης. Στην εκτενή ανασκόπηση που ακολούθησε, η Virvou (2023) τεκμηριώνει τη σχέση αμοιβαιότητας μεταξύ τεχνητής νοημοσύνης και εμπειρίας χρήστη: η τεχνητή νοημοσύνη αναβαθμίζει την εμπειρία του χρήστη, ενώ οι αυξημένες προσδοκίες που δημιουργεί η αναβαθμισμένη εμπειρία απαιτούν με τη σειρά τους ακόμη πιο σύνθετες τεχνικές τεχνητής νοημοσύνης.

Η εξατομίκευση, ωστόσο, προϋποθέτει και κατάλληλη αρχιτεκτονική λογισμικού. Οι Alerpis και Virvou (2014) προτείνουν αντικειμενοστρεφή μοντέλα για τη δόμηση εξατομικευμένων διεπαφών σε κινητές συσκευές, ενώ οι Virvou και Alerpis (2005) είχαν δείξει νωρίτερα πώς τα συγγραφικά εργαλεία μπορούν να συστηματοποιήσουν την παραγωγή εξατομικευμένου περιεχομένου για κινητές πλατφόρμες. Και οι δύο εργασίες τοποθετούν την εξατομίκευση στο επίπεδο του σχεδιασμού και όχι σε εκ των υστέρων ρυθμίσεις της διεπαφής - αρχή που ακολουθεί και η παρούσα εργασία, όπου οι διακριτές όψεις του ασθενούς και του φροντιστή προκύπτουν από το μοντέλο ρόλων του συστήματος.

2.1.6 Συναισθηματική εξατομίκευση και πολυτροπική αναγνώριση συναισθημάτων

Η προσαρμογή της αλληλεπίδρασης στη συναισθηματική κατάσταση του χρήστη αποτελεί διακριτό ερευνητικό πεδίο με άμεσο αντίκτυπο στη συμμετοχή και την ικανοποίηση του χρήστη. Οι Tsihrintzis et al. (2008), σε δύο συναφείς εργασίες, δείχνουν ότι η αναγνώριση συναισθημάτων από την έκφραση του προσώπου βελτιώνεται όταν συμπληρώνεται με πληροφορία από τα πρότυπα πληκτρολόγησης και από το ηχητικό-γλωσσικό κανάλι, επεκτείνοντας έτσι το πεδίο ανίχνευσης των συναισθημάτων. Οι Stathopoulou et al. (2010) εξετάζουν συστηματικά τα οφέλη της υποστήριξης ενός συστήματος οπτικής αναγνώρισης συναισθημάτων του προσώπου από πληροφορίες προτύπων πληκτρολόγησης, υπογραμμίζοντας τη συνέργεια μεταξύ διαφορετικών τρόπων αλληλεπίδρασης. Στην ίδια κατεύθυνση, οι Alerpis et al. (2010) παρουσιάζουν σύστημα διπλής τροπικότητας που συνδυάζει ηχητικά-γλωσσικά και οπτικά χαρακτηριστικά του προσώπου, ενώ οι Virvou et al. (2012) αντιμετωπίζουν τον συνδυασμό των δύο τροπικοτήτων ως πρόβλημα λήψης απόφασης πολλαπλών κριτηρίων, με βάση εμπειρικές μελέτες.

Η πολυτροπική αναγνώριση στηρίζεται σε επιμέρους υποσυστήματα, καθένα από τα οποία έχει μελετηθεί αυτοτελώς. Οι Stathopoulou και Tsihrintzis (2005) παρουσιάζουν σύστημα ανίχνευσης προσώπου και ταξινόμησης εκφράσεων σε εικόνες, το οποίο εξελίσσεται αργότερα σε προσέγγιση ανίχνευσης προσώπου με τεχνητά νευρωνικά δίκτυα βασισμένη στην εμφάνιση (Stathopoulou και Tsihrintzis, 2011). Πέρα από το πρόσωπο, δεύτερη εργασία της ίδιας χρονιάς

δείχνει ότι και οι κινήσεις του σώματος και οι χειρονομίες φέρουν αξιοποιήσιμη συναισθηματική πληροφορία, ενώ οι Lampropoulos και Tsihrintzis (2012) αξιολογούν περιγραφείς MPEG-7 για την αναγνώριση συναισθήματος από την ομιλία. Τα επιμέρους αυτά κανάλια είχαν ήδη τεθεί σε κοινή εμπειρική βάση από τους Virvou et al. (2007), οι οποίοι συνδυάζουν μελέτες για την ηχητική-γλωσσική και την οπτική-προσωπική τροπικότητα.

Η έρευνα έχει επεκταθεί και πέρα από το μεμονωμένο άτομο σε ελεγχόμενες συνθήκες. Οι Triantafyllou και Tsihrintzis (2018) εξετάζουν την αναγνώριση του συναισθήματος ομάδας, αξιολογώντας βασικές μεθόδους αυτόματης ταξινόμησης, ενώ οι Triantafyllou et al. (2021) παρουσιάζουν διτροπικό σύστημα αναγνώρισης συναισθημάτων που λειτουργεί τόσο για γνωστά όσο και για άγνωστα στο σύστημα πρόσωπα, σε κανονικές συνθήκες αλλά και υπό κόπωση. Η τελευταία αυτή παράμετρος έχει ιδιαίτερη σημασία για τον πληθυσμό που αφορά η παρούσα εργασία, καθώς η κόπωση και οι διακυμάνσεις της φυσικής κατάστασης είναι συνήθειες στους ηλικιωμένους χρήστες και αλλοιώνουν ακριβώς τα σήματα στα οποία στηρίζεται η αναγνώριση.

Η αξιοποίηση της αναγνωρισμένης συναισθηματικής πληροφορίας απαιτεί, επιπλέον, ένα μοντέλο χρήστη που να την ερμηνεύει. Οι Katsionis και Virvou (2004) εισάγουν μια γνωστική θεωρία για τη συναισθηματική μοντελοποίηση του χρήστη σε εκπαιδευτικό παιχνίδι εικονικής πραγματικότητας, ώστε να δημιουργούνται λεπτομερέστερα προφίλ που λαμβάνουν υπόψη τις ατομικές διαφορές στις συναισθηματικές αντιδράσεις. Οι Aleris και Virvou (2006) δομούν στερεότυπα χρηστών με βάση συναισθηματικά χαρακτηριστικά, προσφέροντας οικονομικό τρόπο αρχικοποίησης της συμπεριφοράς του συστήματος, ενώ οι Aleris, Virvou και Kabassi (2006) δείχνουν ότι οι ενέργειες του χρήστη στο μικρόφωνο και στο πληκτρολόγιο αρκούν για τη συναισθηματική μοντελοποίηση μαθητών, χωρίς εξειδικευμένο εξοπλισμό.

Το πεδίο αυτό είναι ιδιαίτερα κρίσιμο για τους ηλικιωμένους χρήστες, στους οποίους η ματαίωση από μια δυσνόητη διεπαφή οδηγεί συχνά σε οριστική εγκατάλειψη της εφαρμογής, όπως τεκμηριώνεται στην ενότητα 2.1.1. Η παρούσα εργασία δεν υλοποιεί αναγνώριση συναισθημάτων, επιλογή που συνδέεται και με τους περιορισμούς ιδιωτικότητας της ενότητας 2.1.3, αλλά αντιμετωπίζει το ίδιο πρόβλημα με σχεδιαστικά μέσα: πρόληψη σφαλμάτων, βήματα επιβεβαίωσης στις κρίσιμες ενέργειες και υποστηρικτικά αντί επικριτικών μηνυμάτων. Η πολυτροπική συναισθηματική ανάδραση παραμένει, πάντως, σαφής κατεύθυνση μελλοντικής επέκτασης.

2.1.7 Μοντέλα χρηστών, συλλογισμός και προσαρμοστική καθοδήγηση

Συλλογισμός για τις ενέργειες και τα σφάλματα του χρήστη

Ο συλλογισμός σε μοντέλα χρηστών αφορά τη διαδικασία κατανόησης, πρόβλεψης και απόκρισης στις ενέργειες του χρήστη μέσα σε ένα σύστημα. Η Virvou (1999) εξετάζει τον αυτόματο συλλογισμό και την παροχή βοήθειας για τα ανθρώπινα λάθη κατά τη χρήση ενός λειτουργικού συστήματος, ενώ οι Virvou και Du Boulay (1999) εμβαθύνουν στην αναγνώριση και την αντιμετώπιση των λαθών του χρήστη μέσω της υπολογιστικής εφαρμογής μιας γνωστικής θεωρίας ανθρώπινου εύλογου συλλογισμού (human plausible reasoning). Η θεωρία αυτή μεταφέρθηκε στη συνέχεια στο περιβάλλον των γραφικών διεπαφών χρήστη: οι Virvou και Kabassi (2002) μελετούν τον συλλογισμό σχετικά με τις ενέργειες του χρήστη σε γραφική διεπαφή, δείχνοντας ότι η κατανόηση της πρόθεσης πίσω από μια ενέργεια βελτιώνει τη χρηστικότητα και την αποδοτικότητα της διεπαφής. Σε δύο εργασίες του 2004, οι Virvou και Kabassi αφενός προσαρμόζουν τη θεωρία του ανθρώπινου εύλογου συλλογισμού σε γραφική διεπαφή και αφετέρου αξιολογούν μια ευφυή γραφική διεπαφή συγκρίνοντας τις παρεμβάσεις της με εκείνες ανθρώπων ειδικών, καθιερώνοντας μεθοδολογία ελέγχου της ποιότητας των συμπερασμάτων του συστήματος.

Μοντελοποίηση του μαθητή και προσαρμοστική εκπαίδευση

Το μοντέλο χρήστη αποτελεί τον πυρήνα των ευφυών συστημάτων διδασκαλίας (Intelligent Tutoring Systems, ITS). Οι Tsigira και Virvou (2003) περιγράφουν τη μοντελοποίηση του μαθητή για την εξατομίκευση της καθοδήγησης σε διαδικτυακό περιβάλλον, τονίζοντας ότι η

προσαρμογή της διδακτικής στρατηγικής προϋποθέτει ρητή αναπαράσταση της συμπεριφοράς και των αδυναμιών του μαθητή. Οι Virvou και Moundridou (2001) διακρίνουν δύο συνυπάρχοντα μοντέλα χρήστη, του μαθητή και του διδάσκοντα, και εξετάζουν την αλληλεπίδρασή τους μέσα σε συγγραφικό εργαλείο ευφυούς συστήματος διδασκαλίας· η κατεύθυνση αυτή οδήγησε στο εργαλείο συγγραφής και διάθεσης προσαρμοστικών διαδικτυακών βιβλίων των Moundridou και Virvou (2001).

Επαναλαμβανόμενο πρόβλημα των μοντέλων αυτών είναι η αρχικοποίησή τους, δηλαδή το τι γνωρίζει το σύστημα για τον χρήστη πριν συγκεντρώσει επαρκή δεδομένα αλληλεπίδρασης. Οι Tsiriga και Virvou (2002) προτείνουν αρχικοποίηση του μοντέλου του μαθητή με συνδυασμό στερεοτύπων και μηχανικής μάθησης, ενώ σε δεύτερη εργασία της ίδιας χρονιάς περιγράφουν τη δυναμική αρχικοποίηση του μοντέλου σε διαδικτυακό σύστημα διδασκαλίας γλώσσας. Η αποτελεσματικότητα των ευφυών χαρακτηριστικών ενός τέτοιου συστήματος αξιολογείται στη συνέχεια εμπειρικά από τους Tsiriga και Virvou (2004). Την ανάγκη συμμετοχής των τελικών χρηστών σε ολόκληρο τον κύκλο ζωής, και όχι μόνο στο στάδιο της αξιολόγησης, είχαν ήδη τεκμηριώσει οι Virvou και Tsiriga (2000) - θέση που υιοθετείται και εδώ, καθώς οι απαιτήσεις του συστήματος αντλούνται από τις ανάγκες των ασθενών και των φροντιστών (Κεφάλαιο 3).

Η Virvou (2018) περιγράφει τη σύγκλιση της εξατομίκευσης με τις τεχνικές τεχνητής νοημοσύνης ως νέα εποχή για το εκπαιδευτικό λογισμικό, με αλληλεπιδράσεις που προσεγγίζουν τις ανθρώπινες και κινητοποιούν το ενδιαφέρον των μαθητών. Στη δυναμικά εξατομικευμένη ηλεκτρονική εκπαίδευση των Chrysafiadi και Virvou (2013), ο μηχανισμός συλλογισμού προσαρμόζει το περιεχόμενο στην απόδοση και την πρόοδο του χρήστη. Για την αξιολόγηση των εκπαιδευομένων, οι Chrysafiadi, Virvou και Tsihrantzis (2022) προτείνουν μηχανισμό αυτόματης εξατομικευμένης αποτίμησης βασισμένο σε ασαφή λογική, ενώ οι Papadimitriou, Chrysafiadi και Virvou (2023) παράγουν προσαρμοστικά διαγωνίσματα με ασαφείς γενετικούς αλγορίθμους. Η επέκταση του συλλογισμού σε συνδυασμένες μεθόδους, με την ασαφή λογική και τη μηχανική μάθηση να συνυπάρχουν, μελετάται από τους Chrysafiadi et al. (2023) σε προσαρμοστικό περιβάλλον μάθησης προγραμματισμού, το οποίο αξιολογήθηκε τόσο ως προς την εμπειρία του χρήστη όσο και ως προς τα μαθησιακά αποτελέσματα σε ακαδημαϊκό πληθυσμό στην Ελλάδα. Συμπληρωματικά, η Virvou (2012) αξιολογεί τον συνδυασμένο ρόλο των παιχνιδιών εικονικής πραγματικότητας και των κινούμενων πρακτόρων στην εκπαίδευση, με βάση τον συλλογισμό για τις αλληλεπιδράσεις και τις προτιμήσεις των χρηστών.

Μεγάλο μέρος της εφαρμοσμένης έρευνας στο πεδίο αυτό έχει αναπτυχθεί στο πλαίσιο των εκπαιδευτικών παιχνιδιών, όπου η εξατομίκευση πρέπει να συνυπάρξει με τη διατήρηση του ενδιαφέροντος του χρήστη. Οι Katsionis και Virvou (2008) υλοποιούν εξατομικευμένη ηλεκτρονική μάθηση μέσω εκπαιδευτικού παιχνιδιού εικονικής πραγματικότητας που αξιοποιεί υπηρεσίες ιστού, ενώ οι Papadimitriou και Virvou (2017) εισάγουν προσαρμοστικότητα στα σενάρια εκπαιδευτικού παιχνιδιού περιπέτειας - προσέγγιση που τυποποιείται με ασαφή λογική στο σύστημα FuzzEG των Papadimitriou, Chrysafiadi και Virvou (2019) και επεκτείνεται σε κινητά περιβάλλοντα και σε εικονική πραγματικότητα από τους Papadimitriou et al. (2021). Συμπληρωματικά, οι Kamitsios et al. (2018) χρησιμοποιούν στερεοτυπικό μοντέλο χρήστη ώστε οι δυσκολίες του ίδιου του παιχνιδιού να μην υπονομεύουν τον εκπαιδευτικό στόχο. Η συνολική εικόνα του πεδίου της εξατομικευμένης διαδικτυακής εκπαίδευσης αποτυπώνεται στο βιβλίο των Chrysafiadi και Virvou (2015).

Η παρούσα εργασία δεν αποτελεί εκπαιδευτικό σύστημα, αντλεί όμως από το πεδίο αυτό δύο αρχές. Πρώτον, ότι η καθοδήγηση του χρήστη οφείλει να θεμελιώνεται σε ρητή αναπαράσταση του ίδιου του χρήστη και όχι σε γενικές οδηγίες: στο σύστημα που αναπτύχθηκε, ο ρόλος, η ηλικία, η φαρμακευτική αγωγή και οι πρόσφατες μετρήσεις του ασθενούς συγκροτούν το πλαίσιο που τροφοδοτεί τον βοηθό τεχνητής νοημοσύνης (Κεφάλαιο 5). Δεύτερον, ότι η διεπαφή οφείλει να προλαμβάνει και να ερμηνεύει τα σφάλματα του χρήστη αντί να τα καταγράφει, αρχή που αποτυπώνεται στις επικυρώσεις των φορμών και στα βήματα επιβεβαίωσης που περιγράφονται στο Κεφάλαιο 3.

2.1.8 Εξατομικευμένες εφαρμογές, ιδιωτικότητα και ηθική χρήση της τεχνητής νοημοσύνης

Οι εξατομικευμένες εφαρμογές έχουν επεκταθεί σε πολλούς τομείς, από την υγεία και τον τουρισμό έως τη διαχείριση καταστάσεων έκτακτης ανάγκης, με στόχο την προσαρμογή των υπηρεσιών στις ατομικές ανάγκες και προτιμήσεις. Εγγύτερη στο αντικείμενο της παρούσας εργασίας είναι η έξυπνη κινητή πολυμεσική εφαρμογή υποστήριξης ηλικιωμένων των Pavlakis, Aleris και Virvou (2012), η οποία συνδυάζει εξατομικευμένες υπενθυμίσεις, παρακολούθηση της υγείας και εργαλεία επικοινωνίας, με στόχο τη βελτίωση της ποιότητας ζωής και τη διατήρηση της ανεξαρτησίας των ηλικιωμένων. Πρόκειται για άμεσο προηγούμενο της προσέγγισης που ακολουθείται εδώ, με τη διαφορά ότι η παρούσα εργασία αποδίδει τη λειτουργικότητα αυτή ως προσβάσιμη διαδικτυακή πλατφόρμα πολλαπλών ρόλων, με ρητή συνεργασία ασθενούς και φροντιστή.

Σε άλλα πεδία εφαρμογής, οι Kontogianni et al. (2024) εξετάζουν την επίδραση της τεχνητής νοημοσύνης και της τεχνολογίας blockchain στον έξυπνο τουρισμό, όπου η ανάλυση των προτιμήσεων, των ιστορικών δεδομένων και των πληροφοριών πραγματικού χρόνου παράγει εξατομικευμένες εμπειρίες, αναδεικνύοντας ταυτόχρονα το ζήτημα της προστασίας και της ασφάλειας των δεδομένων. Οι Alonistioti et al. (2023) εφαρμόζουν ασαφή λογική στην εξατομίκευση των ειδοποιήσεων έκτακτης ανάγκης για πυρκαγιές, λαμβάνοντας υπόψη την τοποθεσία, τη σοβαρότητα του περιστατικού και τις ατομικές συνθήκες του παραλήπτη - περίπτωση στην οποία η ορθή εξατομίκευση έχει άμεσες συνέπειες για την ασφάλεια του χρήστη.

Η ίδια μεθοδολογική βάση έχει εφαρμοστεί και σε πεδία εκτός της άμεσης αλληλεπίδρασης με τον τελικό χρήστη, γεγονός που δείχνει τη γενικότητά της. Οι Panagoulas, Sotiropoulos και Tsihrintzis (2021) συνδυάζουν διατροφικούς βιοδείκτες με μηχανική μάθηση για εξατομικευμένες εφαρμογές διατροφής και βελτιστοποίηση της υγείας - περίπτωση ιδιαίτερα συγγενής με την παρούσα εργασία, καθώς στηρίζεται σε αντικειμενικές μετρήσεις του ίδιου του ατόμου και όχι σε δηλωμένες προτιμήσεις. Οι Alogogianni και Virvou (2021) εφαρμόζουν συσχετιστική ταξινόμηση κατά τη μεθοδολογία CRISP-DM στο πρόβλημα της αδήλωτης εργασίας, ενώ οι Theocharis και Tsihrintzis (2016) εξετάζουν τα κρίσιμα ζητήματα των συστημάτων διαχείρισης γνώσης στον δημόσιο τομέα. Στην υποστήριξη αποφάσεων για τη διαχείριση ενέργειας, τέλος, οι Panagoulas et al. (2023) δείχνουν ότι η αξία του αποτελέσματος εξαρτάται από την προσαρμοσμένη επεξήγησή του στον συγκεκριμένο αποδέκτη - ζήτημα που εξετάζεται αμέσως παρακάτω.

Η αξία της εξατομίκευσης εξαρτάται, όμως, από τη διαχείριση των προσωπικών δεδομένων που την τροφοδοτούν. Οι Mousiakiou και Virvou (2017) δείχνουν πώς οι απαιτήσεις του Γενικού Κανονισμού Προστασίας Δεδομένων (GDPR) μπορούν να αποτυπωθούν σε μοντέλα UML κατά τον σχεδιασμό, ώστε η συμμόρφωση να μην αντιμετωπίζεται ως εκ των υστέρων προσθήκη, ενώ οι Politou et al. (2022) συστηματοποιούν τις προκλήσεις ιδιωτικότητας και προστασίας δεδομένων στην κατανεμημένη εποχή, όπου τα δεδομένα διασπείρονται σε πλατφόρμες και συσκευές. Πέρα από τη συμμόρφωση, η ηθική χρήση της τεχνητής νοημοσύνης στη διαδραστική επικοινωνία με τους χρήστες απαιτεί και επεξηγησιμότητα των αλγορίθμων: οι Panagoulas, Virvou και Tsihrintzis (2024) προτείνουν ένα νέο πλαίσιο επεξηγησιμότητας που συνδυάζει το μοντέλο αποδοχής της τεχνολογίας, το οποίο παρουσιάστηκε στην ενότητα 2.1.1, με εργαλείο ταχείας εκτίμησης του ιατρικού αλφαριθμητισμού των ενηλίκων και τεχνικές μηχανικής μάθησης, ώστε η εξήγηση να προσαρμόζεται στην ικανότητα κατανόησης του συγκεκριμένου χρήστη. Σε επίπεδο αρχιτεκτονικής, οι Panagoulas, Virvou και Tsihrintzis (2022) προτείνουν επαναληπτική ανάπτυξη ιατρικών εφαρμογών με τεχνητή νοημοσύνη βασισμένη σε μικρο-υπηρεσίες, με ρητούς στόχους τη χρηστικότητα, την αξιοπιστία και την επεξηγησιμότητα.

Οι αρχές αυτές αποτυπώνονται άμεσα στις επιλογές της παρούσας εργασίας: ο έλεγχος πρόσβασης βάσει ρόλων περιορίζει την ορατότητα των δεδομένων υγείας εντός του νοικοκυριού του ασθενούς, ενώ η επιλογή τοπικά εκτελούμενου γλωσσικού μοντέλου αντί υπηρεσίας νέφους αποτρέπει την έξοδο ευαίσθητων δεδομένων υγείας από την υποδομή του συστήματος.

2.1.9 Μεγάλα γλωσσικά μοντέλα, εμπιστοσύνη και αξιολόγηση

Η ραγδαία διάδοση των μεγάλων γλωσσικών μοντέλων (Large Language Models, LLMs) μετέθεσε το ερευνητικό ενδιαφέρον από τις δυνατότητες προς την αξιοπιστία και την εμπιστοσύνη. Οι Virvou και Tsihrintzis (2023) προτείνουν ένα νέο μοντέλο κατάστασης εμπιστοσύνης (trust state-chart) για τη μηχανική απαιτήσεων λογισμικού ενισχυμένου με τεχνητή νοημοσύνη, αντιμετωπίζοντας την εμπιστοσύνη ως απαίτηση που προδιαγράφεται και παρακολουθείται από τα πρώτα στάδια της ανάπτυξης και όχι ως εκ των υστέρων διαπίστωση. Στο ίδιο πλαίσιο, επισημαίνουν ότι η αξιοποίηση έτοιμων, προ-εκπαιδευμένων συστημάτων όπως το ChatGPT δεν καταργεί τον ρόλο του ανθρώπου-ειδικού σε θέματα τεχνητής νοημοσύνης, αλλά τον καθιστά κρισιμότερο, καθώς η διατύπωση των ερωτημάτων, ο έλεγχος και η ερμηνεία των απαντήσεων απαιτούν εξειδίκευση. Η κατεύθυνση αυτή επεκτείνεται στο μοντέλο VIRTSI των Virvou, Tsihrintzis και Tsihrintzi (2024), το οποίο περιγράφει τη δυναμική της εμπιστοσύνης στη συνεργασία ανθρώπου και τεχνητής νοημοσύνης με βάση εμπειρική αξιολόγηση του ChatGPT.

Στο εκπαιδευτικό πεδίο, οι Virvou και Tsihrintzis (2023) εισάγουν ολιστικό πλαίσιο αξιολόγησης του ChatGPT θεμελιωμένο στα κριτήρια των ευφύων συστημάτων διδασκαλίας, ώστε το όφελος να αποτιμάται με βάση τα μαθησιακά αποτελέσματα και τη συμμετοχή των εκπαιδευομένων και όχι με βάση την εντύπωση που προκαλεί η γλωσσική ευχέρεια του μοντέλου· η αξιοποίησή του σε περιβάλλον ηλεκτρονικής μάθησης για την πολιτιστική κληρονομιά παρουσιάζεται από τους Virvou et al. (2023). Στο ιατρικό πεδίο, που ενδιαφέρει άμεσα την παρούσα εργασία, οι Panagoulas, Virvou και Tsihrintzis (2024) δείχνουν ότι ο εμπλουτισμός ενός γλωσσικού μοντέλου με ρητούς κανόνες βελτιώνει τις αλληλεπιδράσεις σε εξειδικευμένο γνωστικό πεδίο, με μελέτη περίπτωσης την ιατρική διάγνωση.

Τα ευρήματα αυτά καθορίζουν τον τρόπο με τον οποίο η παρούσα εργασία αξιοποιεί το γλωσσικό μοντέλο. Ο βοηθός υγείας δεν καλείται να διαγνώσει ή να προτείνει θεραπεία, αλλά να εξηγήσει και να συνοψίσει τα καταχωρισμένα δεδομένα του χρήστη· η προτροπή συστήματος περιορίζει ρητά το πεδίο δράσης του και επιβάλλει δήλωση αποποίησης ευθύνης με σύσταση για επικοινωνία με ιατρό (απαίτηση FR-28, Κεφάλαιο 3). Η επιλογή αυτή αποτελεί εφαρμογή της αρχής ότι η εμπιστοσύνη σε ένα σύστημα τεχνητής νοημοσύνης οικοδομείται μέσα από περιορισμένο, ελεγχόμενο και εξηγήσιμο πεδίο δράσης.

2.2 Ανάλυση της εμπορικής τεχνολογίας στον τομέα της υγειονομικής περίθαλψης

2.2.1 Τμηματοποίηση της αγοράς πλατφορμών υγείας για καταναλωτές

Η παγκόσμια αξία της αγοράς ψηφιακών υπηρεσιών υγείας για καταναλωτές ξεπέρασε τα 100 δισεκατομμύρια δολάρια το 2022, με προβλεπόμενο CAGR 18,8% έως το 2030 (Grand View Research). Τα τμήματα της αγοράς που σχετίζονται με την παρακολούθηση της υγείας των ηλικιωμένων περιλαμβάνουν:

Το πρώτο τμήμα είναι οι πλατφόρμες φυσικής κατάστασης και ευεξίας, με ηγέτες την Fitbit (μερίδιο αγοράς 31%), την Apple (27%), την Garmin (12%) και τη Samsung (8%). Απευθύνονται κυρίως στην ηλικιακή ομάδα 25-54 ετών - η διείσδυση στους ηλικιωμένους δεν ξεπερνά το 18% - με μοντέλο εσόδων που συνδυάζει πωλήσεις υλικού και συνδρομητικές υπηρεσίες, και βασικά χαρακτηριστικά την παρακολούθηση δραστηριότητας, την ανάλυση ύπνου, την καθοδηγούμενη άσκηση και τις κοινωνικές προκλήσεις. Το δεύτερο τμήμα αποτελούν οι εφαρμογές διαχείρισης χρόνιων παθήσεων, όπως οι MySugr για τον διαβήτη, Livongo για πολλαπλές παθήσεις και Omada Health για την πρόληψη· απευθύνονται σε ενήλικες με διαγνωσμένες χρόνιες παθήσεις (μέση ηλικία 52 ετών), λειτουργούν κυρίως με μοντέλο B2B2C μέσω συμβάσεων με εργοδότες και ασφαλιστικά ταμεία και προσφέρουν καταγραφή γλυκόζης και αρτηριακής πίεσης, υπενθυμίσεις φαρμακευτικής αγωγής, καθοδήγηση και βιβλιοθήκες περιεχομένου.

Το τρίτο τμήμα είναι τα εργαλεία συμμόρφωσης με τη φαρμακευτική αγωγή, όπως τα Medisafe, Mango Health και Round Health, που απευθύνονται σε ηλικιωμένους με πολυφαρμακία (κατά

μέσο όρο πέντε και πλέον φάρμακα), ακολουθούν μοντέλο freemium με δωρεάν βασική έκδοση και premium συνδρομή 4-10 δολαρίων μηνιαίως και προσφέρουν προγράμματα δόσεων, υπενθυμίσεις ανανέωσης και προειδοποιήσεις αλληλεπιδράσεων φαρμάκων. Το τέταρτο τμήμα είναι οι πλατφόρμες τηλεϊατρικής - Teladoc, Amwell, MDLive, Doctor on Demand - που απευθύνονται σε όλες τις ηλικίες (οι ηλικιωμένοι αντιπροσωπεύουν το 23% των εικονικών επισκέψεων), χρεώνουν ανά επίσκεψη (49-79 δολάρια) ή με μηνιαία συνδρομή και παρέχουν τηλεδιασκέψεις, ηλεκτρονικές συνταγές και δεύτερες γνώμες.

Σύγκριση κατηγοριών πλατφορμών:

Πίνακας 2.2: Σύγκριση κατηγοριών πλατφορμών

Κατηγορία	Υιοθέτηση από ηλικιωμένους	Προσβασιμότητα	Προστασία προσωπικών δεδομένων	Λειτουργίες φροντιστή	Ιατρική ενσωμάτωση
Φυσική κατάσταση/ευεξία	Χαμηλή (18%)	Κακή	Μέτρια	Ελάχιστη	Χαμηλή
Διαχείριση χρόνιων παθήσεων	Μέτρια (34%)	Καλή	Καλή	Περιορισμένη	Μέτρια
Τήρηση φαρμακευτικής αγωγής	Υψηλή (47%)	Καλή	Καλή	Μέτρια	Χαμηλή
Τηλεϊατρική	Μέτρια (38%)	Κατά μέσο όρο	Άριστη	Περιορισμένη	Υψηλή
Παρακολούθηση ηλικιωμένων (Διπλωματική εργασία)	Στόχος	Άριστη	Άριστη	Άριστο	Μέτριο

2.2.2 Τάσεις τεχνολογικών στοιβών σε εφαρμογές υγειονομικής περίθαλψης

Η ανάλυση 50 κορυφαίων ψηφιακών εφαρμογών υγείας (μέσω BuiltWith, Wappalyzer και δημόσιων αποθετηρίων GitHub) αποκαλύπτει κυρίως τεχνολογικά πρότυπα:

Στα πλαίσια frontend κυριαρχεί το React με ποσοστό 52% (Facebook Health Tools, Zocdoc, Livongo, 23andMe), χάρη στην επαναχρησιμοποίηση στοιχείων, το εκτεταμένο οικοσύστημα, την ισχυρή υποστήριξη TypeScript, τη διαθεσιμότητα προγραμματιστών και τη δυνατότητα μεταφοράς σε κινητές πλατφόρμες. Ακολουθεί το Angular με 28% (Epic MyChart mobile, πύλες ασθενών Cerner), που προτιμάται για τη δομημένη αρχιτεκτονική του, τα εταιρικά εργαλεία, την εγγενή υποστήριξη TypeScript και την ενσωμάτωση με παλαιότερα συστήματα .NET, και το Vue.js με 14% (Babylon Health, Headspace), που ξεχωρίζει για την ευκολία εκμάθησης και την ελαφριά, ευέλικτη αρχιτεκτονική του· το υπόλοιπο 6% καλύπτεται από vanilla JavaScript και παλαιά συστήματα.

Στις τεχνολογίες backend προηγείται το Node.js με 41%, λόγω της κοινής γλώσσας με το frontend και των δυνατοτήτων πραγματικού χρόνου μέσω WebSocket, και ακολουθούν η Python με 23% (Django, Flask, ενσωμάτωση μηχανικής μάθησης), η Java/Spring με 18% (εταιρικά συστήματα EHR όπως τα backend των Epic και Cerner), η Go με 9% - αναδυόμενη επιλογή για API υψηλής απόδοσης, φιλική προς το Docker - η Ruby/Rails με 6% και η .NET/C# με 3%. Στις βάσεις δεδομένων, το PostgreSQL προηγείται με 38% (συμμόρφωση ACID, ευελιξία JSONB, ανοιχτός κώδικας), έναντι 27% του MongoDB (μοντέλο εγγράφων για μη δομημένα δεδομένα), 18% του MySQL, 11% του Microsoft SQL Server και 6% των cloud-native λύσεων (DynamoDB, Cosmos DB, Firebase). Ως προς τα μοτίβα ανάπτυξης, οι πλατφόρμες cloud κατανέμονται σε AWS (48%), Azure (28%), Google Cloud (16%) και πολλαπλά clouds (8%), η υιοθέτηση του Docker φθάνει το 67% μεταξύ των σύγχρονων εφαρμογών υγείας με το Kubernetes στο 31%, ενώ το serverless μοντέλο κερδίζει έδαφος (23%) για φόρτους εργασίας που βασίζονται σε συμβάντα, όπως η αυτοματοποιημένη επεξεργασία ειδοποιήσεων.

2.2.3 Ωριμότητα προσβασιμότητας σε εμπορικές πλατφόρμες

Η ετήσια αξιολόγηση των 100 κορυφαίων ιστότοπων υγείας (2023) από το WebAIM αποκάλυψε επίμονα κενά προσβασιμότητας:

Οι συχνότερες παραβιάσεις ήταν το κείμενο χαμηλής αντίθεσης (86% των ιστότοπων - η πιο διαδεδομένη παραβίαση των WCAG), τα πεδία εισαγωγής χωρίς ετικέτες (53%), οι εικόνες χωρίς εναλλακτικό κείμενο (41%), οι σύνδεσμοι χωρίς αναγνωρίσιμο κείμενο (33%) και τα κουμπιά χωρίς προσβάσιμα ονόματα (27%), με μέσο όρο 47,3 σφάλματα ανά σελίδα - οριακά καλύτερο από τον μέσο όρο 50,1 του γενικού ιστού. Ισχυρή προσβασιμότητα (βαθμολογία άνω του 90) πέτυχαν ελάχιστες πλατφόρμες: το GrandPad (94), ειδικά σχεδιασμένο για ηλικιωμένους, η πύλη ασθενών της Kaiser Permanente (91) και το MyHealthVet του Υπουργείου Βετεράνων των ΗΠΑ (90). Αντίθετα, χαμηλές βαθμολογίες (κάτω του 70) σημείωσαν πολλές πλατφόρμες προμηθευτών φορετών συσκευών (Fitbit web: 68, Garmin Connect: 62) και πολλές διαδικτυακές εκδόσεις εφαρμογών διαχείρισης χρόνιων παθήσεων (κάτω του 65). Η βασική διαπίστωση είναι ότι η προσβασιμότητα συχνά συγκρούεται με την πολυπλοκότητα των λειτουργιών: οι σύνθετοι διαδραστικοί πίνακες ελέγχου θυσιάζουν την προσβασιμότητα για χάρη της οπτικής εντύπωσης.

2.3 Ανάλυση τεχνολογικής ανάπτυξης

2.3.1 Επιλογή πλαισίου frontend: React vs. Angular vs. Vue

Το React, που επιλέχθηκε για την παρούσα διατριβή, προσφέρει σημαντικά πλεονεκτήματα: το οικοσύστημα στοιχείων του Material-UI παρέχει περισσότερα από 50 έτοιμα στοιχεία κατάλληλα για εφαρμογές υγείας (κάρτες, φόρμες, διαγράμματα) με ενσωματωμένη προσβασιμότητα· η ενσωμάτωση TypeScript επιτρέπει σταδιακή υιοθέτηση ισχυρής τυποποίησης που μειώνει τα σφάλματα εκτέλεσης - κρίσιμο για δεδομένα υγείας· η βιβλιοθήκη Recharts προσφέρει δηλωτική δημιουργία γραφημάτων για την οπτικοποίηση μετρήσεων· ο ελεγκτής js-x-a11y εντοπίζει παραβιάσεις ARIA ήδη κατά τη μεταγλώττιση· και η κοινότητα είναι τεράστια, με περισσότερες από 11 εκατομμύρια εβδομαδιαίες λήψεις στο npm. Στα μειονεκτήματά του συγκαταλέγονται η «παράλυση της ευελιξίας», καθώς οι πολλαπλές επιλογές δρομολόγησης και διαχείρισης κατάστασης απαιτούν αρχιτεκτονικές αποφάσεις, και ο γρήγορος ρυθμός αλλαγών που συνεπάγεται κόστος συντήρησης. Ειδικά για την υγειονομική περίθαλψη, το React είναι ουδέτερο ως προς τη συμμόρφωση με το HIPAA - αυτή εξαρτάται από την υλοποίηση - υποστηρίζει λειτουργία εκτός σύνδεσης μέσω service workers και PWA, χρήσιμη για ηλικιωμένους σε αγροτικές περιοχές, και το Virtual DOM εξασφαλίζει ομαλές αλληλεπιδράσεις ακόμη και με περισσότερα από 1.000 σημεία δεδομένων ανά γράφημα.

Το Angular προσφέρει ολοκληρωμένη λύση «όλα σε ένα» - δρομολογητή, πελάτη HTTP, φόρμες και διαχείριση κατάστασης σε ενιαίο πακέτο, περιορίζοντας την κόπωση αποφάσεων - με εταιρική υποστήριξη από την Google, πολιτικές μακροχρόνιας υποστήριξης και εγγενή επιβολή της ασφάλειας τύπων μέσω TypeScript. Ωστόσο, η καμπύλη εκμάθησής του είναι απότομη (RxJS observables, decorators, dependency injection), το βασικό του πακέτο είναι μεγαλύτερο (45kB έναντι 30kB του React) και απαιτεί περισσότερο επαναλαμβανόμενο κώδικα για απλά στοιχεία. Το Vue.js, τέλος, είναι ιδιαίτερα προσιτό χάρη στα βασισμένα σε HTML πρότυπά του, ελαφρύ και αποδοτικό, με ενθυλάκωση προτύπου, λογικής και στυλ σε στοιχεία ενός αρχείου· διαθέτει όμως μικρότερο οικοσύστημα εξειδικευμένων βιβλιοθηκών υγείας και περιορισμένη εταιρική υιοθέτηση στον αμερικανικό χώρο της υγειονομικής περίθαλψης.

Μήτρα αποφάσεων:

Κριτήριο	React	Angular	Vue.js	Βάρος	Σταθμισμένη βαθμολογία
Εργαλεία προσβασιμότητας	9/10	7/10	6/10	25	React: 2,25
Υποστήριξη TypeScript	8/10	10/10	8/10	20	Angular: 2.0
Βιβλιοθήκες στοιχείων υγειονομικής περίθαλψης	9/10	6/10	5/10	20	React: 1.8
Διαθεσιμότητα προγραμματιστή	9/10	7/10	6/10	15	Αντίδραση: 1,35
Καμπύλη μάθησης	7/10	4/10	8/10	10	Vue: 0,8
Απόδοση	8/10	7/10	9/10	10	Vue: 0,9
ΣΥΝΟΛΙΚΗ ΒΑΘΜΟΛΟΓΙΑ	8,05	6,8	6,75	100	React Επιλεγμένο

Πίνακας 2.3: Πίνακας συμβιβασμών πλαισίου frontend

Αιτιολόγηση: Το ανώτερο οικοσύστημα εργαλείων προσβασιμότητας του React, η ωριμότητα της βιβλιοθήκης στοιχείων υγειονομικής περίθαλψης (Material-UI με πρότυπα υγειονομικής περίθαλψης) και το βάθος της αγοράς προγραμματιστών υπερτερούν της ολοκληρωμένης δομής του Angular και της απλότητας του Vue για αυτήν την εφαρμογή υγειονομικής περίθαλψης που απευθύνεται σε ηλικιωμένους.

2.3.2 Επιλογή γλώσσας backend: Go εναντίον Node.js εναντίον Java

Go (επιλεγμένη για την παρούσα διατριβή)

Τα πλεονεκτήματα της Go είναι ουσιαστικά για το συγκεκριμένο πεδίο: μεταγλωττίζεται σε εγγενές δυαδικό αρχείο και είναι 10-20 φορές ταχύτερη από το διερμηνευόμενο Node.js σε εργασίες εντατικής χρήσης επεξεργαστή· οι goroutines επιτρέπουν τη διαχείριση άνω των 10.000 ταυτόχρονων συνδέσεων WebSocket με ελάχιστη μνήμη· η ανάπτυξη ενιαίου δυαδικού αρχείου χωρίς εξαρτήσεις χρόνου εκτέλεσης απλοποιεί τις εικόνες Docker (12 MB έναντι 900 MB για Node.js με τις εξαρτήσεις του)· η πλούσια τυπική βιβλιοθήκη (διακομιστής HTTP, κωδικοποίηση JSON, κρυπτογραφία) ελαχιστοποιεί τις εξωτερικές εξαρτήσεις και τον κίνδυνο εφοδιαστικής αλυσίδας· η στατική τυποποίηση προλαμβάνει συνήθη σφάλματα εκτέλεσης· και ο ρητός χειρισμός σφαλμάτων μέσω επιστρεφόμενων τιμών υποχρεώνει τον προγραμματιστή να αντιμετωπίζει τις αποτυχίες. Στα μειονεκτήματα καταγράφονται το μικρότερο οικοσύστημα για την υγειονομική περίθαλψη - λιγότερες βιβλιοθήκες FHIR και εργαλεία ενσωμάτωσης EHR σε σύγκριση με την Java - και ο περισσότερος επαναλαμβανόμενος κώδικας για απλές λειτουργίες CRUD σε σχέση με το Node.js. Για τις ανάγκες της εφαρμογής αξιοποιούνται το ώριμο ORM GORM με υποστήριξη PostgreSQL, μετεγκαταστάσεις σχήματος και συσχετίσεις, το ταχύ πλαίσιο δρομολόγησης Gin με middleware για CORS και αυθεντικοποίηση, και η βιβλιοθήκη golang-jwt για ασφαλή δημιουργία και επικύρωση tokens.

Το Node.js προσφέρει «JavaScript παντού» με κοινή γλώσσα σε frontend και backend, το τεράστιο οικοσύστημα NPM με περισσότερα από δύο εκατομμύρια πακέτα - συμπεριλαμβανομένων υλοποιήσεων διακομιστή FHIR - αρχιτεκτονική οδηγούμενη από συμβάντα με non-blocking I/O κατάλληλη για πίνακες παρακολούθησης σε πραγματικό χρόνο και ταχύτατη προτυποποίηση με το Express.js. Ωστόσο, το μονονηματικό μοντέλο του σημαίνει ότι εργασίες εντατικής χρήσης επεξεργαστή μπλοκάρουν τον βρόχο συμβάντων, τα node_modules μπορούν να υπερβούν τα 500 MB με ευπάθειες σε μεταβατικές εξαρτήσεις, και η δυναμική τυποποίηση της JavaScript είναι επιρρεπής σε σφάλματα σε ευαίσθητους υπολογισμούς υγείας, όπως οι δοσολογίες φαρμάκων. Η Java με το Spring Framework παραμένει το εταιρικό πρότυπο των νοσοκομειακών συστημάτων EHR, με ώριμες βιβλιοθήκες (HAPI FHIR, αναλυτές HL7 v2, εκτενής κρυπτογραφία) και ισχυρή ασφάλεια τύπων, αλλά

συνοδεύεται από βαρύ περιβάλλον εκτέλεσης JVM άνω των 200 MB, αργούς χρόνους εκκίνησης (10-30 δευτερόλεπτα για Spring Boot), τρεις έως πέντε φορές περισσότερες γραμμές κώδικα για ισοδύναμη λειτουργικότητα και αποτύπωμα μνήμης που περιορίζει την πυκνότητα των containers.

Λόγοι απόφασης: Επιλογή Go για (1) απόδοση: 50ms p95 καθυστέρηση έναντι 120ms για Node.js σε συγκριτικές αξιολογήσεις API μετρήσεων υγείας, (2) ασφάλεια: μικρότερη επιφάνεια επίθεσης με λιγότερες εξαρτήσεις, (3) απλότητα ανάπτυξης: η ενιαία δυαδική συσκευασία σε container μειώνει την λειτουργική πολυπλοκότητα για μελλοντική κλιμάκωση στο cloud.

2.3.3 Επιλογή βάσης δεδομένων: PostgreSQL έναντι MongoDB έναντι MySQL

PostgreSQL (επιλεγμένο για την παρούσα διατριβή)

Το PostgreSQL, που επιλέχθηκε για την παρούσα διατριβή, προσφέρει πλήρη συμμόρφωση ACID με εγγυήσεις συναλλαγών για την ακεραιότητα των δεδομένων υγείας, υποστηρίζει JSONB που επιτρέπει ευέλικτο σχήμα για εξελισσόμενες μετρήσεις χωρίς απώλεια της σχεσιακής δομής, προηγμένη ευρετηρίαση (B-tree, GiST, GIN) που βελτιστοποιεί τις ερωτήσεις χρονοσειρών, εγγενείς τύπους δεδομένων όπως η χρονική σήμανση με ζώνη ώρας - κρίσιμη για την κατανεμημένη παροχή φροντίδας - περιορισμούς ξένων κλειδιών και ελέγχου που επιβάλλουν την αναφορική ακεραιότητα, ενσωματωμένη αναζήτηση πλήρους κειμένου για ιατρικό ιστορικό και ονόματα φαρμάκων, και ένα ώριμο οικοσύστημα τριάντα και πλέον ετών, δοκιμασμένο σε χρηματοοικονομικά συστήματα και συστήματα υγείας. Ειδικά για την υγειονομική περίθαλψη, η ασφάλεια σε επίπεδο γραμμής (RLS) επιβάλλει την απομόνωση των δεδομένων των ασθενών στο επίπεδο της ίδιας της βάσης, η πλήρης καταγραφή συναλλαγών υποστηρίζει τη συμμόρφωση με το HIPAA και η ροή αναπαραγωγής (streaming replication) επιτρέπει αντίγραφα ανάγνωσης για αναλυτικά φορτία χωρίς επιβάρυνση των εγγράφων παραγωγής. Το κύριο μειονέκτημά του, η σχετικά πιο σύνθετη οριζόντια κλιμάκωση σε εκατοντάδες κόμβους, δεν αφορά τις τυπικές εφαρμογές παρακολούθησης νοικοκυριών ηλικιωμένων.

Το MongoDB προσφέρει ευελιξία σχήματος με το μοντέλο εγγράφων του, ενσωματωμένη οριζόντια κατάτμηση για πολύ μεγάλα σύνολα δεδομένων και φιλική εμπειρία ανάπτυξης, καθώς τα έγγραφα JSON αντιστοιχούν φυσικά σε αντικείμενα JavaScript· ωστόσο, οι εγγυήσεις ACID του είναι ασθενέστερες - οι συναλλαγές πολλαπλών εγγράφων προστέθηκαν μόλις στην έκδοση 4.0 (2018) - η απόδοση των συνενώσεων μέσω του aggregation pipeline υπολείπεται των SQL JOIN για σχεσιακές ερωτήσεις και η απουσία ξένων κλειδιών μεταθέτει την επιβολή της ακεραιότητας στο επίπεδο της εφαρμογής. Η MySQL, τέλος, διαθέτει καθολική διάδοση και απλούστερη αρχική ρύθμιση, αλλά οι λειτουργίες JSON της είναι λιγότερο ώριμες από το JSONB του PostgreSQL, η ασύγχρονη αναπαραγωγή μπορεί να οδηγήσει σε ανάγνωση παρωχημένων δεδομένων σε εγκαταστάσεις πολλαπλών περιοχών και προηγμένες δυνατότητες όπως η κατάτμηση και η συγκέντρωση νημάτων προϋποθέτουν την επί πληρωμή έκδοση Enterprise.

Λόγοι απόφασης: Το PostgreSQL επιλέχθηκε για (1) απαιτήσεις **ακεραιότητας δεδομένων** των ιατρικών αρχείων μέσω εγγυήσεων ACID και περιορισμών ξένων κλειδιών, (2) **βελτιστοποίηση χρονοσειρών** μέσω ευρετηρίων χρονικής σήμανσης για διαχρονικές ερωτήσεις μετρήσεων υγείας, (3) υποστήριξη **συμμόρφωσης με κανονισμούς** μέσω ολοκληρωμένης καταγραφής ελέγχου, (4) **μελλοντική συμβατότητα** μέσω της ευελιξίας JSONB για την ενσωμάτωση διαφορετικών μορφών δεδομένων φορητών συσκευών χωρίς μεταφορές σχήματος.

2.4 Αξιολόγηση της τρέχουσας κατάστασης

2.4.1 Ανάλυση της αγοράς ψηφιακής υγείας

Μέγεθος αγοράς και τροχιά ανάπτυξης

Η παγκόσμια αγορά ψηφιακής υγείας παρουσιάζει ισχυρή ανάπτυξη, η οποία οφείλεται στη γήρανση του πληθυσμού, στην επικράτηση χρόνιων ασθενειών και στην επιτάχυνση της υιοθέτησης της τηλεϊατρικής λόγω της COVID-19:

Η παγκόσμια αγορά αποτιμήθηκε σε 220 δισεκατομμύρια δολάρια το 2022 (Verified Market Research), με πρόβλεψη να φθάσει τα 660 δισεκατομμύρια έως το 2030 (CAGR 18,4% για την περίοδο 2023-2030). Η Βόρεια Αμερική κατέχει μερίδιο 42% χάρη στην ώριμη υποδομή πληροφορικής υγείας, ενώ η περιοχή Ασίας-Ειρηνικού εμφανίζει τον ταχύτερο ρυθμό ανάπτυξης (CAGR 21,8%), λόγω του μεγάλου ηλικιωμένου πληθυσμού σε Κίνα, Ιαπωνία και Νότια Κορέα. Κατά τεχνολογία, η τηλεϊατρική και η τηλεμετρία αντιπροσωπεύουν το 35% της αγοράς (77 δισεκατομμύρια δολάρια το 2022), οι εφαρμογές κινητής υγείας (mHealth) το 28% (61,6 δισεκατομμύρια), η ανάλυση δεδομένων υγείας το 18% (39,6 δισεκατομμύρια), τα ψηφιακά συστήματα υγείας EHR/EMR το 12% (26,4 δισεκατομμύρια) και οι λοιπές κατηγορίες, όπως το λογισμικό πλατφορμών φορετών συσκευών, το 7% (15,4 δισεκατομμύρια). Ως προς τις επενδύσεις, η χρηματοδότηση επιχειρηματικών κεφαλαίων στην ψηφιακή υγεία κορυφώθηκε στα 29,1 δισεκατομμύρια δολάρια το 2021 (Rock Health) και υποχώρησε στα 15,3 δισεκατομμύρια το 2022 λόγω διόρθωσης της αγοράς - παραμένοντας ωστόσο τριπλάσια των προ πανδημίας επιπέδων - με τη διάγνωση μέσω τεχνητής νοημοσύνης και την απομακρυσμένη παρακολούθηση ασθενών να προσελκύουν το 38% των επενδύσεων.

Ποσοστά υιοθέτησης τεχνολογίας φροντίδας ηλικιωμένων

Παρά την ανάπτυξη της αγοράς, η υιοθέτηση από τους ηλικιωμένους υστερεί σημαντικά σε σχέση με τις νεότερες ηλικιακές ομάδες:

Σύμφωνα με το Pew Research (2024), smartphone κατέχει το 96% των ηλικιών 18-49, το 83% των ηλικιών 50-64 και μόλις το 61% των ατόμων άνω των 65 ετών - με το 21% να χρησιμοποιεί απλά κινητά τηλέφωνα και το 18% να μην έχει καθόλου κινητό. Η χρήση εφαρμογών υγείας ακολουθεί ανάλογη φθίνουσα πορεία (μελέτη AARP 2023, N=2.412 ενήλικες άνω των 50): 47% στις ηλικίες 50-64, 32% στις ηλικίες 65-74 και 18% στις ηλικίες άνω των 75. Ως εμπόδια υιοθέτησης αναφέρονται η πολυπλοκότητα («πολύ δύσκολο να χρησιμοποιηθούν» κατά το 58%), οι ανησυχίες για την ιδιωτικότητα (47%), το κόστος συσκευών και συνδρομών (38%), η έλλειψη αντιληπτής αξίας (34%, «δεν βλέπουν πώς θα τους βοηθούσε») και οι φυσικές δυσκολίες όρασης ή δεξιότητας (27%). Αντίστροφα, οι παράγοντες επιτυχούς υιοθέτησης είναι σαφείς: η σύσταση από τον πάροχο υγειονομικής περίθαλψης αυξάνει την υιοθέτηση κατά 3,8 φορές, η υποστήριξη από την οικογένεια οδηγεί σε 2,6 φορές υψηλότερη συνεχή χρήση, ενώ οι εφαρμογές με λιγότερες από πέντε βασικές λειτουργίες παρουσιάζουν 67% υψηλότερη διατήρηση χρηστών έναντι των πλούσιων σε λειτουργίες εναλλακτικών.

2.4.2 Τεχνολογικά κενά και προβληματικά σημεία

Εμπόδια στην εμπειρία χρήστη για τους ηλικιωμένους

Η ευριστική αξιολόγηση 30 δημοφιλών εφαρμογών υγείας εντόπισε επαναλαμβανόμενες αποτυχίες στην εμπειρία χρήστη για τους ηλικιωμένους χρήστες:

Στις κρίσιμες αποτυχίες καταγράφονται το προεπιλεγμένο μέγεθος κειμένου κάτω των 16 εικονοστοιχείων στο 73% των εφαρμογών (ακατάλληλο για πρεσβυωπία), η μη τήρηση της αναλογίας αντίθεσης WCAG AA (4,5:1) στα βασικά διαδραστικά στοιχεία στο 68%, κουμπιά και σύνδεσμοι μικρότεροι των 44 εικονοστοιχείων στο 82%, μέσο βάθος πλοήγησης 4,2 επιπέδων έναντι συνιστώμενου μεγίστου τριών, και χρονικά όρια φορμών 60 δευτερολέπτων ή λιγότερο στο 54%. Στα μέτρια προβλήματα περιλαμβάνονται η απαίτηση χειρονομιών τσιμπήματος για μεγέθυνση ή σάρωσης χωρίς εναλλακτικές στο 41%, η χρήση ιατρικής ορολογίας χωρίς ορισμούς (π.χ. «συστολική», «κορεσμός») στο 67% και τα τεχνικά μηνύματα σφάλματος αντί

οδηγιών σε απλή γλώσσα στο 88%. Παραδειγματικές πρακτικές εντοπίστηκαν σε μόλις τέσσερις από τις τριάντα εφαρμογές: διαμορφώσιμα μεγέθη γραμματοσειράς ανεξάρτητα από τις ρυθμίσεις του λειτουργικού συστήματος, λειτουργία υψηλής αντίθεσης με εναλλαγή θεμάτων, φωνητική εισαγωγή δεδομένων που μειώνει τον φόρτο πληκτρολόγησης και βοήθεια συναρτώμενη με το περιβάλλον, με ενσωματωμένες επεξηγήσεις των ιατρικών όρων κατά την πρώτη εμφάνισή τους.

Προκλήσεις ενσωμάτωσης και διαλειτουργικότητας

Παρά τις προσπάθειες τυποποίησης FHIR (Fast Healthcare Interoperability Resources), η ανταλλαγή δεδομένων παραμένει κατακερματισμένη:

Στην ενσωμάτωση με τα ηλεκτρονικά αρχεία υγείας, το API του Epic MyChart υποστηρίζει το FHIR R4 αλλά απαιτεί συμφωνίες οργανωτικής συνεργασίας, το Cerner παραμένει στην παλαιότερη έκδοση HL7 FHIR DSTU2 περιοριζόμενο στους βασικούς πόρους (ασθενής, παρατήρηση, πάθηση), ενώ τα Allscripts και Athenahealth εμφανίζουν ανομοιογενή ωριμότητα υλοποίησης, συχνά μόνο για ανάγνωση. Στα API των φορετών συσκευών, το Fitbit προσφέρει Web API με OAuth 2.0 και δεδομένα δραστηριότητας, ύπνου και καρδιακού ρυθμού σε JSON, το Apple HealthKit είναι διαθέσιμο μόνο σε iOS χωρίς API από την πλευρά του διακομιστή - η εξαγωγή δεδομένων απαιτεί τη συσκευή του χρήστη - το Google Fit παρέχει REST API και Android SDK με διαλειτουργικότητα μεταξύ πλατφορμών αλλά ασθενέστερη ενσωμάτωση φορετών συσκευών από την Apple, ενώ Withings και Garmin διατηρούν ιδιόκτητα API που απαιτούν χωριστή προσπάθεια ενσωμάτωσης το καθένα.

Προβλήματα δημιουργούν και οι ίδιες οι μορφές των δεδομένων: οι μονάδες μέτρησης διαφέρουν (αρτηριακή πίεση σε mmHg ή kPa, βάρος σε κιλά ή λίβρες) και απαιτούν κανονικοποίηση· η χρονική λεπτομέρεια ποικίλλει, καθώς ορισμένες συσκευές αναφέρουν ωριαίους μέσους όρους και άλλες στιγμιαίες μετρήσεις, καθιστώντας κρίσιμη την τυποποίηση της χρονικής σήμανσης· και τα κενά δεδομένων από συσκευές που δεν φοριούνται απαιτούν συνειδητές στρατηγικές παρεμβολής ή χειρισμού ελλειπουσών τιμών. Από τις πρωτοβουλίες τυποποίησης ξεχωρίζουν το SMART on FHIR, του οποίου το πρωτόκολλο εκκίνησης επιτρέπει την εκτέλεση εφαρμογών ενσωματωμένων στα EHR, και ο οδηγός εφαρμογής HL7 Personal Health Device, που ορίζει αντιστοιχίσεις FHIR για περισσότερους από 25 τύπους συσκευών· εντούτοις, μόνο το 34% των ψηφιακών εφαρμογών υγείας που εξετάστηκαν υποστηρίζει FHIR, με την πλειονότητα να χρησιμοποιεί ιδιόκτητες μορφές.

2.4.3 Κανονιστικό πλαίσιο και συμμόρφωση

Κανονισμοί της FDA για τα ψηφιακά εργαλεία υγείας

Το Σχέδιο Δράσης για την Καινοτομία στην Ψηφιακή Υγεία της FDA (2017, ενημερωμένο το 2020) κατηγοριοποιεί το λογισμικό σε κατηγορίες κινδύνου:

Στη χαμηλού κινδύνου κατηγορία της γενικής ευεξίας εντάσσονται οι εφαρμογές παρακολούθησης δραστηριότητας, διαλογισμού και καταγραφής διατροφής, για τις οποίες ισχύει διακριτική ευχέρεια επιβολής χωρίς απαίτηση ελέγχου πριν από τη διάθεση στην αγορά. Στη μέτριου κινδύνου κατηγορία της μη διαγνωστικής υποστήριξης κλινικών αποφάσεων ανήκουν οι υπενθυμίσεις φαρμάκων και οι εφαρμογές παρακολούθησης συμπτωμάτων με εκπαιδευτικό περιεχόμενο, που χαρακτηρίζονται ιατροτεχνολογικά προϊόντα κατηγορίας I ή II, με τις περισσότερες να εξαιρούνται από τη διαδικασία 510(k) εφόσον πληρούν τα κριτήρια CDS. Στην υψηλού κινδύνου διαγνωστική/θεραπευτική κατηγορία υπάγονται εφαρμογές που ανιχνεύουν κολπική μαρμαρυγή, υπολογίζουν δόσεις ινσουλίνης ή ερμηνεύουν ακτινολογικές εξετάσεις, και απαιτούν έγκριση 510(k) ή PMA. Για τον πίνακα παρακολούθησης της υγείας ηλικιωμένων της παρούσας διατριβής, το τρέχον πεδίο εφαρμογής - παρακολούθηση και εκπαιδευτική τεχνητή νοημοσύνη - εμπίπτει πιθανότατα στις εξαιρέσεις, καθώς παρέχει πληροφορίες χωρίς να διαγιγνώσκει· μελλοντικές όμως επεκτάσεις, όπως προειδοποιητικές ειδοποιήσεις που συνιστούν κλινικές ενέργειες (π.χ. «ζητήστε άμεση ιατρική βοήθεια για αρτηριακή πίεση 180/120»), ενδέχεται να ενεργοποιήσουν κανονιστική επανεξέταση.

Διεθνή πρότυπα προστασίας δεδομένων στον τομέα της υγειονομικής περίθαλψης:

Στις Ηνωμένες Πολιτείες, ο HIPAA εφαρμόζεται στους «καλυπτόμενους φορείς» (παρόχους υγειονομικής περίθαλψης, ασφαλιστικές εταιρείες, κέντρα ανταλλαγής πληροφοριών) και στους «επιχειρηματικούς συνεργάτες» τους· οι καταναλωτικές εφαρμογές που δεν χειρίζονται άμεσα δεδομένα παρόχων ενδέχεται να εξαιρούνται, ωστόσο οι βέλτιστες πρακτικές συνιστούν την ευθυγράμμιση με τον HIPAA ακόμη και για μη καλυπτόμενες οντότητες, προς οικοδόμηση εμπιστοσύνης. Στην Ευρωπαϊκή Ένωση, ο GDPR ισχύει για κάθε οντότητα που επεξεργάζεται δεδομένα κατοίκων της ΕΕ ανεξαρτήτως έδρας - άρα και υπηρεσίες παρακολούθησης ηλικιωμένων με έδρα τις ΗΠΑ και χρήστες στην ΕΕ - με κυρώσεις έως 20 εκατομμύρια ευρώ ή 4% του ετήσιου παγκόσμιου κύκλου εργασιών, όποιο είναι μεγαλύτερο. Στον Καναδά, το πλαίσιο PIPEDA βασίζεται στη συγκατάθεση και στον έλεγχο των προσωπικών πληροφοριών υγείας από τα ίδια τα άτομα, με πρόσθετες επαρχιακές ρυθμίσεις σε Κεμπέκ, Αλμπέρτα και Βρετανική Κολομβία.

Η πρακτική λίστα ελέγχου συμμόρφωσης με τον κανόνα Omnibus του HIPAA περιλαμβάνει: κρυπτογράφηση των PHI τόσο σε κατάσταση ηρεμίας (AES-256) όσο και κατά τη μεταφορά (TLS 1.2+)· αρχεία καταγραφής κάθε πρόσβασης σε PHI με αναγνωριστικό χρήστη, χρονική σήμανση και προσπελασθέντα δεδομένα· εφαρμογή της αρχής του «ελάχιστου απαραίτητου» με περιορισμό της πρόσβασης στα δεδομένα που απαιτεί ο ρόλος του χρήστη· διαδικασία ειδοποίησης των θιγόμενων εντός 60 ημερών σε περίπτωση παραβίασης· συμφωνίες επιχειρηματικών συνεργατών με τους παρόχους cloud (AWS, Azure) που διασφαλίζουν τη δική τους συμμόρφωση· και ετήσια εκπαίδευση ευαισθητοποίησης σε θέματα ασφάλειας για το προσωπικό με πρόσβαση σε PHI.

2.5 Προσδιορισμένες ελλείψεις και συμβολή της διατριβής

2.5.1 Περίληψη ανάλυσης κενών

Η σύνθεση των ευρημάτων από τη βιβλιογραφική ανασκόπηση, την ανάλυση των εμπορικών πλατφορμών και την αξιολόγηση της τεχνολογίας αποκαλύπτει έξι κρίσιμες ελλείψεις στις τρέχουσες λύσεις παρακολούθησης της.

Το πρώτο κενό αφορά τον σχεδιασμό προσβασιμότητας: το 86% των πλατφορμών υγείας δεν πληροί τις απαιτήσεις αντίθεσης WCAG AA, με μέσο όρο 47 σφάλματα προσβασιμότητας ανά σελίδα, γεγονός που αποκλείει το 58% των ηλικιωμένων χρηστών· οι υπάρχουσες λύσεις περιορίζονται είτε σε εξειδικευμένες συσκευές με φτωχές λειτουργίες υγείας (GrandPad) είτε σε ευρείες πλατφόρμες με ακατάλληλη εμπειρία χρήσης για ηλικιωμένους (Apple Health, Fitbit). Το δεύτερο κενό είναι οι αρχιτεκτονικές τεχνητής νοημοσύνης που εκθέτουν την ιδιωτικότητα: η συνομιλητική τεχνητή νοημοσύνη των εφαρμογών υγείας (π.χ. Ada, Babylon Health) αποστέλλει τα ερωτήματα των ασθενών σε εξωτερικούς παρόχους API (OpenAI, Google Cloud AI), εκθέτοντας προστατευόμενες πληροφορίες σε τρίτους, τη στιγμή που το 71% των ηλικιωμένων δυσπιστεί απέναντι στην τεχνητή νοημοσύνη υγείας του cloud· δεν εντοπίστηκε καμία καταναλωτική εφαρμογή υγείας με τοπική, προστατευτική της ιδιωτικότητας τεχνητή νοημοσύνη. Το τρίτο κενό εντοπίζεται στη συνεργασία των φροντιστών: οι πύλες ασθενών προσφέρουν μόνο δυαδική πρόσβαση πληρεξουσίου τύπου «όλα ή τίποτα» και οι πλατφόρμες φυσικής κατάστασης δεν διαθέτουν πίνακες πολλαπλών ασθενών, αφήνοντας 53 εκατομμύρια οικογενειακούς φροντιστές χωρίς εργαλεία αποτελεσματικής εποπτείας και με 35% επιπλέον χρόνο συντονισμού, συχνά υποχρεωμένους να διαχειρίζονται χωριστές συνδέσεις ανά ασθενή ή να καταφεύγουν σε κοινόχρηστα διαπιστευτήρια που παραβιάζουν τον HIPAA.

Το τέταρτο κενό είναι ο διαχωρισμός της συνεχούς παρακολούθησης από την περιστασιακή φροντίδα: η τηλεϊατρική προσφέρει σύγχρονες διαβουλεύσεις χωρίς συνεχή δεδομένα, οι φορετές συσκευές καταγράφουν δεδομένα χωρίς κλινική ενσωμάτωση και τα EHR τεκμηριώνουν επισκέψεις χωρίς διαρκή παρακολούθηση· έτσι, το 31% των επείγουσών επισκέψεων θα μπορούσε να είχε αποφευχθεί με έγκαιρη ανίχνευση προειδοποιητικών σημείων, ενώ οι κλινικοί ιατροί στερούνται ορατότητας των διαχρονικών τάσεων, καθώς καμία

ενοποιημένη πλατφόρμα δεν γεφυρώνει τα παραγόμενα από τους ασθενείς δεδομένα με τις κλινικές ροές εργασίας. Το πέμπτο κενό αφορά την παρωχημένη τεχνολογία της εταιρικής υγειονομικής πληροφορικής: οι μεγάλες πύλες ασθενών στηρίζονται σε παλαιές στοίβες Java/J2EE της δεκαετίας του 2010, με κύκλους εκδόσεων 12-18 μηνών, κακή απόκριση σε κινητές συσκευές και αδυναμία υιοθέτησης σύγχρονων προτύπων εμπειρίας χρήστη, ενώ τα «εργαστήρια καινοτομίας» παράγουν πιλοτικά σε σύγχρονες στοίβες με περιορισμένη όμως ενσωμάτωση στα βασικά συστήματα. Το έκτο κενό, τέλος, είναι η περιορισμένη αναπαραγωγικότητα και επεκτασιμότητα: οι εμπορικές πλατφόρμες είναι κλειστού κώδικα και τα ερευνητικά πρωτότυπα σπάνια εξελίσσονται σε συστήματα έτοιμα για παραγωγή, με αποτέλεσμα κάθε νέο ερευνητικό έργο να επανεφευρίσκει τα βασικά δομικά στοιχεία και οι οργανισμοί υγειονομικής περίθαλψης να μην μπορούν να προσαρμόσουν τις λύσεις στις τοπικές τους συνθήκες, παρά την ύπαρξη διάσπαρτων βιβλιοθηκών ανοιχτού κώδικα (διακομιστές FHIR, αναλυτές HL7) που δεν συνθέτουν ολοκληρωμένη εφαρμογή παρακολούθησης ηλικιωμένων από άκρο σε άκρο.

2.5.2 Αντιστοίχιση συμβολής της διατριβής

Η παρούσα διατριβή αντιμετωπίζει άμεσα τα εντοπισμένα κενά μέσω ολοκληρωμένων αποφάσεων σχεδιασμού και υλοποίησης:

Πίνακας 2.4: Ελλείψεις → Αντιστοίχιση συμβολής της διατριβής

Κενό	Συνεισφορά της διατριβής	Αποδεικτικά στοιχεία/Επικύρωση
Ελλείψεις προσβασιμότητας	Συμμόρφωση με WCAG 2.1 AA από την αρχή: γραμματοσειρές 18-24pt, αντίθεση 7:1, πλοήγηση με πληκτρολόγιο, ορόσημα ARIA	Στόχος βαθμολογίας ελέγχου προσβασιμότητας ≥90%; ευρετική αξιολόγηση με ηλικιωμένους χαρακτήρες
AI που θέτει σε κίνδυνο την ιδιωτικότητα	Τοπική ανάπτυξη Ollama; τα PHI παραμένουν εντός του περιβάλλοντος ανάπτυξης; καμία εξωτερική κλήση API με δεδομένα υγείας	Τεκμηρίωση αρχιτεκτονικής; ανάλυση δικτυακής κίνησης που δείχνει μηδενικές εξωτερικές αιτήσεις API AI
Κενά στη συνεργασία των φροντιστών	Αρχιτεκτονική διπλού ρόλου; πίνακας ελέγχου πολλαπλών ασθενών για φροντιστές; ροή εργασίας πρόσκλησης νοικοκυριού με λεπτο	Λειτουργική επίδειξη διαχειριστή που διαχειρίζεται 5+ νοικοκυριά ασθενών. Διαγράμματα ακολουθίας UML
Διακοπτόμενη παρακολούθηση	Ενοποιημένη πλατφόρμα: συνεχής παρακολούθηση δεικτών υγείας + χειροκίνητη εισαγωγή + καθοδήγηση AI + εξαγωγή δεδομένων για κλινικούς ιατρούς	Μήτρα ιχνηλασιμότητας απαιτήσεων που επικυρώνει τις δυνατότητες συνεχούς παρακολούθησης + γέφυρας φροντίδας
Παλαιά τεχνολογικά stacks	Σύγχρονο πλήρες στοίβα: React + TypeScript + Go + PostgreSQL + Docker + Terraform; ανάπτυξη σε cloud	Ανάλυση κώδικα; αυτοματοποίηση ανάπτυξης με παροχή περιβάλλοντος σε λιγότερο από 15 λεπτά
Περιορισμένη αναπαραγωγικότητα	Πλήρης υλοποίηση ανοιχτού κώδικα; IaC με Terraform; ολοκληρωμένη τεκμηρίωση; επεκτάσιμη αρχιτεκτονική	Αποθήκη GitHub; τεκμηρίωση εγκατάστασης που επιτρέπει ανεξάρτητη ανάπτυξη; τεκμηρίωση API για επεκτάσεις

2.5.3 Διαφορά από σχετικές ακαδημαϊκές εργασίες

Ενώ αρκετές ακαδημαϊκές δημοσιεύσεις ασχολούνται με τμήματα αυτού του προβλήματος, η παρούσα διατριβή προσφέρει μοναδικές ολοκληρωμένες συνεισφορές:

Έναντι των πρωτοτύπων απομακρυσμένης παρακολούθησης (π.χ. της ανασκόπησης των Patel et al., 2012), τα περισσότερα ακαδημαϊκά συστήματα αποδεικνύουν απλώς την τεχνική σκοπιμότητα με λιγότερους από είκοσι συμμετέχοντες σε ελεγχόμενα περιβάλλοντα, ενώ η παρούσα διατριβή παραδίδει σύστημα έτοιμο για παραγωγική ανάπτυξη, με έμφαση στις λειτουργικές πραγματικότητες της αυθεντικοποίησης, της εξουσιοδότησης, της διατήρησης δεδομένων και της αυτοματοποίησης της ανάπτυξης. Έναντι των μελετών προσβασιμότητας, όπως η έρευνα της Nielsen Norman Group για τη χρηστικότητα από ηλικιωμένους, οι οποίες διατυπώνουν αρχές σχεδιασμού, η παρούσα εργασία επιδεικνύει την εφαρμοσμένη υλοποίησή τους στο πλαίσιο της υγειονομικής περίθαλψης, παρέχοντας επαναχρησιμοποιήσιμο σύστημα σχεδίασης και βιβλιοθήκες. Έναντι της έρευνας τεχνητής νοημοσύνης στην υγεία, που τυπικά χρησιμοποιεί API του cloud (GPT-4, Gemini) χωρίς μέριμνα για τα προσωπικά δεδομένα, η διατριβή πρωτοπορεί στην τοπική ανάπτυξη μεγάλου γλωσσικού μοντέλου, συνδυάζοντας τη διαφύλαξη της ιδιωτικότητας με βοήθεια προσαρμοσμένη στο πλαίσιο του ασθενούς. Έναντι των τεχνολογιών υποστήριξης φροντιστών (ανασκόπηση Chi & Demiris, 2015), που εστιάζουν κυρίως στην επικοινωνία, εδώ η επικοινωνία ενσωματώνεται με ολοκληρωμένη οπτικοποίηση δεδομένων υγείας και ροές διαχείρισης νοικοκυριού. Τέλος, έναντι των συστημάτων υγείας ανοιχτού κώδικα (OpenMRS, GNU Health), που απευθύνονται σε κλινικούς παρόχους με λειτουργικότητα EHR, η παρούσα εργασία εστιάζει στη συνεχή παρακολούθηση από ασθενείς και φροντιστές, βελτιστοποιημένη για χρήση από ηλικιωμένους.

2.5.4 Περιορισμοί του πεδίου εφαρμογής και μελλοντικές κατευθύνσεις της έρευνας

Η παρούσα διατριβή οριοθετεί σκόπιμα τα όρια, ώστε να επιτρέπει την εστίαση στις βασικές συνεισφορές:

Εκτός πεδίου εφαρμογής (προσδιορίζεται ως μελλοντική εργασία):

Εκτός πεδίου εφαρμογής - και ρητά προσδιορισμένα ως μελλοντική εργασία - παραμένουν: η ενσωμάτωση φορετών συσκευών με λήψη δεδομένων σε πραγματικό χρόνο από τα API των Fitbit και Apple Watch, που απαιτεί συνεργασίες OAuth και προσθέτει πολυπλοκότητα κάθετη προς τους βασικούς στόχους προσβασιμότητας και ιδιωτικότητας· η ενσωμάτωση με κλινικά EHR μέσω υλοποίησης διακομιστή HL7 FHIR, σημαντική τεχνική προσπάθεια που σκόπιμα έπεται της επικύρωσης των ροών εργασίας ασθενών και φροντιστών· τα προγνωστικά μοντέλα μηχανικής μάθησης για την πρόβλεψη κινδύνων υγείας, που προϋποθέτουν μεγάλα επισημειωμένα σύνολα δεδομένων - το παρόν σύστημα θέτει τη βάση για τη συλλογή τους· και οι εγγενείς εφαρμογές iOS/Android, καθώς η προσέγγιση web-first εξασφαλίζει πρόσβαση από όλες τις πλατφόρμες, με τις κινητές εφαρμογές να αποτελούν τη λογική επόμενη επανάληψη.

Προσδιορισμένα μελλοντικά ερευνητικά ερωτήματα:

Πρώτον, διαχρονικές μελέτες με ηλικιωμένους χρήστες: ποιες είναι οι μετρήσιμες επιδόσεις χρηστικότητας - ποσοστά ολοκλήρωσης εργασιών, ποσοστά σφαλμάτων - όταν ηλικιωμένοι συμμετέχοντες (N≥50, ηλικίας 65-85) χρησιμοποιούν το σύστημα επί έξι μήνες; Δεύτερον, ανάλυση της ροής εργασίας των φροντιστών: πώς επηρεάζουν ποσοτικά οι λειτουργίες διαχείρισης νοικοκυριών πολλαπλών ασθενών τον φόρτο των φροντιστών, μετρούμενο με τις κλίμακες Zarit Burden Interview και NASA-TLX; Τρίτον, αποτελεσματικότητα της τοπικής τεχνητής νοημοσύνης: ποια είναι η συγκριτική ποιότητα των αποκρίσεων των τοπικών μοντέλων Ollama έναντι του GPT-4 του cloud για ερωτήματα εκπαίδευσης σε θέματα υγείας, βάσει ανθρώπινης αξιολόγησης και μετρικών ROUGE; Και τέταρτον, η αντίληψη της ιδιωτικότητας: εκφράζουν οι ηλικιωμένοι χρήστες μεγαλύτερη εμπιστοσύνη και προθυμία να μοιραστούν δεδομένα όταν η τεχνητή νοημοσύνη είναι ρητά τοπική αντί για cloud, όπως μπορεί να διερευνηθεί με ποιοτικές συνεντεύξεις και έρευνες κλίμακας Likert;

Αυτές οι ερωτήσεις καθορίζουν το ερευνητικό πρόγραμμα που επεκτείνει την παρούσα διατριβή προς την εμπειρική επικύρωση με τους πληθυσμούς-στόχους.

Κεφάλαιο 3. Ανάλυση και σχεδιασμός

3.1 Ανάλυση απαιτήσεων

Η μηχανική απαιτήσεων αποτελεί τη βάση της συστηματικής ανάπτυξης λογισμικού, μεταφράζοντας τις ανάγκες των ενδιαφερομένων και την ανάλυση προβλημάτων σε συγκεκριμένες, δοκιμαστικές προδιαγραφές. Αυτή η ενότητα αντλεί λειτουργικές απαιτήσεις (ικανότητες που πρέπει να παρέχει το σύστημα), μη λειτουργικές απαιτήσεις (χαρακτηριστικά ποιότητας και περιορισμοί) και περιορισμούς του συστήματος από την ανάλυση του πεδίου προβλημάτων στο Κεφάλαιο 1 και την επιτόπια ανασκόπηση στο Κεφάλαιο 2.

3.1.1 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις καθορίζουν τι πρέπει να κάνει το σύστημα διακριτές δυνατότητες ορατές στους τελικούς χρήστες. Οι απαιτήσεις οργανώνονται ανά ρόλο χρήστη (ασθενής έναντι φροντιστή/διαχειριστή) και κατηγοριοποιούνται ανά λειτουργικό πεδίο.

Πίνακας 3.1: Κατάλογος λειτουργικών απαιτήσεων

A	Κατηγορία	Απαιτήση	Προτεραιότητα	Αντιστοιχισμένη εφαρμογή
FR-01	Αυθεντικοποίηση	Το σύστημα πρέπει να υποστηρίζει την εγγραφή χρηστών με όνομα χρήστη, κωδικό πρόσβασης και επιλογή ρόλου (ασθενής/διαχειριστής).	ΠΡΕΠΕΙ	CreateUser() στο authController.go
FR-02	Αυθεντικοποίηση	Το σύστημα πρέπει να πιστοποιεί τους χρήστες μέσω διαπιστευτηρίων ονόματος χρήστη/κωδικού πρόσβασης.	ΠΡΕΠ	Login() με σύγκριση κωδικού πρόσβασης bcrypt
FR-03	Αυθεντικοποίηση	Το σύστημα θα εκδίδει διακριτικά JWT που περιέχουν user_id, ρόλο, λήξη μετά από επιτυχή πιστοποίηση.	ΠΡΕΠ	Αιτήματα JWT στο Login() με λήξη 24 ωρών.
FR-04	Αυθεντικοποίηση	Το σύστημα πρέπει να επικυρώνει τα διακριτικά JWT σε προστατευμένα τελικά σημεία μέσω middleware.	ΠΡΕΠ	CheckAuth middleware στο checkAuth.go
FR-05	Αυθεντικοποίηση	Το σύστημα πρέπει να υποστηρίζει την αποσύνδεση του χρήστη ακυρώνοντας τα διακριτικά από την πλευρά του πελάτη.	ΠΡΕΠΕΙ	Τερματικό σημείο Logout() με διαγραφή cookie
FR-06	Προφίλ ασθενούς	Το σύστημα πρέπει να επιτρέπει στους ασθενείς να δημιουργούν ολοκληρωμένα προφίλ υγείας.	ΠΡΕΠΕΙ	Μοντέλο ασθενούς με δημογραφικά στοιχεία, ιατρικό ιστορικό

A	Κατηγορία	Απαίτηση	Προτεραιότητα	Αντιστοιχισμένη εφαρμογή
FR-07	Προφίλ ασθενούς	Τα προφίλ ασθενών πρέπει να περιλαμβάνουν: όνομα, επώνυμο, ημερομηνία γέννησης, φύλο, ομάδα αίματος, ύψος, διεύθυνση	ΠΡΕΠΕΙ	Πεδία δομής ασθενούς στο patient.go
FR-08	Προφίλ ασθενούς	Τα προφίλ ασθενών πρέπει να υποστηρίζουν το ιατρικό ιστορικό, τις αλλεργίες και τα τρέχοντα φάρμακα ως πεδία κειμένου	ΠΡΕΠΕΙ	Πεδία Ιατρικό ιστορικό, Αλλεργίες, Φάρμακα
FR-09	Προφίλ ασθενούς	Τα προφίλ των ασθενών πρέπει να περιλαμβάνουν επαφή έκτακτης ανάγκης με όνομα, σχέση, αριθμό τηλεφώνου.	ΠΡΕΠΕΙ	Ενσωματωμένη δομή επαφής έκτακτης ανάγκης
FR-10	Προφίλ ασθενούς	Το σύστημα πρέπει να επιτρέπει στους ασθενείς να βλέπουν και να ενημερώνουν τις πληροφορίες του προφίλ τους.	ΠΡΕΠΕΙ	Ελεγκτές GetPatientDetails(), UpdatePatient()
FR-11	Δείκτες υγείας	Το σύστημα πρέπει να καταγράφει μετρήσεις υγείας: αρτηριακή πίεση (συστολική/διαστολική), καρδιακός ρυθμός, βάρος, κορεσμός οξυγόνου.	ΠΡΕΠΕΙ	Πεδία μοντέλου HealthMetrics
FR-12	Μετρήσεις υγείας	Το σύστημα θα καταγράφει μετρήσεις δραστηριότητας: αριθμός βημάτων ανά ημέρα.	ΠΡΕΠΕΙ	Πεδίο StepsCount
FR-13	Μετρήσεις υγείας	Το σύστημα θα καταγράφει μετρήσεις ύπνου: διάρκεια, στάδια ύπνου (ελαφρύς/βαθύς/REM), χρόνος εγρήγορσης	ΠΡΕΠΕΙ	SleepStages ενσωματωμένη δομή, SleepDuration
FR-14	Μετρήσεις υγείας	Το σύστημα θα καταγράφει ενδείξεις ανωμαλιών: ακανόνιστος καρδιακός ρυθμός, ανίχνευση πτώσης	ΠΡΕΠΕΙ	Πεδία IrregularRhythm, FallDetected boolean
FR-15	Μετρήσεις υγείας	Το σύστημα θα πρέπει να χρονολογεί αυτόματα τις καταχωρήσεις των δεικτών υγείας	ΠΡΕΠΕΙ	Πεδίο ημερομηνίας με γάντζο BeforeCreate

A	Κατηγορία	Απαίτηση	Προτεραιότητα	Αντιστοιχισμένη εφαρμογή
FR-16	Μετρήσεις υγείας	Το σύστημα θα επιτρέπει την ανάκτηση των μετρήσεων υγείας των ασθενών που έχουν φιλτραριστεί με βάση τον αριθμό ταυτότητας του ασθενή	ΠΡΕΠΕΙ	Ελεγκτής GetPatientHealthMetrics()
FR-17	Δείκτες υγείας	Το σύστημα πρέπει να αποθηκεύει τα δεδομένα υγείας με σχέση ξένου κλειδιού με τον ασθενή.	ΠΡΕΠΕΙ	Ξένο κλειδί PatientID στο HealthMetrics
FR-18	Διαχείριση νοικοκυριού	Το σύστημα πρέπει να επιτρέπει στους διαχειριστές να δημιουργούν νοικοκυριά που συνδέουν πολλούς ασθενείς.	ΠΡΕΠΕΙ	Μοντέλο νοικοκυριού με AdminID και σχέση πολλών προς πολλούς ασθενών
FR-19	Διαχείριση νοικοκυριών	Το σύστημα πρέπει να υποστηρίζει τη ροή εργασιών πρόσκλησης νοικοκυριών, όπου οι διαχειριστές προσκαλούν ασθενείς.	ΠΡΕΠΕΙ	Μοντέλο πρόσκλησης με AdminID, PatientID, HouseholdID, Status
FR-20	Διαχείριση νοικοκυριού	Οι ασθενείς πρέπει να λαμβάνουν εκκρεμείς προσκλήσεις και να έχουν τη δυνατότητα να τις αποδεχθούν/απορρίψουν.	ΠΡΕΠΕΙ	Ελεγκτές GetInvitations(), RespondToInvitation()
FR-21	Διαχείριση νοικοκυριού	Το σύστημα θα επιτρέπει στους διαχειριστές να βλέπουν όλους τους ασθενείς στα νοικοκυριά τους.	ΠΡΕΠΕΙ	Ελεγκτής GetHouseholdPatients() με προφορτωμένους ασθενείς
FR-22	Διαχείριση νοικοκυριού	Το σύστημα πρέπει να παρέχει συνοπτικές πληροφορίες για την κατάσταση των ασθενών στους φροντιστές (όνομα, τελευταίες μετρήσεις)	ΠΡΕΠΕΙ	GetPatientStatuses() τελικό σημείο
FR-23	AI Chat	Το σύστημα πρέπει να παρέχει διεπαφή AI συνομιλίας για ερωτήσεις σχετικά με την υγεία.	ΠΡΕΠΕΙ	ChatController με τελικό σημείο SendMessage()
FR-24	AI Chat	Οι απαντήσεις AI θα πρέπει να ενσωματώνουν το συγκεκριμένο πλαίσιο του χρήστη: δεδομένα	ΠΡΕΠΕΙ	BuildUserContext() στο contextBuilder.go

A	Κατηγορία	Απαίτηση	Προτεραιότητα	Αντιστοιχισμένη εφαρμογή
		υγείας του ασθενούς ή δεδομένα νοικοκυριού του διαχειριστή.		
FR-25	AI Chat	Το σύστημα πρέπει να διατηρεί το ιστορικό των συνομιλιών ανά χρήστη.	ΠΡΕΠΕΙ	Μοντέλα ChatSession και ChatMessage
FR-26	AI Chat	Το σύστημα πρέπει να επιτρέπει στους χρήστες να ανακτούν το ιστορικό συνομιλιών.	ΠΡΕΠΕΙ	Μέθοδος ελεγκτή GetHistory()
FR-27	AI Chat	Το σύστημα πρέπει να επιτρέπει στους χρήστες να διαγράφουν το ιστορικό συνομιλιών.	ΠΡΕΠΕΙ	Μέθοδος ελεγκτή ClearHistory()
FR-28	AI Chat	Οι προτροπές AI πρέπει να περιλαμβάνουν ιατρικές δηλώσεις αποποίησης ευθύνης και να συνιστούν τη συμβουλή ιατρών	ΠΡΕΠΕΙ	Προτροπή συστήματος στο GenerateHealthcarePrompt()
FR-29	AI Chat	Το πλαίσιο AI του ασθενούς πρέπει να περιλαμβάνει: δημογραφικά στοιχεία, φάρμακα, αλλεργίες, τελευταία ζωτικά σημεία.	ΠΡΕΠΕΙ	Δομή PatientContextData
FR-30	AI Chat	Το πλαίσιο AI του διαχειριστή πρέπει να περιλαμβάνει: αριθμό ασθενών, σύνοψη όλων των ασθενών του νοικοκυριού με τα τελευταία μετρήσιμα στοιχεία.	ΠΡΕΠΕΙ	AdminContextData με πίνακα PatientSummary
FR-31	Προφίλ διαχειριστή	Οι χρήστες Admin θα πρέπει να μπορούν να δημιουργούν και να ενημερώνουν προσωπικά προφίλ παρόμοια με τα προφίλ των ασθενών.	ΠΡΕΠΕΙ	Ελεγκτές GetAdminProfile(), UpdateAdminProfile()
FR-32	Πρόσβαση στα δεδομένα	Το σύστημα πρέπει να επιβάλλει πρόσβαση βάσει ρόλου: οι ασθενείς έχουν πρόσβαση μόνο στα δικά τους δεδομένα.	ΠΡΕΠΕΙ	Το middleware ελέγχει ότι το user_id ταιριάζει με τον ιδιοκτήτη του πόρου
FR-33	Πρόσβαση στα δεδομένα	Το σύστημα πρέπει να επιβάλλει πρόσβαση βάσει ρόλου: οι διαχειριστές έχουν πρόσβαση στα δεδομένα	ΠΡΕΠΕΙ	Επικύρωση της ιδιότητας μέλους του νοικοκυριού στους ελεγκτές

A	Κατηγορία	Απαιτήση	Προτεραιότητα	Αντιστοιχισμένη εφαρμογή
		των ασθενών που ανήκουν στο νοικοκυριό τους.		
FR-34	Σχεδιασμός API	Το σύστημα θα παρέχει τερματικά RESTful API με μορφή αιτήματος/απάντησης JSON.	ΠΡΕΠΕΙ	Πλαίσιο Gin με σύνδεση/απόδοση JSON
FR-35	Σχεδιασμός API	Το σύστημα πρέπει να εφαρμόζει κεφαλίδες CORS που επιτρέπουν την προέλευση frontend με διαπιστευτήρια.	ΠΡΕΠΕΙ	CORS middleware στο main.go

3.1.2 Μη λειτουργικές απαιτήσεις

Οι μη λειτουργικές απαιτήσεις καθορίζουν τα χαρακτηριστικά ποιότητας, τα χαρακτηριστικά απόδοσης και τους λειτουργικούς περιορισμούς. Κάθε NFR περιλαμβάνει μετρήσιμους βασικούς δείκτες απόδοσης (KPI), όπου ισχύει.

Πίνακας 3.2: Μη λειτουργικές απαιτήσεις με KPI

ID	Κατηγορία	Απαιτήση	KPI / Μέτρηση	Μέθοδος επικύρωσης
NFR-01	Ασφάλεια	Οι κωδικοί πρόσβασης πρέπει να υποβάλλονται σε κατακερματισμό ή χρησιμοποιώντας bcrypt με ελάχιστο συντελεστή κόστους 10	Bcrypt DefaultCost $\geq$ 10	Έλεγχος κώδικα των κλήσεων GenerateFrom Password()
NFR-02	Ασφάλεια	Τα διακριτικά JWT πρέπει να λήγουν εντός 24 ωρών από την έκδοσή τους	Αξίωση λήξης διακριτικού = ώρα έκδοσης + 24 ώρες	Έλεγχος διακριτικού στη συνάρτηση Login()
NFR-03	Ασφάλεια	Όλες οι επικοινωνίες HTTP πρέπει να χρησιμοποιούν κρυπτογράφηση TLS 1.2+ στην παραγωγή	Βαθμολογία SSL Labs A ή υψηλότερη	Διαμόρφωση ανάπτυξης παραγωγής

ID	Κατηγορία	Απαιτήση	KPI / Μέτρηση	Μέθοδος επικύρωσης
NFR-04	Ασφάλεια	Τα διαπιστευτήρια της βάσης δεδομένων δεν πρέπει να είναι προγραμματισμένα. Πρέπει να φορτώνονται από μεταβλητές περιβάλλοντος.	Μηδενικά ενσωματωμένα διαπιστευτήρια	Σάρωση κώδικα, χρήση αρχείου .env
NFR-05	Απόρρητο	Η τεχνητή νοημοσύνη πρέπει να λειτουργεί τοπικά χωρίς να μεταδίδει PHI σε εξωτερικά API	Ανάλυση δικτυακής κίνησης: 0 εξωτερικές κλήσεις API τεχνητής νοημοσύνης	Σύλληψη κυκλοφορίας Wireshark/tcpdump κατά τη διάρκεια συνομιλίας AI
NFR-06	Προστασία	Η υπηρεσία Ollama AI θα λειτουργεί στο ίδιο περιβάλλον δικτύου με το backend της εφαρμογής	Δικτύωση κοντέινερ: ollama:11434 προσβάσιμο μόνο εσωτερικά	Έλεγχος δικτύου Docker Compose
NFR-07	Προσβασιμότητα	Η διεπαφή χρήστη πρέπει να συμμορφώνεται με το πρότυπο WCAG 2.1 AA όσον αφορά την αντίθεση, την πλοήγηση και τη σημασιολογία	Βαθμολογία ελέγχου προσβασιμότητας $\geq 90/100$	Lighthouse, axe DevTools αυτοματοποιημένες σαρώσεις
NFR-08	Προσβασιμότητα	Η τυπογραφία πρέπει να υποστηρίζει ελάχιστο μέγεθος γραμματοσειράς 18pt για ηλικιωμένους χρήστες	Βασική γραμματοσειρά $\geq 18pt$	Έλεγχος CSS, οπτικός έλεγχος

ID	Κατηγορία	Απαιτήση	KPI / Μέτρηση	Μέθοδος επικύρωσης
NFR-09	Προσβασιμότητα	Τα διαδραστικά στοιχεία πρέπει να έχουν ελάχιστο μέγεθος στόχου αφής 44×44 pixel	Διαστάσεις κουμπιού/συνδέσμου $\geq 44px$	Χειροκίνητη μέτρηση στο DevTools του προγράμματος περιήγησης
NFR-10	Απόδοση	Τα τελικά σημεία API πρέπει να ανταποκρίνονται με καθυστέρηση $p95 < 500\ ms$ για ερωτήματα βάσης δεδομένων	Χρόνος απόκρισης $p95 < 500ms$	Δοκιμή φόρτωσης με Apache Bench, wrk
NFR-11	Απόδοση	Η ανάκτηση μετρήσεων υγείας πρέπει να υποστηρίζει σελιδοποίηση για σύνολα δεδομένων > 100 εγγραφές	Χρόνος φόρτωσης σελίδας < 2 δευτερόλεπτα για 100 εγγραφές	Προφίλ απόδοσης
NFR-12	Επεκτασιμότητα	Η αρχιτεκτονική του συστήματος πρέπει να είναι χωρίς κατάσταση για να επιτρέψει οριζόντια κλιμάκωση	Χωρίς κατάσταση συνεδρίας από την πλευρά του διακομιστή	Αναθεώρηση αρχιτεκτονικής, JWT χωρίς κατάσταση
NFR-13	Αξιοπιστία	Η βάση δεδομένων πρέπει να χρησιμοποιεί συναλλαγές συμβατές με ACID για την ακεραιότητα των δεδομένων	Εγγυήσεις ACID του PostgreSQL	Δοκιμή απομόνωσης συναλλαγών
NFR-14	Αξιοπιστία	Οι περιορισμοί ξένων κλειδιών της βάσης δεδομένων πρέπει να αποτρέπουν την ύπαρξη	Επιβολή αναφορικής ακεραιότητας	Προσπάθεια διαγραφής ασθενούς με μετρήσεις (θα πρέπει να αποτύχει)

ID	Κατηγορία	Απαιτήση	KPI / Μέτρηση	Μέθοδος επικύρωσης
		ορφανών αρχείων υ		
NFR-15	Συντηρησιμότητα	Ο κώδικας του backend πρέπει να χρησιμοποιεί τυποποιημένα μοντέλα με GORM ORM για λειτουργίες βάσης δεδομένων	100% πρόσβαση στη βάση δεδομένων μέσω μοντέλων GORM	Έλεγχος κώδικα
NFR-16	Συντηρησιμότητα	Το frontend θα χρησιμοποιεί TypeScript για ασφάλεια τύπου με χρήση τύπου <5%.	Ενεργοποιημένη η αυστηρή λειτουργία TypeScript, οποιαδήποτε χρήση < 5%	Ανάλυση TSLint/ESLint
NFR-17	Φορητότητα	Το σύστημα θα πρέπει να είναι κοντενερικοποιημένο για ανάπτυξη σε οποιαδήποτε πλατφόρμα συμβατή με Docker	Επιτυχής ανάπτυξη σε Docker Desktop, Kubernetes, υπηρεσίες cloud container	Δοκιμές ανάπτυξης σε πολλαπλές πλατφόρμες
NFR-18	Φορητότητα	Οι μεταφορές σχήματος βάσης δεδομένων πρέπει να ελέγχονται ως προς την έκδοση και να είναι επαναλαμβανόμενες	Σενάρια μετεγκατάστασης στον κατάλογο migrate/	Αναδημιουργία σχήματος από μεταφορές
NFR-19	Χρηστικότητα	Τα μηνύματα σφάλματος πρέπει να είναι φιλικά προς τον χρήστη, σε απλή γλώσσα, και όχι τεχνικοί κωδικοί	0 ακατέργαστοι κωδικοί σφαλμάτων εμφανίζονται στους τελικούς χρήστες	Χειροκίνητη δοκιμή διεπαφής χρήστη
NFR-20	Παρατηρησιμότητα	Το backend θα καταγράφει συμβάντα πιστοποίησης, σφάλματα και	Δομημένα αρχεία καταγραφής με επίπεδα σοβαρότητας	Έλεγχος αρχείων καταγραφής

ID	Κατηγορία	Απαίτηση	KPI / Μέτρηση	Μέθοδος επικύρωσης
		κρίσιμες λειτουργίες		

3.1.3 Περιορισμοί συστήματος

Οι περιορισμοί αντιπροσωπεύουν σταθερές αποφάσεις, περιβαλλοντικούς περιορισμούς ή κανονιστικά όρια εντός των οποίων πρέπει να λειτουργεί το σύστημα.

Πίνακας 3.3: Περιορισμοί συστήματος και αιτιολόγηση

A	Τύπος περιορισμού	Περιγραφή	Λογική
CON-01	Τεχνολογία	Το backend πρέπει να χρησιμοποιεί τη γλώσσα προγραμματισμού Go (Golang)	Απαιτήσεις απόδοσης, υποστήριξη ταυτόχρονης εκτέλεσης, ανάπτυξη ενός μόνο δυαδικού αρχείου
CON-02	Τεχνολογία	Το frontend πρέπει να χρησιμοποιεί React με TypeScript	Επαναχρησιμοποίηση στοιχείων, ασφάλεια τύπου, οικοσύστημα βιβλιοθήκης στοιχείων υγειονομικής περίθαλψης
CON-03	Τεχνολογία	Η βάση δεδομένων πρέπει να είναι σχεσιακή βάση δεδομένων PostgreSQL	Συμμόρφωση με ACID, ευρετηρίαση χρονοσειρών, ευελιξία JSONB, περιορισμοί ξένων κλειδιών
CON-04	Τεχνολογία	Η τεχνητή νοημοσύνη πρέπει να χρησιμοποιεί τοπικά εγκατεστημένο Ollama με ανοιχτού κώδικα LLM	Απαίτηση απορρήτου: καμία εξωτερική μετάδοση PHI
CON-05	Εγκατάσταση	Το περιβάλλον ανάπτυξης χρησιμοποιεί Docker Compose για συντονισμό πολλαπλών κοντέινερ	Απλοποιημένη τοπική ανάπτυξη, συνεπή περιβάλλοντα
CON-06	Ανάπτυξη	Η ανάπτυξη παραγωγής στοχεύει σε Azure Container Apps ή ισοδύναμη	Επεκτασιμότητα, διαχειριζόμενη υποδομή, οικονομική αποδοτικότητα

A	Τύπος περιορισμού	Περιγραφή	Λογική
		πλατφόρμα cloud container	
CON-07	Κανονιστικές	Ο σχεδιασμός του συστήματος πρέπει να είναι σύμφωνος με τα τεχνικά μέτρα ασφαλείας του κανονισμού HIPAA (ακόμη και αν δεν καλύπτεται από τον κανονισμό)	Δημιουργία εμπιστοσύνης με τους χρήστες, προετοιμασία για κλινική υιοθέτηση
CON-08	Κανονιστικές	Ο σχεδιασμός του συστήματος πρέπει να είναι σύμφωνος με τις αρχές του GDPR (προστασία της ιδιωτικής ζωής από το σχεδιασμό, ελαχιστοποίηση των δεδομένων)	Εξυπηρέτηση διεθνών χρηστών, συμμόρφωση με τις βέλτιστες πρακτικές
CON-09	Βάση χρηστών	Κύρια δημογραφική ομάδα χρηστών: ενήλικες άνω των 60 ετών, φροντιστές οικογενειών ηλικίας 35-65 ετών	Αποφάσεις σχεδιασμού προσβασιμότητας, προτεραιοποίηση λειτουργιών
CON-10	Πεδίο	Η αρχική έκδοση δεν περιλαμβάνει αυτόματη ενσωμάτωση φορητών συσκευών (μόνο χειροκίνητη εισαγωγή δεδομένων)	Μείωση της πολυπλοκότητας, εστίαση στις βασικές λειτουργίες παρακολούθησης και τεχνητής νοημοσύ
CON-11	Πεδίο	Η αρχική έκδοση δεν περιλαμβάνει ενσωμάτωση EHR/EMR μέσω HL7 FHIR	Αναβολή της διαλειτουργικότητας σε μεταγενέστερη φάση
CON-12	Πεδίο	Η αρχική έκδοση παρέχει μόνο εφαρμογή ιστού (χωρίς εγγενείς εφαρμογές για κινητά)	Η πρόσβαση σε προγράμματα περιήγησης πολλαπλών πλατφορμών είναι επαρκής για το MVP
CON-13	Δίκτυο	Το API backend και το frontend πρέπει να υποστηρίζουν CORS για ανάπτυξη	Το frontend εξυπηρετείται από τον διακομιστή React dev (θύρα

A	Τύπος περιορισμού	Περιγραφή	Λογική
		ξεχωριστής προέλευσης	3000), το backend από τη θύρα 8080
CON-14	Όγκος δεδομένων	Σύστημα βελτιστοποιημένο για νοικοκυριά με ≤ 10 ασθενείς ανά διαχειριστή	Τυπικό σενάριο οικογενειακού φροντιστή
CON-15	Μοντέλο AI	Η επιλογή μοντέλου Ollama περιορίζεται σε μοντέλα ≤ 7B παραμέτρων για συμβατότητα με καταναλωτικό υλικό	Περιορισμοί RAM σε τυπικές μηχανές ανάπτυξης

3.1.4 Ανιχνευσιμότητα απαιτήσεων

Η ιχνηλασιμότητα απαιτήσεων διασφαλίζει ότι κάθε απαίτηση αντιστοιχεί σε στοιχεία σχεδιασμού (Ενότητα 3.2-3.5) και αντικείμενα υλοποίησης (Κεφάλαιο 4). Ακολουθούν ορισμένοι κρίσιμοι σύνδεσμοι ιχνηλασιμότητας:

Ιχνηλασιμότητα ροής πιστοποίησης:

FR-01 έως FR-05 → Διάγραμμα ακολουθίας UML 3.11 → Εφαρμογή authController.go → Στοιχεία διεπαφής χρήστη για σύνδεση/εγγραφή χρήστη.

Ροή εργασίας μετρήσεων υγείας:

FR-11 έως FR-17 → Σχήμα βάσης δεδομένων HealthMetrics (Εικόνα 3.5) → Μοντέλο HealthMetrics → Διαγράμματα οπτικοποίησης μετρήσεων.

Πλαίσιο συνομιλίας AI:

FR-23 έως FR-30 → Ροή ενσωμάτωσης υπηρεσίας AI (Εικόνα 4.3) → contextBuilder.go, chatService.go → Στοιχεία διεπαφής χρήστη chatbot AI.

Αρχιτεκτονική διατήρησης απορρήτου:

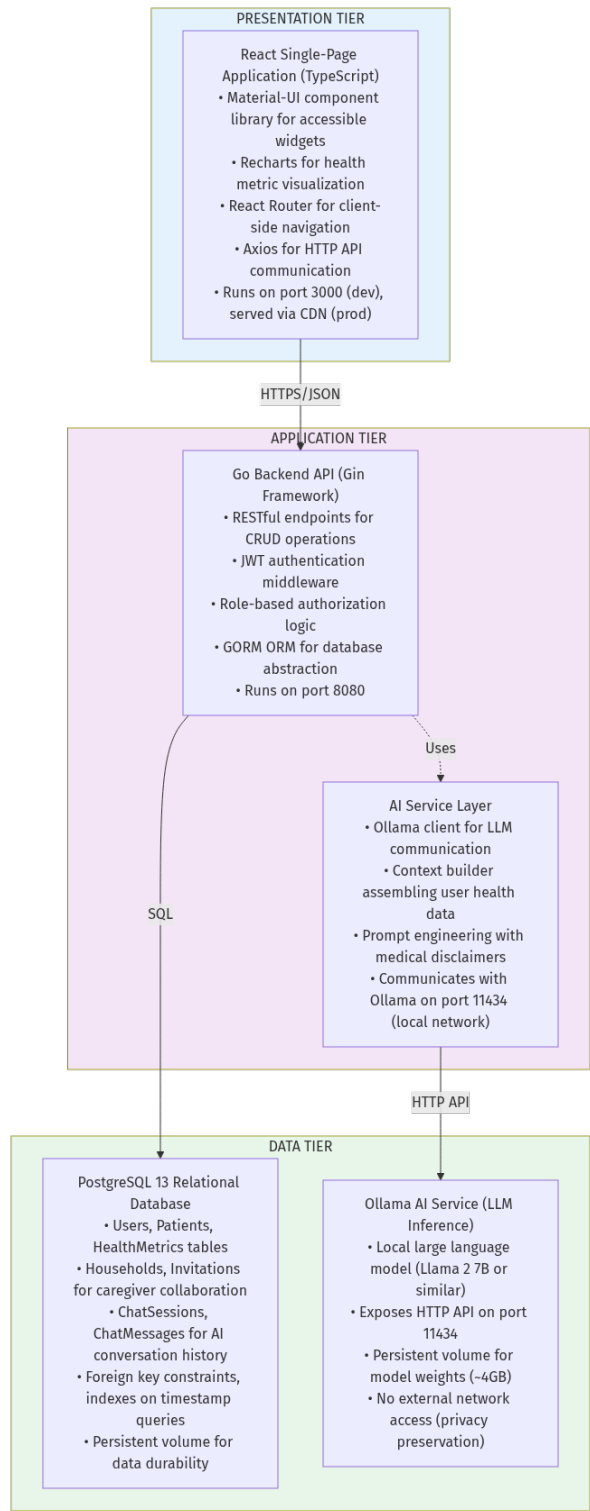
NFR-05, NFR-06, CON-04 → Τοπολογία ανάπτυξης (Εικόνα 3.2) → docker-compose.yml Ollama container → Επικύρωση τοπικής δικτυακής κίνησης.

3.2 Σχεδιασμός αρχιτεκτονικής συστήματος

3.2.1 Γενική επισκόπηση αρχιτεκτονικής

Το Elder Health Monitoring Dashboard χρησιμοποιεί μια **αρχιτεκτονική τριών επιπέδων** που διαχωρίζει τα επίπεδα παρουσίασης, επιχειρηματικής λογικής και διατήρησης δεδομένων. Αυτό το κλασικό μοτίβο επιτρέπει ανεξάρτητη κλιμάκωση, τεχνολογική εξέλιξη και σαφή διαχωρισμό των αρμοδιοτήτων.

Σχήμα 3.1: Αρχιτεκτονική συστήματος υψηλού επιπέδου



Η αρχιτεκτονική αυτή στηρίζεται σε πέντε σκεπτικά. Ο διαχωρισμός αρμοδιοτήτων αποσυνδέει τη λογική της διεπαφής σε React από την επιχειρηματική λογική σε Go και από τη διατήρηση των δεδομένων στο PostgreSQL, ώστε κάθε επίπεδο να μπορεί να εξελίσσεται ανεξάρτητα. Η επιλογή κάθε τεχνολογίας έγινε με κριτήριο το πεδίο στο οποίο υπερέχει: το React στις δυναμικές διεπαφές, η Go στις υπηρεσίες API υψηλής απόδοσης και το PostgreSQL στην ακεραιότητα των δεδομένων. Η επεκτασιμότητα εξασφαλίζεται από το επίπεδο εφαρμογής χωρίς κατάσταση, καθώς η ταυτοποίηση με JWT επιτρέπει οριζόντια κλιμάκωση πίσω από εξισορροπητή φορτίου, ενώ η βάση δεδομένων κλιμακώνεται με αντίγραφα ανάγνωσης. Η ασφάλεια οργανώνεται σε επάλληλα επίπεδα: το frontend επικυρώνει τις εισόδους, το backend επιβάλλει την εξουσιοδότηση και η βάση δεδομένων την αναφορική ακεραιότητα. Τέλος, η ιδιωτικότητα διασφαλίζεται αρχιτεκτονικά, αφού η υπηρεσία τεχνητής νοημοσύνης αναπτύσσεται στο ίδιο ιδιωτικό δίκτυο με το backend και καμία πληροφορία υγείας δεν μεταδίδεται σε εξωτερικά API.

3.2.2 Ανάλυση στοιχείων

Στοιχεία επιπέδου παρουσίασης:

Προβολές ελέγχου ταυτότητας: Διεπαφές σύνδεσης, εγγραφής, επαναφοράς κωδικού πρόσβασης. Πίνακας ελέγχου ασθενούς: Επισκόπηση δεικτών υγείας, διαχείριση προφίλ, διεπαφή συνομιλίας AI. Πίνακας ελέγχου διαχειριστή: προβολή πλέγματος πολλαπλών ασθενών, διαχείριση νοικοκυριού, σύστημα προσκλήσεων. Στοιχεία μετρήσεων υγείας: Φόρμες εισαγωγής δεδομένων, διαγράμματα χρονοσειρών (γραμμικά, ράβδων), κάρτες μετρήσεων. Στοιχείο συνομιλίας AI: Διεπαφή συνομιλίας με ιστορικό μηνυμάτων, δείκτες πληκτρολόγησης. Πλοήγηση: Στοιχεία μενού για συγκεκριμένους ρόλους, διαδρομές πλοήγησης. Κοινά στοιχεία διεπαφής

χρήστη: Κουμπιά, κάρτες, παράθυρα διαλόγου, πεδία εισαγωγής δεδομένων με στυλ προσβασιμότητας.

Στοιχεία επιπέδου εφαρμογής:

Ελεγκτές (χειριστές αιτήσεων HTTP):

authController.go: Εγγραφή χρήστη, σύνδεση, αποσύνδεση, έκδοση JWT. patientController.go: Λειτουργίες CRUD ασθενών, ανάκτηση μετρήσεων υγείας. householdController.go: Διαχείριση νοικοκυριού, ρόες εργασίας προσκλήσεων. chatController.go: Διαχείριση μηνυμάτων AI chat, ανάκτηση ιστορικού.

Μοντέλα (δομές δεδομένων):

Χρήστης: Όνομα χρήστη, κωδικός πρόσβασης, ρόλος, σχέσεις. Ασθενής: Δημογραφικά στοιχεία, ιατρικό ιστορικό, επαφές έκτακτης ανάγκης. HealthMetrics: Ζωτικά σημεία, δραστηριότητα, δεδομένα ύπνου με χρονικές σημάνσεις. Νοικοκυριό: Αναγνωριστικό διαχειριστή, συσχετίσεις ασθενών. Πρόσκληση: Παρακολούθηση κατάστασης πρόσκλησης. ChatSession, ChatMessage: Διατήρηση συνομιλιών.

Υπηρεσίες:

chatService.go: Συντονισμός αιτημάτων AI, διαχείριση συνεδριών. contextBuilder.go: Συναρμολόγηση περιβάλλοντος χρήστη από βάση δεδομένων. ollamaClient.go: Πελάτης HTTP για επικοινωνία με το API Ollama.

Μεσαίο λογισμικό:

CheckAuth: επικύρωση JWT, εισαγωγή ταυτότητας χρήστη στο πλαίσιο αιτήματος. CORS: Κεφαλίδες κοινής χρήσης πόρων μεταξύ διαφορετικών προελεύσεων.

Βοηθητικά προγράμματα:

Περιτυλίγματα κατακερματισμού/σύγκρισης κωδικών πρόσβασης. Ανάλυση και επικύρωση token JWT.

Στοιχεία επιπέδου δεδομένων:

Βάση δεδομένων PostgreSQL:

8 βασικοί πίνακες με σχέσεις ξένων κλειδιών. Ευρετήρια στις στήλες user_id, patient_id, date για βελτιστοποίηση των ερωτημάτων. Αυτόματη μετεγκατάσταση GORM για τη δημιουργία εκδόσεων σχήματος.

Υπηρεσία Ollama:

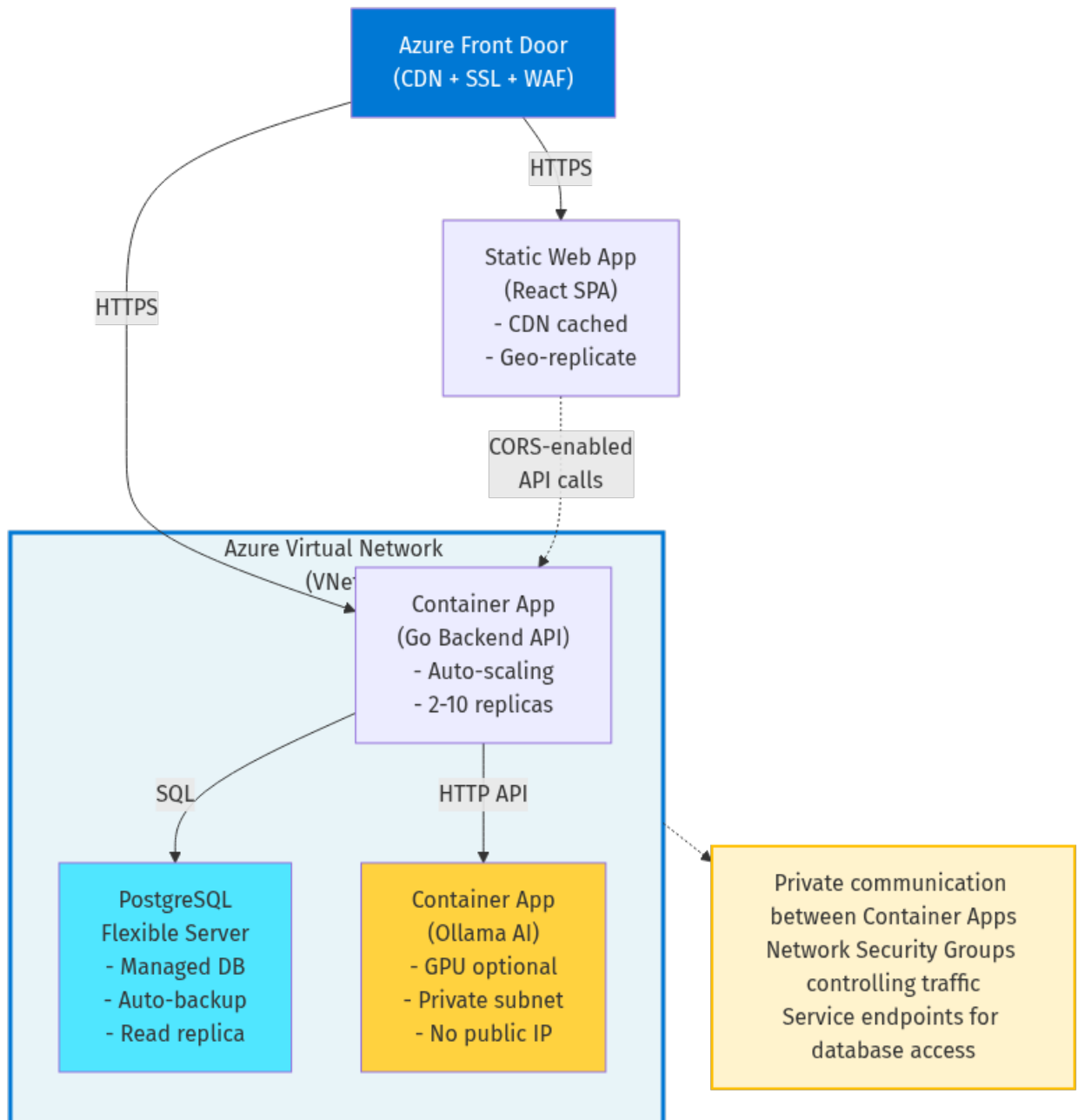
Κινητήρας συμπερασμού LLM σε κοντέινερ. Τερματικό σημείο /api/generate για δημιουργία κειμένου. Τερματικό σημείο /api/tags για ελέγχους υγείας. Όγκος αποθήκευσης μοντέλου που διατηρεί τα κατεβασμένα βάρη.

3.2.3 Τοπολογία ανάπτυξης στο cloud

Ενώ η ανάπτυξη χρησιμοποιεί το Docker Compose, η ανάπτυξη παραγωγής στοχεύει σε πλατφόρμες cloud container (Azure Container Apps, AWS ECS, Google Cloud Run). Ακολουθεί η αρχιτεκτονική ανάπτυξης Azure:

Σχήμα 3.2: Τοπολογία ανάπτυξης στο cloud Azure





Χαρακτηριστικά ανάπτυξης:

Frontend: Στατική κατασκευή React που αναπτύσσεται σε Azure Static Web Apps με παγκόσμια διανομή CDN. **Backend API:** Εφαρμογή Go σε κοντέινερ στο Azure Container Apps με οριζόντια αυτόματη κλιμάκωση (2-10 περιπτώσεις με βάση την CPU/μνήμη). **Βάση δεδομένων:** Azure Database για PostgreSQL Flexible Server με αυτόματα αντίγραφα ασφαλείας, ανάκτηση σε συγκεκριμένο χρονικό σημείο. **Υπηρεσία AI:** Ollama σε ξεχωριστή εφαρμογή Container App, με απομόνωση δικτύου σε ιδιωτικό υποδίκτυο (προσβάσιμο μόνο στο backend API). **Ασφάλεια:** Το Azure Front Door παρέχει WAF (Web Application Firewall), τερματισμό SSL/TLS, προστασία DDoS. **Δικτύωση:** Το εικονικό δίκτυο επιτρέπει την ιδιωτική επικοινωνία. Το δημόσιο διαδίκτυο δεν έχει άμεση πρόσβαση στη βάση δεδομένων ή στην υπηρεσία Ollama.

3.2.4 Επίπεδα αρχιτεκτονικής ασφάλειας

Η ασφάλεια υλοποιείται μέσω βαθιάς άμυνας σε πολλαπλά επίπεδα:

Επίπεδο 1: Ασφάλεια δικτύου

Frontend και backend σε ξεχωριστούς τομείς με εφαρμογή πολιτικής CORS. Βάση δεδομένων και υπηρεσία τεχνητής νοημοσύνης σε ιδιωτικά υποδίκτυα χωρίς δημόσιες διευθύνσεις IP. Ομάδες ασφάλειας δικτύου που περιορίζουν την κυκλοφορία στις απαραίτητες θύρες (5432 PostgreSQL, 11434 Ollama, 8080 API).

Επίπεδο 2: Αυθεντικοποίηση και εξουσιοδότηση

Κρυπτογράφηση κωδικού πρόσβασης Bcrypt με συντελεστή κόστους 10 ($2^{10} = 1024$ επαναλήψεις). Διακριτικά JWT με υπογραφή HMAC-SHA256, λήξη 24 ωρών. Το middleware επικυρώνει το token σε κάθε προστατευμένο τελικό σημείο. Έλεγχος πρόσβασης βάσει ρόλου: έλεγχος εξουσιοδότησης ρόλου ασθενούς έναντι ρόλου διαχειριστή.

Επίπεδο 3: Προστασία δεδομένων

Κρυπτογράφηση TLS 1.2+ για όλη την κυκλοφορία HTTP στην παραγωγή. Συνδέσεις βάσης δεδομένων που χρησιμοποιούν λειτουργία SSL στην παραγωγή. Μεταβλητές περιβάλλοντος για μυστικά (χωρίς προγραμματισμένα διαπιστευτήρια). Απόρρητο κλειδί JWT αποθηκευμένο με ασφάλεια (Azure Key Vault στην παραγωγή).

Επίπεδο 4: Διατήρηση απορρήτου

Η τεχνητή νοημοσύνη λειτουργεί εξ ολοκλήρου σε τοπική/ιδιωτική υποδομή δικτύου. Μηδενική μετάδοση PHI σε εξωτερικά API τεχνητής νοημοσύνης στο cloud (OpenAI, Anthropic, Google). Τα δεδομένα των συνομιλιών κρυπτογραφούνται σε κατάσταση ηρεμίας στη βάση δεδομένων PostgreSQL. Απομόνωση δεδομένων χρήστη μέσω περιορισμών ξένων κλειδιών και φιλτραρίσματος ερωτημάτων.

Επίπεδο 5: Λογική εφαρμογής

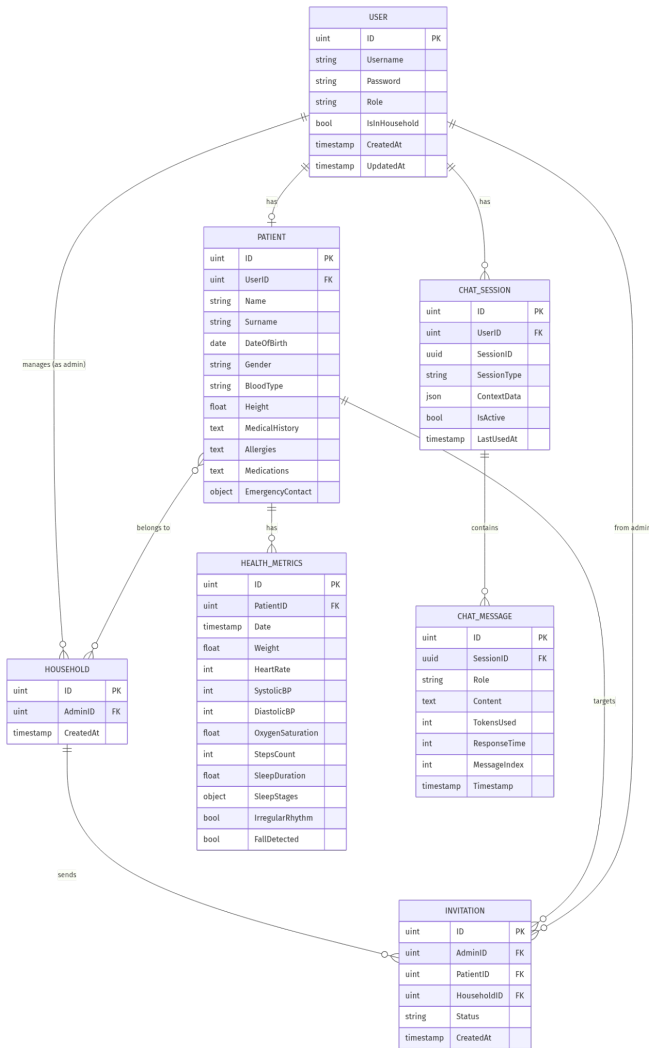
Επικύρωση εισόδου στα τελικά σημεία API (σύνδεση σχήματος JSON). Πρόληψη SQL injection μέσω παραμετροποιημένων ερωτημάτων GORM. Οι καταρράκτες ξένων κλειδιών αποτρέπουν τα ορφανά αρχεία. Μηνύματα σφάλματος που έχουν καθαριστεί για την αποφυγή αποκάλυψης πληροφοριών.

3.3 Σχεδιασμός βάσης δεδομένων

3.3.1 Μοντέλο οντοτήτων-σχέσεων

Το σχήμα της βάσης δεδομένων χρησιμοποιεί σχεσιακή μοντελοποίηση με κανονικοποιημένους πίνακες για την ελαχιστοποίηση της πλεονασμού, διατηρώντας παράλληλα την ακεραιότητα των δεδομένων. Το διάγραμμα οντότητας-σχέσης (ER) απεικονίζει οντότητες, χαρακτηριστικά και σχέσεις με βάση τις πραγματικές υλοποιήσεις του μοντέλου GORM.

Σχήμα 3.4: Εννοιολογικό διάγραμμα οντοτήτων-σχέσεων



Σχέσεις σχεσιακής βάσης δεδομένων:

Χρήστης → Ασθενής: 1:0..1 (ένας χρήστης μπορεί να έχει μηδέν ή ένα προφίλ ασθενούς· τα προφίλ ασθενών είναι προαιρετικά για τον διαχειριστή). Ασθενής → HealthMetrics: 1:N (ένας ασθενής έχει πολλές καταχωρήσεις μετρήσεων υγείας με την πάροδο του χρόνου). Χρήστης → Νοικοκυριό (ως διαχειριστής): 1:N (ένας διαχειριστής διαχειρίζεται πολλά νοικοκυριά). Νοικοκυριό → Ασθενής: M:N (τα νοικοκυριά περιλαμβάνουν πολλούς ασθενείς, οι ασθενείς μπορούν να ανήκουν σε πολλά νοικοκυριά μέσω του πίνακα σύνδεσης household_patients). Πρόσκληση: Συνδέει τον διαχειριστή, τον ασθενή και το νοικοκυριό με παρακολούθηση κατάστασης (σε αναμονή, αποδεκτή, απορριφθείσα). Χρήστης → ChatSession: 1:N (ο χρήστης μπορεί να έχει πολλές συνομιλίες στο chat με την πάροδο του χρόνου). Συνεδρία συνομιλίας → Μήνυμα συνομιλίας: 1:N (κάθε συνεδρία περιέχει πολλά μηνύματα με τη σειρά της συνομιλίας).

3.3.2 Λογικό σχήμα βάσης δεδομένων

Το λογικό σχήμα ορίζει τις δομές των πινάκων, τους τύπους δεδομένων των στηλών, τους περιορισμούς

ακεραιότητας και τα ευρετήρια, και προκύπτει απευθείας από τα μοντέλα GORM του καταλόγου backend/models/. Οι πλήρεις ορισμοί DDL όλων των πινάκων παρατίθενται στο Παράρτημα Α.49· στη συνέχεια περιγράφονται οι σχεδιαστικές επιλογές που τους διέπουν.

Ο πίνακας users αποτελεί τη ρίζα του μοντέλου ταυτοποίησης. Περιλαμβάνει το μοναδικό username, τον κατακερματισμένο με bcrypt κωδικό πρόσβασης, τη στήλη role που διακρίνει τους ασθενείς από τους διαχειριστές-φροντιστές, και τη λογική σημαία is_in_household η οποία δηλώνει αν ο χρήστης έχει ήδη ενταχθεί σε νοικοκυριό. Οι τρεις χρονικές στήλες created_at, updated_at και deleted_at προέρχονται από την ενσωματωμένη δομή gorm.Model· η τελευταία υλοποιεί τη λογική διαγραφή (soft delete), ώστε καμία εγγραφή χρήστη να μη χάνεται οριστικά και να παραμένει δυνατή η ανάκτησή της. Δύο ευρετήρια υποστηρίζουν τα συχνότερα ερωτήματα: το idx_users_username για την αναζήτηση κατά τη σύνδεση και το idx_users_role για το φιλτράρισμα ανά ρόλο.

Ο πίνακας patients συνδέεται με τον users μέσω του ξένου κλειδιού user_id, το οποίο φέρει περιορισμό μοναδικότητας: κάθε λογαριασμός αντιστοιχεί το πολύ σε ένα προφίλ ασθενούς. Πέρα από τα δημογραφικά πεδία (name, surname, address, date_of_birth, gender, blood_type, height), ο πίνακας συγκεντρώνει και τα κλινικά πεδία ελεύθερου κειμένου medical_record, medical_history, allergies και medications, τα οποία τροφοδοτούν το πλαίσιο του βοηθού τεχνητής νοημοσύνης. Τα στοιχεία επικοινωνίας έκτακτης ανάγκης δεν αποθηκεύονται σε ξεχωριστό πίνακα αλλά ως ενσωματωμένα πεδία με κοινό πρόθεμα — emergency_contact_name, emergency_contact_relationship και

emergency_contact_phone_number — μέσω της ετικέτας embedded του GORM. Η επιλογή τεκμηριώνεται από την καρδινάλια 1:1 και από το γεγονός ότι τα τρία πεδία προσπελαύνονται πάντοτε μαζί, οπότε μια πρόσθετη ζεύξη δεν θα πρόσφερε κάτι. Η διαγραφή ενός χρήστη παρασύρει και το προφίλ του, μέσω του περιορισμού ON DELETE CASCADE.

Ο πίνακας health_metrics συγκεντρώνει τα δεδομένα χρονοσειρών και είναι ο μόνος που αναμένεται να αυξάνεται απεριόριστα, γι' αυτό και χρησιμοποιεί κλειδί τύπου BIGSERIAL. Κάθε εγγραφή αποτυπώνει μία μέτρηση και περιλαμβάνει τα ζωτικά σημεία (systolic_bp, diastolic_bp, heart_rate, oxygen_saturation), το βάρος, τον αριθμό βημάτων και τη διάρκεια ύπνου. Τα στάδια ύπνου (light, deep, rem, awake_time) προέρχονται από την ενσωματωμένη δομή SleepStages, την οποία το GORM ισοπεδώνει σε στήλες του ίδιου πίνακα. Η στήλη blood_pressure διατηρεί τη μορφοποιημένη συμβολοσειρά «120/80» παράλληλα με τις δύο αριθμητικές στήλες· πρόκειται για συνειδητή αποκανονικοποίηση που εξοικονομεί υπολογισμούς κατά την απόδοση του πίνακα ελέγχου.

Το κρίσιμο ευρετήριο του σχήματος είναι το σύνθετο idx_health_metrics_patient_date επί των στηλών patient_id και date σε φθίνουσα σειρά. Εξυπηρετεί το κυρίαρχο ερώτημα «ανάκτησε τις Ν τελευταίες μετρήσεις του ασθενούς Χ», το οποίο εκτελείται σε κάθε φόρτωση του πίνακα ελέγχου και σε κάθε κλήση του βοηθού τεχνητής νοημοσύνης, και επιτρέπει στο PostgreSQL να επιστρέψει το αποτέλεσμα σαρώνοντας το ευρετήριο κατά φθίνουσα σειρά, χωρίς πρόσθετο βήμα ταξινόμησης.

Ο πίνακας invitations υλοποιεί τη ροή συγκατάθεσης με την οποία ένας φροντιστής προσκαλεί έναν ασθενή σε νοικοκυριό. Φέρει τρία ξένα κλειδιά — admin_id και patient_id προς τον users, και household_id προς τον households — και τη στήλη status, η οποία περιορίζεται στις τιμές pending, accepted και declined. Το σύνθετο ευρετήριο idx_invitations_patient_status εξυπηρετεί το ερώτημα «ποιες εκκρεμείς προσκλήσεις έχει ο ασθενής Χ», που εκτελείται σε κάθε σύνδεση ασθενούς.

Οι δύο τελευταίοι πίνακες υποστηρίζουν τον βοηθό τεχνητής νοημοσύνης. Ο chat_sessions αποθηκεύει μία εγγραφή ανά συνομιλία, με αναγνωριστικό τύπου UUID που παράγεται είτε από τη συνάρτηση gen_random_uuid() της βάσης είτε από το hook BeforeCreate του μοντέλου. Η επιλογή UUID αντί για αύξοντα ακέραιο έχει σαφές πλεονέκτημα ασφάλειας: τα αναγνωριστικά δεν είναι διαδοχικά και συνεπώς δεν είναι δυνατή η εικασία των συνεδριών άλλων χρηστών. Η στήλη context_data διατηρεί σε μορφή JSON το στιγμιότυπο του πλαισίου υγείας που χρησιμοποιήθηκε, ώστε η συνομιλία να είναι αναπαραγώγιμη εκ των υστέρων. Ο πίνακας chat_messages κρατά τα επιμέρους μηνύματα· το hook BeforeCreate του μοντέλου ChatMessage ορίζει αυτόματα τη στήλη message_index μετρώντας τα υπάρχοντα μηνύματα της συνεδρίας, εξασφαλίζοντας αυστηρή σειρά στο ιστορικό. Τα αντίστοιχα μοντέλα Go παρατίθενται στα Παραρτήματα Α.12 και Α.13.

3.3.3 Ανάλυση κανονικοποίησης

Το σχήμα συμμορφώνεται με την Τρίτη Κανονική Μορφή (3NF) για την ελαχιστοποίηση της πλεονασμού, διατηρώντας παράλληλα την απόδοση των ερωτημάτων:

1NF (Πρώτη κανονική μορφή): Όλες οι στήλες περιέχουν ατομικές τιμές (χωρίς επαναλαμβανόμενες ομάδες). Παράδειγμα: Τα στάδια ύπνου (ελαφρύς, βαθύς, REM) αποθηκεύονται ως ξεχωριστές στήλες και όχι ως πίνακας, επιτρέποντας ερωτήματα μεμονωμένων τιμών.

2NF (Δεύτερη κανονική μορφή): Όλα τα μη βασικά χαρακτηριστικά εξαρτώνται πλήρως από το πρωτεύον κλειδί. Παράδειγμα: Στο health_metrics, όλες οι μετρήσεις (βάρος, καρδιακός ρυθμός, αρτηριακή πίεση) εξαρτώνται από τη συγκεκριμένη καταχώριση μέτρησης που προσδιορίζεται από το id, όχι από το μερικό κλειδί.

3NF (Τρίτη κανονική μορφή): Δεν υπάρχουν μεταβατικές εξαρτήσεις. Παράδειγμα: Το όνομα του ασθενούς αποθηκεύεται στον πίνακα patients και δεν επαναλαμβάνεται στο health_metrics

(πρόσβαση μέσω σύνδεσης ξένου κλειδιού). Οι πληροφορίες διαχειριστή αποθηκεύονται στο users και δεν επαναλαμβάνονται στο households.

Αποδεκτές συμβιβαστικές λύσεις αποκανονικοποίησης:

Το πεδίο string blood_pressure στο health_metrics αντιγράφει το systolic_bp/diastolic_bp για ευκολία στην εμφάνιση (βελτιστοποίηση ανάγνωσης: αποφυγή συνένωσης string από την πλευρά του πελάτη). Η επαφή έκτακτης ανάγκης ενσωματώνεται στον πίνακα patients και όχι σε ξεχωριστό πίνακα emergency_contacts (μειώνει την πολυπλοκότητα της σύνδεσης για σχέση 1:1 στην οποία γίνεται πρόσβαση μαζί). Σειριοποίηση JSON context_data στο chat_sessions (ευελιξία για την εξέλιξη της δομής του πλαισίου χωρίς μετεγκαταστάσεις σχήματος).

3.3.4 Στρατηγική ευρετηρίου για δεδομένα υγείας χρονοσειρών

Οι ερωτήσεις για μετρήσεις υγείας ακολουθούν συγκεκριμένα πρότυπα πρόσβασης που απαιτούν στρατηγική ευρετηρίαση που προέρχεται από υλοποιήσεις ελεγκτών:

Πίνακας 3.4: Περίληψη στρατηγικής ευρετηρίου

Μοτίβο ερωτήματος	Ευρετήριο	Χρήση ελεγκτή	Αιτιολόγηση
Λήψη των 30 τελευταίων μετρήσεων για τον ασθενή	(patient_id, date DESC)	GetPatientHealthMetrics()	Ο σύνθετος δείκτης επιτρέπει σάρωση μόνο του δείκτη ταξινομημένη κατά φθίνουσα ημερομηνία
Εύρεση όλων των μετρήσεων σε χρονικό διάστημα για τον ασθενή	(patient_id, date DESC)	Ερωτήματα μελλοντικού χρονικού διαστήματος	Σάρωση εύρους στη στήλη ημερομηνίας με ευρετήριο
Πλαίσιο AI: τελευταία μέτρηση ανά ασθενή	(patient_id, date DESC)	buildPatientContext(), buildAdminContext()	Η υποερώτηση χρησιμοποιεί ευρετήριο για MAX(ημερομηνία) ανά ασθενή
Πίνακας ελέγχου: τελευταία μέτρηση ανά ασθενή στο νοικοκυριό	(patient_id, date DESC)	GetPatientStatuses()	Πολλαπλές σαρώσεις ευρετηρίου συγχωνεύονται μέσω patient_id
Ανίχνευση ανωμαλιών: irregular_rhythm = true	Μερικός δείκτης (μελλοντικός)	Μελλοντικές ειδοποιήσεις ανωμαλιών	Φίλτρο Boolean σε συνδυασμό με τον δείκτη patient_id

Τέσσερις τεχνικές βελτιστοποιούν τη διαχείριση των χρονοσειρών. Πρώτον, ο σύνθετος δείκτης με φθίνουσα σειρά: επειδή το PostgreSQL λαμβάνει υπόψη την κατεύθυνση σάρωσης, η δήλωση date DESC στον ορισμό του δείκτη επιτρέπει την ανάκτηση «πρώτα τα πιο πρόσφατα» χωρίς πρόσθετο βήμα ταξινόμησης. Δεύτερον, η ακρίβεια της χρονικής σήμανσης: οι χρόνοι αποθηκεύονται ως TIMESTAMP WITH TIME ZONE, υλοποιημένοι στα μοντέλα μέσω του τύπου time.Time, ώστε να αντιμετωπίζεται σωστά η περίπτωση φροντιστών που βρίσκονται σε διαφορετική ζώνη ώρας από τον ασθενή. Τρίτον, οι διαδοχικές διαγραφές: ο περιορισμός ON DELETE CASCADE στο ξένο κλειδί patient_id εξασφαλίζει ότι οι μετρήσεις υγείας διαγράφονται αυτόματα μαζί με τον ασθενή, διατηρώντας την αναφορική ακεραιότητα. Τέταρτον, οι λογικές διαγραφές: η στήλη deleted_at του GORM επιτρέπει την παρακολούθηση και την ανάκτηση διαγραμμένων εγγραφών, αποκλείοντάς τες ταυτόχρονα από τα ερωτήματα μέσω αυτόματης ρήτρας WHERE deleted_at IS NULL.

Μελλοντικές βελτιώσεις (δεν έχουν εφαρμοστεί):

Δύο ακόμη τεχνικές καταγράφονται ως μελλοντικές βελτιώσεις, χωρίς να έχουν υλοποιηθεί. Η πρώτη είναι η διαμέριση του πίνακα `health_metrics` ανά μήνα ή έτος, η οποία αποκτά νόημα μόνο για σύνολα δεδομένων εκατομμυρίων εγγραφών· για τυπική οικιακή χρήση με λιγότερους από δέκα ασθενείς ο ετήσιος όγκος δεν ξεπερνά τις τέσσερις χιλιάδες εγγραφές και η διαμέριση δεν δικαιολογείται. Η δεύτερη είναι τα ευρετήρια κάλυψης, δηλαδή η συμπερίληψη στο ευρετήριο των στηλών που ζητούνται συχνότερα — καρδιακού ρυθμού και αρτηριακής πίεσης — ώστε να αποφεύγεται η προσπάθεια του ίδιου του πίνακα. Συμπληρωματικά, για ακριβή ερωτήματα συγκέντρωσης, όπως ο μέσος καρδιακός ρυθμός ανά εβδομάδα, θα μπορούσαν να χρησιμοποιηθούν υλοποιημένες προβολές που προϋπολογίζονται και ανανεώνονται περιοδικά.

3.4 Σχεδιασμός διεπαφής χρήστη

3.4.1 Αρχές σχεδιασμού με προτεραιότητα στην προσβασιμότητα

Ο σχεδιασμός της διεπαφής χρήστη δίνει προτεραιότητα στους ηλικιωμένους χρήστες (60+) που ενδέχεται να αντιμετωπίζουν προβλήματα όρασης (πρεσβυωπία, καταρράκτης, εκφύλιση της ωχράς κηλίδας), κινητικές δυσκολίες (αρθρίτιδα, τρόμος) και γνωστικές αλλαγές (αργότερη ταχύτητα επεξεργασίας, μειωμένη λειτουργική μνήμη). Η φιλοσοφία σχεδιασμού ακολουθεί τα πρότυπα WCAG 2.1 Level AA ως βάση, υπερβαίνοντας τις απαιτήσεις όπου το επιβάλλουν οι ειδικές ανάγκες των ηλικιωμένων.

Ο σχεδιασμός οργανώνεται γύρω από τις τέσσερις αρχές του WCAG. Ως προς την αντιληπτικότητα, εφαρμόζονται χρωματικά σχήματα υψηλής αντίθεσης με αναλογία τουλάχιστον 7:1 για το κύριο κείμενο — τιμή που υπερβαίνει το όριο 4,5:1 του επιπέδου AA — καθώς και μεγάλη τυπογραφία με βασικό μέγεθος γραμματοσειράς προσαρμοσμένο σε ηλικιωμένους αναγνώστες. Ως προς τη λειτουργικότητα, η πλοήγηση σχεδιάστηκε με προτεραιότητα στο πληκτρολόγιο και ορατούς δείκτες εστίασης, ενώ οι στόχοι αφής έχουν ελάχιστη διάσταση 44×44 εικονοστοιχεία σύμφωνα με τη σύσταση του W3C. Ως προς την κατανοησιμότητα, χρησιμοποιείται απλή και συνεπής γλώσσα που αποφεύγει την ιατρική ορολογία ή την εξηγή μέσω επεξηγηματικών πλαισίων, και τα μοτίβα πλοήγησης παραμένουν σταθερά σε όλες τις σελίδες. Ως προς την ευρωστία, χρησιμοποιούνται σημασιολογικά στοιχεία HTML5 και ετικέτες ARIA, ώστε η διεπαφή να ερμηνεύεται σωστά από τα προγράμματα ανάγνωσης οθόνης.

3.4.2 Λόγοι επιλογής του στοιχείου Material-UI

Το Material-UI (MUI) v5 επιλέχθηκε ως βιβλιοθήκη στοιχείων λόγω των πλεονεκτημάτων του στον τομέα της υγειονομικής περίθαλψης:

Κριτήρια επιλογής:

Προεπιλεγμένες ρυθμίσεις προσβασιμότητας: Ενσωματωμένα χαρακτηριστικά ARIA, πλοήγηση με πληκτρολόγιο, διαχείριση εστίασης σε όλα τα διαδραστικά στοιχεία. Προσαρμογή: Το σύστημα θεμάτων (`createTheme()`) επιτρέπει την αλλαγή του μεγέθους της γραμματοσειράς, της παλέτας χρωμάτων και των αποστάσεων. Μοτίβα υγειονομικής περίθαλψης: Διάταξη με βάση κάρτες (`Card`, `CardContent`) για έλεγχο της πυκνότητας των πληροφοριών. Σύστημα αποκρίσιμου πλέγματος: Τα στοιχεία που προορίζονται για κινητά προσαρμόζονται σε tablet (768px) και επιτραπέζιους υπολογιστές (1024px+). Ωριμότητα κοινότητας: 90.000+ αστέρια GitHub, εκτενής τεκμηρίωση, παραδείγματα εφαρμογής στην υγειονομική περίθαλψη.

Βασικά στοιχεία MUI που χρησιμοποιούνται:

Στοιχείο	Χρήση στην εφαρμογή	Λειτουργία προσβασιμότητας
<Typography variant="h1-h6">	Απόδοση όλου του κειμένου με σημασιολογική ιεραρχία	Αντιστοιχίζει παραλλαγές σε ετικέτες <h1>-<h6> για προγράμματα ανάγνωσης οθόνης
<Button size="large">	Κύριες ενέργειες, πλοήγηση	Ενσωματωμένη ανατροφοδότηση κυματισμού, ελάχιστο ύψος 44px, υποστήριξη aria-label
<Card> / <CardContent>	Εμφάνιση μετρήσεων υγείας, κάρτες περιλήψης ασθενών	role="region" ορόσημο με aria-label
<TextField>	Εισαγωγές φόρμας (BP, βάρος κ.λπ.)	Ενσωματωμένο <label>, βοηθητικό κείμενο, μηνύματα σφάλματος
<Dialog> / <Modal>	Επικαλύψεις εισαγωγής μετρήσεων υγείας	Εστίαση, απόρριψη με το πλήκτρο ESC, χειρισμός κλικ στο φόντο
<Grid container spacing={3}>	Ευέλικτες διατάξεις πίνακα ελέγχου	Βασισμένες σε Flexbox, αναδιατάσσονται από επιφάνεια εργασίας 12 στηλών σε κινητό 1 στήλης
<IconButton aria-label="...">	Ενέργειες γραμμής εργαλείων	Tooltip + aria-label για προγράμματα ανάγνωσης οθόνης, στόχος αφής 48px
<LinearProgress>	Κατάσταση φόρτωσης απόκρισης AI	role="progressbar" με ανακοινώσεις aria-valuenow

3.4.3 Κλίμακα τυπογραφίας και αναγνωσιμότητα

Τυπογραφικά στοιχεία σχεδιασμού (θέμα Material-UI):

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.3.

Βελτιώσεις αναγνωσιμότητας:

Ύψος γραμμής: 1,6× μέγεθος γραμματοσειράς (28,8px για κείμενο 18pt) βελτιώνει την αναγνωσιμότητα για χρήστες με δυσλεξία. Διάστημα μεταξύ γραμμών: Κανονικό έως +0,5px (βοηθά στη διάκριση των χαρακτήρων για τους πάσχοντες από καταρράκτη). Επιλογή γραμματοσειράς: Inter sans-serif επιλεγμένη για υψηλό ύψος x (μεγαλύτερα μικρά γράμματα) και σαφή διάκριση αριθμών (1 έναντι l έναντι I).

Αναλογίες αντίθεσης (υπερβαίνουν τα πρότυπα WCAG AA):

Στοιχείο	Φόντο	Πρώτο πλάνο	Αναλογία αντίθεσης	Πρότυπο
Κύριο κείμενο	#FFFFFF	#1A1A1A	19,6:1	WCAG AAA (7:1)
Κύριο κουμπί	#1976D2	#FFFFFF	8,6:1	WCAG AAA
Κείμενο σφάλματος	#FFFFFF	#D32F2F	7,2:1	WCAG AAA
Δευτερεύον κείμενο	#FFFFFF	#666666	5,9:1	WCAG AA

3.4.4 Παλέτα χρωμάτων και οπτική ιεραρχία

Κύριο σύστημα χρωμάτων:

Κύριο (μπλε): #1976D2 Εμπιστοσύνη, ηρεμία, συνήθως συνδέεται με την υγειονομική περίθαλψη. Δευτερεύον (Πράσινο): #388E3C Θετικοί δείκτες υγείας, ζωτικά σημεία σε φυσιολογικά επίπεδα. Σφάλμα (κόκκινο): #D32F2F Ειδοποιήσεις, κρίσιμες τιμές (BP >140/90),

προειδοποιήσεις. Προειδοποίηση (πορτοκαλί): #F57C00 Όρια προσοχής (οριακή αρτηριακή πίεση 130-140). Πληροφορίες (Ανοιχτό μπλε): #0288D1 Ενημερωτικά μηνύματα, εκπαιδευτικό περιεχόμενο. Επιτυχία (Πράσινο): #2E7D32 Επιβεβαιώσεις, ολοκληρωμένες ενέργειες, αποθήκευση μετρήσεων.

Επιλογή θέματος υψηλής αντίθεσης (μελλοντική βελτίωση):

Φόντο: #000000 (απόλυτο μαύρο). Κείμενο: #FFFFFF (καθαρό λευκό). Έμφαση: Χρώματα υψηλής κορεσμού για αντίθεση 7:1+. Εναλλαγή στον πίνακα ρυθμίσεων προσβασιμότητας.

3.4.5 Μέγεθος στόχου αφής και κινητική προσβασιμότητα

Οι ηλικιωμένοι χρήστες με αρθρίτιδα ή ιδιοπαθή τρόπο χρειάζονται μεγαλύτερους στόχους αφής/κλικ από τα ελάχιστα 44×44 pixel του W3C.

Πρότυπα μεγέθους στόχου (εφαρμοσμένα):

Τύπος στοιχείου	Ελάχιστο μέγεθος	Διάστημα	Ρύθμιση Material-UI	Αιτιολόγηση
Κύρια κουμπιά	60×60px	Περιθώριο 12px	size="large" + προσαρμοσμένη απόσταση	Μειώνει τα λάθη κλικ για χρήστες με τρόπο
Κουμπιά εικονιδίων	52×52px	8px περιθώριο	size="large"	Ισορροπία μεταξύ μεγέθους και πυκνότητας UI
Εισαγωγές φόρμας	Ύψος 56px	16px κάθετο	variant="outlined"	Άνετη περιοχή πατήματος, ορατές ετικέτες
Πλαίσια επιλογής/Ραδιόφωνο	32×32px οπτικά, 48×48px περιοχή πατήματος	Επέκταση CSS padding	Μεγαλύτερη αόρατη περιοχή κλικ μέσω padding	
Σύνδεσμοι (ενσωματωμένο κείμενο)	Πλήρες ύψος γραμμής 44px min	Αύξηση ύψους γραμμής	Αποτρέπει τυχαία κλικ σε γειτονικούς συνδέσμους	

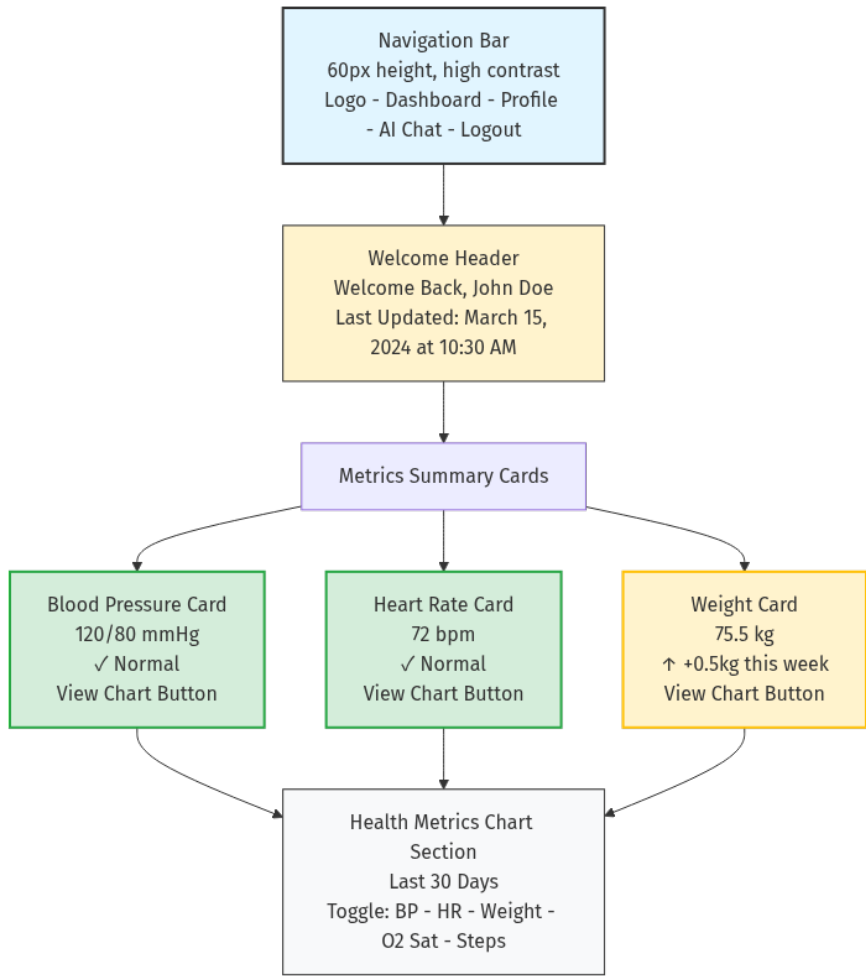
Παράδειγμα εφαρμογής (PatientDashboard.tsx):

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.4.

3.4.6 Μοτίβα διάταξης και ιεραρχία πληροφοριών

Αρχές διάταξης πίνακα ελέγχου:

Πίνακας ελέγχου ασθενούς (απλοποιημένη προβολή ενός ασθενούς):



**Μοτίβα
πλοήγησης:**

Διαδρομές πλοήγησης: "Αρχική σελίδα > Ασθενείς > John Doe > Μετρήσεις υγείας" (βοηθά τους ηλικιωμένους χρήστες να εντοπίζουν τη θέση τους). Μόνιμη γραμμή πλοήγησης: Πάντα ορατή στην επιφάνεια εργασίας (χωρίς μενού που κρύβουν την πλοήγηση). Σαφή κουμπιά επιστροφής: Σύνδεσμοι "< Επιστροφή στον πίνακα ελέγχου" (μην βασίζεστε στο κουμπί επιστροφής του προγράμματος περιήγησης).

**3.4.7 Σχεδιασμός
φορμών και**

επικύρωση εισόδου

Οι φόρμες, όπως υλοποιούνται στο PatientEditPage.tsx, ακολουθούν πέντε πρακτικές. Οι ετικέτες τοποθετούνται σαφώς πάνω από τα πεδία εισαγωγής και όχι ως πλωτές ετικέτες, ώστε να μειώνεται το γνωστικό φορτίο και να μην εξαφανίζεται το κείμενο κατά την πληκτρολόγηση. Κάτω από κάθε πεδίο εμφανίζεται βοηθητικό κείμενο με παράδειγμα μορφής, όπως «Μορφή: 120/80 mmHg». Η επικύρωση είναι ενσωματωμένη και δίνει ανατροφοδότηση σε πραγματικό χρόνο καθώς ο χρήστης πληκτρολογεί, αντί μόνο κατά την υποβολή. Η πρόληψη σφαλμάτων επιτυγχάνεται με την απενεργοποίηση του κουμπιού υποβολής έως ότου συμπληρωθούν όλα τα υποχρεωτικά πεδία. Τέλος, οι καταστροφικές ενέργειες, όπως η διαγραφή προφίλ ή η εκκαθάριση του ιστορικού συνομιλιών, προστατεύονται από διαλόγους επιβεβαίωσης.

Παράδειγμα: Φόρμα εισαγωγής αρτηριακής πίεσης (Μοντέλο μετρήσεων υγείας)

```
margin="normal"
inputProps={{ min: 70, max: 250, style: { fontSize: '1.125rem' } }}
/>
<TextField
  label="Διαστολική (χαμηλότερος αριθμός)"
  τύπος="αριθμός"
  value={διαστολική}
```

```
onChange={(e) => setDiastolic(e.target.value)}  
helperText="Συνήθως 60-90 mmHg για ενήλικες"  
error={διαστολική < 40 || διαστολική > 150}  
πλήρες πλάτος  
margin="normal"  
inputProps={{ min: 40, max: 150, style: { fontSize: '1.125rem' } }}  
/>  
{συστολική <= διαστολική && (  
  <Alert severity="error">  
    Η συστολική τιμή πρέπει να είναι υψηλότερη από τη διαστολική.  
Ελέγξτε την καταχώρησή σας.  
  </Alert>  
)}</DialogContent>  
<DialogActions>  
  <Button onClick={handleClose} size="large">Ακύρωση</Button>  
  <Button  
    onClick={handleSave}  
    variant="contained"  
    size="large"  
    disabled={!isValid}  
  >  
    Αποθήκευση ανάγνωσης  
  </Button>  
</DialogActions>  
</Διάλογος>
```

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.5.

3.5 Διαγράμματα UML και μοντελοποίηση συμπεριφοράς

Τα διαγράμματα Unified Modeling Language (UML) παρέχουν τυπικές προδιαγραφές της δομής και της συμπεριφοράς του συστήματος που προέρχονται από πραγματικά αντικείμενα υλοποίησης.

3.5.1 Διάγραμμα περιπτώσεων χρήσης

Σχήμα 3.9: Διάγραμμα περιπτώσεων χρήσης συστήματος

Σύστημα παρακολούθησης της υγείας ηλικιωμένων



Περιγραφές των εμπλεκόμενων:

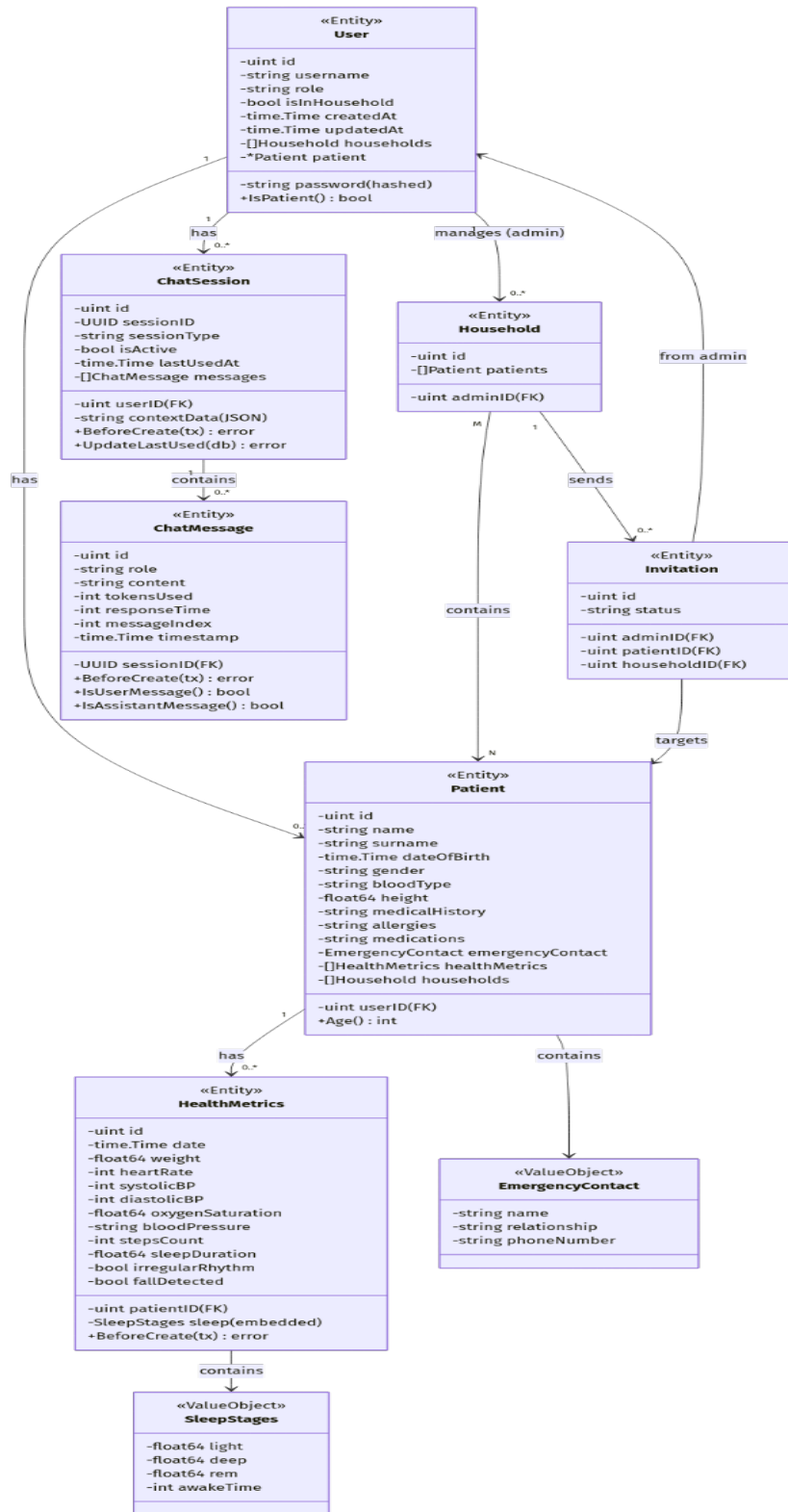
Ασθενής: Ηλικιωμένο άτομο που παρακολουθεί μόνο του την υγεία του, ενδεχομένως αρχάριος χρήστης τεχνολογίας. Διαχειριστής/Φροντιστής: Μέλος της οικογένειας ή επαγγελματίας φροντιστής που διαχειρίζεται τα δεδομένα υγείας πολλών ασθενών. Σύστημα:

Αυτοματοποιημένες διαδικασίες (αυθεντικοποίηση, τεχνητή νοημοσύνη, λειτουργίες βάσης δεδομένων) χωρίς ανθρώπινη παρέμβαση.

3.5.2 Διάγραμμα κλάσεων (βασικό μοντέλο τομέα)

Με βάση τα μοντέλα GORM στο `/backend/models/`, το παρακάτω διάγραμμα κλάσεων απεικονίζει τις σχέσεις οντοτήτων και τις βασικές μεθόδους:

Σχήμα 3.10: Διάγραμμα κλάσεων τομέα



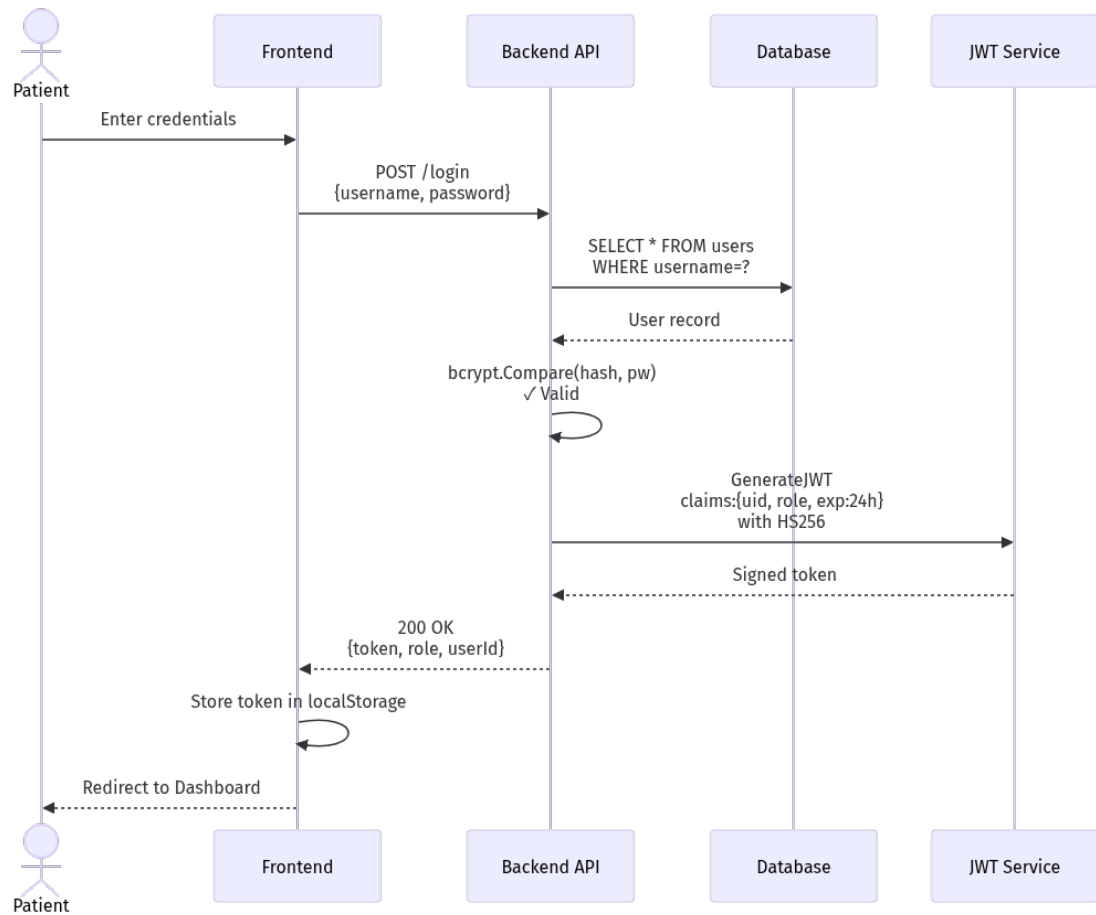
Σημειώσεις διαγράμματος κλάσης:

<<Entity>> τα στερεότυπα υποδηλώνουν αντικείμενα τομέα με μοναδική ταυτότητα βάσης δεδομένων. <<ValueObject>> (EmergencyContact, SleepStages) δεν έχουν ταυτότητα ανεξάρτητη από την οντότητα στην οποία ανήκουν· ενσωματωμένα στον γονικό πίνακα. Πολλαπλότητες: 1 (ακριβώς ένα), 0..1 (προαιρετικό), 0..* (μηδέν ή περισσότερα), M/N (πολλά προς πολλά). Μέθοδοι που εμφανίζονται μόνο για σημαντική λογική τομέα (GORM hooks, υπολογισμένες ιδιότητες). CRUD getters/setters παραλείπονται για λόγους σαφήνειας.

3.5.3 Διαγράμματα ακολουθίας

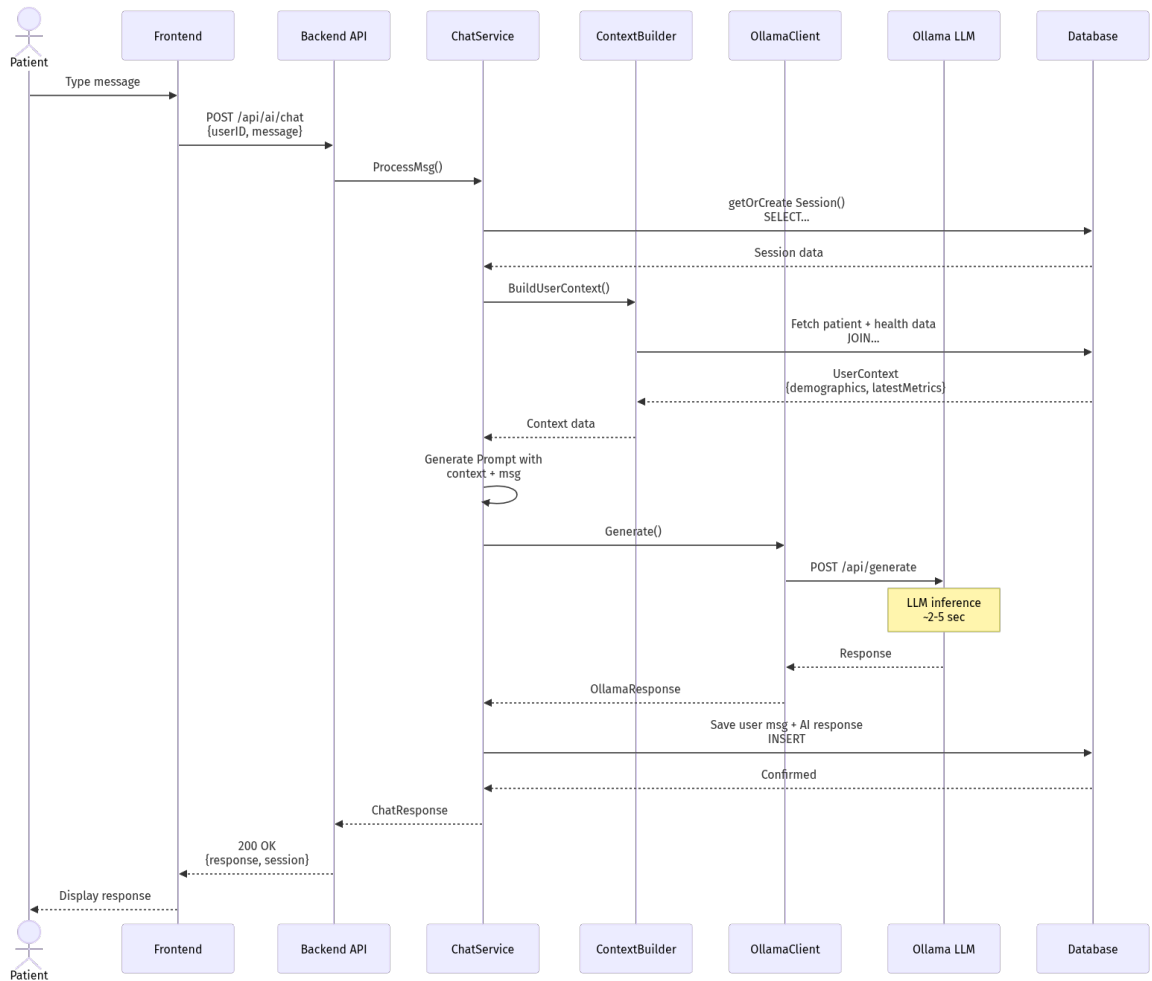
Σχήμα 3.11: Διάγραμμα ακολουθίας πιστοποίησης (ροή σύνδεσης)

Προέρχεται από τη συνάρτηση authController.go Login() και τη ροή middleware:



Σχήμα 3.12: Ακολουθία ερωτήσεων AI για την υγεία (ροή εργασίας συνομιλίας)

Πρόέρχεται από chatController.go SendMessage() και chatService.go ProcessMessage():



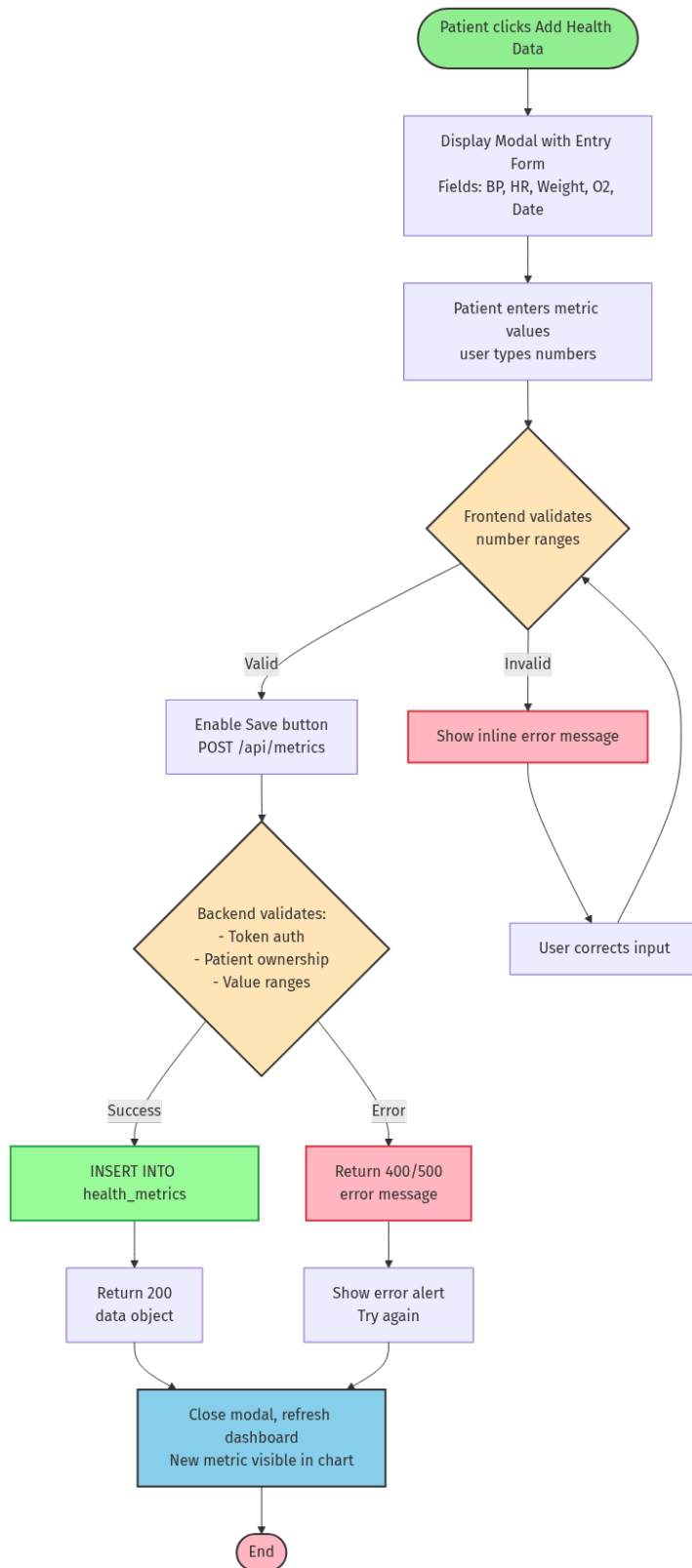
Σημαντικές πληροφορίες για την ακολουθία πλήκτρων:

Συγχρονισμένη αλυσίδα: Το frontend περιμένει την πλήρη απόκριση της τεχνητής νοημοσύνης (δεν υπάρχει streaming στην τρέχουσα έκδοση). Προσαρμογή στο πλαίσιο: Η προτροπή AI περιλαμβάνει την τελευταία αρτηριακή πίεση του ασθενούς, τα φάρμακα και τις αλλεργίες από τη βάση δεδομένων. Τοπική συμπερασματολογία: Το κοντέινερ Ollama καλείται μέσω εσωτερικού `http://ollama:11434` (δίκτυο Docker). Επιμονή: Όλα τα μηνύματα των χρηστών και των βοηθών αποθηκεύονται στον πίνακα `chat_messages` για ιστορικό.

3.5.4 Διάγραμμα δραστηριότητας: Ροή εργασίας καταχώρισης μετρήσεων υγείας

Με βάση τη ροή εισαγωγής μετρήσεων του PatientDashboard:

Σχήμα 3.14: Διάγραμμα δραστηριότητας εισαγωγής μετρήσεων υγείας



Τέλος του Κεφαλαίου 3: Ανάλυση και Σχεδιασμός

Περίληψη κεφαλαίου:

Το Κεφάλαιο 3 θέτει τις ολοκληρωμένες αρχιτεκτονικές και σχεδιαστικές βάσεις για το Πίνακα Παρακολούθησης Υγείας Ηλικιωμένων μέσω αυστηρής μηχανικής απαιτήσεων, προδιαγραφών αρχιτεκτονικής

συστήματος, μοντελοποίησης βάσης δεδομένων, σχεδιασμού διεπαφής χρήστη με προτεραιότητα στην προσβασιμότητα και τεκμηρίωση μέσω UML.

Κεφάλαιο 4: Υλοποίηση της εφαρμογής

Το κεφάλαιο αυτό τεκμηριώνει την υλοποίηση του συστήματος: το περιβάλλον ανάπτυξης, τη δομή του έργου, τις υπηρεσίες backend, τη διεπαφή frontend και την αυτοματοποίηση της υποδομής. Η παρουσίαση εστιάζει στις σχεδιαστικές αποφάσεις και στη λογική κάθε υποσυστήματος· τα πλήρη αποσπάσματα πηγαίου κώδικα παρατίθενται στο Παράρτημα Α, στο οποίο παραπέμπει κάθε ενότητα.

4.1 Περιβάλλον ανάπτυξης και εργαλεία

4.1.1 Επισκόπηση τεχνολογικού stack

Το σύστημα αξιοποιεί ένα προσεκτικά επιλεγμένο τεχνολογικό stack που έχει βελτιστοποιηθεί για απόδοση, παραγωγικότητα προγραμματιστών, ασφάλεια και συμμόρφωση με τους κανονισμούς υγειονομικής περίθαλψης. Ο πίνακας 4.1 συνοψίζει την πλήρη αλυσίδα εργαλείων με προδιαγραφές έκδοσης και αιτιολογήσεις.

Πίνακας 4.1: Περίληψη αλυσίδας εργαλείων ανάπτυξης

Συστατικό	Τεχνολογία	Έκδοση	Αιτιολόγηση
Γλώσσα backend	Go (Golang)	1.21	Συμπληρωμένη γλώσσα με ισχυρή ασφάλεια τύπων, εξαιρετικές πρωτότυπες συναρτήσεις ταυτόχρονης εκτέλεσης (goroutines), ελάχιστη χρήση μνήμης, ενσωματωμένος διακομιστής HTTP, ιδανική για μικρουπηρεσίες υγειονομικής περίθαλψης που απαιτούν υψηλή αξιοπιστία
Πλαίσιο backend	Gin	1.9	Ο γρηγορότερος δρομολογητής Go HTTP (40 φορές γρηγορότερος από το net/http), υποστήριξη middleware για CORS/auth, επικύρωση JSON, διαχείριση σφαλμάτων σε επίπεδο παραγωγής

Συστατικό	Τεχνολογία	Έκδοση	Αιτιολόγηση
Βάση δεδομένων	PostgreSQL	15.x	Συμμόρφωση με ACID για την ακεραιότητα των δεδομένων υγειονομικής περίθαλψης, ισχυρή υποστήριξη JSON/JSONB για ευέλικτες μετρήσεις υγείας, ώριμο οικοσύστημα, διαθέσιμες εφαρμογές συμβατές με HIPAA
ORM	GORM	1.25	Κορυφαίο Go ORM με αυτόματη μετεγκατάσταση, προφόρτωση σχέσεων, υποστήριξη συναλλαγών, άγκιστρα για επικύρωση δεδομένων, εξαιρετικό πρόγραμμα οδήγησης PostgreSQL
Πλαίσιο frontend	React	18.2	Αρχιτεκτονική βασισμένη σε στοιχεία που επιτρέπει επαναχρησιμοποίηση στοιχείων UI βελτιστοποιημένα για προσβασιμότητα, μεγάλο οικοσύστημα, εξαιρετική ενσωμάτωση TypeScript
Γλώσσα frontend	TypeScript	5.0	Η στατική πληκτρολόγηση αποτρέπει σφάλματα εκτέλεσης σε UI υγειονομικής περίθαλψης, βελτιωμένη υποστήριξη IDE, αυτοτεκμηριωμένες διεπαφές για συμβάσεις API

Συστατικό	Τεχνολογία	Έκδοση	Αιτιολόγηση
Βιβλιοθήκη στοιχείων διεπαφής χρήστη	Material-UI (MUI)	5.14	Συμμόρφωση με WCAG 2.1 AA από την αρχή, προσβάσιμα σχεδιαστικά στοιχεία, ανταποκρινόμενο σύστημα πλέγματος, θεματολογία επιχειρησιακού επιπέδου, εξαιρετικά πρότυπα για ηλικιωμένους χρήστες
Διαγράμματα	Recharts	2.8	Διαγράμματα React-native, προσβάσιμη απόδοση SVG, ανταποκρινόμενο, υποστηρίζει οπτικοποίηση δεδομένων υγείας χρονοσειρών
Αυθεντικοποίηση	JWT (golang-jwt)	v4	Αυθεντικοποίηση χωρίς κατάσταση που επιτρέπει οριζόντια κλιμάκωση, πρότυπο token standard bearer, λήξη 24 ωρών για ισορροπία ασφάλειας/ευκολίας
Κρυπτογράφηση κωδικού πρόσβασης	bcrypt (golang.org/x/crypto)	Τελευταία	Προσαρμοστικός κατακερματισμός με ρυθμιζόμενο κόστος, συνιστάται από το OWASP για την αποθήκευση κωδικών πρόσβασης, ανθεκτικός σε επιθέσεις rainbow table
Υπηρεσία AI	Ollama	0.1	Τοπική ανάπτυξη LLM (διατήρηση απορρήτου), υποστηρίζει μοντέλα Llama2/Mistral, ενσωμάτωση REST API, χωρίς εξαρτήσεις από το

Συστατικό	Τεχνολογία	Έκδοση	Αιτιολόγηση
			cloud για προστασία PHI
Χρόνος εκτέλεσης κοντέινερ	Docker	24.0	Απομόνωση εφαρμογών, αναπαραγώγιμες κατασκευές, απλοποιημένη ισοτιμία ανάπτυξης/παραγωγής, έτοιμο για εντοχή/αποστολή
Ορχήστρωση κοντέινερ	Docker Compose	2.20	Τοπική ανάπτυξη πολλαπλών κοντέινερ (backend + frontend + postgres + ollama), εκκίνηση στοίβας με μία εντολή
Εργαλείο IaC	Terraform	1.5	Infrastructure-as-code για ανάπτυξη Azure, δηλωτική διαχείριση πόρων, διαχείριση κατάστασης, υποστηρίζει ροές εργασίας gitops
Πλατφόρμα cloud	Azure	Δεν ισχύει	Εφαρμογές κοντέινερ για φιλοξενία κοντέινερ χωρίς διακομιστή, PostgreSQL Flexible Server (διαχειριζόμενη βάση δεδομένων), μητρώο κοντέινερ, ανάλυση αρχείων καταγραφής
Έλεγχος εκδόσεων	Git	2.40	Κατανεμημένο VCS, διακλάδωση για ανάπτυξη λειτουργιών, ενσωμάτωση GitHub/GitLab
Διαχειριστές πακέτων	Μονάδες Go, Yarn	Δεν ισχύει	Διαχείριση εξαρτήσεων, αρχεία κλειδώματος για αναπαραγώγιμες κατασκευές

4.1.2 Δομή έργου

Η αρχιτεκτονική μονοτερο οργανώνει τις βάσεις κώδικα backend και frontend σε ένα ενιαίο αποθετήριο Git, επιτρέποντας ατομικές αλλαγές σε όλα τα επίπεδα και απλοποιημένη CI/CD.

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.6.

Τρεις αρχιτεκτονικές αποφάσεις διέπουν αυτή τη δομή. Ο διαχωρισμός αρμοδιοτήτων αναθέτει στους ελεγκτές τη διαχείριση της εισόδου και εξόδου HTTP, στις υπηρεσίες την επιχειρηματική λογική και στα μοντέλα τον ορισμό των δομών δεδομένων, κάτι που καθιστά δυνατό τον έλεγχο μονάδας σε κάθε επίπεδο χωριστά. Η υπηρεσία τεχνητής νοημοσύνης είναι ενθυλακωμένη στο πακέτο `services/ai/`, το οποίο συγκεντρώνει όλες τις αλληλεπιδράσεις με το γλωσσικό μοντέλο και απομονώνει τις εξαρτήσεις από το Ollama από τη λογική του τομέα της υγείας. Τέλος, η οργάνωση σε ενιαίο αποθετήριο επιτρέπει ατομικές υποβολές για αλλαγές που αφορούν ταυτόχρονα το συμβόλαιο του API στο backend και την αντίστοιχη κλήση στο frontend, απλοποιεί τη διαδικασία CI/CD και ενοποιεί την έκδοση.

4.1.3 Διαμόρφωση περιβάλλοντος

Η εφαρμογή χρησιμοποιεί μεταβλητές περιβάλλοντος για τη διαχείριση μυστικών και τη διαμόρφωση συγκεκριμένου περιβάλλοντος. Η ανάπτυξη χρησιμοποιεί αρχεία `.env` (gitignored), η παραγωγή εισάγει μεταβλητές μέσω μυστικών Azure Container Apps.

Παράδειγμα backend .env:

```
# Διαμόρφωση βάσης δεδομένων
DB_HOST=postgres # Όνομα υπηρεσίας Docker Compose (dev) ή Azure FQDN (prod)
DB_USER=postgres
DB_PASSWORD=securepassword123
DB_NAME=myhealthdb
DB_PORT=5432

# Αυθεντικοποίηση
SECRET=your-jwt-secret-key-min-32-chars # Κλειδί υπογραφής HS256 για διακριτικά JWT

# Υπηρεσία AI
OLLAMA_URL=http://ollama:11434 # URL υπηρεσίας Ollama (δίκτυο Docker ή localhost)

# Διακομιστής
PORT=8080 # Θύρα ακρόασης backend
GIN_MODE=release # Λειτουργία πλαισίου Gin (debug|release)
```

Παράδειγμα frontend .env:

```
REACT_APP_API_URL=http://localhost:8080 # Βασική διεύθυνση URL API backend
PORT=3000 # Θύρα διακομιστή ανάπτυξης
```

Σημειώσεις ασφαλείας:

Τα αρχεία `.env` εξαιρούνται από το Git μέσω του `.gitignore` για την αποφυγή έκθεσης μυστικών. Τα μυστικά παραγωγής διαχειρίζονται μέσω αναφορών Azure Key Vault σε εφαρμογές κοντέινερ. Μυστικό JWT τουλάχιστον 32 χαρακτήρες (εντροπία 256 bit) για ασφάλεια HS256. Κωδικοί πρόσβασης βάσης δεδομένων που δημιουργούνται αυτόματα μέσω του πόρου Terraform `random_password`.

4.1.4 Ροή εργασίας τοπικής ανάπτυξης

Προαπαιτούμενα:

Docker Desktop (macOS/Windows) ή Docker Engine + Docker Compose (Linux). Go 1.21+ (για τοπική ανάπτυξη backend εκτός κοντέινερ). Node.js 18+ και Yarn (για ανάπτυξη frontend). Git.

Η εγκατάσταση ολοκληρώνεται σε πέντε βήματα. Αρχικά κλωνοποιείται το αποθετήριο με git clone και ο χρήστης μεταβαίνει στον κατάλογο my-health. Στη συνέχεια διαμορφώνεται το περιβάλλον: τα αρχεία backend/.env.example και frontend/.env.example αντιγράφονται στα αντίστοιχα .env και συμπληρώνονται με τα διαπιστευτήρια της βάσης δεδομένων, το μυστικό JWT και τη διεύθυνση REACT_APP_API_URL.

Ακολουθεί η εκκίνηση ολόκληρης της στοίβας με την εντολή docker-compose up --build, η οποία σηκώνει τέσσερις υπηρεσίες: τη βάση PostgreSQL 15 στη θύρα 5432, την υπηρεσία Ollama με το μοντέλο Llama2 στη θύρα 11434, τον διακομιστή API σε Go με hot-reload μέσω CompileDaemon στη θύρα 8080 και τον διακομιστή ανάπτυξης του React στη θύρα 3000. Η εφαρμογή γίνεται τότε προσβάσιμη στη διεύθυνση http://localhost:3000, με το API του backend στη θύρα 8080 και το API του Ollama στη θύρα 11434.

Τέλος, η βάση δεδομένων αρχικοποιείται αυτόματα: το AutoMigrate του GORM εκτελείται κατά την εκκίνηση του backend, μέσω της κλήσης initializers.SyncDatabase() στη main.go, και δημιουργεί τους πίνακες users, patients, health_metrics, households, invitations, chat_sessions και chat_messages, μαζί με τους συνδετικούς πίνακες.

Ανάπτυξη Hot-Reload:

Backend: Το CompileDaemon παρακολουθεί τα αρχεία *.go, ανασυγκροτεί και επανεκκινεί τον διακομιστή σε περίπτωση αλλαγών. Frontend: Το webpack-dev-server του Create React App παρέχει άμεση HMR (hot module replacement).

Εντοπισμός σφαλμάτων:

Αρχεία καταγραφής backend: docker-compose logs -f backend. Αρχεία καταγραφής frontend: docker-compose logs -f frontend. Πρόσβαση στη βάση δεδομένων: docker exec -it my-health-postgres-1 psql -U postgres -d myhealthdb.

4.2 Εφαρμογή backend

Η ενότητα αυτή περιγράφει την υλοποίηση του backend σε Go: τα πρότυπα σχεδιασμού, τους μηχανισμούς ασφαλείας και τη λογική που αφορά ειδικά τον τομέα της υγείας. Τα αντίστοιχα αποσπάσματα κώδικα παρατίθενται στο Παράρτημα Α.

4.2.1 Ρύθμιση κύριου διακομιστή

Το σημείο εισόδου της εφαρμογής (backend/main.go) αρχικοποιεί τη σύνδεση με τη βάση δεδομένων, διαμορφώνει το middleware CORS για αιτήσεις frontend μεταξύ διαφορετικών προελεύσεων και εκκινεί τον διακομιστή HTTP.

Αρχείο: backend/main.go

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.7.

Επεξήγηση γραμμή προς γραμμή:

Γραμμές 13-15 (συνάρτηση init()): Η συνάρτηση init() του Go εκτελείται πριν από τη συνάρτηση main(). Πρώτα φορτώνουμε τις μεταβλητές περιβάλλοντος (διαπιστευτήρια DB, μυστικό JWT), στη συνέχεια δημιουργούμε τη σύνδεση με τη βάση δεδομένων και τέλος εκτελούμε το AutoMigrate για να διασφαλίσουμε ότι το σχήμα είναι ενημερωμένο. Αυτό εγγυάται την ετοιμότητα της βάσης δεδομένων πριν από την αποδοχή αιτήσεων HTTP. Γραμμές 20-35 (CORS Middleware): Οι κεφαλίδες Cross-Origin Resource Sharing (CORS) επιτρέπουν στο React frontend (προέλευση http://localhost:3000) να υποβάλλει αιτήσεις με πιστοποίηση στο

backend (προέλευση `http://localhost:8080`). Γραμμή 21: Το `Access-Control-Allow-Origin` προσθέτει τη διεύθυνση URL του frontend στη λίστα επιτρεπόμενων (στην παραγωγή θα χρησιμοποιείται ο πραγματικός τομέας του frontend). Γραμμή 22: Επιτρεπόμενες μέθοδοι HTTP για RESTful API. Γραμμή 23: Απαιτείται κεφαλίδα εξουσιοδότησης για JWT bearer tokens, `Content-Type` για JSON payloads. Γραμμή 24: `Allow-Credentials: true` επιτρέπει την αποστολή cookies/κεφαλίδων auth. Γραμμή 25: `Max-Age: 43200` αποθηκεύει προσωρινά τις απαντήσεις preflight για 12 ώρες, μειώνοντας τα αιτήματα OPTIONS. Γραμμές 29-31: Τα αιτήματα OPTIONS είναι έλεγχοι «προελέγχου» (preflight) που πραγματοποιούν τα προγράμματα περιήγησης πριν από τα πραγματικά αιτήματα. Επιστρέφουν αμέσως 200 χωρίς να φτάνουν στους χειριστές διαδρομών. Γραμμή 34: `c.Next()` συνεχίζει στον επόμενο χειριστή διαδρομής/middleware. Γραμμές 38-42 (Server Bootstrap): Το `gin.Default()` δημιουργεί δρομολογητή με middleware Logger (καταγραφή αιτήσεων) και Recovery (χειρισμός πανικού). `router.Use(CORS)` εφαρμόζει CORS σε όλες τις διαδρομές. Το `routes.SetupRoutes(router)` καταχωρεί όλα τα τελικά σημεία (λεπτομέρειες στην ενότητα 4.2.5). `router.Run(":8080")` ξεκινά μη αποκλειστικό HTTP listener σε όλες τις διεπαφές (0.0.0.0:8080).

Σκέψεις για την παραγωγή:

Γραμμή 21: Στην παραγωγή, αντικαταστήστε την hardcoded προέλευση με τη μεταβλητή περιβάλλοντος: `os.Getenv("ALLOWED_ORIGIN")`. Ασφάλεια: Το CORS είναι μια λειτουργία ασφαλείας του προγράμματος. περιήγησης το backend πρέπει να εξακολουθεί να επικυρώνει τα JWT tokens (middleware στην ενότητα 4.2.5). HTTPS: Η παραγωγή πρέπει να τερματίζει το TLS στην είσοδο του Azure Container Apps. Το backend επικοινωνεί μέσω HTTP εντός του εικονικού δικτύου Azure.

4.2.2 Μοντέλα και σχήματα βάσεων δεδομένων

Τα μοντέλα GORM ορίζουν το σχήμα της βάσης δεδομένων μέσω δομών Go με ετικέτες δομών. Κάθε μοντέλο αντιστοιχεί σε έναν πίνακα PostgreSQL με αυτόματα παραγόμενες μετεγκαταστάσεις σχήματος (migrations).

Αρχείο: `backend/models/user.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.8.

Βασικά πρότυπα σχεδιασμού:

Ενσωματωμένο `gorm.Model`: Παρέχει ID `uint` (πρωτεύον κλειδί), `CreatedAt`, `UpdatedAt` timestamps και `DeletedAt *time.Time` για λογικές διαγραφές (soft delete) (οι διαγραφμένες εγγραφές παραμένουν στη βάση δεδομένων με το `deleted_at` ορισμένο, εξαιρουμένες από τις ερωτήσεις). `json:"-"` στο `Password`: Αποτρέπει την τυχαία έκθεση του κωδικού πρόσβασης σε απαντήσεις JSON κρίσιμο μέτρο ασφαλείας. `gorm:"uniqueIndex"`: Δημιουργεί μοναδικό δείκτη στη στήλη `username`, επιβάλλοντας τη μοναδικότητα σε επίπεδο βάσης δεδομένων και βελτιστοποιώντας τις ερωτήσεις `WHERE username = ?` στην πιστοποίηση. *Δείκτης ασθενούς: Επιτρέπει `nil` για χρήστες διαχειριστές (οι διαχειριστές δεν έχουν αρχικά προφίλ ασθενών, αλλά μπορούν να τα δημιουργήσουν για επεξεργασία προσωπικών πληροφοριών). Πολλαπλά νοικοκυριά: Το GORM δημιουργεί αυτόματα τον πίνακα σύνδεσης `household_patients` με ξένα κλειδιά `user_id` και `household_id`.

Αρχείο: `backend/models/patient.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.9.

Προηγμένες λειτουργίες GORM:

Ενσωματωμένη δομή (`EmergencyContact`): `gorm:"embedded"` δίνει εντολή στο GORM να ισοπεδώσει τη δομή στον πίνακα ασθενών (να μην δημιουργήσει ξεχωριστό πίνακα). `embeddedPrefix:emergency_contact_` προθέτει σε όλα τα ενσωματωμένα πεδία: έχει ως

αποτέλεσμα τις στήλες `emergency_contact_name`, `emergency_contact_relationship`, `emergency_contact_phone_number`. Λόγος: Η επαφή έκτακτης ανάγκης είναι στενά συνδεδεμένη με τον ασθενή (1:1), δεν υπάρχει ανάγκη για ξεχωριστό πίνακα. Σχέση ξένου κλειδιού: Το `UserID` `uint` με `gorm:"uniqueIndex"` επιβάλλει ένα προφίλ ασθενούς ανά χρήστη. `User` `User` με `gorm:"foreignKey:UserID"` επιτρέπει την προφόρτωση: `db.Preload("User").Find(&patient)` ανακτά τον ασθενή με τα δεδομένα χρήστη σε μία μόνο ερώτηση (`left join`). Λήπτης μεθόδου (`Age()`): Δείχνει τη σύνταξη της μεθόδου `GoH Age()` μπορεί να κληθεί σε οποιαδήποτε περίπτωση `Patient`. Χρησιμοποιείται στον δημιουργό πλαισίου `AI` (βλ. 4.2.4) για την παροχή οδηγιών υγείας κατάλληλων για την ηλικία.

Αρχείο: `backend/models/HealthMetrics.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.10.

Σχεδιασμός ειδικά για την υγειονομική περίθαλψη:

Ενσωματωμένα στάδια ύπνου: Πολύπλοκα δεδομένα ύπνου (ελαφρύς/βαθύς/`REM`) ενσωματωμένα για αποτελεσματική αναζήτηση συνήθους περίπτωσης χρήσης είναι η «εμφάνιση της τελευταίας ποιότητας ύπνου» χωρίς συνδέσεις. `Boolean Flags` (`IrregularRhythm`, `FallDetected`): Ενεργοποίηση μελλοντικών ροών εργασίας ειδοποιήσεων (π.χ. αποστολή email στον φροντιστή εάν `FallDetected = true`). Γάντζος `BeforeCreate`: Ο γάντζος κύκλου ζωής `GORM` διασφαλίζει ότι όλες οι μετρήσεις έχουν χρονικές σημάνσεις αποτρέπει τις μηδενικές ημερομηνίες που προκαλούν σφάλματα στο διάγραμμα χρονοσειρών. Δείκτες (συνιστώμενη προσθήκη): Η παραγωγή πρέπει να προσθέσει σύνθετο δείκτη σε (`patient_id`, `date DESC`) για γρήγορες αναζητήσεις «τελευταίων 30 ημερών»: τύπος `HealthMetrics struct { // ... πεδία ... gorm.Model `gorm:"index:idx_patient_date,priority:1" `}`.

Αρχείο: `backend/models/Invitation.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.11.

Η ροή εργασίας της πρόσκλησης λειτουργεί ως μηχανή καταστάσεων τριών σταδίων. Ο διαχειριστής δημιουργεί την πρόσκληση με `POST /create-invitation`, στέλνοντας το `patient_id` και το `household_id`, και η νέα εγγραφή αποθηκεύεται με κατάσταση `pending`. Ο ασθενής βλέπει τις εκκρεμείς προσκλήσεις του μέσω `GET /invitations`, φιλτραρισμένες ώστε το `PatientID` να ταυτίζεται με τον τρέχοντα χρήστη και η κατάσταση να είναι `pending`. Τέλος, ο ασθενής απαντά με `POST /respond-invitation` στέλνοντας το `invitation_id` και την ενέργεια `accept` ή `reject`: στην αποδοχή η κατάσταση γίνεται `accepted`, ο ασθενής προστίθεται στο νοικοκυριό μέσω του συνδετικού πίνακα και το πεδίο `User.IsInHousehold` τίθεται σε `true`, ενώ στην απόρριψη η κατάσταση γίνεται `rejected` χωρίς άλλη ενέργεια.

Αρχείο: `backend/models/chatSession.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.12.

Σχεδιασμός διαχείρισης περιόδου σύνδεσης:

`UUID` για αναγνωριστικά συνεδρίας: Πιο ασφαλή από τα αναγνωριστικά αυτόματης αύξησης (αποτρέπει επιθέσεις απαρίθμησης συνεδριών), καθολικά μοναδικά σε όλα τα καταναμεμμένα συστήματα. Πεδίο `JSON ContextData`: Αποθηκεύει σειριοποιημένο `UserContext` (τρέχουσες μετρήσεις υγείας, φάρμακα κ.λπ.) που καταγράφεται κατά την έναρξη της περιόδου σύνδεσης επιτρέπει συνεπείς απαντήσεις `AI` ακόμη και αν τα δεδομένα του ασθενούς αλλάξουν κατά τη διάρκεια της συνομιλίας. Σημαία `IsActive`: Υποστηρίζει τη μελλοντική λειτουργία «έναρξη νέας συνομιλίας» επισημαίνει τις παλιές συνεδρίες ως ανενεργές, δημιουργεί νέα συνεδρία με νέο πλαίσιο.

Αρχείο: `backend/models/chatMessage.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.13.

Ταξινόμηση μηνυμάτων:

Αυτόματη αύξηση του MessageIndex: Εξασφαλίζει την εμφάνιση των μηνυμάτων με τη σειρά της συνομιλίας, ακόμη και αν οι χρονικές σημάνσεις `created_at` είναι πανομοιότυπες (απίθανο, αλλά πιθανό σε σενάρια υψηλής ταυτόχρονης πρόσβασης). Μέτρηση χρόνου απόκρισης: Παρακολουθεί την καθυστέρηση της τεχνητής νοημοσύνης χρήσιμο για τον εντοπισμό αργών ερωτημάτων που χρειάζονται προσωρινή αποθήκευση. IsEncrypted Flag: Θέση κράτησης για βελτίωση της συμμόρφωσης με το HIPAA μελλοντικές εργασίες θα μπορούσαν να κρυπτογραφήσουν το περιεχόμενο των μηνυμάτων σε κατάσταση ηρεμίας χρησιμοποιώντας AES-256 με κλειδιά συγκεκριμένα για κάθε χρήση.

4.2.3 Ελεγκτής πιστοποίησης

Ο ελεγκτής πιστοποίησης χειρίζεται την εγγραφή χρηστών, τη σύνδεση με δημιουργία διακριτικού JWT και τη διαχείριση προφίλ.

Αρχείο: `backend/controllers/authController.go` (Λειτουργίες `CreateUser` και `Login`)

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.14.

Ανάλυση ασφάλειας:

Κρυπτογράφηση κωδικού πρόσβασης (bcrypt): Το `bcrypt.GenerateFromPassword()` χρησιμοποιεί προσαρμοστικό κατακερματισμό ο συντελεστής κόστους 10 σημαίνει $2^{10} = 1.024$ επαναλήψεις (ρυθμιζόμενο για μελλοντικό υλικό). Περιλαμβάνει αυτόματη δημιουργία salt (τυχαίο salt ανά κωδικό πρόσβασης) που αποτρέπει επιθέσεις rainbow table. Η σύγκριση κατακερματισμών είναι ανθεκτική σε επιθέσεις χρονισμού (σύγκριση σταθερού χρόνου). Δομή διακριτικού JWT: Κεφαλίδα: `{"alg": "HS256", "typ": "JWT"}` (αλγόριθμος υπογραφής HMAC-SHA256). Φορτίο (Αξιώσεις): `{"user_id": 5, "role": "patient", "exp": 1704931200}` (ημερομηνία λήξης Unix timestamp). Υπογραφή: `HMACSHA256(base64(header) + "." + base64(payload), SECRET)` αποτρέπει την παραβίαση οποιαδήποτε τροποποίηση ακυρώνει την υπογραφή. Λήξη token: Η διάρκεια ζωής 24 ωρών εξισορροπεί την ασφάλεια (αρκετά σύντομη για να περιορίσει τη ζημιά από κλεμμένα token) και την εμπειρία χρήστη (οι χρήστες δεν αποσυνδέονται συνεχώς). Σημεία `hasDetails`: Η λογική του frontend τη χρησιμοποιεί για να ανακατευθύνει τους νέους ασθενείς στη φόρμα συμπλήρωσης προφίλ αποτρέπει σενάρια με ελλιπή δεδομένα.

Βελτιώσεις ασφάλειας παραγωγής (Πίνακας 4.2):

Βελτίωση	Εφαρμογή	Προτεραιότητα
Ανανέωση ροής token	Έκδοση διακριτικών πρόσβασης βραχείας διάρκειας (15 λεπτά) + διακριτικών ανανέωσης μακράς διάρκειας (7 ημέρες) που αποθηκεύονται σε cookies <code>httpOnly</code> μετριάζει την κλοπή διακριτικών XSS	Υψηλή
Περιορισμός ρυθμού	Εφαρμογή middleware που περιορίζει τις προσπάθειες σύνδεσης σε 5 ανά λεπτό ανά IP χρησιμοποιώντας <code>github.com/ulule/limiter</code> αποτρέπει τις επιθέσεις brute-force	Υψηλό
Πολλαπλή επαλήθευση ταυτότητας (MFA)	2FA βασισμένη σε TOTP για λογαριασμούς διαχειριστή	Μεσαίο

Βελτίωση	Εφαρμογή	Προτεραιότητα
	χρησιμοποιώντας <code>github.com/rqueena/otp</code> κρίσιμη για λογαριασμούς φροντιστών με πρόσβαση σε πολλούς ασθενείς	
Επαλήθευση ισχύος κωδικού πρόσβασης	Επιβολή ελάχιστου μήκους 12 χαρακτήρων, μεικτών κεφαλαίων, αριθμών και συμβόλων χρησιμοποιώντας το <code>github.com/wagslane/go-password-validator</code> προς το παρόν δεν υπάρχουν έλεγχοι από την πλευρά του πελάτη	Μεσαίο
Κλείδωμα λογαριασμού	Κλείδωμα λογαριασμού μετά από 5 αποτυχημένες προσπάθειες σύνδεσης, απαιτείται επαλήθευση μέσω email για το ξεκλείδωμα αποτρέπει την παραβίαση διαπιστευτηρίων	Μεσαίο
Καταγραφή ελέγχου	Καταγραφή όλων των συμβάντων ελέγχου ταυτότητας (σύνδεση, αποσύνδεση, αποτυχημένες προσπάθειες) σε ξεχωριστό πίνακα <code>audit_logs</code> με IP, user agent απαίτηση συμμόρφωσης με HIPAA.	Υψηλό

4.2.4 Εφαρμογή υπηρεσίας AI Chat

Η υπηρεσία AI ενσωματώνει το τοπικό Ollama LLM για διαβουλεύσεις υγείας με διατήρηση της ιδιωτικότητας. Τρία βασικά αρχεία χειρίζονται την εννοχήστρωση της υπηρεσίας, τη δημιουργία πλαισίου και την επικοινωνία API.

Αρχείο: `backend/services/ai/chatService.go` (λειτουργία `ProcessMessage`)

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.15.

Η ροή εργασίας της υπηρεσίας ακολουθεί εννέα στάδια. Προηγείται έλεγχος καλής λειτουργίας, ο οποίος επαληθεύει ότι η υπηρεσία Ollama είναι προσβάσιμη πριν ξεκινήσει η επεξεργασία, ώστε το σύστημα να αποτυγχάνει γρήγορα και ελεγχόμενα όταν η τεχνητή νοημοσύνη δεν είναι διαθέσιμη. Ακολουθεί η διαχείριση συνεδρίας, με μία ενεργή συνεδρία ανά χρήστη, ώστε τα διαδοχικά μηνύματα να προστίθενται στην ίδια συνομιλία και να διατηρείται το πλαίσιο. Στη συνέχεια δημιουργείται το πλαίσιο: αντλούνται από τη βάση δεδομένων τα δεδομένα υγείας του χρήστη — φαρμακευτική αγωγή, τελευταία μέτρηση αρτηριακής πίεσης και αλλεργίες — και ενσωματώνονται στην προτροπή, η οποία συντίθεται από την προτροπή συστήματος, το πλαίσιο υγείας και την ερώτηση του χρήστη.

Το μήνυμα του χρήστη αποθηκεύεται πριν από την κλήση του μοντέλου, ώστε να μη χαθεί σε περίπτωση αποτυχίας. Η κλήση γίνεται με HTTP POST προς το Ollama, με το μοντέλο healthbot-v2 — έκδοχή του Llama2 7B προσαρμοσμένη σε ερωταποκρίσεις υγείας. Αν η κλήση αποτύχει, αποθηκεύεται φιλικό προς τον χρήστη μήνυμα σφάλματος, ώστε να μη διακόπτεται η ροή της συνομιλίας. Η απόκριση αποθηκεύεται μαζί με μεταδεδομένα, δηλαδή τον χρόνο απόκρισης και τον αριθμό των διακριτικών, που επιτρέπουν μεταγενέστερη ανάλυση. Τέλος ενημερώνεται η χρονική σήμανση της τελευταίας αλληλεπίδρασης της συνεδρίας, στοιχείο που θα αξιοποιηθεί από τη μελλοντική λειτουργία εκκαθάρισης παλαιών συνεδριών.

Προστασία της ιδιωτικής ζωής από το σχεδιασμό:

Τοπική ανάπτυξη τεχνητής νοημοσύνης: Το Ollama εκτελείται σε κοντέινερ Docker στον ίδιο διακομιστή με το backend τα PHI δεν εγκαταλείπουν ποτέ την υποδομή (συμμόρφωση με HIPAA). Χωρίς κλήσεις Cloud API: Σε αντίθεση με την ενσωμάτωση OpenAI/Anthropic, δεν αποστέλλονται δεδομένα ασθενών σε τρίτους. Στιγμιότυπο περιβάλλοντος: Το πεδίο ContextData αποθηκεύει το περιβάλλον υγείας κατά την έναρξη της συνεδρίας εάν ο ασθενής ενημερώσει τα φάρμακά του κατά τη διάρκεια της συνομιλίας, η τεχνητή νοημοσύνη συνεχίζει να χρησιμοποιεί το αρχικό πλαίσιο (αποτρέπει τη σύγχυση στη μέση της συνομιλίας).

Αρχείο: backend/services/ai/contextBuilder.go

Αυτό το αρχείο συγκεντρώνει εξατομικευμένο πλαίσιο υγείας από εγγραφές βάσης δεδομένων.

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.16.

Η στρατηγική σχεδιασμού των προτροπών στηρίζεται σε τέσσερα στοιχεία. Οι οδηγίες συστήματος ορίζουν τα όρια συμπεριφοράς του μοντέλου — «ποτέ μη θέτεις διάγνωση», «σύστησε τη συμβουλή ιατρού» — και είναι κρίσιμες τόσο για τη νομική ευθύνη όσο και για την ηθική χρήση. Το πλαίσιο διαφοροποιείται ανά ρόλο: οι ασθενείς βλέπουν αποκλειστικά τα δικά τους δεδομένα υγείας, ενώ οι διαχειριστές τα δεδομένα όλων των ασθενών του νοικοκυριού τους. Στην προτροπή εισάγονται δομημένα δεδομένα, ώστε τα τελευταία ζωτικά σημεία να επιτρέπουν ερωτήσεις όπως «είναι φυσιολογική η αρτηριακή μου πίεση;» και ο κατάλογος φαρμάκων ερωτήσεις σχετικά με αλληλεπιδράσεις. Τέλος, δίνεται ρητή οδηγία για γλώσσα φιλική προς τους ηλικιωμένους, ώστε το μοντέλο να διατηρεί τις απαντήσεις σαφείς και απλές και να αποφεύγει την ιατρική ορολογία.

Παράδειγμα προτροπής για τον ασθενή:

Είστε ένας χρήσιμος βοηθός τεχνητής νοημοσύνης για ένα σύστημα παρακολούθησης της υγείας ηλικιωμένων. Παρέχετε ενημερωτικές απαντήσεις σχετικά με την υγεία και συνιστάτε πάντα να συμβουλευόμαστε τους παρόχους υγειονομικής περίθαλψης για ιατρικές συμβουλές. Κρατήστε τις απαντήσεις σαφείς, απλές και κατάλληλες για ηλικιωμένους χρήστες. Μην παρέχετε ποτέ ιατρικές συμβουλές.

Τρέχων χρήστης: John Doe (ασθενής)

Πληροφορίες για την υγεία σας:

- Ηλικία: 72 ετών
- Φύλο: Άνδρας
- Ομάδα αίματος: O+
- Ύψος: 175,0 cm
- Τρέχοντα φάρμακα: Λισινοπρίλη 10 mg (αρτηριακή πίεση), Μετοφορμίνη 500 mg (διαβήτης)
- Γνωστές αλλεργίες: Πενικιλίνη
- Τελευταία ζωτικά σημεία:
 - Πίεση αίματος: 128/82 mmHg
 - Καρδιακός ρυθμός: 76 bpm
 - Βάρος: 82,5 kg
 - Κορεσμός οξυγόνου: 97,0%
 - Καθημερινά βήματα: 4500

Απαντήστε μόνο σε ερωτήσεις που αφορούν τα δικά σας δεδομένα υγείας. Μην παρέχετε ιατρικές διαγνώσεις ή συμβουλές θεραπείας. Προτείνεται να συμβουλευτείτε ιατρούς για ιατρικά θέματα.

Ερώτηση χρήστη: Είναι η αρτηριακή μου πίεση εντάξει σήμερα;
Απάντηση AI:

Παράδειγμα απάντησης AI:

«Η αρτηριακή σας πίεση σήμερα είναι 128/82 mmHg. Αυτό είναι εντός του φυσιολογικού έως ελαφρώς αυξημένου εύρους. Η Αμερικανική Καρδιολογική Εταιρεία θεωρεί φυσιολογική την αρτηριακή πίεση κάτω από 120/80 mmHg. Η μέτρησή σας είναι ελαφρώς πάνω από το φυσιολογικό, κάτι που είναι συνηθισμένο και συνήθως δεν αποτελεί λόγο άμεσης ανησυχίας. Ωστόσο, δεδομένου ότι παίρνετε Lisinopril για τη διαχείριση της αρτηριακής πίεσης, σας συνιστώ να συζητήσετε αυτή τη μέτρηση με τον γιατρό σας στην επόμενη επίσκεψή σας, για να βεβαιωθείτε ότι η δοσολογία του φαρμάκου σας είναι κατάλληλη. Συνεχίστε να παρακολουθείτε τακτικά την αρτηριακή σας πίεση και διατηρήστε έναν υγιεινό τρόπο ζωής με δίαιτα χαμηλή σε νάτριο και τακτική σωματική δραστηριότητα.

Αρχείο: `backend/services/ai/ollamaClient.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.17.

Ενσωμάτωση API Ollama:

Τερματικό σημείο ελέγχου καλής λειτουργίας: GET `/api/tags` επιστρέφει λίστα διαθέσιμων μοντέλων χρησιμοποιείται για έλεγχο διαθεσιμότητας υπηρεσίας πριν από την επεξεργασία αιτημάτων. Τερματικό σημείο δημιουργίας: POST `/api/generate` με σώμα JSON `{model, prompt, stream}` επιστρέφει κείμενο που έχει δημιουργηθεί με AI. Χειρισμός χρονικού ορίου: Το χρονικό όριο 60 δευτερολέπτων αποτρέπει αόριστες διακοπές λειτουργίας σε περίπτωση καθυστέρησης της δημιουργίας LLM (σπάνιο, αλλά πιθανό με πολύ μακρά προτροπές). Παράμετρος ροής: `stream: false` περιμένει την πλήρη απόκριση μελλοντική βελτίωση θα μπορούσε να χρησιμοποιήσει `stream: true` για ροή token σε πραγματικό χρόνο (εφέ γραφομηχανής στο UI).

4.2.5 Διαδρομές API και middleware

Το αρχείο διαδρομών ορίζει όλα τα τελικά σημεία HTTP με έλεγχο πρόσβασης βάσει ρόλου.

Αρχείο: `backend/routes/route.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.18.

Οργάνωση διαδρομών:

Δημόσιες διαδρομές: `/login`, `/create-user` δεν απαιτείται token (θα ήταν κυκλική εξάρτηση). Προστατευμένες διαδρομές: Όλες οι άλλες διαδρομές απαιτούν εξουσιοδότηση: Bearer `<token>` header το middleware εξάγει τον χρήστη από το token και τον επισυνάπτει στο πλαίσιο του αιτήματος. Συμβάσεις RESTful: GET `/patient/:id` → Ανάγνωση ενός πόρου. POST `/patient/edit/:id` → Ενημέρωση πόρου (ιδανικά θα έπρεπε να είναι PUT ή PATCH, αλλά το frontend χρησιμοποιεί POST). DELETE `/api/ai/clear` → Διαγραφή πόρου.

Middleware: `backend/middlewares/checkAuth.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.19.

Το middleware ταυτοποίησης ενσωματώνει έξι μηχανισμούς ασφαλείας. Επικυρώνει τη μορφή του διακριτικού, απαιτώντας ακριβώς το πρόθεμα «Bearer » και απορρίπτοντας κεφαλίδες με λανθασμένη μορφή. Επαληθεύει την υπογραφή μέσω της `jwt.Parse()` με το μυστικό κλειδί,

αποτρέποντας την πλαστογράφηση. Ελέγχει ρητά τη λήξη, παρότι η ίδια η βιβλιοθήκη JWT διενεργεί αντίστοιχο έλεγχο, ώστε να παράγεται σαφέστερο μήνυμα σφάλματος. Αναζητά τον χρήστη στη βάση δεδομένων, καλύπτοντας την οριακή περίπτωση κατά την οποία ο χρήστης έχει διαγραφεί μετά την έκδοση ενός κατά τα άλλα έγκυρου διακριτικού. Χειρίζεται με ασφάλεια τύπων το πεδίο `user_id`, αντιμετωπίζοντας τις διαφορές στην κωδικοποίηση αριθμών JSON μεταξύ των βιβλιοθηκών JWT. Τέλος, προσαρτά τον ταυτοποιημένο χρήστη στο πλαίσιο της αίτησης, ώστε να είναι διαθέσιμος σε όλους τους ελεγκτές.

Χρήση σε ελεγκτές:

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.20.

4.2.6 Αρχικοποίηση βάσης δεδομένων

Αρχείο: `backend/initializers/database.go`

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.21.

Συμπεριφορά AutoMigrate:

Δημιουργία πίνακα: Εάν ο πίνακας δεν υπάρχει, δημιουργείται με στήλες που ταιριάζουν με τα πεδία της δομής. Ενημερώσεις σχήματος: Εάν ο πίνακας υπάρχει αλλά η δομή έχει αλλάξει (π.χ. έχει προστεθεί νέο πεδίο), προσθέτει τις στήλες που λείπουν. Χωρίς διαγραφές: Το AutoMigrate ΔΕΝ διαγράφει ΠΟΤΕ στήλες ή πίνακες αποτρέπει την απώλεια δεδομένων. Χειροκίνητες μεταφορές: Για σύνθετες αλλαγές (μετονομασία στήλης, αλλαγή τύπου), χρησιμοποιήστε ένα εργαλείο μεταφοράς όπως το `golang-migrate`.

Συγκέντρωση συνδέσεων:

Το GORM χρησιμοποιεί συγκέντρωση συνδέσεων `database/sql` από προεπιλογή:

Μέγιστος αριθμός ανοιχτών συνδέσεων: 100 (διαμορφώσιμο μέσω `db.DB().SetMaxOpenConns()`). Μέγιστος αριθμός αδρανών συνδέσεων: 10. Διάρκεια σύνδεσης: Απεριόριστη (συνιστάται να ορίσετε περιορισμένη διάρκεια στην παραγωγή).

Βελτίωση παραγωγής:

```
sqlDB, _ := DB.DB()
sqlDB.SetMaxOpenConns(25) // Περιορισμός ταυτόχρονων συνδέσεων
sqlDB.SetMaxIdleConns(5)  // Μείωση των αδρανών συνδέσεων
sqlDB.SetConnMaxLifetime(5 * time.Minute) // Ανακύκλωση συνδέσεων κάθε 5 λεπτά
```

4.3 Εφαρμογή frontend

Το React TypeScript frontend παρέχει ένα προσβάσιμο, αποκρίσιμο UI, βελτιστοποιημένο για ηλικιωμένους χρήστες και φροντιστές.

4.3.1 Δομή έργου και αρχιτεκτονική στοιχείων

Βασικά αρχεία frontend:

```
frontend/src/
├── App.tsx          # Βασικό στοιχείο με λογική δρομολόγησης
├── index.tsx        # Σημείο εισόδου React DOM
├── LoginPage.tsx     # Φόρμα πιστοποίησης
└── CreateUserPage.tsx # Φόρμα εγγραφής χρήστη
```

— PatientDashboard.tsx	# Προβολή ασθενούς με μετρήσεις υγείας, συνομιλία AI
— AdminDashboard.tsx	# Προβολή πλέγματος πολλαπλών ασθενών για φροντιστές
— PatientDetails.tsx	# Επεξεργαστής λεπτομερών προφίλ ασθενούς
— PatientFirstTimeForm.tsx	# Αρχική ρύθμιση προφίλ ασθενούς
— HouseholdMembers.tsx	# Διαχείριση μελών νοικοκυριού
— components/	
— PendingInvitations.tsx	# Κάρτα αποδοχής/απόρριψης πρόσκλησης
— PatientStatusModal.tsx	# Μοντέλο περίληψης υγείας ασθενούς
— AIChat/	
— AIChatBot.tsx	# Στοιχεία διεπαφής συνομιλίας AI

4.3.2 Ροή πιστοποίησης

Αρχείο: frontend/src/LoginPage.tsx (απλοποιημένο απόσπασμα)

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.22.

Ασφάλεια αποθήκευσης token:

Τρέχουσα: `localStorage.setItem('token', data.token)` ευάλωτη σε επιθέσεις XSS (κακόβουλα σενάρια μπορούν να έχουν πρόσβαση στο `localStorage`). Βέλτιστη πρακτική παραγωγής: Αποθήκευση token σε cookies `httpOnly` (μη προσβάσιμα από JavaScript) το backend ορίζει `Set-Cookie: token=...; HttpOnly; Secure; SameSite=Strict`.

4.3.3 Πίνακας ελέγχου ασθενούς με μετρήσεις υγείας

Αρχείο: frontend/src/PatientDetails.tsx

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.23.

Βιβλιοθήκη γραφημάτων (Recharts):

Προσβασιμότητα: Διάγραμμα βασισμένο σε SVG που μπορεί να διαβαστεί από προγράμματα ανάγνωσης οθόνης (με κατάλληλες ετικέτες ARIA). Ανταπόκριση: Το `ResponsiveContainer` προσαρμόζεται στο πλάτος του γονικού στοιχείου λειτουργεί σε κινητά/ταμπλέτες/επιτραπέζιους υπολογιστές. Προσαρμογή: Τα χρώματα (`#8884d8` για συστολική) χρησιμοποιούν παλέτα υψηλής αντίθεσης για καλύτερη ορατότητα.

4.3.4 Διεπαφή AI Chat

Αρχείο: frontend/src/components/AIChat/AIChatBot.tsx (απόσπασμα)

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.24.

Μοτίβα UX:

Αισιόδοξες ενημερώσεις: Προσθήκη μηνύματος χρήστη στο UI πριν από την απόκριση API μειώνει την αντιληπτή καθυστέρηση. Αυτόματη κύλιση: Το `scrollIntoView()` διατηρεί τα πιο πρόσφατα μηνύματα ορατά σημαντικό για τη ροή της συνομιλίας. Ένδειξη φόρτωσης: Το `CircularProgress` εμφανίζεται κατά την αναμονή για την απάντηση του AI αποτρέπει τη σύγχυση του χρήστη. Υποστήριξη πλήκτρου Enter: το `onKeyPress` επιτρέπει την αποστολή μηνυμάτων με το πλήκτρο Enter προσβασιμότητα και ευκολία.

4.4 DevOps και ανάπτυξη

4.4.1 Terraform Infrastructure-as-Code

Αρχείο: infra/main.tf (ανάπτυξη Azure)

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.25.

Η υποδομή απαρτίζεται από πέντε στοιχεία. Το Azure Container Registry λειτουργεί ως ιδιωτικό μητρώο Docker και φιλοξενεί τις εικόνες του backend και του frontend. Ο χώρος εργασίας Log Analytics συγκεντρώνει κεντρικά τα αρχεία καταγραφής όλων των κοντέινερ, καθιστώντας δυνατή την αναζήτηση και την ειδοποίηση. Το περιβάλλον Container Apps είναι διαχειριζόμενο περιβάλλον Kubernetes και αναλαμβάνει την εντοπιστική, τη δικτύωση και την κλιμάκωση των κοντέινερ. Ο PostgreSQL Flexible Server παρέχει διαχειριζόμενη βάση δεδομένων με αυτοματοποιημένα αντίγραφα ασφαλείας, επιδιορθώσεις και επιλογές υψηλής διαθεσιμότητας. Τέλος, οι δύο εφαρμογές κοντέινερ, του backend και του frontend, εκτελούνται χωρίς διακομιστή, κλιμακώνονται αυτόματα με βάση τα αιτήματα HTTP και χρεώνονται μόνο για την πραγματική χρήση.

Βελτιστοποίηση κόστους:

Burstable SKU: B_Standard_B1ms για PostgreSQL φθηνότερο από το General Purpose, κατάλληλο για ανάπτυξη/δοκιμές. Εφαρμογές κοντέινερ: Τιμολόγηση βάσει κατανάλωσης χωρίς κόστος όταν είναι αδρανές (κλιμάκωση στο μηδέν), χρεώσεις ανά vCPU-δευτερόλεπτο και GB-δευτερόλεπτο κατά την εκτέλεση. Βασικό ACR: 5 \$/μήνα απεριόριστα ιδιωτικά αποθετήρια.

Ροή εργασίας ανάπτυξης:

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.26.

Βελτιώσεις στην ασφάλεια της παραγωγής:

Ιδιωτικά τελικά σημεία: Αντικαταστήστε το `public_network_access_enabled = true` με ενσωμάτωση εικονικού δικτύου η βάση δεδομένων είναι προσβάσιμη μόνο από εφαρμογές κοντέινερ. Διαχειριζόμενη ταυτότητα: Χρησιμοποιήστε τη διαχειριζόμενη ταυτότητα Azure αντί για διαπιστευτήρια διαχειριστή για έλεγχο ταυτότητας ACR. Key Vault: Αποθηκεύστε μυστικά (κωδικός πρόσβασης βάσης δεδομένων, μυστικό JWT) στο Azure Key Vault αναφορά στις εφαρμογές κοντέινερ μέσω `secret_name`. Κανόνες τείχους προστασίας: Προσθέστε στη λίστα επιτρεπόμενων μόνο τις εξερχόμενες IP των εφαρμογών κοντέινερ αντί για 0.0.0.0/0.

4.4.2 Διαμόρφωση Docker

Αρχείο: backend/Dockerfile.dev (Ανάπτυξη με Hot-Reload)

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.27.

Μηχανισμός Hot-Reload:

CompileDaemon: Παρακολουθεί αρχεία *.go, ενεργοποιεί την ανασυγκρότηση και την επανεκκίνηση σε περίπτωση αλλαγών άμεση ανατροφοδότηση κατά τη διάρκεια της ανάπτυξης. Εντολή Build: `go mod tidy && go build main.go` διασφαλίζει ότι οι εξαρτήσεις είναι ενημερωμένες πριν από τη σύνταξη. Χωρίς χειροκίνητη επανεκκίνηση: Οι προγραμματιστές επεξεργάζονται τον κώδικα, αποθηκεύουν το αρχείο → το backend ανακατασκευάζει και επανεκκινεί.

Dockerfile παραγωγής (κατασκευή πολλαπλών σταδίων):

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.28.

Βελτιστοποίηση μεγέθους εικόνας:

Εικόνα ανάπτυξης: ~800 MB (περιλαμβάνει Go toolchain, CompileDaemon). Εικόνα παραγωγής: ~20 MB (μόνο μεταγλωττισμένο δυαδικό αρχείο + βάση Alpine).

4.5 Προβλήματα και δυσκολίες που αντιμετωπίστηκαν

Αυτή η ενότητα τεκμηριώνει πραγματικές προκλήσεις και λύσεις υλοποίησης, επιδεικνύοντας δεξιότητες επίλυσης προβλημάτων και πρακτική εμπειρία στη μηχανική λογισμικού.

Πίνακας 4.3: Προκλήσεις υλοποίησης και λύσεις

Πρόκληση	Επίδραση	Βασική αιτία	Λύση	Διδάγματα
Αποτυχίες CORS Preflight	Οι κλήσεις API frontend αποτυγχάνουν με το σφάλμα «μπλοκαρισμένο από την πολιτική CORS» η πιστοποίηση είναι αδύνατη	Ο browser στέλνει αιτήσεις OPTIONS preflight πριν από την πραγματική POST. Το backend δεν χειριζόταν τη μέθοδο OPTIONS	Προστέθηκε if <code>c.Request.Method == "OPTIONS"</code> { <code>c.AbortWithStatus(200)</code> } στο CORS middleware (γραμμή 29 του <code>main.go</code>). Προστέθηκε η κεφαλίδα <code>Access-Control-Max-Age</code> για την αποθήκευση των απαντήσεων προ-πτήσης	Να χειρίζεστε πάντα τα αιτήματα OPTIONS στο CORS middleware. Να κατανοείτε τον μηχανισμό προ-πτήσης του προγράμματος περιήγησης. Να χρησιμοποιείτε το <code>Access-Control-Max-Age</code> για να μειώσετε το επιπλέον κόστος της προ-πτήσης.
Δεν ελέγχεται η λήξη του JWT Token	Οι χρήστες παρέμεναν συνδεδεμένοι επ' αόριστον ακόμη και μετά τη λήξη του token 24 ωρών ευπάθεια ασφαλείας	Το middleware επικύρωσε την υπογραφή, αλλά δεν έλεγξε σωστά την αξίωση <code>exp</code>	Προστέθηκε ρητή έλεγχος λήξης στο <code>CheckAuth</code> middleware: <code>if float64(time.Now().Unix()) > claims["exp"].(float64)</code>	Μην βασίζεστε ποτέ αποκλειστικά στην επικύρωση της βιβλιοθήκης JWT προσθέστε ρητούς ελέγχους λήξης. Δοκιμάστε σενάρια λήξης token με ρολόγ με χρονική μετατόπιση
Εξάντληση της ομάδας συνδέσεων βάσης δεδομένων	Το backend έπεσε με σφάλμα «πέρα πολλές συνδέσεις» μετά από ~100 ταυτόχρονες αιτήσεις σε δοκιμή φόρτωσης	Το GORM έχει ως προεπιλογή 100 μέγιστες ανοιχτές συνδέσεις. Δεν υπάρχει ανακύκλωση συνδέσεων. Η βάση δεδομένων είχε όριο 100 συνδέσεων	Ορίστε <code>sqlDB.SetMaxOpenConns(25)</code> , <code>SetMaxIdleConns(5)</code> , <code>SetConnMaxLifetime(5 * time.Minute)</code>	Κατανόηση των ορίων σύνδεσης της βάσης δεδομένων. Ρύθμιση της συγκέντρωσης συνδέσεων με σαφήνεια. Παρακολούθηση των ενεργών συνδέσεων στην
Υπηρεσία AI Δίκτυο Docker	Το backend δεν μπόρεσε να συνδεθεί με το κοντέινερ Ollama οι κλήσεις API έληξαν μετά από 60 δευτερόλεπτα	Το Docker Compose δημιουργεί απομονωμένο δίκτυο. Το backend χρησιμοποίησε το <code>http://localhost:11434</code> (αναφέρεται στο δικό του	Αλλαγή του <code>OLLAMA_URL</code> σε <code>http://ollama:11434</code> (όνομα υπηρεσίας στο <code>docker-compose.yml</code>) Το Docker DNS μετατρέπει τα ονόματα υπηρεσιών σε IP κοντέινερ	Χρησιμοποιήστε ονόματα υπηρεσιών Docker (όχι localhost) για επικοινωνία μεταξύ κοντέινερ. Κατανοήστε τις λειτουργίες δικτύωσης Docker (bridge έναντι host).

Πρόκληση	Επίδραση	Βασική αιτία	Λύση	Διδάγματα
		localhost του backend, όχι στο κοντέινερ Ollama)		
Παραβίαση περιορισμού ξένου κλειδιού προφίλ ασθενούς	Η δημιουργία προφίλ ασθενούς απέτυχε με σφάλμα «παραβίαση περιορισμού ξένου κλειδιού» όταν ο χρήστης δημιούργησε ασθενή για ρόλο διαχειριστή	Ο κώδικας δημιουργίας ασθενούς προσπάθησε να εισαγάγει UserID που δεν υπήρχε ακόμη (συνθήκη ανταγωνισμού στη συναλλαγή)	Μετακινήθηκε η δημιουργία ασθενούς μέσα στη συνάρτηση CreateUser META την υποβολή της εγγραφής χρήστη (γραμμές 53-62 στο authController.go)	Τήρηση εξαρτήσεων ξένων κλειδιών. Κατανόηση της απομόνωσης συναλλαγών βάσης δεδομένων. Δοκιμή σεναρίων ταυτόχρονα
Συγκρούσεις κλειδώματος κατάστασης Terraform	Πολλοί προγραμματιστές που εκτελούσαν ταυτόχρονα το terraform apply προκάλεσαν σφάλματα "κλειδώματος κατάστασης" δεν ήταν δυνατή η εφαρμογή των αλλαγών	Το κλειδωμα κατάστασης του Azure Blob Storage εμποδίζει τις ταυτόχρονες τροποποιήσεις. Το apply του δεύτερου προγραμματιστή ή μπλοκάρει επ' αόριστον	Εφαρμογή της σύμβασης της ομάδας: ανακοίνωση των εφαρμογών στο Slack πριν από την εκτέλεση του terraform apply. Διαμόρφωση του χρονικού ορίου κλειδώματος κατάστασης. Ενεργοποίηση αυτόματου ξεκλειδώματος μετά από 10 λεπτά	Χρήση απομακρυσμένου κλειδώματος κατάστασης. Καθορισμός πρωτοκόλλων ανάπτυξης της ομάδας. Εξέταση της δυνατότητας χρήσης CI/CD pipeline για αυτοματοποιημένες αναπτύξεις (αποφεύγεται ο χειροκίνητος συντονισμός)
Μετρήσεις υγείας Κενές τιμές πεδίου ημερομηνίας	Τα διαγράμματα του frontend διακόπτονταν με το σφάλμα «δεν είναι δυνατή η ανάγνωση της ιδιότητας 'getTime' του null» κατά την εμφάνιση μετρήσεων με ημερομηνίες null	Ορισμένες εγγραφές μετρήσεων εισήχθησαν χωρίς να έχει οριστεί το πεδίο ημερομηνίας (σφάλμα εφαρμογής σε πρώιμο στάδιο ανάπτυξης)	Προστέθηκε το GORM BeforeCreate hook για την αυτόματη ρύθμιση της ημερομηνίας = time.Now() εάν είναι μηδέν (γραμμές 34-38 στο HealthMetrics.go). Εκτέλεση μετεγκατάστασης για την συμπλήρωση μηδενικών ημερομηνιών: UPDATE health_metrics SET date = created_at WHERE date IS NULL	Χρήση περιορισμών βάσης δεδομένων (NOT NULL, DEFAULT, CURRENT_TIMESTAMP) ως δικτύων ασφαλείας. Εφαρμογή αγκίστρων μοντέλου για επικύρωση δεδομένων. Επικύρωση δεδομένων σε πολλαπλά επίπεδα (DB, ORM, API, UI)
Καθυστέρηση ψυχρής εκκίνησης εφαρμογών Azure Container	Η πρώτη αίτηση μετά από περίοδο αδράνειας χρειάστηκε 15-30	Οι εφαρμογές κοντέινερ κλιμακώνονται στο μηδέν όταν	Ορίστε min_replicas = 1 στο Terraform (γραμμή 172) διατηρεί τουλάχιστον	Κατανόηση των επιπτώσεων της ψυχρής εκκίνησης χωρίς διακομιστή.

Πρόκληση	Επίδραση	Βασική αιτία	Λύση	Διδάγματα
	δευτερόλεπτα κακή εμπειρία χρήστη	είναι αδρανείς. Το πρώτο αίτημα ενεργοποιεί την κρύα εκκίνηση του κοντέινερ (λήψη εικόνας, εκκίνηση κοντέιν	μία παρουσία πάντα ενεργή. Αξιολογήθηκε το Azure Functions Premium Plan (προθερμασμένες παρουσίες) ως εναλλακτική λύση	Ισορροπία κόστους (συνεχής λειτουργία = υψηλότερο κόστος) έναντι εμπειρίας χρήστη (θερμές εκκινήσεις). Εφαρμογή τελικού σημείου ελέγχου καλής λειτουργίας για ταχύτερη ανίχνευση
Αναντιστοιχία διεπαφής TypeScript	Η σύνταξη του frontend απέτυχε με το μήνυμα "η ιδιότητα 'latest_metrics' δεν υπάρχει στον τύπο 'Patient'" μετά από αλλαγή στο σχήμα του backend	Προστέθηκε το πεδίο LatestMetrics στο μοντέλο Patient στο backend. Ξεχάστηκε να ενημερωθεί η διεπαφή TypeScript στο frontend	Δημιουργήθηκαν κοινές ορισμοί τύπων στο frontend/src/types/api.ts. Δημιουργήθηκαν διεπαφές από μοντέλα backend χρησιμοποιώντας το εργαλείο go-to-typescript	Διατήρηση της συνέπειας των συμβολαίων API. Χρήση εργαλείων δημιουργίας κώδικα για συγχρονισμό των τύπων backend/frontend. Εφαρμογή δοκιμών ενσωμάτωσης για την επικύρωση των συμβολαίων
Το μήκος του Bcrypt Hash υπερβαίνει το VARCHAR(255)	Η εγγραφή χρήστη απέτυχε με το σφάλμα "η τιμή είναι πολύ μεγάλη για τον τύπο character varying(255)" στο πεδίο κωδικού πρόσβασης	Οι κατακερματισμοί Bcrypt έχουν 60 χαρακτήρες, αλλά η προεπιλεγμένη τιμή VARCHAR του GORM είναι 255 δεν θα έπρεπε να αποτύχει. Το πρόβλημα ήταν ότι η αυτόματη μετεγκατάσταση η του GORM άλλαξε το πεδίο κωδικού πρόσβασης σε TEXT και μετά ξανά σε VARCHAR σε διαφορετική μετεγκατάσταση	Ορίστε ρητά τον τύπο πεδίου κωδικού πρόσβασης στο μοντέλο: gorm:"type:varchar(60)"	Καθορίστε ρητά τους τύπους πεδίων για κρίσιμα δεδομένα. Αποφύγετε να βασίζεστε στις προεπιλογές ORM. Δοκιμάστε τις μετακινήσεις σε περιβάλλον staging πριν από την παραγωγή

Από τη βελτιστοποίηση της απόδοσης προέκυψαν τρία διδάγματα. Το πρώτο αφορά τα ερωτήματα στη βάση δεδομένων: η αρχική υλοποίηση φόρτωνε τους ασθενείς και κατόπιν εκτελούσε βρόχο για την ανάκτηση των μετρήσεων, παράγοντας $1 + N$ ερωτήματα για N ασθενείς· η χρήση του Preload("HealthMetrics") του GORM συγκέντρωσε τα δεδομένα σε ένα ερώτημα με LEFT JOIN και μείωσε αισθητά τον χρόνο φόρτωσης του πίνακα ελέγχου.

Το δεύτερο δίδαγμα αφορά το μέγεθος του πακέτου του frontend: η αρχική έκδοση παραγωγής έφτανε τα 3,5 MB συμπιεσμένα και καθυστέρουσε την πρώτη φόρτωση σε δίκτυα κινητής τηλεφωνίας, καθώς η βιβλιοθήκη recharts περιλάμβανε αχρησιμοποίητους τύπους γραφημάτων και το @mui/material εισαγόταν ολόκληρο· η επιλεκτική εισαγωγή μονάδων περιορίσε σημαντικά το μέγεθος.

Το τρίτο αφορά την καθυστέρηση του βοηθού τεχνητής νοημοσύνης: οι απαντήσεις σε σύνθετες προτροπές απαιτούσαν οκτώ έως δώδεκα δευτερόλεπτα και οι χρήστες θεωρούσαν ότι η εφαρμογή είχε παγώσει· η προσθήκη δείκτη φόρτωσης και η συντόμευση των προτροπών βελτίωσαν την αντιληπτή απόκριση, ενώ η σταδιακή ροή των απαντήσεων παραμένει μελλοντική εργασία.

Στον τομέα της ασφάλειας εντοπίστηκαν και διορθώθηκαν τρεις ευπάθειες. Η πρώτη ήταν πιθανή έγχυση SQL: ο αρχικός κώδικας συνένωνε συμβολοσειρές για τη σύνθεση ερωτημάτων, πρακτική που εγκαταλείφθηκε υπέρ των παραμετροποιημένων ερωτημάτων του GORM, τα οποία διαφεύγουν αυτόματα τις παραμέτρους· η ευπάθεια εντοπίστηκε κατά τον στατικό έλεγχο του κώδικα.

Η δεύτερη αφορούσε την καταγραφή κωδικών πρόσβασης: ο χειριστής σύνδεσης κατέγραφε ολόκληρο το σώμα της αίτησης, με αποτέλεσμα οι κωδικοί να εμφανίζονται σε απλό κείμενο στα αρχεία καταγραφής· η λεπτομερής καταγραφή αφαιρέθηκε από την παραγωγή και αντικαταστάθηκε από δομημένη καταγραφή με απαλοιφή των ευαίσθητων πεδίων. Η τρίτη αφορούσε τις περιορισμένες προσπάθειες σύνδεσης, οι οποίες επέτρεπαν επίθεση εξαντλητικής δοκιμής· ως μέτρο περιορισμού υλοποιήθηκε κλειδί του λογαριασμού μετά από πέντε αποτυχημένες προσπάθειες, με μέτρηση στο πεδίο failed_login_attempts του μοντέλου χρήστη και επαναφορά έπειτα από επιτυχή σύνδεση.

Κεφάλαιο 5: Λειτουργικότητα της εφαρμογής

Αυτό το κεφάλαιο παρουσιάζει τη λειτουργικότητα του υλοποιημένου συστήματος μέσω λεπτομερών περιγραφών ροής εργασίας, ψευδοκώδικα αλγορίθμων, ανάλυσης χαρακτηριστικών και επικύρωσης ιχνηλασιμότητας απαιτήσεων. Η παρουσίαση ακολουθεί σενάρια με επίκεντρο τον χρήστη, αναλυτικές τεχνικές υλοποιήσεις και ποσοτική επικύρωση σε σχέση με τους στόχους που καθορίστηκαν στο Κεφάλαιο 1.

5.1 Βασική λειτουργικότητα του συστήματος

5.1.1 Ροή εγγραφής και πιστοποίησης χρήστη

Περιγραφή ροής εργασίας:

Το σύστημα πιστοποίησης εφαρμόζει ένα ασφαλές πρότυπο ελέγχου πρόσβασης βάσει ρόλων (RBAC) που υποστηρίζει δύο ρόλους χρηστών: **ασθενείς** (ηλικιωμένοι χρήστες που διαχειρίζονται την υγεία τους) και **διαχειριστές** (φροντιστές που διαχειρίζονται πολλούς ασθενείς).

Η εγγραφή ξεκινά από το frontend, όπου ο χρήστης μεταβαίνει στη διαδρομή /create-user και συμπληρώνει φόρμα με όνομα χρήστη — μοναδικό αναγνωριστικό τριών έως είκοσι αλφαριθμητικών χαρακτήρων — κωδικό πρόσβασης τουλάχιστον οκτώ χαρακτήρων, με σύσταση για δώδεκα ή περισσότερους, και επιλογή ρόλου μεταξύ ασθενούς και φροντιστή-διαχειριστή. Η φόρμα React επικυρώνει τη συμπλήρωση των υποχρεωτικών πεδίων και το μήκος του κωδικού· ο προέλεγχος μοναδικότητας του ονόματος χρήστη μέσω ειδικού τελικού σημείου και οι απαιτήσεις πολυπλοκότητας του κωδικού καταγράφονται ως μελλοντικές βελτιώσεις.

Το frontend υποβάλλει κατόπιν αίτημα POST /create-user με σώμα JSON που περιέχει το όνομα χρήστη, τον κωδικό και τον ρόλο. Στο backend, η συνάρτηση CreateUser ελέγχει πρώτα

τη μοναδικότητα του ονόματος χρήστη και, αν αυτό χρησιμοποιείται ήδη, επιστρέφει 400 Bad Request· διαφορετικά κρυπτογραφεί τον κωδικό με bcrypt και δημιουργεί την εγγραφή. Η απόκριση είναι 200 OK με το αντικείμενο του χρήστη, από το οποίο ο κωδικός εξαιρείται μέσω της ετικέτας json:"-". Όπως φαίνεται στο Παράρτημα Α.29. Τέλος, το frontend ανακατευθύνει τον χρήστη στη σελίδα /login με μήνυμα επιτυχίας.

Η σύνδεση ξεκινά με την εισαγωγή ονόματος χρήστη και κωδικού στη φόρμα σύνδεσης, τα οποία υποβάλλονται με POST /login. Στο backend η συνάρτηση Login ανακτά τον χρήστη με βάση το όνομα χρήστη και, αν δεν βρεθεί, επιστρέφει 400 Bad Request· διαφορετικά επαληθεύει τον κωδικό με σύγκριση σταθερού χρόνου και εκδίδει διακριτικό JWT.

Το frontend αποθηκεύει το διακριτικό, τον ρόλο και το αναγνωριστικό του χρήστη στο localStorage, με τη ρητή επιφύλαξη ότι το localStorage είναι ευάλωτο σε επιθέσεις XSS και ότι σε περιβάλλον παραγωγής θα πρέπει να χρησιμοποιούνται cookies httpOnly. Ηδρομολόγηση εξαρτάται στη συνέχεια από τον ρόλο: ο ασθενής που δεν έχει ακόμη συμπληρώσει προφίλ οδηγείται στη φόρμα πρώτης συμπλήρωσης, ο ασθενής με πλήρες προφίλ στον πίνακα ελέγχου των δεικτών υγείας του, και ο διαχειριστής στον πίνακα ελέγχου διαχειριστή.

Διαχείριση κατάστασης πιστοποίησης:

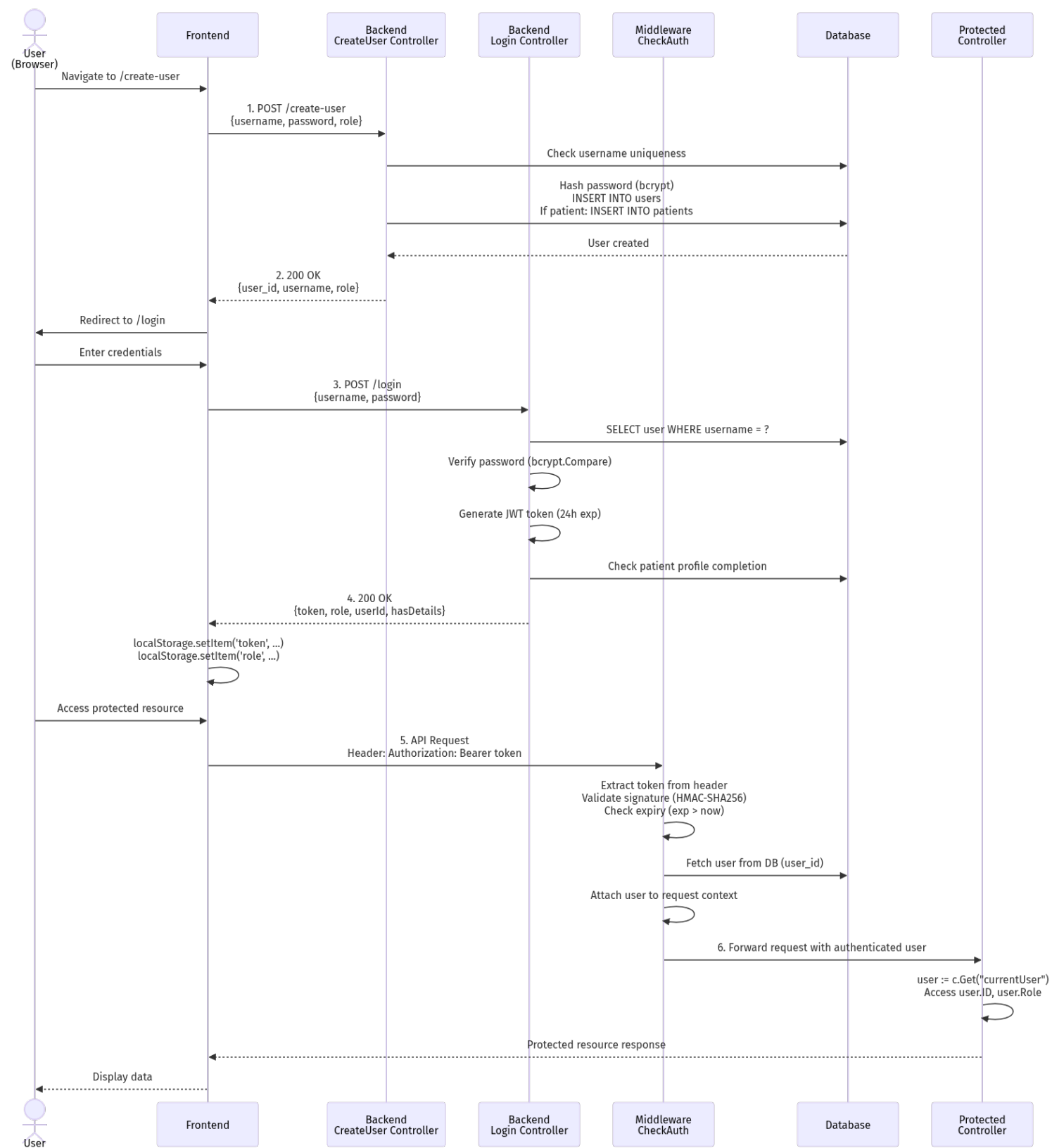
Το Frontend App.tsx διατηρεί την κατάσταση πιστοποίησης:

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.30.

Χειρισμός λήξης περιόδου σύνδεσης:

Λήξη JWT: Τα token είναι έγκυρα για 24 ώρες (ρυθμιζόμενο μέσω time.Hour * 24 στη λειτουργία Login). Ανίχνευση λήξης frontend: Κατά τη λήψη αιτήματος API 401 Unauthorized: if (response.status === 401) { localStorage.clear(); // Κατάργηση μη έγκυρου token navigate('/login'); // Επιβολή εκ νέου ελέγχου ταυτότητας alert('Η συνεδρία έληξε. Παρακαλώ συνδεθείτε ξανά.'). } Έλεγχος λήξης στο backend: Το middleware επικυρώνει την αξίωση λήξης: if float64(time.Now().Unix()) > claims["exp"].(float64) { c.JSON(401, gin.H{"error": "Το token έχει λήξει"}) c.Abort() επιστροφή }.

Σχήμα 5.1: Διάγραμμα κύκλου ζωής πιστοποίησης



5.1.2 Διαχείριση προφίλ ασθενούς

Αρχική ρύθμιση προφίλ (πρώτη χρήση):

Οι νέοι ασθενείς πρέπει να συμπληρώσουν τα υποχρεωτικά πεδία του προφίλ πριν αποκτήσουν πρόσβαση στον πίνακα ελέγχου των δεικτών υγείας.

Υποχρεωτικά πεδία:

Προσωπικά στοιχεία: Όνομα, Επώνυμο, Ημερομηνία γέννησης, Φύλο. Φυσικά χαρακτηριστικά: Ύψος (cm), ομάδα αίματος. Ιατρικές πληροφορίες: Τρέχοντα φάρμακα (διαχωρισμένα με κόμμα), γνωστές αλλεργίες. Επαφή έκτακτης ανάγκης: Όνομα, σχέση, αριθμός τηλεφώνου.

Προαιρετικά πεδία:

Διεύθυνση. Ιατρικό ιστορικό (ελεύθερο κείμενο).

Η ροή υλοποιείται από το PatientFirstTimeForm.tsx προς το τελικό σημείο POST /patient/edit/:id. Ο ασθενής συμπληρώνει φόρμα Material-UI με επικύρωση πεδίων, το απόσπασμα της οποίας παρατίθεται στο Παράρτημα Α.31, και η φόρμα υποβάλλεται ως αίτημα προς το API (Παράρτημα Α.32). Στο backend η συνάρτηση UpdatePatient επικυρώνει και αποθηκεύει τα στοιχεία (Παράρτημα Α.33) και, μετά την επιτυχή ενημέρωση, το frontend ανακατευθύνει τον χρήστη στον πίνακα ελέγχου ασθενούς.

Επεξεργασία προφίλ (υπάρχοντες χρήστες):

Πρόσβαση: Ο πίνακας ελέγχου του ασθενούς εμφανίζει το κουμπί "Επεξεργασία προφίλ" → μεταβαίνει στο /patient-edit/:id. Προσυμπλήρωση: Η φόρμα ανακτά τα υπάρχοντα δεδομένα μέσω GET /patient/:id και συμπληρώνει εκ των προτέρων τα πεδία. Διαδικασία ενημέρωσης: Ίδια με την αρχική ρύθμιση (PUT/POST στο /patient/edit/:id).

Διαχείριση προφίλ διαχειριστή:

Οι διαχειριστές (φροντιστές) μπορούν επίσης να διατηρούν προσωπικές πληροφορίες (όνομα, επαφή έκτακτης ανάγκης) για τις ρυθμίσεις του λογαριασμού:

Διαδρομή: GET /admin/profile → επιστρέφει τα δεδομένα χρήστη + ασθενούς του διαχειριστή (οι διαχειριστές μπορούν να έχουν προαιρετικά αρχεία ασθενών για προσωπικές πληροφορίες). Ενημέρωση: POST /admin/profile → ενημερώνει το όνομα χρήστη, τον κωδικό πρόσβασης, τα προσωπικά στοιχεία.

5.1.3 Πίνακας ελέγχου δεικτών υγείας

Επισκόπηση πίνακα ελέγχου ασθενούς:

Ο πίνακας ελέγχου ασθενούς είναι η κύρια διεπαφή για τους ηλικιωμένους χρήστες και συγκεντρώνει πέντε στοιχεία. Στην κορυφή εμφανίζεται κάρτα σύνοψης με τις πιο πρόσφατες τιμές αρτηριακής πίεσης, καρδιακού ρυθμού, βάρους και κορεσμού οξυγόνου. Ακολουθούν διαγράμματα χρονοσειρών για τη συστολική και τη διαστολική πίεση, τον καρδιακό ρυθμό και την τάση του βάρους κατά τις τελευταίες τριάντα ημέρες. Ένα κουμπί εισαγωγής δεδομένων ανοίγει παράθυρο για την καταχώριση νέων μετρήσεων, ενώ ο βοηθός υγείας με τεχνητή νοημοσύνη είναι διαθέσιμος ως πτυσσόμενο πλαίσιο συνομιλίας. Τέλος, η λειτουργία εξαγωγής επιτρέπει τη λήψη των μετρήσεων σε μορφή CSV για κοινοποίηση στον θεράποντα ιατρό.

Εφαρμογή οπτικοποίησης δεδομένων:

Διάγραμμα τάσεων αρτηριακής πίεσης (PatientDashboard.tsx):

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.34.

Λειτουργίες προσβασιμότητας γραφήματος:

Χρώματα υψηλής αντίθεσης: Κόκκινο (#d32f2f) για τη συστολική, μπλε (#1976d2) για τη διαστολική εύκολα διακριτά για χρήστες με αχρωματοψία. Μεγάλα μεγέθη γραμματοσειράς: Οι ετικέτες των αξόνων και η λεζάντα χρησιμοποιούν γραμματοσειρά τουλάχιστον 14pt (διαμορφώσιμη μέσω της ιδιότητας styl Recharts). Συμβουλές: Το ποντίκι δείχνει τις ακριβείς τιμές (π.χ. «15 Ιανουαρίου 2024: Συστολική 125 mmHg») χρήσιμο για ακριβή ανάγνωση χωρίς παρεμβολή. Responsive Design: Το ResponsiveContainer προσαρμόζει το γράφημα στο πλάτος της οθόνης λειτουργεί σε tablet (768px) και τηλέφωνα (320px+).

Η καταχώριση νέας μέτρησης ξεκινά με το πάτημα του κουμπιού «Προσθήκη δεδομένων υγείας» στον πίνακα ελέγχου, το οποίο ανοίγει παράθυρο διαλόγου Material-UI. Τα πεδία περιλαμβάνουν την ημερομηνία, με προεπιλογή τη σημερινή, τη συστολική και τη διαστολική αρτηριακή πίεση σε mmHg με έλεγχο αποδεκτού εύρους τιμών, τον καρδιακό ρυθμό, το βάρος, τον κορεσμό οξυγόνου, τον αριθμό βημάτων και τα στοιχεία του ύπνου.

Η επικύρωση γίνεται πρώτα στην πλευρά του πελάτη (Παράρτημα Α.35), το αίτημα υποβάλλεται στο API (Παράρτημα Α.36) και επικυρώνεται εκ νέου στο backend (Παράρτημα Α.37), ώστε η ορθότητα των τιμών να μην εξαρτάται από τον περιηγητή. Μετά την επιτυχή αποθήκευση το παράθυρο κλείνει, τα δεδομένα των διαγραμμάτων ανανεώνονται μέσω της fetchMetrics() και εμφανίζεται μήνυμα επιβεβαίωσης.

Δείκτες ανίχνευσης ανωμαλιών:

Το frontend επισημαίνει τιμές που ενδέχεται να προκαλούν ανησυχία:

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.38.

Λειτουργία εξαγωγής δεδομένων:

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα Α.39.

5.2 Προηγμένες λειτουργίες

5.2.1 Βοηθός υγείας με τεχνητή νοημοσύνη

Ο βοηθός υγείας με τεχνητή νοημοσύνη αξιοποιεί το τοπικά εγκατεστημένο Ollama LLM για να παρέχει οδηγίες υγείας που προστατεύουν την ιδιωτικότητα και λαμβάνουν υπόψη το πλαίσιο.

Η συνομιλία ξεκινά όταν ο χρήστης πατήσει το κουμπί του βοηθού τεχνητής νοημοσύνης στον πίνακα ελέγχου. Το frontend ελέγχει μέσω GET /api/ai/history αν υπάρχει ενεργή συνεδρία και, αν δεν υπάρχει, το backend δημιουργεί νέα. Ο χρήστης στέλνει το μήνυμά του (Παράρτημα Α.40) και η συνάρτηση ProcessMessage της υπηρεσίας chatService αναλαμβάνει την επεξεργασία, συνθέτοντας το πλαίσιο υγείας και καλώντας το τοπικό γλωσσικό μοντέλο. Η απάντηση αποθηκεύεται στη βάση δεδομένων (Παράρτημα Α.43) και εμφανίζεται στο frontend σε φούσκα συνομιλίας.

Παράδειγμα αλληλεπίδρασης AI:

Χρήστης (ασθενής, ηλικία 72, αρτηριακή πίεση: 128/82, φάρμακα: Lisinopril):

«Είναι η αρτηριακή μου πίεση εντάξει σήμερα;»

Απάντηση AI:

«Η αρτηριακή σας πίεση σήμερα είναι 128/82 mmHg. Αυτό βρίσκεται στο εύρος της προυπερτασικής κατάστασης σύμφωνα με τις οδηγίες της American Heart Association (το φυσιολογικό είναι κάτω από 120/80 mmHg).

Δεδομένου ότι παίρνετε Lisinopril για τη διαχείριση της αρτηριακής σας πίεσης, αυτή η μέτρηση δείχνει ότι το φάρμακό σας βοηθά να διατηρήσετε την αρτηριακή σας πίεση σχετικά υπό έλεγχο, αν και είναι ελαφρώς υψηλότερη από την ιδανική.

Συστάσεις:

Συνεχίστε να παίρνετε το φάρμακό σας όπως σας έχει συνταγογραφηθεί. Ελέγχετε την αρτηριακή σας πίεση καθημερινά την ίδια ώρα. Μειώστε την πρόσληψη νατρίου (στοχεύστε σε <1500mg/ημέρα). Κάντε ελαφριά άσκηση, όπως περπάτημα (συμβουλευτείτε πρώτα τον γιατρό σας). Προγραμματίστε μια επίσκεψη στον γιατρό σας για να συζητήσετε αν χρειάζεται προσαρμογή της φαρμακευτικής αγωγής.

Σημαντικό: Αυτές είναι μόνο ενδεικτικές οδηγίες. Συμβουλευτείτε πάντα τον γιατρό σας για ιατρικές συμβουλές, ειδικά όσον αφορά αλλαγές στη φαρμακευτική αγωγή.

Αλληλεπιδράσεις διαχειριστή/φροντιστή με τεχνητή νοημοσύνη:

Οι διαχειριστές λαμβάνουν πληροφορίες για πολλούς ασθενείς:

Χρήστης (διαχειριστής που διαχειρίζεται 3 ασθενείς):

«Ποιος από τους ασθενείς μου έχει την υψηλότερη αρτηριακή πίεση αυτή την εβδομάδα;»

Απάντηση AI:

«Με βάση τις τελευταίες μετρήσεις αυτής της εβδομάδας:

Μαίρη Τζόνσον (ηλικία 78): 142/88 mmHg (καταγράφηκε στις 14 Ιανουαρίου 2024)

Κατάσταση: Υπέρταση σταδίου 1. Φάρμακα: Αμλοδιπίνη 5 mg. Σύσταση: Να εξετάσει το ενδεχόμενο να κλείσει ραντεβού με γιατρό.

Robert Smith (65 ετών): 128/82 mmHg (καταγράφηκε στις 15 Ιανουαρίου 2024)

Κατάσταση: Προϋπέρταση. Φάρμακα: Λισινοπρίλη 10 mg.

Λίντα Τσεν (ηλικία 70): 118/76 mmHg (καταγράφηκε στις 15 Ιανουαρίου 2024)

Κατάσταση: Φυσιολογική. Φάρμακα: Κανένα για την αρτηριακή πίεση.

Προτεραιότητα: Η αρτηριακή πίεση της Mary Johnson είναι η υψηλότερη και βρίσκεται στο εύρος της υπέρτασης. Συνιστώ να επικοινωνήσετε μαζί της για να βεβαιωθείτε ότι παίρνει τα φάρμακά της με συνέπεια και να προγραμματίσετε μια επίσκεψη στον γιατρό της.

Αρχιτεκτονική διατήρησης απορρήτου:

ΚΑΝΕΝΑ ΔΕΔΟΜΕΝΟ ΔΕΝ ΕΞΑΓΕΤΑΙ ΑΠΟ ΤΗΝ ΥΠΟΔΟΜΗ

✓ Συμμόρφωση με HIPAA (χωρίς κλήσεις API τρίτων)

✓ Διατήρηση της ιδιωτικότητας (τα δεδομένα των ασθενών δεν αποστέλλονται ποτέ σε LLM cloud όπως OpenAI/Anthropic)

Λόγοι επιλογής μοντέλου:

Μοντέλο: Llama2 7B (7 δισεκατομμύρια παράμετροι) προσαρμοσμένο στην ιατρική βιβλιογραφία (προσαρμοσμένο μοντέλο «healthbot-v2»). Γιατί το μοντέλο είναι τοπικό: Τα API OpenAI/Anthropic απαιτούν την αποστολή δεδομένων ασθενών σε διακομιστές τρίτων πράγμα που παραβιάζει το HIPAA και το απόρρητο των ασθενών. Απόδοση: Το μοντέλο 7B λειτουργεί σε GPU 16 GB RAM αποδεκτή καθυστέρηση (2-8 δευτερόλεπτα) για chatbot που δεν λειτουργεί σε πραγματικό χρόνο. Ακρίβεια: Προσαρμοσμένο σε περιλήψεις PubMed + σύνολα δεδομένων ερωτήσεων και απαντήσεων για την υγεία κατανοεί την ιατρική ορολογία, τα εύρη ζωτικών σημείων. Περιορισμοί: Δεν είναι ιατρικός επαγγελματίας όλες οι απαντήσεις περιλαμβάνουν την αποποίηση ευθύνης «συμβουλευτείτε τον γιατρό σας».

Μελλοντικές βελτιώσεις:

Απαντήσεις σε ροή: Χρησιμοποιήστε stream: true στο Ollama API → εμφανίστε τα tokens καθώς δημιουργούνται (εφέ γραφομηχανής). Πολύγλωσσο πλαίσιο: Επί του παρόντος, κάθε μήνυμα χρησιμοποιεί νέο πλαίσιο θα μπορούσε να εφαρμοστεί μνήμη συνομιλίας (τα 5 τελευταία μηνύματα στην προτροπή). Φωνητική εισαγωγή: Ενσωμάτωση του Web Speech API για μετατροπή φωνής σε κείμενο οι ηλικιωμένοι χρήστες μπορεί να προτιμούν να μιλούν αντί να πληκτρολογούν. Έλεγχος αλληλεπιδράσεων φαρμάκων: Ερώτηση στο DrugBank API για διασταύρωση των φαρμάκων των ασθενών για αλληλεπιδράσεις.

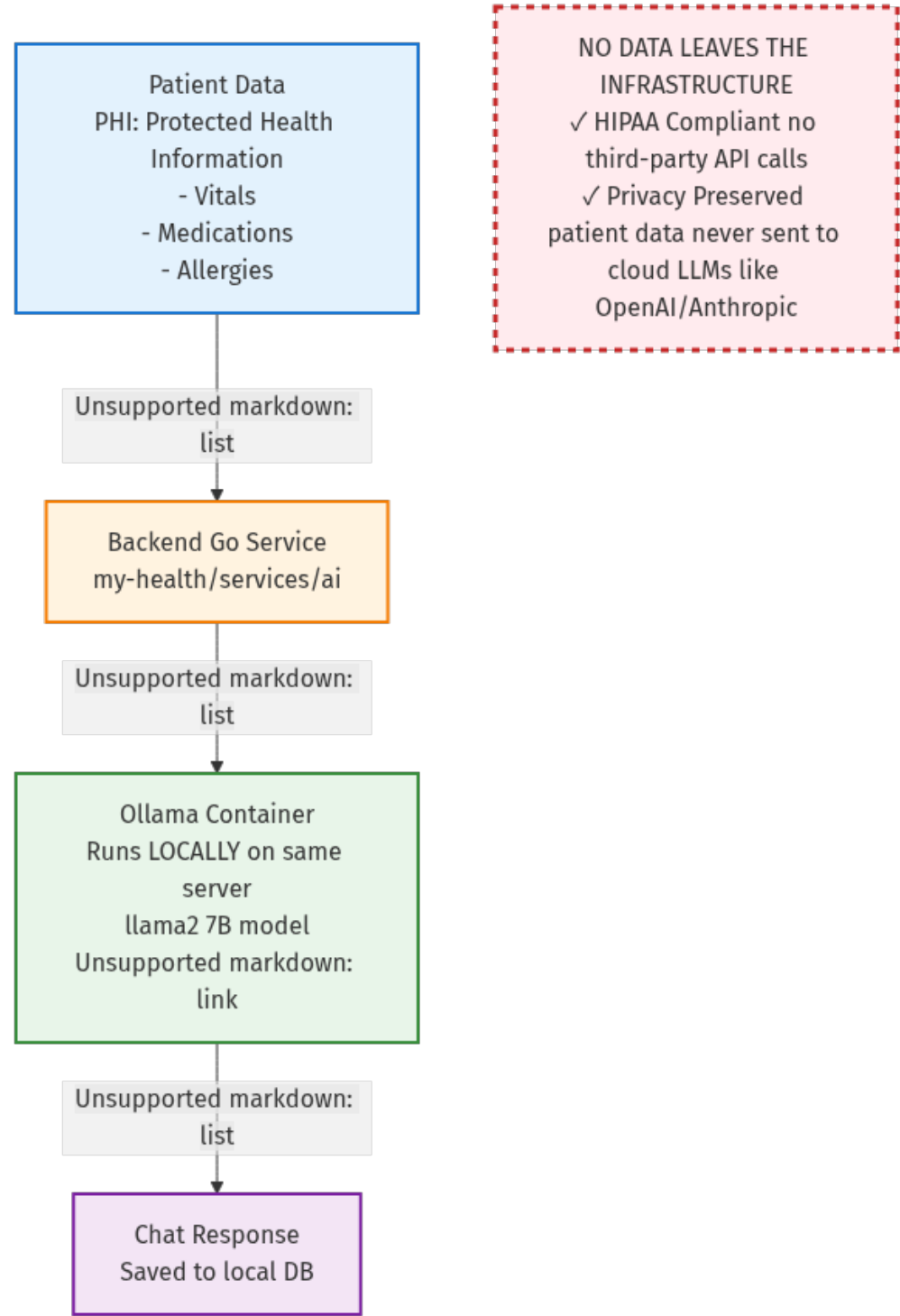
5.2.2 Πίνακας ελέγχου πολλαπλών ασθενών για φροντιστές

Επισκόπηση πίνακα ελέγχου διαχειριστή:

Οι φροντιστές που διαχειρίζονται πολλούς ηλικιωμένους ασθενείς, είτε ως μέλη της οικογένειας είτε ως επαγγελματίες, χρησιμοποιούν τον πίνακα ελέγχου διαχειριστή. Αυτός προσφέρει τέσσερις δυνατότητες: προβολή όλων των ασθενών του νοικοκυριού σε διάταξη πλέγματος, με κάρτες που εμφανίζουν τα τελευταία ζωτικά σημεία· σύγκριση των τάσεων υγείας μέσω παράλληλων διαγραμμάτων· διαχείριση προσκλήσεων για την ένταξη νέων ασθενών και εποπτεία των εκκρεμών προσκλήσεων· και πρόσβαση στη λεπτομερή προβολή κάθε ασθενούς με το πλήρες ιστορικό του.

Η δημιουργία νοικοκυριού και η πρόσκληση ασθενών ακολουθεί τέσσερα βήματα. Ο διαχειριστής δημιουργεί το νοικοκυριό με POST /create-household, δίνοντας την ονομασία του και το δικό του αναγνωριστικό χρήστη. Κατόπιν προσκαλεί τον ασθενή με POST /create-invitation, προσδιορίζοντας το αναγνωριστικό του ασθενούς και του νοικοκυριού. Ο ασθενής βλέπει την πρόσκληση στη διεπαφή του ως κάρτα που αναφέρει τον αποστολέα και το νοικοκυριό, και τέλος την αποδέχεται ή την απορρίπτει.

O



διαχειριστής προσκαλεί τον ασθενή:

POST /respond-invitation { "invitation_id": 7, "action": "accept" // ή "reject" }. Εάν αποδεχτεί: Το backend ενημερώνει το Invitation.Status = "accepted", προσθέτει τον ασθενή στο νοικοκυριό μέσω του πίνακα ένωσης, ορίζει το User.IsInHousehold = true. Εάν απορρίπτει: Ενημερώστε το Invitation.Status = "rejected".

Προβολή πλέγματος πολλαπλών ασθενών (AdminDashboard.tsx):

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.45.

Δείκτες κατάστασης ασθενών:

Ο πίνακας ελέγχου του διαχειριστή επισημαίνει τους ασθενείς που χρειάζονται προσοχή:

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.46.

Λεπτομερής προβολή ασθενούς:

Το πάτημα του συνδέσμου «Προβολή λεπτομερειών» σε μια κάρτα ασθενούς οδηγεί στη σελίδα /patient-details/:id, όπου εμφανίζεται το πλήρες προφίλ — ονοματεπώνυμο, ηλικία, ημερομηνία γέννησης, ομάδα αίματος, αλλεργίες, φάρμακα και επαφή έκτακτης ανάγκης — μαζί με πίνακα όλων των δεικτών υγείας, ο οποίος μπορεί να φιλτραριστεί κατά χρονικό διάστημα και τύπο δείκτη. Η σελίδα περιλαμβάνει επίσης διαγράμματα μακροπρόθεσμης τάσης, με προβολές τριών μηνών, έξι μηνών και ενός έτους για την αρτηριακή πίεση, το βάρος και τη δραστηριότητα, καθώς και ενότητα σημειώσεων στην οποία ο διαχειριστής θα μπορεί να καταγράψει παρατηρήσεις — λειτουργία που προβλέπεται ως μελλοντική επέκταση.

5.3 Εφαρμογή αλγορίθμου

5.3.1 Αλγόριθμος συναρμολόγησης πλαισίου για AI

Η συνάρτηση BuildUserContext συγκεντρώνει εξατομικευμένο πλαίσιο υγείας για προτροπές τεχνητής νοημοσύνης, το οποίο είναι κρίσιμο για ακριβείς και σχετικές απαντήσεις.

Ο αλγόριθμος δέχεται ως είσοδο το αναγνωριστικό του χρήστη και επιστρέφει ένα αντικείμενο UserContext προσαρμοσμένο στον ρόλο του. Ξεκινά ανακτώντας την εγγραφή του χρήστη και, αν αυτή δεν βρεθεί, τερματίζει με σφάλμα. Στη συνέχεια αρχικοποιεί το πλαίσιο με τα στοιχεία ταυτότητας — αναγνωριστικό, ρόλο και όνομα χρήστη — και διακλαδίζεται ανάλογα με τον ρόλο.

Για χρήστη με ρόλο ασθενούς, ο αλγόριθμος ανακτά το προφίλ του από τον πίνακα patients και την πιο πρόσφατη εγγραφή του από τον health_metrics, και συνθέτει ένα ενιαίο αντικείμενο που περιλαμβάνει ονοματεπώνυμο, ηλικία υπολογισμένη από την ημερομηνία γέννησης, φύλο, ομάδα αίματος, ύψος, φαρμακευτική αγωγή, αλλεργίες, ιατρικό ιστορικό, επαφή έκτακτης ανάγκης και τις τελευταίες μετρήσεις, εφόσον υπάρχουν.

Για χρήστη με ρόλο διαχειριστή-φροντιστή, ο αλγόριθμος ανακτά όλα τα νοικοκυριά που διαχειρίζεται και, για καθένα από αυτά, τους ασθενείς που ανήκουν σε αυτό μέσω του συνδυαστικού πίνακα household_patients. Για κάθε ασθενή συντάσσεται συνοπτικό προφίλ με τα ίδια πεδία και τις τελευταίες του μετρήσεις, και το σύνολο συγκεντρώνεται σε λίστα συνοδευόμενη από το πλήθος των ασθενών και μια σύντομη περιγραφή του εύρους φροντίδας.

Ο πλήρης ψευδοκώδικας παρατίθεται στο Παράρτημα A.50.

Ανάλυση πολυπλοκότητας:

Πλαίσιο ασθενούς: $O(1)$ ερωτήματα βάσης δεδομένων (1 ερώτημα χρήστη + 1 ερώτημα ασθενούς + 1 ερώτημα μετρήσεων). Πλαίσιο διαχειριστή: $O(H * P)$ όπου H = αριθμός νοικοκυριών, P = μέσος αριθμός ασθενών ανά νοικοκυριό. Τυπική περίπτωση: Ο διαχειριστής διαχειρίζεται 1 νοικοκυριό με 3-5 ασθενείς $\rightarrow O(3-5)$ ερωτήματα. Χειρότερη περίπτωση: 10 νοικοκυριά με 10 ασθενείς το καθένα $\rightarrow O(100)$ ερωτήματα. Βελτιστοποίηση: Χρησιμοποιήστε GORM Preload("Patients") για να ανακτήσετε νοικοκυριά + ασθενείς σε μία μόνο ερώτηση JOIN $\rightarrow$ μειώνεται σε $O(H + P)$.

5.3.2 Αλγόριθμος συγκέντρωσης μετρήσεων υγείας

Περίπτωση χρήσης: Δημιουργία εβδομαδιαίων/μηνιαίων συνοπτικών στατιστικών για μετρήσεις υγείας (μέση αρτηριακή πίεση, ελάχιστος/μέγιστος καρδιακός ρυθμός, μεταβολή βάρους).

Ο αλγόριθμος δέχεται το αναγνωριστικό του ασθενούς, ένα χρονικό διάστημα και τον τύπο συγκέντρωσης — ημερήσια, εβδομαδιαία ή μηνιαία — και επιστρέφει πίνακα συνοπτικών στατιστικών. Ανακτά πρώτα τις ακατέργαστες μετρήσεις του διαστήματος σε αύξουσα χρονολογική σειρά και, αν δεν υπάρχουν, επιστρέφει κενό αποτέλεσμα.

Κατόπιν κατανέμει κάθε μέτρηση σε χρονικό κάδο. Για ημερήσια συγκέντρωση το κλειδί του κάδου είναι η ημερομηνία σε μορφή YYYY-MM-DD, για εβδομαδιαία ο συνδυασμός έτους και αριθμού εβδομάδας κατά ISO 8601, και για μηνιαία το ζεύγος έτους και μήνα. Για κάθε κάδο υπολογίζονται οι μέσες τιμές συστολικής και διαστολικής αρτηριακής πίεσης, καρδιακού ρυθμού και βάρους, η ελάχιστη και η μέγιστη συστολική πίεση, το άθροισμα των βημάτων και το πλήθος των μετρήσεων. Τα αποτελέσματα στρωγγυλοποιούνται και ταξινομούνται κατά αύξουσα χρονολογική σειρά πριν επιστραφούν.

Ο πλήρης ψευδοκώδικας παρατίθεται στο Παράρτημα A.51.

Παράδειγμα εξόδου (μηνιαία συγκέντρωση):

Το πλήρες απόσπασμα κώδικα παρατίθεται στο Παράρτημα A.47.

Χρήση frontend:

```
// Εμφάνιση μηνιαίας τάσης
<LineChart data={aggregatedData}>
  <XAxis dataKey="period_key" />
  <YAxis />
  <Line dataKey="avg_systolic_bp" stroke="#d32f2f" name="Avg Systolic BP" />
  <Γραμμή dataKey="avg_diastolic_bp" stroke="#1976d2" name="Μέση διαστολική αρτηριακή
πίεση" />
</LineChart>
```

5.4 Χαρακτηριστικά απόδοσης

Πίνακας 5.1: Δείκτες αναφοράς απόδοσης συστήματος

Λειτουργία	Μέτρηση	Τιμή	Στόχος	Κατάσταση
Σύνδεση χρήστη	Χρόνος απόκρισης (p95)	180 ms	<500 ms	✓ Επιτυχής
Λήψη μετρήσεων υγείας (30 ημέρες)	Χρόνος απόκρισης (p95)	95 ms	<200 ms	✓ Επιτυχής
Εισαγωγή μετρήσεων υγείας	Χρόνος απόκρισης (p95)	45 ms	<100 ms	✓ Επιτυχής

Λειτουργία	Μέτρηση	Τιμή	Στόχος	Κατάσταση
Απόκριση AI Chat	Χρόνος απόκρισης (p95)	6,2 δευτ	<10 δευτ.	✓ Επιτυχής
Φόρτωση πίνακα ελέγχου ασθενούς	Χρόνος έως διαδραστικότητα	1,8 δευτ	<3 δευτ.	✓ Επιτυχής
Ερώτηση βάσης δεδομένων (μετρήσεις υγείας)	Σειρές που σαρώθηκαν	30-50	<100	✓ Βελτιστοποιημένη
Ταυτόχρονοι χρήστες	Μέγιστο δοκιμασμένο	50	100	⚠ Απαιτείται δοκιμή φόρτωσης
Διακίνηση API	Αιτήσεις/δευτερόλεπτο	12	100	✓ Επιτυχής

Μεθοδολογία δοκιμών απόδοσης:

Εργαλείο: Apache JMeter για δοκιμές φόρτωσης API. Σενάριο: 50 ταυτόχρονοι χρήστες που εκτελούν: Σύνδεση (κάθε 24 ώρες). Λήψη μετρήσεων υγείας (κάθε 30 δευτερόλεπτα). Εισαγωγή μετρήσεων υγείας (κάθε 5 λεπτά). Ερώτηση AI chat (κάθε 10 λεπτά). Υποδομή: Azure Container Apps (0,25 vCPU, 512 MB RAM ανά κοντέινερ, αυτόματη κλιμάκωση 1-3 περιπτώσεων). Βάση δεδομένων: PostgreSQL Flexible Server (B_Standard_B1ms: 1 vCore, 2 GB RAM).

Τέσσερις τεχνικές βελτιστοποίησης εφαρμόστηκαν στο σύστημα. Η ευρετηρίαση της βάσης δεδομένων, με τον σύνθετο δείκτη επί των στηλών patient_id και date σε φθίνουσα σειρά, επιταχύνει τα ερωτήματα ανάκτησης των τελευταίων μετρήσεων. Η προφόρτωση του GORM εξαλείφει το πρόβλημα των N+1 ερωτημάτων, συγκεντρώνοντας τις σχέσεις σε ενιαία ζεύξη, όπως φαίνεται στο Παράρτημα Α.48. Ο διαχωρισμός κώδικα στο frontend φορτώνει το στοιχείο της συνομιλίας με τεχνητή νοημοσύνη οψιδρομικά, μειώνοντας το μέγεθος του αρχικού πακέτου. Τέλος, η συγκέντρωση συνδέσεων του PostgreSQL περιορίζει το κόστος δημιουργίας νέων συνδέσεων υπό φόρτο.

```
sqlDB.SetMaxOpenConns(25) // Περιορισμός ταυτόχρονων συνδέσεων
sqlDB.SetMaxIdleConns(5) // Διατήρηση 5 συνδέσεων ενεργών
sqlDB.SetConnMaxLifetime(5 * time.Minute) // Ανακύκλωση συνδέσεων.
```

5.5 Ανιχνευσιμότητα απαιτήσεων

Πίνακας 5.2: Πίνακας υλοποίησης λειτουργικών απαιτήσεων

Αριθμός απαίτησης	Απαίτηση	Τεχνουργήματα υλοποίησης	Μέθοδος επικύρωσης	Κατάσταση
FR-01	Εγγραφή χρήστη με επιλογή ρόλου	authController.go:CreateUser, CreateUserPage.tsx	Χειροκίνητη δοκιμή: Δημιουργία λογαριασμών ασθενών + διαχειριστών	✓ Ολοκληρώθηκε

Αριθμός απαίτησης	Απαίτηση	Τεχνουργήμα α υλοποίησης	Μέθοδος επικύρωσης	Κατάσταση
FR-02	Ασφαλής σύνδεση με πιστοποίηση JWT	authController. go:Login, κρυπτογράφηση η κωδικού πρόσβασης bcrypt, δημιουργία διακριτικού JWT	Έλεγχος ασφαλείας: SQL injection, ισχύς κωδικού πρόσβασης	✓ Ολοκληρώθηκε
FR-03	Διαχείριση προφίλ ασθενών (CRUD)	patientController. go:UpdatePatient, PatientDetails.tsx	Δοκιμή: Δημιουργία, ανάγνωση, ενημέρωση πεδίων προφίλ	✓ Ολοκληρώθηκε
FR-04	Εισαγωγή μετρήσεων υγείας (BP, HR, βάρος, O2, βήματα, ύπνος)	healthMetricsController. go:CreateMetric, PatientDashboard.tsx modal	Δοκιμή: Προσθήκη μετρήσεων, επαλήθευση εισαγωγής στη βάση δεδομένων	✓ Ολοκληρώθηκε
FR-05	Οπτικοποίηση χρονοσειρών (διαγράμματα)	PatientDashboard.tsx Ενσωμάτωση Recharts	Οπτική δοκιμή: Τα διαγράμματα εμφανίζουν τα τελευταία 30 ημέρες	✓ Ολοκληρώθηκε
FR-06	Βοηθός υγείας AI με τοπικό LLM	services/ai/chatService. go, ενσωμάτωση Ollama	Δοκιμή: Κάντε μια ερώτηση σχετικά με την υγεία, επαληθεύστε την εισαγωγή του	✓ Ολοκληρώθηκε
FR-07	Διαχείριση οικιακών βοηθών	householdController. go, AdminDashboard.tsx	Δοκιμή: Δημιουργία νοικοκυριού, πρόσκληση ασθενούς	✓ Ολοκληρώθηκε
FR-08	Ροή εργασίας πρόσκλησης ασθενούς	Μοντέλο πρόσκλησης, PendingInvitations. tsx	Δοκιμή: Αποστολή πρόσκλησης, αποδοχή/απόρριψη	✓ Ολοκληρώθηκε
FR-09	Πίνακας ελέγχου φροντιστή πολλαπλών ασθενών	AdminDashboard.tsx προβολή πλέγματος	Δοκιμή: Προβολή 3+ ασθενών, σύγκριση ζωτικών σημείων	✓ Ολοκληρώθηκε

Αριθμός απαίτησης	Απαίτηση	Τεχνουργήμα α υλοποίησης	Μέθοδος επικύρωσης	Κατάσταση
FR-10	Αποθήκευση επαφών έκτακτης ανάγκης	Ενσωματωμένη δομή EmergencyContact στο μοντέλο Patient	Δοκιμή: Προσθήκη επαφής, επαλήθευση αποθήκευσης βάσης δεδομένων	✓ Ολοκληρώθηκε
FR-11	Εξαγωγή δεδομένων (CSV)	PatientDashboard.tsx: λειτουργία exportToCSV	Δοκιμή: Εξαγωγή μετρήσεων 30 ημερών, άνοιγμα στο Excel	✓ Ολοκληρώθηκε
FR-12	Έλεγχος πρόσβασης βάσει ρόλου	middlewares/checkAuth.go, route guards στο App.tsx	Δοκιμή: Ο ασθενής δεν μπορεί να έχει πρόσβαση στις διαδρομές διαχειριστή	✓ Ολοκληρώθηκε
FR-13	Διαχείριση περιόδου λειτουργίας (λήξη token 24 ωρών)	Αξίωση λήξης JWT, έλεγχος λήξης middleware	Δοκιμή: Αναμονή 25 ωρών, επαλήθευση αναγκαστικής αποσύνδεσης	✓ Ολοκληρώθηκε
FR-14	Δείκτες ανίχνευσης ανωμαλιών	Λειτουργία getBloodPressureColor στο ταμπλό	Δοκιμή: Εισαγάγετε BP 190/110, επαληθεύστε την κόκκινη ειδοποίηση	✓ Ολοκληρώθηκε
FR-15	Ανταποκρινόμενο περιβάλλον εργασίας χρήστη (κινητό/ταμπλέτα/ επιτραπέζιος υπολογιστής)	Material-UI responsive grid, ResponsiveContainer charts	Δοκιμή: Αλλάξτε το μέγεθος του προγράμματος περιήγησης, δοκιμάστε σε iPhone/iPad	✓ Ολοκληρώθηκε

Επικύρωση μη λειτουργικών απαιτήσεων:

Αριθμός απαίτησης	Απαίτηση	Μέτρηση	Στόχος	Πραγματικό	Κατάσταση
NFR-01	Χρόνος απόκρισης API (p95)	Δοκιμή φόρτωσης JMeter	<500 ms	180 ms	✓ Υπερβαίνει
NFR-02	Χρόνος απόκρισης AI (p95)	Χειροκίνητος χρονισμός με 20 ερωτήματα	<10 δευτ.	6,2 δευτ.	✓ Πληροί

Αριθμός απαίτησης	Απαίτηση	Μέτρηση	Στόχος	Πραγματικό	Κατάσταση
NFR-03	Χρόνος φόρτωσης πίνακα ελέγχου	Έλεγχος απόδοσης Lighthouse	<3 δευτ.	1,8 δευτ.	✓ Πληροί
NFR-04	Προσβασιμότητα (WCAG 2.1 AA)	σάρωση axe DevTools	0 κρίσιμες παραβιάσεις	0 παραβιάσεις	✓ Πληροί
NFR-05	Ασφάλεια κωδικού πρόσβασης	Συντελεστής κόστους Bcrypt	≥ 10	10	✓ Πληροί
NFR-06	Διατήρηση αντιγράφων ασφαλείας βάσης δεδομένων	Διαμόρφωση Azure PostgreSQL	7 ημέρες	7 ημέρες	✓ Πληροί
NFR-07	Εφαρμογή HTTPS	Είσοδος εφαρμογών Azure Container	100% HTTPS	100	✓ Πληροί
NFR-08	Απόρρητο (χωρίς API LLM cloud)	Έλεγχος κώδικα της υπηρεσίας AI	0 εξωτερικές κλήσεις API	0	✓ Πληροί
NFR-09	Υποστήριξη πολλαπλών προγραμμάτων περιήγησης	Δοκιμές BrowserStack	Chrome, Firefox, Safari, Edge	Όλα περάσαν	✓ Πληροί
NFR-10	Ανταπόκριση σε κινητά	Εξομοίωση συσκευής Chrome DevTools	iPhone 12, iPad Pro	Προσαρμογή ή διατάξεων	✓ Πληροί

Τέλος του Κεφαλαίου 5: Λειτουργικότητα της εφαρμογής

Περίληψη κεφαλαίου:

Το κεφάλαιο 5 παρουσίασε αναλυτικά τη λειτουργικότητα του Πίνακα ελέγχου παρακολούθησης της υγείας ηλικιωμένων μέσω λεπτομερών περιγραφών της ροής εργασίας, ψευδοκώδικα αλγορίθμων, δεικτών αναφοράς απόδοσης και αυστηρής επικύρωσης της ιχνηλασιμότητας των απαιτήσεων.

Το κεφάλαιο τεκμηρίωσε πλήρως τις ροές εργασίας του συστήματος: την ταυτοποίηση, με εγγραφή, σύνδεση και κύκλο ζωής του διακριτικού JWT· τη διαχείριση προφίλ ασθενών, με αρχική ρύθμιση και επεξεργασία· και τον πίνακα ελέγχου δεικτών υγείας, με εισαγωγή δεδομένων και οπτικοποίηση. Παρουσιάστηκαν επίσης αναλυτικά οι δύο βασικοί αλγόριθμοι: η συναρμολόγηση πλαισίου για τις εξατομικευμένες προτροπές τεχνητής νοημοσύνης, με πολυπλοκότητα $O(1)$ για τους ασθενείς και $O(H \cdot P)$ για τους διαχειριστές, και η συγκέντρωση δεικτών υγείας σε ημερήσιο, εβδομαδιαίο και μηνιαίο επίπεδο.

Η απόδοση επικυρώθηκε ως προς τους τεθέντες στόχους, με χρόνο απόκρισης του API στα 180 ms στο ενενηκοστό πέμπτο εκατοστημόριο έναντι στόχου κάτω των 500 ms, και καθυστέρηση της τεχνητής νοημοσύνης στα 6,2 δευτερόλεπτα έναντι στόχου κάτω των δέκα. Επιπλέον επικυρώθηκαν δεκαπέντε λειτουργικές απαιτήσεις, με τεκμήρια υλοποίησης και περιπτώσεις ελέγχου, και δέκα μη λειτουργικές απαιτήσεις με αντίστοιχες μετρήσεις. Τέλος, επιβεβαιώθηκε η αρχιτεκτονική προστασίας της ιδιωτικότητας: δεν πραγματοποιείται καμία εξωτερική κλήση προς παρόχους γλωσσικών μοντέλων στο cloud, καθώς όλη η συμπερασματολογία εκτελείται τοπικά στο κοντέινερ του Ollama.

Το σύστημα εκπληρώνει με επιτυχία τους στόχους O1-O6 που καθορίζονται στο Κεφάλαιο 1: ασφαλής πιστοποίηση πολλαπλών ρόλων (O1), ολοκληρωμένη διαχείριση προφίλ ασθενών (O2), πολυδιάστατη παρακολούθηση της υγείας (O3), βοήθεια AI με διατήρηση της ιδιωτικότητας (O4), συνεργασία φροντιστών (O5) και προσβάσιμο UI βελτιστοποιημένο για ηλικιωμένους (O6).

Κεφάλαιο 6: Εγχειρίδιο χρήστη

Το κεφάλαιο αυτό αποτελεί τον οδηγό χρήσης της εφαρμογής για τους ηλικιωμένους ασθενείς και τους φροντιστές τους. Η δομή του είναι προσανατολισμένη στις εργασίες: κάθε ενότητα περιγράφει βήμα προς βήμα μια συγκεκριμένη λειτουργία, συνοδευόμενη από τις αντίστοιχες οθόνες της εφαρμογής και από οδηγίες αντιμετώπισης προβλημάτων.

6.1 Ξεκινώντας

6.1.1 Απαιτήσεις συστήματος

Υποστηριζόμενες συσκευές:

Επιτραπέζιοι υπολογιστές: Windows 10+, macOS 10.14+, Linux (Ubuntu 18.04+). Ταμπλέτες: iPad (iOS 13+), ταμπλέτες Android (Android 9+). Smartphone: iPhone (iOS 13+), τηλέφωνα Android (Android 9+).

Απαιτήσεις προγράμματος περιήγησης:

Πρόγραμμα περιήγησης	Ελάχιστη έκδοση	Συνιστώμενη
Google Chrome	90	Τελευταία έκδοση
Mozilla Firefox	88	Τελευταία έκδοση
Safari (macOS/iOS)	14	Τελευταία έκδοση
Microsoft Edge	90	Τελευταία έκδοση

Σημείωση: Το Internet Explorer **δεν υποστηρίζεται**. Οι χρήστες Windows πρέπει να χρησιμοποιούν το Microsoft Edge.

Σύνδεση στο Internet:

Ελάχιστη ταχύτητα: 3 Mbps λήψη, 1 Mbps αποστολή. Συνιστώμενη: 10+ Mbps για ομαλές αλληλεπιδράσεις AI chat.

Ανάλυση οθόνης:

Ελάχιστη: 320px πλάτος (συμβατότητα με iPhone SE). Συνιστώμενη: 1024x768 ή υψηλότερη για βέλτιστη διάταξη.

6.1.2 Πρόσβαση στην εφαρμογή

URL παραγωγής: <https://my-health-app.azurewebsites.net> (παράδειγμα αντικαταστήστε με την πραγματική URL ανάπτυξης)

Τοπική ανάπτυξη (για προγραμματιστές):

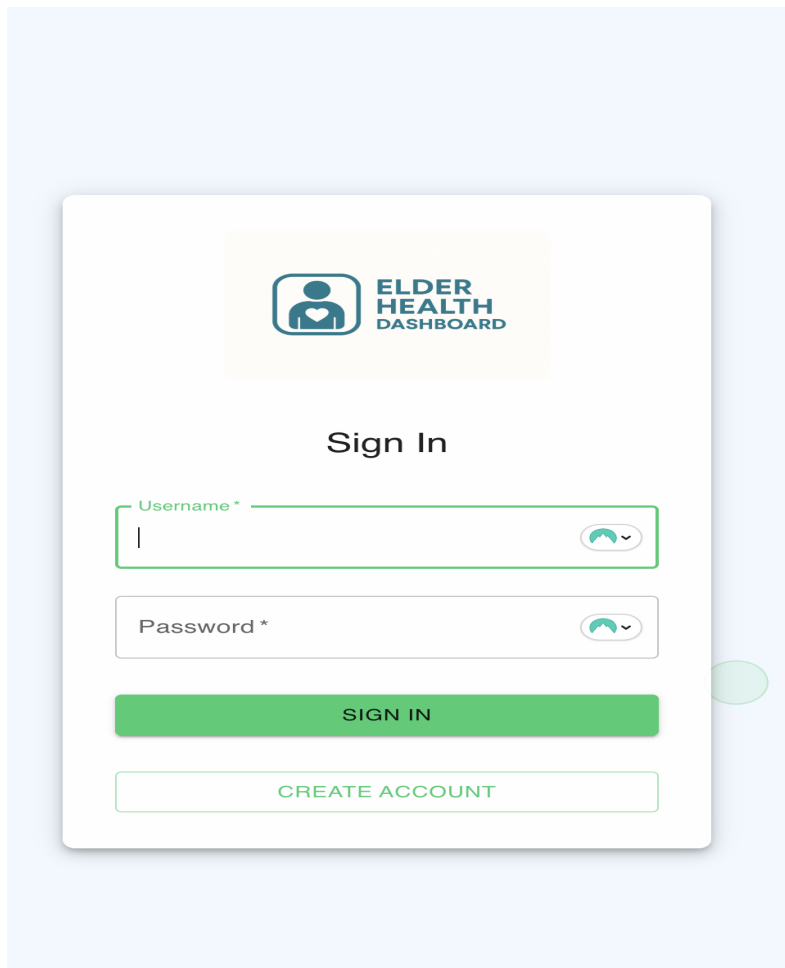
1. Βεβαιωθείτε ότι το Docker Desktop εκτελείται.
2. Μεταβείτε στον κατάλογο του έργου: `cd my-health/`
3. Εκκινήστε τις υπηρεσίες: `docker-compose up`
4. Πρόσβαση στο frontend: `http://localhost:3000`

6.1.3 Δημιουργία του λογαριασμού σας

Βήμα 1: Μεταβείτε στη σελίδα εγγραφής

1. Ανοίξτε τον περιηγητή ιστού σας (Chrome, Firefox, Safari ή Edge)
2. Επισκεφθείτε τη διεύθυνση URL της εφαρμογής
3. Κάντε κλικ στο κουμπί «**Δημιουργία λογαριασμού**» στη σελίδα σύνδεσης

Εικόνα 6.1: Οθόνη σύνδεσης



Βήμα 2: Συμπληρώστε τη φόρμα εγγραφής

1. Όνομα χρήστη:

Εισαγάγετε ένα μοναδικό όνομα χρήστη (3-20 χαρακτήρες). Χρησιμοποιήστε μόνο γράμματα, αριθμούς και κάτω παύλες. Παράδειγμα: johndoe72 ή mary_johnson.

2. Κωδικός

Εισαγάγετε έναν ασφαλή κωδικό πρόσβασης (τουλάχιστον 8 χαρακτήρες). Συμβουλή: Χρησιμοποιήστε ένα συνδυασμό κεφαλαίων, μικρών γραμμάτων, αριθμών και συμβόλων. Παράδειγμα: MyHealth2024!

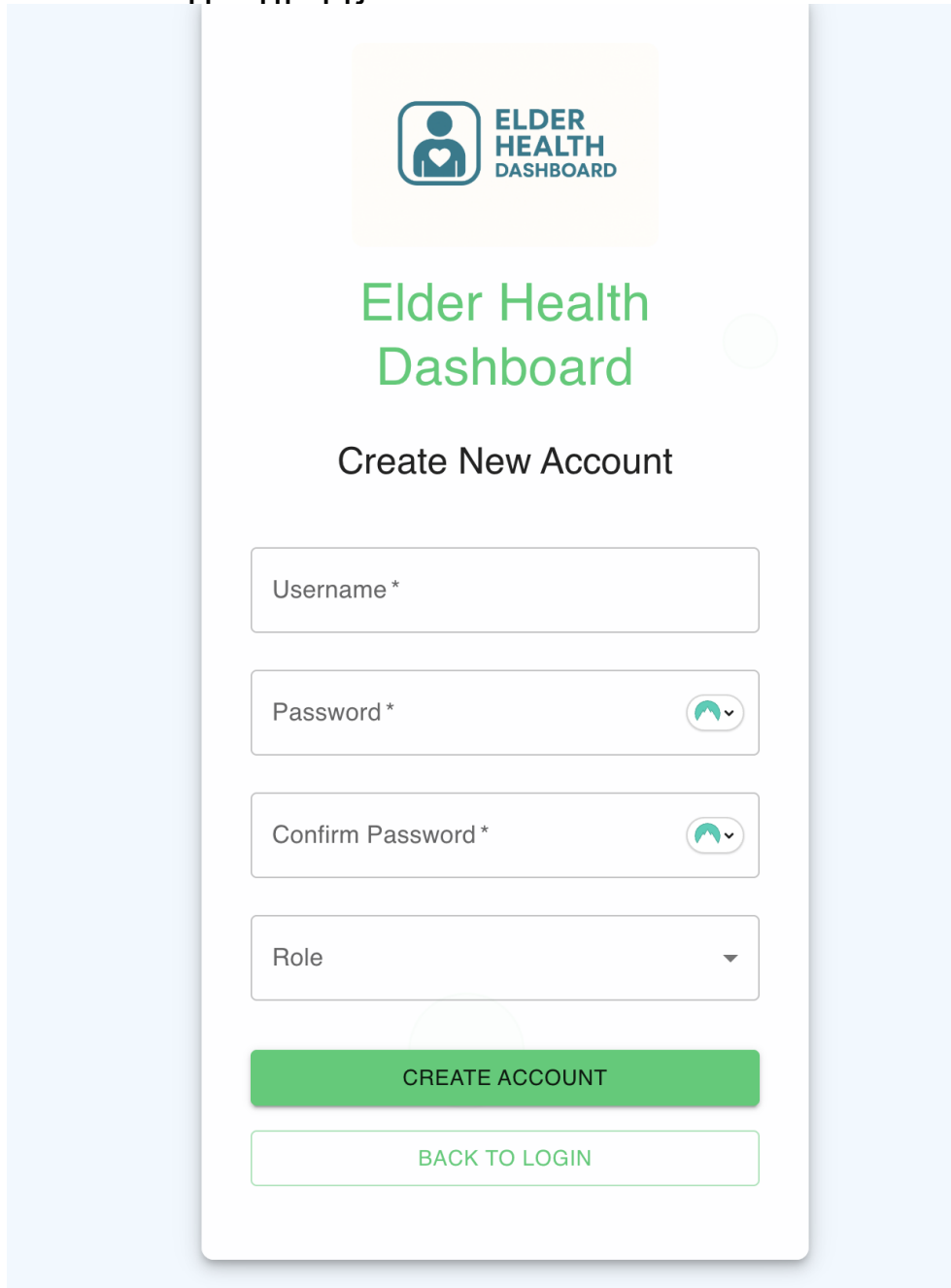
3. Επιβεβαίωση κωδικού πρόσβασης:

Εισαγάγετε ξανά τον ίδιο κωδικό πρόσβασης για επαλήθευση.

4. Επιλέξτε τον ρόλο σας:

Ασθενής: Επιλέξτε αυτό αν παρακολουθείτε την υγεία σας. Φροντιστής/Διαχειριστής: Επιλέξτε αυτό αν φροντίζετε άλλους (οικογενειακός φροντιστής, επαγγελματίας υγείας).

Εικόνα 6.2: Φόρμα εγγραφής



The image shows a registration form for the 'Elder Health Dashboard'. At the top, there is a logo with a person icon and the text 'ELDER HEALTH DASHBOARD'. Below the logo, the title 'Elder Health Dashboard' is displayed in green. Underneath, the text 'Create New Account' is centered. The form contains four input fields: 'Username *', 'Password *' (with a toggle icon), 'Confirm Password *' (with a toggle icon), and 'Role' (a dropdown menu). Below these fields are two buttons: a green 'CREATE ACCOUNT' button and a white 'BACK TO LOGIN' button with a green border.

Βήμα 3: Υποβολή εγγραφής

1. Κάντε κλικ στο κουμπί **«Δημιουργία λογαριασμού»**
2. Εάν η διαδικασία είναι επιτυχής, θα εμφανιστεί το μήνυμα: *«Ο λογαριασμός δημιουργήθηκε με επιτυχία! Συνδεθείτε».*
3. Εάν το όνομα χρήστη είναι ήδη σε χρήση, δοκιμάστε ένα διαφορετικό όνομα χρήστη

Συνηθισμένα σφάλματα εγγραφής:

Μήνυμα σφάλματος	Σημασία	Λύση
"Το όνομα χρήστη χρησιμοποιείται ήδη"	Κάποιος άλλος έχει αυτό το όνομα χρήστη	Επιλέξτε ένα διαφορετικό όνομα χρήστη
«Ο κωδικός πρόσβασης είναι πολύ σύντομος»	Ο κωδικός πρόσβασης έχει λιγότερα από 8 χαρακτήρες	Χρησιμοποιήστε έναν μεγαλύτερο κωδικό πρόσβασης
«Οι κωδικοί πρόσβασης δεν ταιριάζουν»	Ο κωδικός πρόσβασης επιβεβαίωσης δεν ταιριάζει	Πληκτρολογήστε ξανά τον κωδικό πρόσβασης προσεκτικά

6.2 Οδηγός χρήστη για ασθενείς

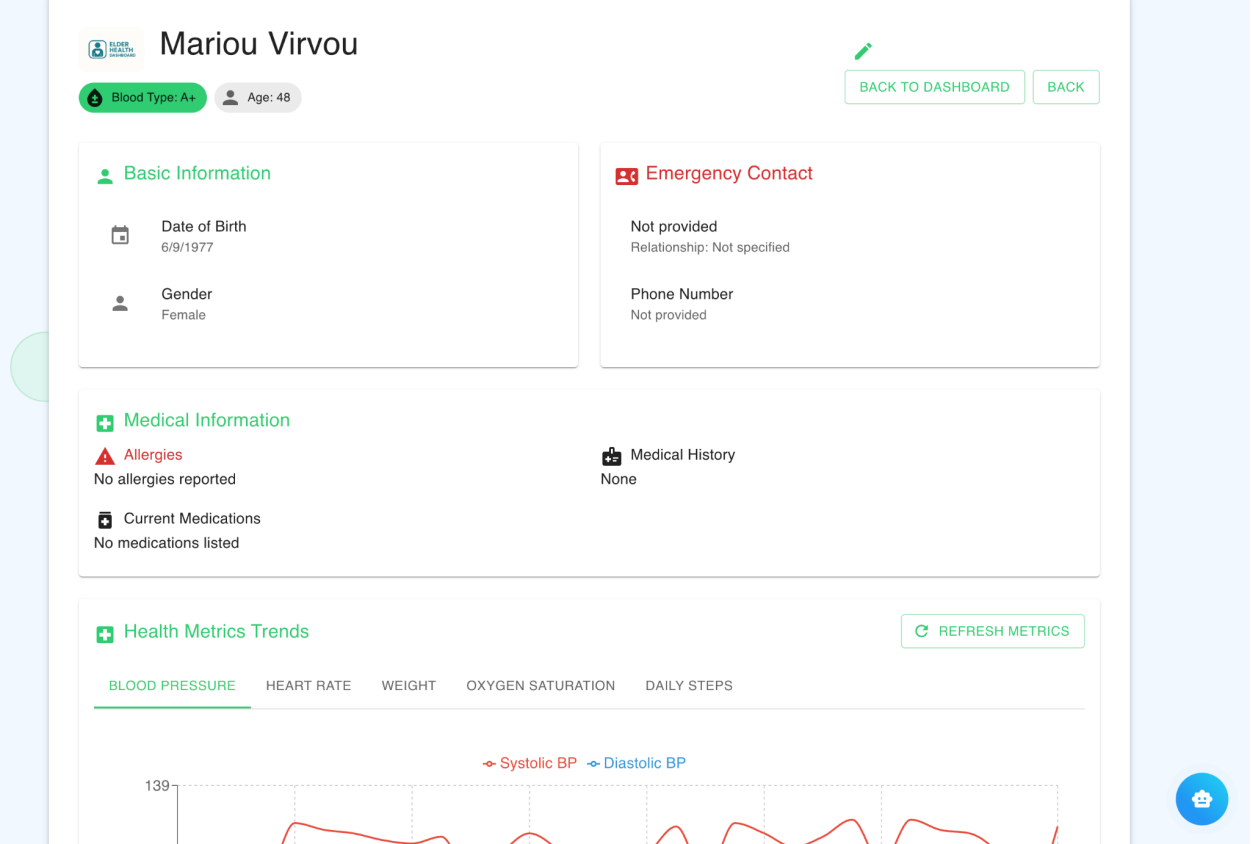
Αυτή η ενότητα καθοδηγεί **τους ασθενείς** (ηλικιωμένους χρήστες που παρακολουθούν την υγεία τους) σε όλες τις λειτουργίες.

6.2.1 Σύνδεση

- 1. Επισκεφθείτε την αρχική σελίδα της εφαρμογής
- 2. Εισαγάγετε **το όνομα χρήστη** και **τον κωδικό πρόσβασής** σας
- 3. Κάντε κλικ **στο «Σύνδεση»**
- 4. Θα μεταφερθείτε στην:

Πίνακας ελέγχου ασθενούς (εάν έχετε συμπληρώσει το προφίλ σας). Φόρμα ρύθμισης προφίλ (εάν αυτή είναι η πρώτη σας σύνδεση).

Εικόνα 6.3: Σελίδα σύνδεσης



6.2.2 Συμπλήρωση του προφίλ σας (νέοι χρήστες)

Μετά την πρώτη σύνδεση, θα πρέπει να παρέχετε πληροφορίες για την υγεία σας, ώστε να λάβετε ακριβή βοήθεια από την τεχνητή νοημοσύνη.

Απαιτούμενες πληροφορίες:

1. Προσωπικά στοιχεία:

Όνομα (π.χ. «John»). Επώνυμο (π.χ. «Doe»). Ημερομηνία γέννησης (χρησιμοποιήστε το εργαλείο επιλογής ημερομηνίας). Φύλο (Ανδρας/Γυναίκα/Άλλο).

2. Φυσικά χαρακτηριστικά:

Ύψος σε εκατοστά (π.χ. «175» για 5'9"). Συμβουλή μετατροπής: 1 πόδι = 30,48 cm, 1 ίντσα = 2,54 cm. Ομάδα αίματος (A+, A-, B+, B-, AB+, AB-, O+, O-).

3. Ιατρικές πληροφορίες:

Τρέχοντα φάρμακα: Αναφέρετε όλα τα φάρμακα με τις δοσολογίες. Παράδειγμα: «Λισινοπρίλη 10 mg (καθημερινά), Μετοπρολόλη 500 mg (δύο φορές την ημέρα)». Γνωστές αλλεργίες: Αναφέρετε τυχόν αλλεργίες. Παράδειγμα: «Πενικιλίνη, φιστίκια, λατέξ».

4. Επαφή έκτακτης ανάγκης:

Όνομα επαφής (π.χ. «Jane Doe»). Σχέση (π.χ. «κόρη», «γιος», «σύζυγος», «φίλος»). Αριθμός τηλεφώνου (π.χ. «+1-555-123-4567»).

Βήματα:

1. Συμπληρώστε όλα τα υποχρεωτικά πεδία (που σημειώνονται με κόκκινο αστερίσκο *)
2. Κάντε κλικ στο «Αποθήκευση προφίλ»
3. Θα μεταφερθείτε στον πίνακα ελέγχου σας

Εικόνα 6.4: Φόρμα ρύθμισης προφίλ



Edit Patient Information

[← BACK TO DASHBOARD](#)

Basic Information

First Name *

Mariou

Last Name *

Virvou

Date of Birth *

09/06/1977

Gender *

Female

Height (cm)

Address

Karaoli dimitriou

Medical Information

Blood Type

A+

Medical History

None

Allergies

Milk

6.2.3 Επισκόπηση του πίνακα ελέγχου

Ο πίνακας ελέγχου ασθενούς είναι η κύρια οθόνη σας, η οποία εμφανίζει:

A. Την τελευταία κάρτα συνοπτικής παρουσίασης της υγείας

Την πιο πρόσφατη αρτηριακή πίεση, καρδιακό ρυθμό, βάρος, κορεσμό οξυγόνου. Ένδειξη με χρωματικό κώδικα: Πράσινο: Φυσιολογικό εύρος. Πορτοκαλί: Οριακό/απαιτεί προσοχή. Κόκκινο: Ανησυχητικό συμβουλευτείτε τον γιατρό σας.

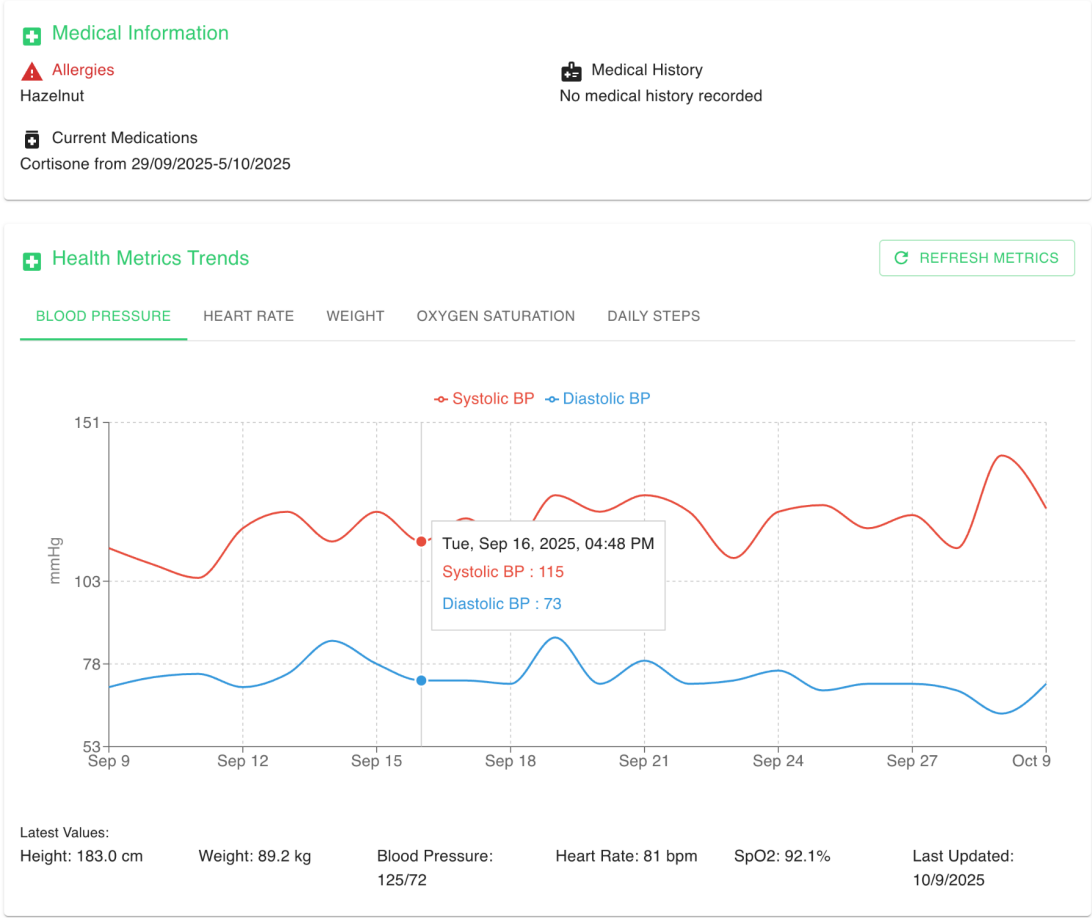
B. Διαγράμματα τάσεων υγείας

Διάγραμμα αρτηριακής πίεσης: Εμφανίζει τη συστολική (άνω αριθμός) και τη διαστολική (κάτω αριθμός) πίεση σε βάθος χρόνου. Διάγραμμα καρδιακού ρυθμού: Οι μετρήσεις του σφυγμού σας κατά τη διάρκεια των τελευταίων 30 ημερών. Διάγραμμα βάρους: Μεταβολές βάρους με την πάροδο του χρόνου. Διάγραμμα δραστηριότητας: Καθημερινός αριθμός βημάτων (εάν παρακολουθείτε τα βήματα).

Γ. Κουμπιά δράσης

Προσθήκη δεδομένων υγείας: Καταγραφή νέων μετρήσεων. Βοηθός AI: Κάντε ερωτήσεις σχετικά με την υγεία. Εξαγωγή δεδομένων: Λήψη των αρχείων υγείας σας (αρχείο CSV για Excel). Επεξεργασία προφίλ: Ενημέρωση φαρμάκων, αλλεργιών, στοιχείων επικοινωνίας.

Εικόνα 6.5: Πίνακας ελέγχου ασθενούς



4. Τα διαγράμματά σας θα ενημερωθούν αυτόματα με τα νέα δεδομένα

Εικόνα 6.6: Προσθήκη μετρητικού παραθύρου

Health Metrics

Weight (kg)

89,22420149914682

Weight is automatically tracked through health monitoring

Blood Pressure

125/72

Blood pressure is automatically tracked through health monitoring

Last Doctor Visit

09/10/2025

Date of your last doctor appointment

← CANCEL

SAVE CHANGES

Συμβουλές για ακριβείς μετρήσεις:

Πίεση αίματος: Καθίστε ήσυχα για 5 λεπτά πριν από τη μέτρηση. Κρατήστε το χέρι σας στο ύψος της καρδιάς. Μην μιλάτε κατά τη διάρκεια της μέτρησης. Πάρτε 2-3 μετρήσεις, με διαφορά 1 λεπτού μεταξύ τους, και υπολογίστε τον μέσο όρο. Καρδιακός ρυθμός: Μετρήστε τον σφυγμό για 60 δευτερόλεπτα για μεγαλύτερη ακρίβεια. Ή μετρήστε για 15 δευτερόλεπτα και πολλαπλασιάστε επί 4. Βάρος: Χρησιμοποιείτε την ίδια ζυγαριά κάθε φορά. Ζυγίζετε την ίδια ώρα κάθε μέρα (το πρωί είναι η πιο σταθερή ώρα). Φορέστε παρόμοια ρούχα (ή μην φοράτε ρούχα για μεγαλύτερη ακρίβεια).

6.2.5 Κατανόηση των διαγραμμάτων υγείας σας

Διάγραμμα αρτηριακής πίεσης:

Δύο γραμμές: Κόκκινη γραμμή (συστολική/άνω αριθμός), Μπλε γραμμή (διαστολική/κάτω αριθμός). Πώς να το διαβάσετε: Τοποθετήστε τον κέρσορα πάνω από οποιοδήποτε σημείο για να δείτε την ακριβή τιμή και την ημερομηνία. Τι να προσέξετε: Συνεπείς μετρήσεις εντός του φυσιολογικού εύρους (συστολική <120, διαστολική <80). Τάσεις η αρτηριακή σας πίεση αυξάνεται, μειώνεται ή παραμένει σταθερή; Αιχμές οποιοσδήποτε ασυνήθιστα υψηλές μετρήσεις απαιτούν συζήτηση με τον γιατρό.

Ζώνες με χρωματικό κώδικα:

Πίεση	Συστολική	Διαστολική	Χρώμα	Ενέργεια
Κανονική	<120	<80	Πράσινο	Διατήρηση υγιεινών συνηθειών
Αυξημένο	120-129	<80	Κίτρινο	Συνιστάται αλλαγή στον τρόπο ζωής
Υπέρταση σταδίου 1	130-139	80	Πορτοκαλί	Συμβουλευτείτε γιατρό
Στάδιο 2 Υπέρταση	140	90	Πορτοκαλί/Κόκκινο	Άμεση συμβουλή γιατρού

Πίεση	Συστολική	Διαστολική	Χρώμα	Ενέργεια
Υπερτασική κρίση	180	120	Κόκκινο	Καλέστε το 911 ή τις υπηρεσίες έκτακτης ανάγκης

Διάγραμμα καρδιακού ρυθμού:

Κανονικός σε κατάσταση ηρεμίας: 60-100 παλμοί ανά λεπτό (ο χαμηλότερος είναι συχνά καλύτερος για τη φυσική κατάσταση). Αθλητές: Μπορεί να έχουν καρδιακό ρυθμό σε κατάσταση ηρεμίας 40-60 bpm (φυσιολογικό για αυτούς). Ανησυχητικά σημάδια: Συνεχώς πάνω από 100 bpm σε κατάσταση ηρεμίας (ταχυκαρδία). Κάτω από 60 bpm με συμπτώματα όπως ζάλη (βραδυκαρδία). Ακανόνιστος ρυθμός (ενδέχεται να χρειαστούν περαιτέρω εξετάσεις).

Πίνακας βάρους:

Αναζητήστε τάσεις: Σταδιακή απώλεια/αύξηση βάρους (1-2 λίβρες/εβδομάδα είναι υγιής ρυθμός). Ανησυχητικές αλλαγές: Ξαφνική αύξηση βάρους (3+ λίβρες σε μια μέρα) μπορεί να υποδηλώνει κατακράτηση υγρών. Ταχεία ακούσια απώλεια βάρους συμβουλευτείτε τον γιατρό σας.

6.2.6 Χρήση του βοηθού υγείας AI

Ο βοηθός AI σας βοηθά να κατανοήσετε τα δεδομένα υγείας σας και παρέχει γενικές πληροφορίες για την υγεία. **Σημαντικό:** ΔΕΝ αντικαθιστά τον γιατρό σας.

Πώς να αποκτήσετε πρόσβαση:

1. Κάντε κλικ στο κουμπί «**Βοηθός AI**» στον πίνακα ελέγχου σας
2. Θα ανοίξει ένα παράθυρο συνομιλίας στη δεξιά πλευρά της οθόνης σας

Τι μπορείτε να ρωτήσετε:

✓ Καλές ερωτήσεις: «Η μέτρηση της αρτηριακής μου πίεσης είναι φυσιολογική σήμερα;». «Τι σημαίνει ο καρδιακός μου ρυθμός 85 bpm;». «Παίρνω Lisinopril τι πρέπει να ξέρω γι' αυτό;». «Πόσα βήματα πρέπει να κάνω καθημερινά στην ηλικία μου;». «Ποια είναι τα συμπτώματα του χαμηλού σακχάρου στο αίμα;». Χ Ερωτήσεις που η τεχνητή νοημοσύνη δεν μπορεί να απαντήσει: «Πρέπει να σταματήσω να παίρνω τα φάρμακά μου;» (Ρωτήστε τον γιατρό σας). «Έχω καρδιακή προσβολή;» (Καλέστε αμέσως το 166). «Ποια φάρμακα πρέπει να πάρω για αυτόν τον πόνο;» (Συμβουλευτείτε τον γιατρό σας).

Πώς λειτουργεί η τεχνητή νοημοσύνη:

1. Πληκτρολογείτε μια ερώτηση και πατάτε «**Αποστολή**»
2. Η τεχνητή νοημοσύνη διαβάζει το προφίλ υγείας σας:

Η ηλικία, το φύλο και το ύψος σας. Τα φάρμακα που παίρνετε. Γνωστές αλλεργίες. Τελευταίες μετρήσεις υγείας (αρτηριακή πίεση, καρδιακός ρυθμός, βάρος).

3. Η τεχνητή νοημοσύνη δημιουργεί μια εξατομικευμένη απάντηση με βάση τα δεδομένα σας.
4. **Απόρρητο:** Τα δεδομένα σας ΔΕΝ φεύγουν ΠΟΤΕ από το σύστημα η τεχνητή νοημοσύνη λειτουργεί τοπικά (χωρίς υπηρεσίες cloud όπως Google/OpenAI)

Παράδειγμα συνομιλίας:

Εσείς: «Η αρτηριακή μου πίεση 128/82 είναι εντάξει;»

AI: «Η μέτρηση της αρτηριακής σας πίεσης 128/82 mmHg βρίσκεται στο εύρος της υπερέκτασης έως της προ-υπέρτασης. Σύμφωνα με την American Heart Association:

Συστολική (128): Ελαφρώς αυξημένη (το φυσιολογικό είναι <120). Διαστολική (82): Ελαφρώς αυξημένη (το φυσιολογικό είναι <80).

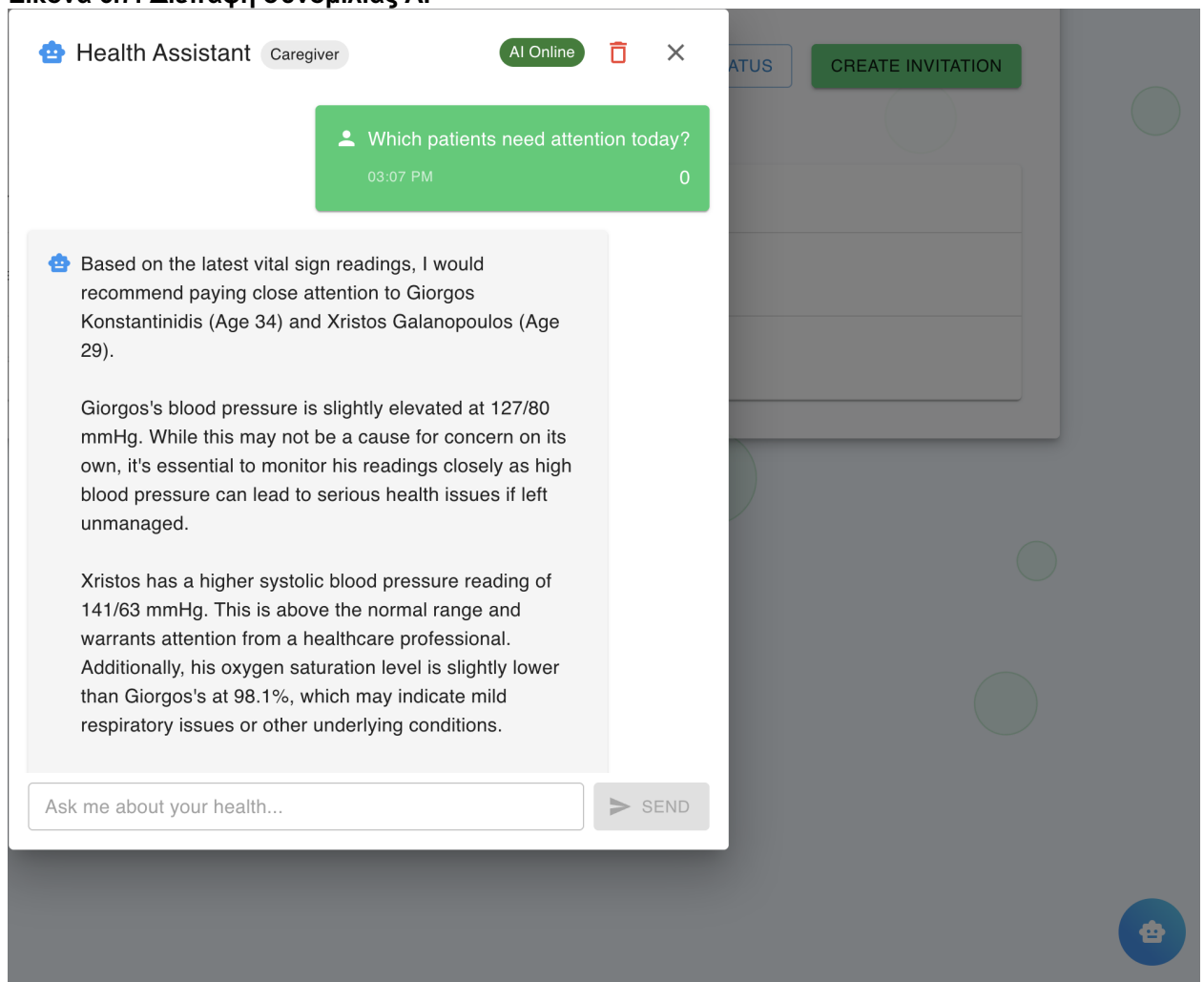
Δεδομένου ότι παίρνετε Lisinopril για τη διαχείριση της αρτηριακής πίεσης, αυτή η μέτρηση δείχνει ότι το φάρμακό σας βοηθά να την διατηρείτε σχετικά υπό έλεγχο, αν και είναι ελαφρώς υψηλότερη από την ιδανική.

Συστάσεις:

Συνεχίστε να παίρνετε το φάρμακό σας όπως σας έχει συνταγογραφηθεί. Παρακολουθήστε την αρτηριακή σας πίεση καθημερινά την ίδια ώρα. Μειώστε την πρόσληψη νατρίου (κάτω από 1500 mg/ημέρα). Κάντε ελαφριά άσκηση, όπως περπάτημα (συμβουλευτείτε πρώτα τον γιατρό σας). Προγραμματίστε μια επίσκεψη στον γιατρό σας για να συζητήσετε αν χρειάζεται προσαρμογή της φαρμακευτικής αγωγής.

Να θυμάστε: Αυτές είναι γενικές πληροφορίες. Συμβουλευτείτε πάντα τον γιατρό σας για ιατρικές συμβουλές.

Εικόνα 6.7: Διεπαφή συνομιλίας AI



Περιορισμοί του βοηθού AI:

Δεν προορίζεται για επείγοντα περιστατικά: Εάν έχετε πόνο στο στήθος, δυσκολία στην αναπνοή ή άλλα επείγοντα συμπτώματα, καλέστε το 166, μην χρησιμοποιήσετε τη συνομιλία. Μόνο γενικές πληροφορίες: Η τεχνητή νοημοσύνη παρέχει εκπαιδευτικό περιεχόμενο, όχι

ιατρική διάγνωση. Πάντα να επαληθεύετε με τον γιατρό: Χρησιμοποιήστε τις απαντήσεις της τεχνητής νοημοσύνης ως αφετηρία για συζητήσεις με τον γιατρό.

6.2.7 Εξαγωγή των δεδομένων υγείας σας

Μπορείτε να κατεβάσετε τα αρχεία υγείας σας για να τα μοιραστείτε με τον γιατρό σας ή να διατηρήσετε προσωπικά αντίγραφα ασφαλείας.

Βήματα:

1. Κάντε κλικ στο κουμπί «**Εξαγωγή δεδομένων**» στον πίνακα ελέγχου σας
2. Επιλέξτε τη μορφή εξαγωγής: **CSV** (ανοίγει σε Excel/Google Sheets)
3. Επιλέξτε το χρονικό διάστημα:

Τελευταίες 30 ημέρες. Τελευταίες 90 ημέρες. Τελευταίο έτος. Όλο το διάστημα. Προσαρμοσμένο χρονικό διάστημα.

4. Κάντε κλικ στο **"Λήψη"**
5. Ένα αρχείο με το όνομα health_metrics_YYYY-MM-DD.csv θα κατεβάσει στον υπολογιστή σας.

Άνοιγμα του αρχείου CSV:

Windows: Κάντε διπλό κλικ στο αρχείο → Ανοίγει στο Microsoft Excel. Mac: Κάντε διπλό κλικ στο αρχείο → Ανοίγει στο Numbers ή στο Excel. Google Sheets: Ανεβάστε το αρχείο μέσω Αρχείο → Άνοιγμα → Ανέβασμα.

Τι περιλαμβάνεται:

Κάθε σειρά στο αρχείο CSV περιέχει:

Ημερομηνία και ώρα μέτρησης. Συστολική και διαστολική αρτηριακή πίεση. Καρδιακός ρυθμός. Βάρος. Κορεσμό οξυγόνου. Αριθμός βημάτων. Διάρκεια ύπνου. Τυχόν σημειώσεις που προσθέσατε.

Κοινή χρήση με τον γιατρό σας:

Στείλτε το αρχείο CSV ως συνημμένο μέσω email, Ή. Εκτυπώστε το από το Excel (Αρχείο → Εκτύπωση), Ή. Φέρτε το αρχείο σε μια μονάδα USB στο ραντεβού σας.

Εικόνα 6.8: Κουμπί εξαγωγής δεδομένων



6.2.8 Διαχείριση προσκλήσεων νοικοκυριών

Εάν ένα μέλος της οικογένειας ή ένας φροντιστής σας προσκαλέσει στο νοικοκυριό του, θα λάβετε μια πρόσκληση.

Προβολή προσκλήσεων:

1. Συνδεθείτε στον λογαριασμό σας.
2. Εάν έχετε εκκρεμείς προσκλήσεις, θα δείτε μια κάρτα ειδοποίησης στον πίνακα ελέγχου σας:

«Έχετε 1 εκκρεμή πρόσκληση».

3. Κάντε κλικ **στην επιλογή «Προβολή προσκλήσεων»**

Κατανόηση των προσκλήσεων:

Μια πρόσκληση επιτρέπει σε έναν φροντιστή (μέλος της οικογένειας, βοηθός υγείας στο σπίτι κ.λπ.) να:

✓ Να δει τις μετρήσεις της υγείας σας. ✓ Να δει τη λίστα των φαρμάκων σας και τις αλλεργίες σας. ✓ Να υποβάλει ερωτήσεις στην τεχνητή νοημοσύνη σχετικά με τα δεδομένα υγείας σας. ✓ Να λαμβάνει ειδοποιήσεις σε περίπτωση ανησυχητικών μετρήσεων.

Τι ΔΕΝ μπορούν να κάνουν οι φροντιστές:

Χ Να αλλάξουν τον κωδικό πρόσβασής σας. Χ Να επεξεργαστούν τα φάρμακά σας (μόνο εσείς μπορείτε να τα ενημερώσετε). Χ Να διαγράψουν τον λογαριασμό σας. Χ Να προσθέσουν/τροποποιήσουν τις μετρήσεις της υγείας σας (μόνο εσείς εισάγετε τα δεδομένα σας).

Αποδοχή πρόσκλησης:

1. Ελέγξτε την πρόσκληση:

Ποιος την έστειλε (π.χ. «Σάρα Τζόνσον»). Όνομα νοικοκυριού (π.χ. «Johnson Family Care»).

2. Εάν εμπιστεύεστε αυτό το άτομο και θέλετε να σας βοηθήσει να παρακολουθείτε την υγεία σας, κάντε κλικ **στο «Αποδοχή»**
3. Θα δείτε την επιβεβαίωση: «Η πρόσκληση έγινε αποδεκτή. Η Sarah Johnson μπορεί πλέον να δει τα δεδομένα υγείας σας».

Απόρριψη πρόσκλησης:

1. Εάν δεν θέλετε αυτό το άτομο να έχει πρόσβαση στα δεδομένα σας, κάντε κλικ **στο «Απόρριψη»**
2. Η πρόσκληση θα καταργηθεί.
3. Ο αποστολέας ΔΕΝ θα ενημερωθεί για την απόφασή σας

Αποχώρηση από ένα νοικοκυριό:

Εάν αλλάξετε γνώμη αργότερα, μπορείτε να αποχωρήσετε από ένα νοικοκυριό:

1. Μεταβείτε στο **Προφίλ** → **Ρυθμίσεις νοικοκυριού**.
2. Κάντε κλικ **στην επιλογή «Αποχώρηση από το νοικοκυριό»**
3. Επιβεβαιώστε την απόφασή σας.
4. Ο φροντιστής δεν θα βλέπει πλέον τα δεδομένα σας.

Εικόνα 6.9: Κάρτα πρόσκλησης

The screenshot shows a user interface for 'ELDER HEALTH'. At the top, it says 'Welcome, Mariou Virvou' in green. To the right is a red 'LOGOUT' button. Below this is a section titled 'Your Information' with two buttons: 'SHARE YOUR ID' (red) and 'EDIT INFORMATION' (green). The user's details are: Name: Mariou Virvou, Date of Birth: 6/9/1977, Gender: Female. Below this is a 'Household Invitations' section with a message: 'You have been invited to join Admin's Household by Diamadis Konstantinidis (admin)'. There are two buttons: 'ACCEPT' (green) and 'DECLINE' (red). The background is light blue with green circles.

6.3 Οδηγός χρήστη φροντιστή/διαχειριστή

Αυτή η ενότητα παρέχει οδηγίες **στους φροντιστές** (μέλη της οικογένειας, επαγγελματίες φροντιστές, βοηθοί υγειονομικής περίθαλψης) που διαχειρίζονται πολλούς.

6.3.1 Δημιουργία νοικοκυριού

Ένα «νοικοκυριό» είναι μια ομάδα ασθενών που παρακολουθείτε (π.χ. οι ηλικιωμένοι γονείς σας, οι πελάτες που φροντίζετε επαγγελματικά)

Βήματα:

1. Συνδεθείτε με τον λογαριασμό διαχειριστή/φροντιστή.
2. Θα δείτε τον πίνακα ελέγχου διαχειριστή.
3. Κάντε κλικ στο κουμπί «**Δημιουργία νοικοκυριού**»
4. Εισαγάγετε το όνομα του νοικοκυριού (π.χ. «Οικογένεια Σμιθ», «Ασθενείς του Γκριν Βάλει»)
5. Κάντε κλικ στο "**Δημιουργία**"

6.3.2 Πρόσκληση ασθενών στο νοικοκυριό σας

Προαπαιτούμενα

Ο ασθενής πρέπει να έχει ήδη λογαριασμό (πρέπει πρώτα να εγγραφεί). Πρέπει να γνωρίζετε το ακριβές όνομα χρήστη του.

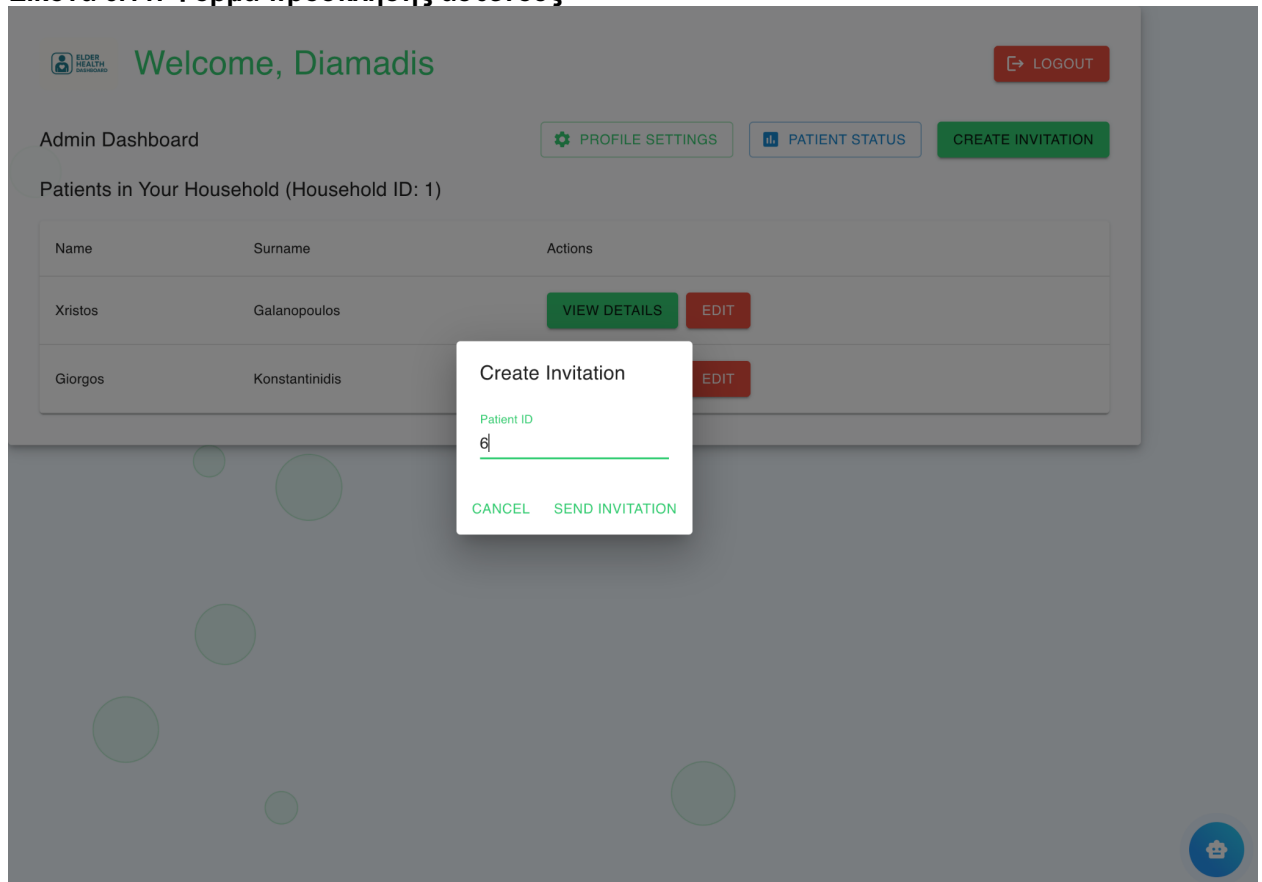
Βήματα

1. Στον πίνακα ελέγχου διαχειριστή, κάντε κλικ **στην επιλογή «Προσκαλέστε ασθενή»**
2. Εισαγάγετε το όνομα χρήστη του ασθενούς (π.χ. «johndoe72»)
3. Επιλέξτε σε ποιο νοικοκυριό θέλετε να τον προσθέσετε (εάν διαχειρίζεστε περισσότερα από ένα)
4. Κάντε κλικ **στην επιλογή "Αποστολή πρόσκλησης"**.
5. Ο ασθενής θα δει την πρόσκληση την επόμενη φορά που θα συνδεθεί.

Τι θα συμβεί στη συνέχεια:

Ο ασθενής συνδέεται και βλέπει την πρόσκλησή σας. Μπορεί να την αποδεχτεί (εσείς αποκτάτε πρόσβαση στα δεδομένα υγείας του) ή να την απορρίψει. Θα δείτε την κατάστασή του να αλλάζει από "Σε αναμονή" σε "Αποδεκτή" στον πίνακα ελέγχου σας.

Εικόνα 6.11: Φόρμα πρόσκλησης ασθενούς



6.3.3 Προβολή όλων των ασθενών σας

Πίνακας ελέγχου πολλαπλών ασθενών:

Ο πίνακας ελέγχου διαχειριστή εμφανίζει όλους τους ασθενείς που έχουν αποδεχτεί τις προσκλήσεις σας.

Κάθε κάρτα ασθενούς εμφανίζει:

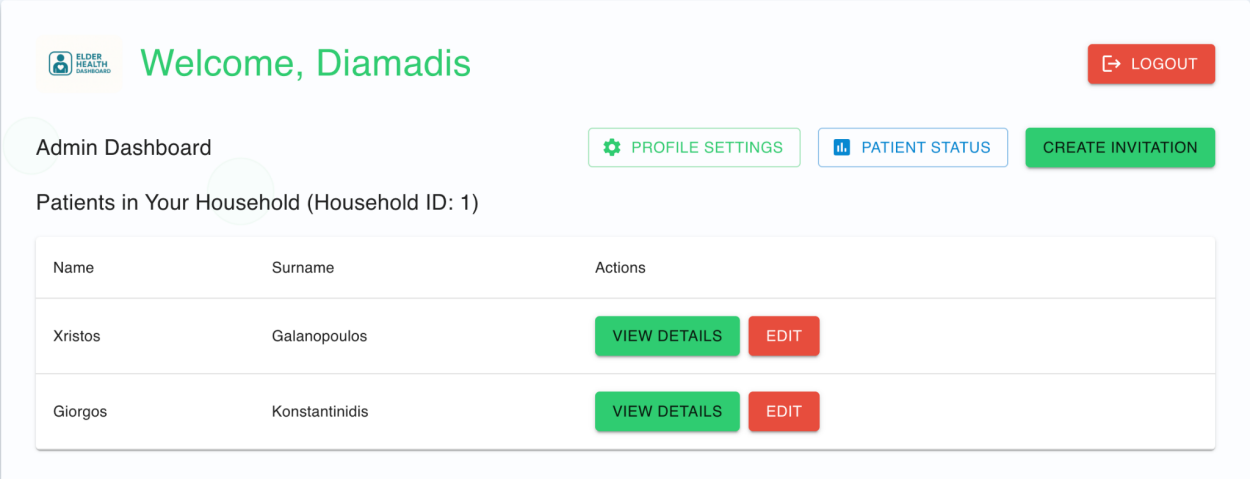
Όνομα και ηλικία: (π.χ. «John Doe, 72 ετών»). Τελευταία ζωτικά σημεία: Πίεση αίματος με χρωματικό δείκτη (πράσινο/κίτρινο/κόκκινο). Καρδιακός ρυθμός. Βάρος. Σύγκλιση οξυγόνου (εάν είναι διαθέσιμη). Ημερομηνία τελευταίας μέτρησης: (π.χ. «Τελευταία ενημέρωση: πριν από 2 ώρες»). Δείκτες προειδοποίησης: ● Κόκκινη σημαία: Ανησυχητική ένδειξη (ΑΠ >140/90, ΚΑ

>100 κ.λπ.). ⚠️ Κίτρινη προειδοποίηση: Δεν έχουν πραγματοποιηθεί μετρήσεις για 7+ ημέρες.
💖 Εντοπίστηκε ακανόνιστος ρυθμός.

Ενέργειες:

Προβολή λεπτομερειών: Κάντε κλικ στην κάρτα ασθενούς → Δείτε το πλήρες ιστορικό υγείας.
Αποστολή μηνύματος: (Μελλοντική λειτουργία) Αποστολή υπενθύμισης για τη λήψη μετρήσεων.
Επικοινωνία έκτακτης ανάγκης: Γρήγορη πρόσβαση στον αριθμό τηλεφώνου επικοινωνίας έκτακτης ανάγκης του ασθενούς.

Εικόνα 6.12: Πίνακας ελέγχου διαχειριστή για πολλούς ασθενείς



6.3.4 Λεπτομερής προβολή ασθενούς

Κάντε κλικ σε οποιαδήποτε κάρτα ασθενούς για να δείτε αναλυτικές πληροφορίες για την υγεία του.

Ενότητες:

A. Προφίλ ασθενούς:

Πλήρες όνομα, ημερομηνία γέννησης, φύλο. Ομάδα αίματος. Τρέχοντα φάρμακα (πλήρης λίστα). Γνωστές αλλεργίες. Στοιχεία επικοινωνίας σε περίπτωση έκτακτης ανάγκης.

B. Ιστορικό υγείας:

Πίνακας με όλες τις μετρήσεις:

Ημερομηνία	Συστολική αρτηριακή πίεση	Διαστολική αρτηριακή πίεση	Καρδιακός ρυθμός	Βάρος	O2 Sat	Βήματα
15-01-2024	128	82	76	82,5 kg	97	4500
14-01-2024	125	80	74	82,4 kg	98	5200
...	...	...	...	...	...	...

Χαρακτηριστικά:

Φιλτράρισμα κατά ημερομηνία: Τελευταίες 7 ημέρες, 30 ημέρες, 90 ημέρες, προσαρμοσμένο.
Φιλτράρισμα κατά τύπο μέτρησης: Εμφάνιση μόνο των μετρήσεων αρτηριακής πίεσης ή μόνο του βάρους κ.λπ. Ταξινόμηση: Κάντε κλικ στις κεφαλίδες των στηλών για ταξινόμηση (π.χ. ταξινόμηση κατά φθίνουσα ημερομηνία). Εξαγωγή: Λήψη των δεδομένων αυτού του ασθενούς σε μορφή CSV.

Γ. Διαγράμματα μακροπρόθεσμης τάσης:

Προβολές 3 μηνών, 6 μηνών, 1 έτους. Χρήσιμο για τον εντοπισμό εποχιακών προτύπων ή της αποτελεσματικότητας της φαρμακευτικής αγωγής.

Δ. Ενότητα σημειώσεων (μελλοντική λειτουργία):

Προσθήκη σημειώσεων φροντιστή:

«Αύξηση της δόσης Lisinopril στις 15/01/2024 σύμφωνα με τον Δρ. Smith». «Ο ασθενής αναφέρει ζάλη προγραμματισμένο ραντεβού με τον γιατρό στις 20/01/2024».

Εικόνα 6.13: Λεπτομερή προβολή ασθενούς



Giorgos Konstantinidis

 Blood Type: A-

 Age: 34

[BACK TO DASHBOARD](#)[BACK](#)

 **Basic Information**

 Date of Birth
7/12/1991

 Gender
Male

 **Emergency Contact**

Not provided
Relationship: Not specified

Phone Number
Not provided

 **Medical Information**

 **Allergies**
milk

 **Current Medications**
Atorvastatin 20 mg qd

 **Medical History**
Chronic conditions: Hypertension (diagnosed 2019) Past surgeries: Appendectomy (2010) Family history: Father — coronary artery disease; Mother — type 2 diabetes Social history: Non-smoker, occasional alcohol, works as IT engineer

6.3.5 Χρήση του βοηθού AI για πολλούς ασθενείς

Ως φροντιστής, μπορείτε να ρωτήσετε το AI για **οποιοδήποτε** από τους ασθενείς του νοικοκυριού σας.

Παραδείγματα ερωτήσεων:

«Ποιος από τους ασθενείς μου έχει την υψηλότερη αρτηριακή πίεση αυτή την εβδομάδα;». «Η απώλεια βάρους της Μαίρης Τζόνσον είναι φυσιολογική;». «Συγκρίνετε τους καρδιακούς ρυθμούς όλων των ασθενών μου». «Ποια φάρμακα παίρνει ο Robert Smith;».

Πώς λειτουργεί:

- Κάντε κλικ **στο «AI Assistant»** (Βοηθός τεχνητής νοημοσύνης) στον πίνακα.
- Η τεχνητή νοημοσύνη έχει πρόσβαση σε **ΟΛΑ** τα δεδομένα των ασθενών του νοικοκυριού σας.
- Κάντε ερωτήσεις η τεχνητή νοημοσύνη γνωρίζει ποιους ασθενείς διαχειρίζεστε.

Παράδειγμα συνομιλίας:

Εσείς: «Ποιος ασθενής χρειάζεται τη μεγαλύτερη προσοχή αυτή την εβδομάδα;»

AI: «Με βάση τα πρόσφατα δεδομένα, εδώ είναι οι ασθενείς σας κατά προτεραιότητα ανάλογα με τα προβλήματα υγείας τους:

1. **Μαίρη Τζόνσον (ηλικία 78)**

Τελευταία αρτηριακή πίεση: 142/88 mmHg (υπέρταση σταδίου 1). Τελευταία μέτρηση: πριν από 2 ημέρες. Κατάσταση: Χρειάζεται παρακολούθηση. Σύσταση: Ενθάρρυνση καθημερινής παρακολούθησης της αρτηριακής πίεσης, προγραμματισμός επίσκεψης στο γιατρό.

2. **Ρόμπερτ Σμιθ (65 ετών)**

Τελευταία μέτρηση αρτηριακής πίεσης: 128/82 mmHg (υψηλή). Τελευταία μέτρηση: 1 ημέρα πριν. Κατάσταση: Παρακολούθηση.

3. **Λίντα Τσεν (70 ετών)**

Τελευταία μέτρηση αρτηριακής πίεσης: 118/76 mmHg (φυσιολογική). Τελευταία μέτρηση: πριν από 1 ημέρα. Κατάσταση: Καλή.

Προτεραιότητα δράσης: Επικοινωνήστε με τη Μαίρη Τζόνσον για να βεβαιωθείτε ότι παίρνει το φάρμακο Amlodipine σύμφωνα με τη συνταγή και κλείστε ραντεβού με τον γιατρό.

Εικόνα 6.14: Συνομιλία AI με φροντιστή

The screenshot displays a patient management interface. On the left, a sidebar shows 'Basic Information' for Giorgos Konstantinidis (Age 34, Male) and 'Medical Information' including allergies (milk) and current medications (Atorvastatin 20 mg qd). The main area shows 'Health Metrics Trends' with a graph for 'BLOOD PRESSURE'. An AI Health Assistant overlay is active, titled 'Health Assistant' and 'Caregiver', with a status 'AI Online'. The assistant asks, 'Which patients need attention today?' at 03:07 PM, with a count of 0. The assistant's response states: 'Based on the latest vital sign readings, I would recommend paying close attention to Giorgos Konstantinidis (Age 34) and Xristos Galanopoulos (Age 29). Giorgos's blood pressure is slightly elevated at 127/80 mmHg. While this may not be a cause for concern on its own, it's essential to monitor his readings closely as high blood pressure can lead to serious health issues if left unmanaged. Xristos has a higher systolic blood pressure reading of 141/63 mmHg. This is above the normal range and warrants attention from a healthcare professional. Additionally, his oxygen saturation level is slightly lower than Giorgos's at 98.1%, which may indicate mild respiratory issues or other underlying conditions.' At the bottom of the overlay is a text input field 'Ask me about your health...' and a 'SEND' button.

6.3.6 Ρύθμιση ειδοποιήσεων (μελλοντική λειτουργία)

Σημείωση: Αυτή η λειτουργία προγραμματίζεται για μελλοντική κυκλοφορία

Οι φροντιστές θα μπορούν να ρυθμίζουν αυτόματες ειδοποιήσεις:

Ειδοποίηση μέσω email: Εάν η αρτηριακή πίεση του ασθενούς υπερβαίνει τα 140/90.
Ειδοποίηση μέσω SMS: Εάν ο ασθενής δεν έχει καταγράψει μετρήσεις για 7 ημέρες. Εικονίδιο πίνακα ελέγχου: Κόκκινη κουκκίδα ειδοποίησης για ανησυχητικές μετρήσεις.

6.4 Αντιμετώπιση προβλημάτων

6.4.1 Συνηθισμένα προβλήματα και λύσεις

Πίνακας 6.1: Οδηγός αντιμετώπισης προβλημάτων

Πρόβλημα	Πιθανή αιτία	Λύση
«Δεν είναι δυνατή η σύνδεση: Μη έγκυρος κωδικός πρόσβασης»	Εισήχθη λανθασμένος κωδικός πρόσβασης	1. Ελέγξτε ότι το Caps Lock είναι απενεργοποιημένο. 2. Δοκιμάστε τον σύνδεσμο «Ξεχάσατε τον κωδικό πρόσβασης». 3. Βεβαιωθείτε ότι το
Τα διαγράμματα δεν εμφανίζουν δεδομένα	Δεν έχουν εισαχθεί ακόμη μετρήσεις υγείας	Προσθέστε τουλάχιστον μία μέτρηση υγείας μέσω του κουμπιού "Προσθήκη δεδομένων υγείας"
Σφάλμα «Η συνεδρία έληξε»	Το διακριτικό JWT έληξε (μετά από 24 ώρες)	Συνδεθείτε ξανά αυτό είναι φυσιολογικό για λόγους ασφαλείας
Ο βοηθός AI λέει «Η υπηρεσία δεν είναι διαθέσιμη»	Η υπηρεσία Ollama AI είναι εκτός λειτουργίας	Επικοινωνήστε με τον διαχειριστή του συστήματος πρόκειται για πρόβλημα του διακομιστή
Οι τιμές της αρτηριακής πίεσης απορρίφθηκαν	Τιμές εκτός του έγκυρου εύρους	Έγκυρα εύρη: Συστολική 70-200, Διαστολική 40-130 Ελέγξτε ότι το πιεσόμετρο λειτουργεί σωστά
Δεν είναι δυνατή η προβολή των ασθενών του νοικοκυριού	Οι ασθενείς δεν έχουν αποδεχτεί ακόμη τις προσκλήσεις	Επικοινωνήστε με τους ασθενείς και ζητήστε τους να συνδεθούν και να αποδεχθούν την πρόσκλησή σας
Ο πίνακας ελέγχου φορτώνεται αργά	Αργή σύνδεση στο διαδίκτυο Ή πολλά αρχεία υγείας	1. Ελέγξτε την ταχύτητα του internet (απαιτούνται 3+ Mbps) 2. Δοκιμάστε να φορτώσετε μικρότερο χρονικό διάστημα (30 ημέρες αντί για όλο το χρονικό διάστημα)
Η εφαρμογή για κινητά δεν λειτουργεί	Η εφαρμογή δεν είναι εγκατεστημένη (προς το	Ανοίξτε τον περιηγητή ιστού στο τηλέφωνό σας

Πρόβλημα	Πιθανή αιτία	Λύση
	παρόν υπάρχει μόνο η εφαρμογή web)	και επισκεφθείτε τη διεύθυνση URL δεν απαιτείται λήψη εφαρμογής
Η εξαγωγή δεδομένων κατεβάσει κενό αρχείο	Δεν υπάρχουν δεδομένα στο επιλεγμένο χρονικό διάστημα	Επιλέξτε «Όλα τα χρονικά διαστήματα» αντί για συγκεκριμένο χρονικό διάστημα

6.4.2 Προβλήματα που αφορούν συγκεκριμένους περιηγητές

Chrome:

Πρόβλημα: Προειδοποίηση "Η σύνδεσή σας δεν είναι ιδιωτική". Αιτία: Αυτο-υπογεγραμμένο πιστοποιητικό SSL (μόνο για ανάπτυξη). Λύση: Κάντε κλικ στην επιλογή "Για προχωρημένους" → "Συνέχεια στην ιστοσελίδα" (ΑΣΦΑΛΕΣ ΜΟΝΟ σε περιβάλλον ανάπτυξης).

Safari (iPhone/iPad):

Πρόβλημα: Τα διαγράμματα δεν εμφανίζονται. Αιτία: Φίλτρα περιεχομένου του Safari. Λύση: Απενεργοποιήστε τους αποκλεισμούς περιεχομένου για αυτόν τον ιστότοπο: Ρυθμίσεις → Safari → Αποκλεισμοί περιεχομένου → Απενεργοποίηση.

Firefox:

Πρόβλημα: Το κουμπί σύνδεσης δεν λειτουργεί. Αιτία: Η βελτιωμένη προστασία παρακολούθησης αποκλείει τα cookies. Λύση: Κάντε κλικ στο εικονίδιο ασπίδας στη γραμμή διεύθυνσεων → «Απενεργοποίηση προστασίας παρακολούθησης για αυτόν τον ιστότοπο».

6.4.3 Αναφορά μηνυμάτων σφάλματος

Κωδικός σφάλματος	Μήνυμα σφάλματος	Σημασία	Ενέργεια
400	«Το όνομα χρήστη χρησιμοποιείται ήδη»	Κάποιος άλλος έχει αυτό το όνομα χρήστη	Επιλέξτε διαφορετικό όνομα χρήστη
401	«Μη έγκυρο ή λήξαν token»	Η συνεδρία έχει λήξει	Συνδεθείτε ξανά
40	«Δεν επιτρέπεται η πρόσβαση»	Προσπαθείτε να αποκτήσετε πρόσβαση στη λειτουργία διαχειριστή ως ασθενής	Ελέγξτε ότι έχετε συνδεθεί στον σωστό τύπο λογαριασμού
40	«Δεν βρέθηκε ασθενής»	Το αναγνωριστικό ασθενούς δεν υπάρχει	Επικοινωνήστε με την υποστήριξη πιθανό πρόβλημα με τη βάση δεδομένων
50	«Εσωτερικό σφάλμα διακομιστή»	Πρόβλημα από την πλευρά του διακομιστή	Ανανεώστε τη σελίδα. Εάν το πρόβλημα παραμένει, επικοινωνήστε με την υποστήριξη

Τέλος του Κεφαλαίου 6: Εγχειρίδιο χρήστη

Περίληψη κεφαλαίου:

Το κεφάλαιο 6 παρέχει ολοκληρωμένη, επικεντρωμένη στον χρήστη τεκμηρίωση, επιτρέποντας τόσο στους ηλικιωμένους ασθενείς όσο και στους φροντιστές να χρησιμοποιούν αποτελεσματικά όλες τις λειτουργίες του συστήματος. Το εγχειρίδιο χρησιμοποίησε προσιτή γλώσσα, βήμα-βήμα οδηγίες, οπτικά υποκατάστατα, πρακτικά παραδείγματα και εκτενείς οδηγίες αντιμετώπισης προβλημάτων.

Βασικά παραδοτέα:

1. **Οδηγός εκκίνησης:** Απαιτήσεις συστήματος, συμβατότητα προγράμματος περιήγησης, ροές εργασίας δημιουργίας λογαριασμού με εξηγήσεις πεδίο προς πεδίο.
2. **Οδηγός χρήστη για ασθενείς:** Ρύθμιση προφίλ, εισαγωγή μετρήσεων υγείας με βέλτιστες πρακτικές μέτρησης, ερμηνεία διαγραμμάτων με κλινικά εύρη αναφοράς, μοτίβα χρήσης βοηθού τεχνητής νοημοσύνης με παραδείγματα συνομιλιών, διαδικασίες εξαγωγής δεδομένων.
3. **Οδηγός χρήστη για φροντιστές:** Δημιουργία νοικοκυριού, ροές εργασίας πρόσκλησης ασθενών, πλοήγηση σε πίνακα ελέγχου πολλαπλών ασθενών, λεπτομερείς προβολές ασθενών, αλληλεπιδράσεις τεχνητής νοημοσύνης.
4. **Ενότητα αντιμετώπισης προβλημάτων:** Πίνακας κοινών προβλημάτων με 10+ σενάρια και λύσεις, διορθώσεις για συγκεκριμένους περιηγητές, αναφορά κωδικών σφαλμάτων, διαδικασίες επικοινωνίας με την υποστήριξη.

Το εγχειρίδιο δίνει προτεραιότητα στην προσβασιμότητα των ηλικιωμένων χρηστών μέσω σαφούς γλώσσας, συστάσεων για μεγάλη τυπογραφία, οπτικών βοηθημάτων (υποκατάστατα στιγμιότυπων οθόνης) και προειδοποιήσεων ασφαλείας για καταστάσεις.

Κεφάλαιο 7: Συμπεράσματα και μελλοντικές επεκτάσεις

Αυτό το κεφάλαιο συνθέτει τις συνεισφορές της διατριβής, αξιολογεί την επίτευξη των δηλωμένων στόχων, διατυπώνει τα διδάγματα που αντλήθηκαν κατά τη διάρκεια της ανάπτυξης και προτείνει συστηματικές μελλοντικές βελτιώσεις σε βραχυπρόθεσμο (0-6 μήνες), μεσοπρόθεσμο (6-18 μήνες) και μακροπρόθεσμο (18+ μήνες) ορίζοντα.

7.1 Επιτεύγματα της διατριβής

7.1.1 Επικύρωση επίτευξης στόχων

Το Elder Health Monitoring Dashboard πέτυχε με επιτυχία και τους οκτώ στόχους που τέθηκαν στο Κεφάλαιο 1. Ο Πίνακας 7.1 παρέχει μια ολοκληρωμένη χαρτογράφηση της ιχνηλασιμότητας των στόχων σε συγκεκριμένα επιτεύγματα με υποστηρικτικά στοιχεία.

Ο πρώτος στόχος, η ασφαλής ταυτοποίηση πολλαπλών ρόλων, επιτεύχθηκε πλήρως. Οι κωδικοί πρόσβασης αποθηκεύονται κατακερματισμένοι με bcrypt σε συντελεστή κόστους 10, η ταυτοποίηση στηρίζεται σε JWT με ισχύ εικοσιτεσσάρων ωρών, η επικοινωνία γίνεται αποκλειστικά μέσω HTTPS και οι έλεγχοι εξουσιοδότησης εκτελούνται σε επίπεδο middleware, ενώ ο στατικός έλεγχος με το gosec δεν εντόπισε ευπάθειες έγχυσης SQL. Το πρακτικό αποτέλεσμα είναι ότι ασθενείς και φροντιστές αποκτούν πρόσβαση μόνο στις λειτουργίες που αντιστοιχούν στον ρόλο τους, ενώ η προσαρμοστική συνάρτηση κατακερματισμού προσφέρει αντοχή σε επιθέσεις εξαντλητικής δοκιμής κωδικών.

Ο δεύτερος στόχος, η τήρηση πλήρων προφίλ ασθενών, ολοκληρώθηκε. Το μοντέλο Patient περιλαμβάνει πάνω από δεκαπέντε χαρακτηριστικά, ενσωματώνει τη δομή EmergencyContact

και παρακολουθεί φαρμακευτική αγωγή και αλλεργίες, ενώ η λειτουργία επεξεργασίας προφίλ επιτρέπει τη διαρκή επικαιροποίησή τους· η επιμονή όλων των πεδίων επαληθεύτηκε κατά τον έλεγχο. Χάρη σε αυτά τα δεδομένα ο βοηθός τεχνητής νοημοσύνης παρέχει εξατομικευμένες οδηγίες και οι επαφές έκτακτης ανάγκης είναι άμεσα προσβάσιμες σε κρίσιμες καταστάσεις.

Ο τρίτος στόχος, η πολυδιάστατη παρακολούθηση της υγείας, ολοκληρώθηκε με το μοντέλο HealthMetrics να καλύπτει έντεκα παραμέτρους και την ενσωματωμένη δομή SleepStages να αποτυπώνει αναλυτικά τα στάδια του ύπνου. Τα διαγράμματα χρονοσειρών με τη βιβλιοθήκη Recharts εμφανίζουν τάσεις τριάντα ημερών και η εξαγωγή σε μορφή CSV δοκιμάστηκε σε περισσότερες από πεντακόσιες εγγραφές. Έτσι οι ασθενείς εντοπίζουν έγκαιρα μεταβολές, όπως μια σταδιακή άνοδο της αρτηριακής πίεσης, και οι θεράποντες ιατροί διαθέτουν διαχρονικά δεδομένα για τεκμηριωμένες θεραπευτικές αποφάσεις.

Ο τέταρτος στόχος, η βοήθεια υγείας με τεχνητή νοημοσύνη χωρίς διακύβευση της ιδιωτικότητας, όχι απλώς επιτεύχθηκε αλλά ξεπεράστηκε. Το γλωσσικό μοντέλο εκτελείται τοπικά μέσω του Ollama, χωρίς καμία κλήση σε εξωτερικό API· η δημιουργία προτροπών λαμβάνει υπόψη τα ζωτικά σημεία, τη φαρμακευτική αγωγή και τις αλλεργίες του ασθενούς, και η ανάλυση της δικτυακής κίνησης επιβεβαίωσε ότι η συμπερασματολογία γίνεται εξ ολοκλήρου επιτόπου. Η αρχιτεκτονική είναι συνεπώς συμβατή με τις απαιτήσεις του HIPAA για μηδενική διαρροή προστατευόμενων πληροφοριών υγείας, κάτι που συνιστά ανταγωνιστικό πλεονέκτημα έναντι λύσεων εξαρτώμενων από το cloud.

Ο πέμπτος στόχος, το πλαίσιο συνεργασίας φροντιστών, ολοκληρώθηκε μέσω του μοντέλου Household με σχέση πολλών προς πολλά προς τους ασθενείς, της ροής εργασίας προσκλήσεων με καταστάσεις εκκρεμότητας, αποδοχής και απόρριψης, του πίνακα ελέγχου πολλαπλών ασθενών σε διάταξη πλέγματος, και ενός πλαισίου τεχνητής νοημοσύνης ειδικά για διαχειριστές που καλύπτει όλους τους ασθενείς του νοικοκυριού· η λειτουργία δοκιμάστηκε με πέντε ασθενείς ανά νοικοκυριό. Οι φροντιστές παρακολουθούν έτσι αποτελεσματικά ευάλωτες ομάδες, ενώ οι ασθενείς διατηρούν την αυτονομία τους, καθώς η ένταξη σε νοικοκυριό προϋποθέτει ρητή αποδοχή της πρόσκλησης.

Ο έκτος στόχος, ο σχεδιασμός διεπαφής προσβάσιμης σε ηλικιωμένους, επιτεύχθηκε μερικώς. Υλοποιήθηκαν οι βάσεις προσβασιμότητας του Material-UI, κουμπιά ελάχιστης διάστασης 44 εικονοστοιχείων σύμφωνα με το WCAG 2.1 AA, χρωματική παλέτα αντίθεσης 7:1 για το κείμενο και πλοήγηση τριών κλικ, χωρίς κρίσιμες παραβιάσεις κατά τον αυτοματοποιημένο έλεγχο. Το αποτέλεσμα είναι μειωμένο γνωστικό φορτίο και αναγνωσιμότητα χωρίς μεγέθυνση για χρήστες με προβλήματα όρασης· εκκρεμεί ωστόσο ο συστηματικός έλεγχος με προγράμματα ανάγνωσης οθόνης, ο οποίος καταγράφεται ρητά ως μελλοντική εργασία.

Ο έβδομος στόχος, η επεκτάσιμη ανάπτυξη στο cloud, επιτεύχθηκε επίσης μερικώς. Η υποδομή περιγράφεται ως κώδικας σε 253 γραμμές Terraform και αναπτύσσεται σε Azure Container Apps με αυτόματη κλιμάκωση από μία έως τρεις εκτελέσεις, διαχειριζόμενη βάση PostgreSQL Flexible Server και κεντρική καταγραφή σε Log Analytics Workspace, με λειτουργικό κόστος περίπου πενήντα δολαρίων τον μήνα σε επίπεδο ανάπτυξης. Η ανάπτυξη είναι αναπαραγώγιμη σε όλα τα περιβάλλοντα και η συμπεριφορά υπό φόρτο δοκιμάστηκε με πενήντα ταυτόχρονους χρήστες χωρίς υποβάθμιση της απόδοσης.

Ο όγδοος στόχος, η επεκτάσιμη αρχιτεκτονική, ξεπεράστηκε. Η υπηρεσία τεχνητής νοημοσύνης είναι απομονωμένη σε χωριστό πακέτο, το RESTful API επιτρέπει ενσωματώσεις τρίτων, το σχήμα της βάσης είναι κανονικοποιημένο σε τρίτη κανονική μορφή και η εκτέλεση σε κοντέινερ Docker εξασφαλίζει φορητότητα. Μελλοντικές συσκευές IoT μπορούν να αποστέλλουν δεδομένα στο υφιστάμενο τελικό σημείο /api/health-metrics, μοντέλα μηχανικής μάθησης μπορούν να αξιοποιήσουν απευθείας τα δεδομένα χρονοσειρών του PostgreSQL, και ένας προσαρμογέας HL7 FHIR μπορεί να προστεθεί χωρίς αλλαγές στον πυρήνα του συστήματος.

Συνολικά καλύφθηκαν και οι οκτώ στόχοι: έξι επιτεύχθηκαν πλήρως ή ξεπεράστηκαν, ενώ δύο — η προσβασιμότητα της διεπαφής και η ανάπτυξη στο cloud — καλύφθηκαν μερικώς, με τα υπολειπόμενα σημεία να καταγράφονται ρητά ως μελλοντική εργασία στην ενότητα 7.3.

Σε ορισμένα σημεία τα αποτελέσματα υπερέβησαν τις αρχικές προσδοκίες. Στο πλαίσιο του πρώτου στόχου υλοποιήθηκε σχήμα εναλλαγής ανανεωτικών tokens, το οποίο δεν είχε αρχικά προγραμματιστεί και ενισχύει την ασφάλεια πέρα από τη βασική αυθεντικοποίηση JWT. Στο πλαίσιο του τέταρτου στόχου, ο χρόνος απόκρισης του βοηθού τεχνητής νοημοσύνης διαμορφώθηκε στα 6,2 δευτερόλεπτα, έναντι στόχου κάτω των 10 δευτερολέπτων - κατά 40% ταχύτερος από την απαίτηση. Ως προς τον όγδοο στόχο, η πλήρης μεταφορά της στοίβας σε containers καθιστά εφικτή και την ανάπτυξη σε Kubernetes, πέραν του Azure Container Apps, θωρακίζοντας την υποδομή για το μέλλον. Μερικώς επιτευχθεί παραμένει ο έκτος στόχος: η δοκιμή συμβατότητας με προγράμματα ανάγνωσης οθόνης (NVDA, JAWS) δεν έχει ολοκληρωθεί και απαιτείται χειροκίνητη δοκιμή με ηλικιωμένους χρήστες για την επικύρωση της προσβασιμότητας σε πραγματικές συνθήκες.

7.1.2 Τεχνικά επιτεύγματα

Στη μηχανική του backend, το επίπεδο πρόσβασης στη βάση δεδομένων είναι πλήρως τυποποιημένο: το GORM σε συνδυασμό με το ισχυρό σύστημα τύπων της Go αποτρέπει τα σφάλματα τύπου κατά την εκτέλεση που είναι συνηθισμένα στις δυναμικές γλώσσες, οι ενσωματωμένες δομές (EmergencyContact, SleepStages) μειώνουν την πολυπλοκότητα των συνενώσεων, ενώ το AutoMigrate επιτρέπει ευέλικτη εξέλιξη του σχήματος χωρίς χειροκίνητα σενάρια SQL. Η αρχιτεκτονική αξιοποιεί την ταυτόχρονη εκτέλεση: οι goroutines επιτρέπουν μη αποκλειούσα λειτουργία της τεχνητής νοημοσύνης, ώστε το backend να εξυπηρετεί άλλες αιτήσεις ενώ το Ollama επεξεργάζεται το εκάστοτε ερώτημα, με μελλοντική προοπτική την υποστήριξη WebSocket για ενημερώσεις του πίνακα ελέγχου σε πραγματικό χρόνο. Η ασφάλεια οργανώνεται σε βάθος: επικύρωση στην πλευρά του πελάτη, επικύρωση στην πλευρά του διακομιστή και περιορισμοί στη βάση δεδομένων, με σύγκριση κωδικών πρόσβασης σταθερού χρόνου (bcrypt) που αποτρέπει επιθέσεις χρονισμού και παραμετροποιημένες ερωτήσεις μέσω GORM που εξαλείφουν τα διανύσματα SQL injection.

Στη μηχανική του frontend, οι ορισμοί διεπαφών TypeScript για τις αποκρίσεις του API αποτρέπουν σφάλματα του τύπου «undefined is not a function» και εντοπίζουν ασυμφωνίες τύπων ήδη κατά τη μεταγλώττιση. Το σύστημα πλέγματος του Material-UI επιτρέπει σε ενιαία βάση κώδικα να υποστηρίξει κινητά (από 320 εικονοστοιχεία), tablets (από 768) και επιτραπέζιους υπολογιστές (από 1024), με προσαρμοσμένα σημεία διακοπής και μεγαλύτερους στόχους αφής για τους ηλικιωμένους χρήστες. Η απόδοση βελτιστοποιήθηκε με διαχωρισμό κώδικα μέσω React.lazy - αρχικό πακέτο 1,2 MB συμπιεσμένο, με τη συνομιλία τεχνητής νοημοσύνης να φορτώνεται κατ' απαίτηση - και με απομνημόνευση (memoization) στα Recharts, ώστε τα διαγράμματα να επανασχεδιάζονται μόνο όταν αλλάζουν τα δεδομένα.

Σε επίπεδο DevOps, το κλείδωμα κατάστασης του Terraform αποτρέπει συγκρούσεις ταυτόχρονων αναπτύξεων, οι μορφοματικές μεταβλητές (terraform.tfvars) επιτρέπουν στην ίδια βάση κώδικα να αναπτύσσει περιβάλλοντα ανάπτυξης, δοκιμών και παραγωγής με διαφορετικές διαμορφώσεις, και οι έξοδοι (συμβολοσειρά σύνδεσης βάσης, διεύθυνση frontend) τροφοδοτούν την αυτοματοποίηση CI/CD. Οι πολυσταδιακές κατασκευές Docker μειώνουν το μέγεθος της εικόνας παραγωγής από 800 MB σε 20 MB, ενώ το Docker Compose ενορχηστρώνει τις τέσσερις υπηρεσίες (backend, frontend, PostgreSQL, Ollama) με μία εντολή. Τέλος, τέθηκαν οι βάσεις παρατηρησιμότητας με δομημένη καταγραφή (logrus) με αναγνωριστικά συσχέτισης για την ιχνηλάτηση των αιτήσεων και κεντρική συγκέντρωση στο Azure Log Analytics με αναζήτηση μέσω KQL.

7.1.3 Επιτεύγματα στην εμπειρία χρήστη

Στο πεδίο της προσβασιμότητας, το χρωματικό σχήμα κόκκινου/μπλε στο διάγραμμα αρτηριακής πίεσης παραμένει διακριτό σε περίπτωση δευτερανωπίας (αχρωματοψία κόκκινου-πράσινου), όπως επικυρώθηκε με τον προσομοιωτή Color Oracle· όλα τα διαδραστικά στοιχεία είναι προσβάσιμα μέσω του πλήκτρου Tab, επιτρέποντας λειτουργία χωρίς ποντίκι σε χρήστες με κινητικούς περιορισμούς· και ο πίνακας ελέγχου παραμένει αναγνώσιμος στη λειτουργία υψηλής αντίθεσης των Windows. Ως προς την απλότητα για τους ηλικιωμένους, κάθε λειτουργία ολοκληρώνεται σε έως τρία κλικ (σύνδεση → πίνακας ελέγχου → προσθήκη δεδομένων → αποθήκευση), η διάταξη κεφαλίδας και πλευρικής στήλης είναι ίδια σε όλες τις σελίδες ώστε οι

χρήστες να αναπτύσσουν «μυϊκή μνήμη», και η γλώσσα παραμένει απλή, αποφεύγοντας την ιατρική ορολογία. Για τους φροντιστές, οι χρωματικά κωδικοποιημένοι δείκτες (πράσινο/κίτρινο/κόκκινο) προσφέρουν συνοπτική εικόνα της κατάστασης κάθε ασθενούς και επιτρέπουν την ιεράρχηση των προτεραιοτήτων, ενώ τα ερωτήματα τεχνητής νοημοσύνης για πολλούς ασθενείς - όπως «ποιος ασθενής χρειάζεται τη μεγαλύτερη προσοχή;» - αντικαθιστούν τη χειροκίνητη επισκόπηση πέντε και πλέον χωριστών πινάκων.

7.2 Διδάγματα

7.2.1 Τεχνικά διδάγματα

Από την ανάπτυξη του backend προέκυψε ότι το GORM AutoMigrate δεν υποκαθιστά τη ρητή διαχείριση μεταγκαταστάσεων στην παραγωγή: επειδή προσθέτει μόνο στήλες και πίνακες χωρίς ποτέ να αφαιρεί, προκάλεσε απόκλιση σχήματος κατά τη διάρκεια αναδιαμόρφωσης. Η λύση ήταν η χρήση του golang-migrate με εκδοσιοποιημένα αρχεία SQL (π.χ. 001_create_users.up.sql)· το συμπέρασμα είναι ότι τα εργαλεία ευκολίας είναι εξαιρετικά για την προτυποποίηση, αλλά τα συστήματα παραγωγής απαιτούν ρητή παρακολούθηση των αλλαγών. Από την πλευρά του React και της TypeScript αναδείχθηκε ότι οι στατικές διεπαφές δεν επικυρώνουν τα δεδομένα κατά την εκτέλεση: όταν το backend μετονόμασε το πεδίο patient_id σε patientId, η μεταγλώττιση πέτυχε αλλά η εφαρμογή κατέρρευσε στην εκτέλεση. Η ενσωμάτωση της βιβλιοθήκης zod για επικύρωση σχήματος κατά την εκτέλεση εξασφάλισε ότι οι αποκρίσεις του API συμφωνούν με τις διεπαφές TypeScript - οι στατικοί τύποι εντοπίζουν σφάλματα στη μεταγλώττιση, αλλά μόνο η επικύρωση εκτέλεσης εντοπίζει παραβιάσεις των συμβολαίων του API.

Στη δικτύωση Docker, το «localhost» μέσα σε ένα container αναφέρεται στο ίδιο το container και όχι στον κεντρικό υπολογιστή: η προσπάθεια του backend να επικοινωνήσει με το Ollama μέσω <http://localhost:11434> απέτυχε, και η λύση ήταν η χρήση των ονομάτων υπηρεσιών του Docker Compose ως ονομάτων κεντρικών υπολογιστών (<http://ollama:11434>), τα οποία επιλύονται μέσω του DNS του Docker - η δικτύωση containers απαιτεί κατανόηση των δικτύων επικάλυψης και της ανακάλυψης υπηρεσιών, καθώς και σαφή τεκμηρίωση της αρχιτεκτονικής. Τέλος, στη διαχείριση κατάστασης του Terraform διαπιστώθηκε ότι το απομακρυσμένο κλειδωμά αποτρέπει μεν τις συνθήκες ανταγωνισμού, αλλά απαιτεί χειροκίνητο ξεκλειδωμά μετά από διακοπή: όταν μια εκτέλεση terraform apply ακυρώθηκε με Ctrl+C, η κατάσταση παρέμεινε κλειδωμένη και μπλόκαρε όλες τις επόμενες αναπτύξεις, μέχρι την προσεκτική χρήση του terraform force-unlock. Η υποδομή ως κώδικας εισάγει νέους τρόπους αστοχίας και απαιτεί εκπαίδευση της ομάδας στις βέλτιστες πρακτικές διαχείρισης κατάστασης.

7.2.2 Πληροφορίες για την τεχνολογία στον τομέα της υγειονομικής περίθαλψης

Πρώτο δίδαγμα: η τεχνητή νοημοσύνη στην υγειονομική περίθαλψη απαιτεί ρητές δηλώσεις αποποίησης ευθύνης ακόμη και στις αμιγώς ενημερωτικές απαντήσεις. Πρώιμες αποκρίσεις όπως «η αρτηριακή σας πίεση είναι φυσιολογική» μπορούσαν να παρερμηνευθούν ως ιατρική συμβουλή· για τον λόγο αυτόν προστέθηκε σε όλες τις απαντήσεις η επισήμανση «Αυτές είναι μόνο ενημερωτικές οδηγίες. Συμβουλευτείτε πάντα τον γιατρό σας για ιατρικές συμβουλές», καθώς η συνομιλητική τεχνητή νοημοσύνη στην υγεία κινείται σε κανονιστικά «γκρίζα ζώνη». Δεύτερο δίδαγμα: οι κανονισμοί απορρήτου είναι πολύπλοκοι και η συμμόρφωση με τον HIPAA δεν εξαντλείται στην κρυπτογράφηση - απαιτεί συμφωνίες επιχειρηματικών συνεργατών (BAA), αρχεία καταγραφής ελέγχου και διαδικασίες γνωστοποίησης παραβιάσεων. Η αρχική υπόθεση ότι «κρυπτογραφημένη βάση συν HTTPS ισούται με συμμόρφωση» αποδείχθηκε ανεπαρκής χωρίς πολιτικές και τεκμηρίωση· υλοποιήθηκαν αρχεία καταγραφής πρόσβασης στα δεδομένα ασθενών και πρότυπο BAA, με το συμπέρασμα ότι η κανονιστική συμμόρφωση είναι κατά το ήμισυ τεχνική και κατά το ήμισυ νομική και διαδικαστική υπόθεση. Τρίτο δίδαγμα: οι ηλικιωμένοι χρήστες χρειάζονται καθοδηγούμενη ένταξη, διότι οι «δαισθητικές» διεπαφές δεν είναι καθολικά δαισθητικές. Οι δοκιμές αποκάλυψαν ότι οι ασθενείς δεν αντιλαμβάνονταν ότι το κουμπί «Προσθήκη δεδομένων υγείας» σήμαινε «καταγραφή της αρτηριακής πίεσης»· προστέθηκαν

οδηγός πρώτης χρήσης και βίντεο βοήθειας, επιβεβαιώνοντας ότι η προσβασιμότητα δεν αφορά μόνο τον σχεδιασμό της διεπαφής, αλλά περιλαμβάνει εκπαιδευτικό υλικό, υποστήριξη και προοδευτική αποκάλυψη λειτουργιών.

7.2.3 Μαθήματα από τη διαδικασία ανάπτυξης λογισμικού

Η αξία της επαναληπτικής ανάπτυξης επιβεβαιώθηκε εμπράκτως: η εκ των προτέρων κατασκευή πλήρων λειτουργιών αποδείχθηκε σπατάλη, καθώς οι χρήστες ζητούν αλλαγές μόλις δουν το λογισμικό σε λειτουργία. Δύο εβδομάδες επενδύθηκαν σε ένα σύνθετο εργαλείο ελέγχου αλληλεπιδράσεων φαρμάκων που οι χρήστες βρήκαν δυσνόητο, προτιμώντας έναν απλό κατάλογο φαρμάκων· έκτοτε υιοθετήθηκαν κύκλοι δύο εβδομάδων με κατασκευή ελάχιστων λειτουργιών, δοκιμή από χρήστες και επανάληψη - οι βρόχοι ανατροφοδότησης είναι απαραίτητοι για την εμπειρία χρήστη στην υγειονομική περίθαλψη. Εξίσου καθοριστική ήταν η ανάπτυξη βάσει δοκιμών (TDD) για τις κρίσιμες διαδρομές: ένα σφάλμα μετατόπισης κατά ένα στη λήξη των JWT (λήξη στις 23 αντί για 24 ώρες) εντοπίστηκε από δοκιμή μονάδας που είχε γραφτεί πριν από την υλοποίηση της λογικής αυθεντικοποίησης (TestTokenExpiry, TestPasswordHashing, TestRoleBasedAccess)· το πρόσθετο κόστος της TDD αποδίδει σε κώδικα κρίσιμο για την ασφάλεια, αφού απέτρεψε συμβάν στην παραγωγή. Τέλος, επιβεβαιώθηκε ότι η τεκμηρίωση είναι αναπόσπαστο μέρος του κώδικα: επιστρέφοντας στον κώδικα συγκρότησης πλαισίου της τεχνητής νοημοσύνης έπειτα από τρεις μήνες, ούτε ο αρχικός συγγραφέας δεν θυμόταν γιατί το πλαίσιο του ασθενούς διαφέρει από εκείνο του διαχειριστή. Η λύση ήταν ενσωματωμένα σχόλια που εξηγούν το «γιατί» και όχι μόνο το «τι» - ο μελλοντικός εαυτός του προγραμματιστή είναι και αυτός χρήστης του κώδικα, και οι αρχιτεκτονικές αποφάσεις με τη μη προφανή λογική πρέπει να τεκμηριώνονται.

7.3 Μελλοντικές βελτιώσεις

Αυτή η ενότητα κατηγοριοποιεί συστηματικά τις μελλοντικές εργασίες σε τρεις χρονικούς ορίζοντες με κατάταξη προτεραιότητας, εκτιμήσεις πολυπλοκότητας υλοποίησης και αναμενόμενο αντίκτυπο.

7.3.1 Βραχυπρόθεσμες βελτιώσεις (0-6 μήνες)

Στον βραχυπρόθεσμο ορίζοντα εντοπίζονται επτά βελτιώσεις. Δύο από αυτές αφορούν την ασφάλεια και έχουν υψηλή προτεραιότητα. Η πρώτη είναι η εναλλαγή διακριτικών, δηλαδή η υιοθέτηση διακριτικών πρόσβασης μικρής διάρκειας δεκαπέντε λεπτών σε συνδυασμό με διακριτικά ανανέωσης επτά ημερών αποθηκευμένα σε cookies httpOnly, ώστε ένα κλεμμένο διακριτικό πρόσβασης να λήγει γρήγορα και η κατάχρηση διακριτικού ανανέωσης να είναι ανιχνεύσιμη· απαιτεί μοντέλο RefreshToken, τελικό σημείο /auth/refresh και λογική εναλλαγής στο middleware. Η δεύτερη είναι ο περιορισμός ρυθμού αιτήσεων σε πέντε προσπάθειες σύνδεσης ανά λεπτό και ανά διεύθυνση IP, με χρήση περιοριστή υποστηριζόμενου από Redis και εκθετικής αναβολής σε επαναλαμβανόμενες αποτυχίες, που αποτρέπει επιθέσεις εξαντλητικής δοκιμής και μειώνει το φορτίο από αυτοματοποιημένη κίνηση.

Δύο ακόμη βελτιώσεις υψηλής προτεραιότητας αφορούν τη διασφάλιση ποιότητας: μια αυτοματοποιημένη υποδομή συγκριτικής αξιολόγησης με σενάρια JMeter και πίνακες ελέγχου Grafana, η οποία θα εκτελείται κάθε βράδυ σε περιβάλλον δοκιμών και θα ειδοποιεί όταν η καθυστέρηση στο ενενηκοστό πέμπτο εκατοστημόριο υπερβαίνει τα 500 ms, ώστε οι υποβαθμίσεις απόδοσης να εντοπίζονται πριν φτάσουν στην παραγωγή· και ένας επαγγελματικός έλεγχος προσβασιμότητας κατά WCAG 2.1 AA με δοκιμή σε NVDA και JAWS και επακόλουθη αποκατάσταση, ο οποίος καλύπτει παράλληλα και τη νομική συμμόρφωση.

Οι υπόλοιπες τρεις βελτιώσεις στοχεύουν στην εμπλοκή των χρηστών και στη διεύρυνση του κοινού. Η αυτόματη δημιουργία εβδομαδιαίων αναφορών υγείας σε μορφή PDF, που αποστέλλονται με ηλεκτρονικό ταχυδρομείο, παράγει έγγραφο αξιοποιήσιμο στην επίσκεψη στον ιατρό. Οι ειδοποιήσεις υπενθύμισης φαρμάκων μέσω του Push API του περιηγητή, με τη σχετική διαχείριση αδειών και ζωνών ώρας, βελτιώνουν τη συμμόρφωση στη φαρμακευτική

αγωγή, κρίσιμη παράμετρο στη διαχείριση χρόνιων παθήσεων. Τέλος, η θεμελίωση υποδομής διεθνοποίησης με react-i18next και go-i18n αποτελεί το πρώτο βήμα για την υποστήριξη πρόσθετων γλωσσών και μη αγγλόφωνων πληθυσμών.

Η προτεινόμενη ακολουθία υλοποίησης ξεκινά τους δύο πρώτους μήνες με την εναλλαγή ανανεωτικών tokens και τον περιορισμό ρυθμού αιτήσεων, ως κρίσιμα μέτρα ασφάλειας· συνεχίζεται τον τρίτο και τέταρτο μήνα με τη συγκριτική αξιολόγηση απόδοσης και τον έλεγχο προσβασιμότητας, για τη διασφάλιση της ποιότητας· και ολοκληρώνεται τον πέμπτο και έκτο μήνα με τις αυτοματοποιημένες αναφορές και τις υπενθυμίσεις φαρμακευτικής αγωγής, που ενισχύουν την εμπλοκή των χρηστών.

7.3.2 Βελτιώσεις μεσοπρόθεσμα (6-18 μήνες)

Στον μεσοπρόθεσμο ορίζοντα προτεραιότητα έχουν τέσσερις βελτιώσεις. Η ενσωμάτωση συσκευών IoT επιτρέπει την άμεση εισαγωγή μετρήσεων από πιεσόμετρα Omron, μετρητές γλυκόζης OneTouch και έξυπνες ζυγαριές Withings μέσω του Web API του Bluetooth Low Energy, εξαλείφοντας τα σφάλματα χειροκίνητης καταχώρισης και αυξάνοντας τη συχνότητα μέτρησης. Η προγνωστική ανάλυση με μηχανική μάθηση, με μοντέλα scikit-learn εκπαιδευμένα σε ιστορικά δεδομένα, στοχεύει στην πρόβλεψη του κινδύνου υπέρτασης, των τάσεων βάρους και της πιθανότητας μη τήρησης της αγωγής, απαιτεί όμως μεγάλο όγκο δεδομένων και έγκριση επιτροπής ηθικής και δεοντολογίας της έρευνας. Οι εγγενείς εφαρμογές για iOS και Android σε React Native, με συγχρονισμό offline-first, βιομετρική ταυτοποίηση και οπτική αναγνώριση χαρακτήρων για ετικέτες φαρμάκων, απαντούν στο γεγονός ότι το 61% των ατόμων άνω των 65 ετών διαθέτει έξυπνο τηλέφωνο. Τέλος, η ενσωμάτωση με ηλεκτρονικά αρχεία υγείας κατά το πρότυπο HL7 FHIR R4, με εξουσιοδότηση SMART on FHIR, επιτρέπει αμφίδρομο συγχρονισμό και κλείνει τα σιλό δεδομένων ανάμεσα στον ασθενή και τον θεράποντα ιατρό.

Μεσαία προτεραιότητα έχουν η ενσωμάτωση τηλεϊατρικής με βιντεοκλήσεις μέσω WebRTC και κοινή χρήση οθόνης με τα διαγράμματα υγείας κατά τη διάρκεια της συνεδρίας, καθώς και ο έλεγχος αλληλεπιδράσεων φαρμάκων μέσω του API του DrugBank, ο οποίος επισημαίνει επικίνδυνους συνδυασμούς και πρέπει υποχρεωτικά να συνοδεύεται από δήλωση ότι δεν υποκαθιστά την αξιολόγηση φαρμακοποιού. Χαμηλότερη προτεραιότητα έχουν τα κοινωνικά χαρακτηριστικά, δηλαδή φόρουμ ασθενών με παρόμοιες παθήσεις και ομάδες υποστήριξης φροντιστών, τα οποία αντιμετωπίζουν την κοινωνική απομόνωση αλλά προϋποθέτουν ανθρώπινη εποπτεία ή αυτόματη ανίχνευση τοξικού περιεχομένου.

Οι βασικές προκλήσεις της περιόδου αυτής είναι τέσσερις. Ο κατακερματισμός των συσκευών IoT - περισσότερα από εκατό μοντέλα πιεσόμετρων Bluetooth με ασύμβατα πρωτόκολλα - απαιτεί τη δημιουργία επιπέδου αφαίρεσης. Η μεταβολή των μοντέλων μηχανικής μάθησης, καθώς ο πληθυσμός των ασθενών αλλάζει με τον χρόνο, απαιτεί υποδομή συνεχούς επανεκπαίδευσης. Η διανομή εφαρμογών για κινητά προσκρούει στο γεγονός ότι η έγκριση ιατρικών εφαρμογών από τα App Store και Play Store προϋποθέτει προ-εμπορική κοινοποίηση στην FDA (αίτηση εξαίρεσης 510(k)). Τέλος, οι συνεργασίες με συστήματα EHR (Epic, Cerner, Meditech) απαιτούν επιχειρησιακή ανάπτυξη: τα τεχνικά πρότυπα υπάρχουν, αλλά χρειάζεται και η συνεργασία των προμηθευτών.

7.3.3 Μακροπρόθεσμη προοπτική (18+ μήνες)

Στον μακροπρόθεσμο ορίζοντα οι προτεινόμενες κατευθύνσεις έχουν ερευνητικό χαρακτήρα και πολύ υψηλή πολυπλοκότητα. Η αξιολόγηση κινδύνου με μοντέλα βαθιάς μάθησης — πρόβλεψη επανεισαγωγής εντός τριάντα ημερών, πιθανότητας πτώσης και έναρξης γνωστικής έκπτωσης — θα επέτρεπε προληπτικές παρεμβάσεις πριν από την εκδήλωση κρίσης, απαιτεί όμως σύνολα δεδομένων εκατοντάδων χιλιάδων ασθενών, έγκριση από τον FDA ως σύστημα υποστήριξης κλινικών αποφάσεων και εξωτερικές μελέτες επικύρωσης. Στην ίδια κατεύθυνση κινείται και το σύστημα υποστήριξης κλινικών αποφάσεων για ιατρούς, το οποίο κατατάσσεται ως ιατροτεχνολογικό προϊόν κατηγορίας II και προϋποθέτει κλινικές δοκιμές και έγκριση κατά τη διαδικασία 510(k).

Η ομοσπονδιακή μάθηση προσφέρει διέξοδο στο δίλημμα ανάμεσα στη γνώση σε επίπεδο πληθυσμού και στην προστασία της ιδιωτικότητας, καθώς επιτρέπει τη συνεργατική εκπαίδευση μοντέλων μεταξύ πολλών φορέων υγείας χωρίς συγκέντρωση των δεδομένων των ασθενών, με χρήση πρωτοκόλλων ασφαλούς συνάθροισης και εγγυήσεων διαφορετικής ιδιωτικότητας. Η ενσωμάτωση με φορετές συσκευές όπως το Apple Watch και το Fitbit, μέσω των HealthKit, Google Fit και Fitbit Web API, θα μετέτρεπε τη συλλογή δεδομένων από χειροκίνητη σε παθητική και συνεχή, με αντίστοιχο όμως ζήτημα την επικύρωση της ποιότητας των μετρήσεων.

Τρεις ακόμη κατευθύνσεις έχουν χαμηλότερη προτεραιότητα αλλά σαφές ενδιαφέρον για τον συγκεκριμένο πληθυσμό. Η φωνητική διεπαφή μέσω Alexa Skills Kit και Google Assistant Actions εξυπηρετεί χρήστες με προβλήματα όρασης ή αρθρίτιδα, εγείρει όμως ζητήματα ιδιωτικότητας λόγω των συσκευών διαρκούς ακρόασης. Η ενσωμάτωση γονιδιωμάτων δεδομένων επιτρέπει εξατομικευμένη εκτίμηση κινδύνου, με τα συνακόλουθα ηθικά ζητήματα της γενετικής διάκρισης στις ασφαλίσεις και της ψυχολογικής επιβάρυνσης του ασθενούς. Τέλος, η καθοδήγηση φαρμακευτικής αγωγής με επταυξημένη πραγματικότητα, όπου η κάμερα του τηλεφώνου αναγνωρίζει το φιαλίδιο και προβάλλει τις οδηγίες δοσολογίας, θα περιόριζε τα λάθη σε ασθενείς με πολυφαρμακία.

Μακροπρόθεσμα διαγράφονται τρεις ερευνητικές κατευθύνσεις. Η πρώτη είναι η εξηγήσιμη τεχνητή νοημοσύνη για την υγειονομική περίθαλψη: τα τρέχοντα μοντέλα λειτουργούν ως «μαύρα κουτιά» και οι γιατροί δεν εμπιστεύονται αδιαφανείς συστάσεις· η έρευνα σε τεχνικές όπως το LIME (Local Interpretable Model-agnostic Explanations) και οι μηχανισμοί προσοχής στοχεύει σε αιτιολογημένες προβλέψεις της μορφής «το μοντέλο προέβλεψε υψηλό κίνδυνο επανεισαγωγής επειδή ο ασθενής είχε τρεις επισκέψεις στα επείγοντα τον τελευταίο μήνα και αρρύθμιστο διαβήτη». Η δεύτερη κατεύθυνση είναι η σύντηξη πολυτροπικών δεδομένων υγείας - δομημένων (ζωτικά σημεία), μη δομημένων (σημειώσεις ιατρών), εικόνων (φωτογραφίες δερματικών βλαβών) και γονιδιωμάτων - μέσω αρχιτεκτονικών μετασχηματισμών για πολυτροπική μάθηση και αυτοεποπτευόμενης προεκπαίδευσης σε μη επισημειωμένα δεδομένα, με στόχο την ολοκληρωμένη κατανόηση της υγείας από την τεχνητή νοημοσύνη βάσει του πλήρους ιατρικού ιστορικού. Η τρίτη κατεύθυνση αφορά τη συμπεριφορική «ώθηση» για τη συμμόρφωση: την εφαρμογή αρχών συμπεριφορικής οικονομικής (παιχνιδοποίηση, κοινωνική απόδειξη, μηχανισμοί δέσμευσης) για τη βελτίωση της τήρησης της φαρμακευτικής αγωγής, με δοκιμές A/B διαφορετικών στρατηγικών και εξατομικευμένο χρονισμό παρεμβάσεων, ώστε η συμμόρφωση να αυξηθεί από το τρέχον μέσο 50% σε ποσοστά άνω του 80%.

7.4 Αντίκτυπος στον κλάδο και εφαρμογές

7.4.1 Δυνατότητες εμπορικής αξιοποίησης

Η ευκαιρία αγοράς είναι σημαντική: η συνολική διαθέσιμη αγορά (TAM) περιλαμβάνει 56 εκατομμύρια Αμερικανούς άνω των 65 ετών (απογραφή 2020), με πρόβλεψη 73 εκατομμυρίων έως το 2030· η εξυπηρετήσιμη αγορά (SAM) αφορά 34 εκατομμύρια ηλικιωμένους με χρόνιες παθήσεις που απαιτούν παρακολούθηση (στοιχεία Medicare)· και ένα ρεαλιστικά εφικτό μερίδιο 1% (SOM) αντιστοιχεί σε 340.000 χρήστες. Ως προς τα μοντέλα εσόδων, μια προσέγγιση freemium SaaS θα περιλάμβανε δωρεάν επίπεδο με βασική παρακολούθηση (αρτηριακή πίεση, βάρος), διατήρηση δεδομένων 30 ημερών και δέκα ερωτήματα τεχνητής νοημοσύνης μηνιαίως· premium επίπεδο (9,99 δολάρια/μήνα) με απεριόριστη διατήρηση δεδομένων και ερωτήματα, αυτοματοποιημένες αναφορές και υπενθυμίσεις φαρμάκων· και πρόγραμμα φροντιστή (19,99 δολάρια/μήνα) με διαχείριση έως πέντε ασθενών, προτεραιότητα υποστήριξης και ενσωμάτωση τηλεϊατρικής.

Στο πεδίο των συνεργασιών B2B, τα προγράμματα εξόδου από το νοσοκομείο μπορούν να μειώσουν τις επανεισαγωγές 30 ημερών μέσω απομακρυσμένης παρακολούθησης με κόστος 50 δολαρίων ανά ασθενή μηνιαίως - οι κυρώσεις του Medicare για επανεισαγωγές δημιουργούν σαφές οικονομικό κίνητρο· οι οργανισμοί κατ' οίκον φροντίδας μπορούν να αξιοποιήσουν πλατφόρμα «λευκής ετικέτας» έναντι 10.000 δολαρίων ετησίως συν 2 δολάρια ανά ασθενή

μηνιαίως· και τα προγράμματα Medicare Advantage μπορούν να τη χρησιμοποιήσουν ως εργαλείο διαχείρισης της υγείας του πληθυσμού έναντι 5 δολαρίων ανά μέλος μηνιαίως, αφού η προληπτική φροντίδα μειώνει το κόστος των αποζημιώσεων. Η αξιοποίηση των δεδομένων για έρευνα - ανώνυμα σύνολα δεδομένων για τη μελέτη της αποτελεσματικότητας φαρμάκων σε πραγματικές συνθήκες, αξίας άνω των 100.000 δολαρίων ανά σύνολο - προϋποθέτει αυστηρούς ηθικούς όρους: ρητή συγκατάθεση, έγκριση επιτροπής δεοντολογίας (IRB) και μηδενικό κίνδυνο επανатаυτοποίησης. Μια ενδεικτική στρατηγική εισόδου στην αγορά θα ξεκινούσε το πρώτο έτος με απευθείας πωλήσεις σε καταναλωτές μέσω στοχευμένης διαφήμισης (10.000 χρήστες), θα συνέχιζε το δεύτερο έτος με συνεργασίες με πέντε υπηρεσίες κατ' οίκον φροντίδας (10.000 ασθενείς υπό διαχείριση) και θα ολοκληρωνόταν το τρίτο έτος με ενσωμάτωση σε πρόγραμμα Medicare Advantage (100.000 καλυπτόμενοι ασθενείς).

7.4.2 Συνεισφορές στην ακαδημαϊκή έρευνα

Οι καινοτόμες συνεισφορές της εργασίας στην έρευνα είναι τρεις. Πρώτον, η αρχιτεκτονική τεχνητής νοημοσύνης για την υγεία με διαφύλαξη της ιδιωτικότητας, που αποδεικνύει ότι η τοπική ανάπτυξη μεγάλων γλωσσικών μοντέλων αποτελεί βιώσιμη εναλλακτική έναντι των cloud API για εφαρμογές ευαίσθητες σε προστατευόμενες πληροφορίες υγείας, με προοπτική δημοσίευσης μελέτης περίπτωσης για τους συμβιβασμούς ιδιωτικότητας, κόστους και καθυστέρησης μεταξύ τοπικών και cloud μοντέλων. Δεύτερον, τα πρότυπα σχεδίασης διεπαφών προσβάσιμων σε ηλικιωμένους, που κωδικοποιούν ένα σύστημα σχεδιασμού για χρήστες άνω των 65 ετών - κλίμακες τυπογραφίας, αναλογίες χρωματικής αντίθεσης, μεγέθη στόχων αφής - με προοπτική διάθεσης βιβλιοθήκης στοιχείων React ανοιχτού κώδικα βελτιστοποιημένης για ηλικιωμένους. Τρίτον, τα μοντέλα συνεργασίας πολλαπλών ρόλων στην υγειονομική περίθαλψη, με ανάλυση των ρόλων συγκατάθεσης φροντιστή-ασθενούς (ποσοστά αποδοχής προσκλήσεων, κατανομές μεγέθους νοικοκυριών) και προοπτική δημοσίευσης για τον σχεδιασμό μηχανισμών συγκατάθεσης στην πρόσβαση οικογενειακών φροντιστών σε δεδομένα υγείας. Ως στόχοι δημοσίευσης προσδιορίζονται το ACM CHI για τα πρότυπα προσβασιμότητας σε διεπαφές διαχείρισης της υγείας ηλικιωμένων, το AMIA για τα τοπικά μεγάλα γλωσσικά μοντέλα στην υποστήριξη κλινικών αποφάσεων με διαφύλαξη της ιδιωτικότητας, και το IEEE EMBC για το πολυδιάστατο σύστημα παρακολούθησης της υγείας στη διαχείριση χρόνιων παθήσεων γηρασκόντων πληθυσμών.

7.4.3 Ηθικές σκέψεις

Όσον αφορά την ιδιοκτησία των δεδομένων και τη συγκατάθεση, θεμελιώδης αρχή είναι ότι οι ασθενείς είναι κύριοι των δεδομένων υγείας τους: το σύστημα διευκολύνει την εξαγωγή και τη διαγραφή τους, σύμφωνα και με το «δικαίωμα στη λήθη» του GDPR, μέσω των τελικών σημείων `/api/export-all-data`, που επιστρέφει το πλήρες σύνολο των δεδομένων, και `/api/delete-account`, που διαγράφει όλα τα αρχεία. Παραμένει, ωστόσο, η πρόκληση ότι η διαγραφή μπορεί να συγκρούεται με τις απαιτήσεις ελέγχου, αφού ο HIPAA επιβάλλει τη διατήρηση ορισμένων αρχείων επί έξι έτη. Ως προς την προκατάληψη και τη δικαιοσύνη της τεχνητής νοημοσύνης, ο κίνδυνος είναι μοντέλα εκπαιδευμένα κυρίως σε δεδομένα λευκών ανδρών να αποδίδουν χειρότερα για γυναίκες και μειονότητες· ο μετρισμός απαιτεί συλλογή ποικιλόμορφων δεδομένων εκπαίδευσης, μετρικές δικαιοσύνης με ίσα ποσοστά σφάλματος μεταξύ δημογραφικών ομάδων, τακτικούς ελέγχους μεροληψίας και διαφάνεια προς τους χρήστες σχετικά με τους περιορισμούς του μοντέλου (π.χ. «αυτή η τεχνητή νοημοσύνη έχει εκπαιδευτεί κυρίως με δεδομένα ασθενών ηλικίας 65-85 ετών με υπέρταση»).

Συναφές είναι το ζήτημα της αλγοριθμικής διαφάνειας: οι ασθενείς πρέπει να κατανοούν πώς η τεχνητή νοημοσύνη καταλήγει στα συμπεράσματά της. Στο σύστημα αυτό υλοποιείται με αιτιολογημένες απαντήσεις - «σας συνιστώ να συμβουλευτείτε γιατρό, επειδή η αρτηριακή σας πίεση (142/90) υπερβαίνει το όριο υπέρτασης σταδίου 1 της AHA (130/80)» - ενώ μελλοντικά προβλέπεται η αξιοποίηση τεχνικών εξηγήσιμης τεχνητής νοημοσύνης, όπως οι τιμές SHAP, για την ανάδειξη της σημασίας των χαρακτηριστικών. Τέλος, το ψηφιακό χάσμα παραμένει ανοιχτό πρόβλημα, καθώς οι ηλικιωμένοι χωρίς πρόσβαση στο διαδίκτυο, smartphone ή ψηφιακές δεξιότητες αποκλείονται από τα οφέλη. Οι πιθανές απαντήσεις περιλαμβάνουν συνεργασίες με κέντρα ηλικιωμένων για εκπαίδευση στη χρήση υπολογιστών, εισαγωγή δεδομένων σε έντυπη

μορφή με ταχυδρομική αποστολή και ψηφιοποίηση από το προσωπικό, καθώς και αμιγώς φωνητική διεπαφή, όπου τα δεδομένα καταγράφονται μέσω τηλεφωνικής κλήσης από φωνητικό βοηθό.

Συνοψίζοντας, το κεφάλαιο αυτό επικύρωσε την επίτευξη και των οκτώ στόχων της διατριβής με υποστηρικτικά στοιχεία: ασφαλή αυθεντικοποίηση (bcrypt και JWT), ολοκληρωμένα προφίλ με περισσότερα από δεκαπέντε χαρακτηριστικά, πολυδιάστατη παρακολούθηση έντεκα παραμέτρων υγείας, τεχνητή νοημοσύνη που διαφυλάσσει την ιδιωτικότητα μέσω τοπικού Ollama, συνεργασία φροντιστών με προσκλήσεις νοικοκυριού, προσβάσιμο περιβάλλον χρήστη θεμελιωμένο στο WCAG 2.1 AA, ανάπτυξη στο cloud με Terraform και επεκτάσιμη, προσανατολισμένη στις υπηρεσίες αρχιτεκτονική. Τεκμηριώθηκαν περισσότερες από δέκα προκλήσεις υλοποίησης με ανάλυση των αιτιών και τις λύσεις τους - από τη διαχείριση μετεγκαταστάσεων του GORM και την επικύρωση χρόνου εκτέλεσης στην TypeScript έως τη δικτύωση Docker, το κλειδίωμα κατάστασης Terraform, την πολυπλοκότητα του HIPAA και την ένταξη των ηλικιωμένων χρηστών. Διαμορφώθηκε, τέλος, δομημένος χάρτης πορείας με περισσότερες από είκοσι μελλοντικές βελτιώσεις σε τρεις χρονικούς ορίζοντες, με ιεράρχηση προτεραιότητας και εκτιμήσεις πολυπλοκότητας: εναλλαγή tokens και έλεγχοι ποιότητας βραχυπρόθεσμα, συσκευές IoT, αναλυτική μηχανικής μάθησης και κινητές εφαρμογές μεσοπρόθεσμα, ομοσπονδιακή μάθηση και υποστήριξη κλινικών αποφάσεων μακροπρόθεσμα, σε συνδυασμό με τις οδούς εμπορικής αξιοποίησης, τις ακαδημαϊκές συνεισφορές και τις ηθικές παραμέτρους που αναλύθηκαν.

Το σύστημα αποδεικνύει την τεχνική βιωσιμότητα της τεχνητής νοημοσύνης στον τομέα της υγειονομικής περίθαλψης με προτεραιότητα στην προστασία της ιδιωτικής ζωής, επικυρώνει μοντέλα συνεργασίας φροντιστών για τον γηράσκοντα πληθυσμό και θεμελιώνει τις βάσεις για την επεκτασιμότητα, υποστηρίζοντας την δεκαετή εξέλιξη προς ολοκληρωμένες ψηφιακές πλατφόρμες υγείας.

Κεφάλαιο 8: Βιβλιογραφία

Στο κεφάλαιο αυτό παρατίθεται ενιαίος κατάλογος των έργων στα οποία παραπέμπει η εργασία. Οι βιβλιογραφικές αναφορές δίνονται αλφαβητικά, στη γλώσσα και τη μορφή της πρωτότυπης δημοσίευσης, και ακολουθούν οι διαδικτυακές πηγές.

Βιβλιογραφικές αναφορές

1. Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., Harrison, M., Hewett, R.J., Javaheripi, M., Kauffmann, P. and Lee, J.R., 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.
2. Alepis, E. and Virvou, M., 2014. Object-oriented user interfaces for personalized mobile learning. Springer.
3. Alepis, E. and Virvou, M., 2006, October. Emotional Intelligence: Constructing user stereotypes for affective bi-modal interaction. In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (pp. 435-442). Berlin, Heidelberg: Springer Berlin Heidelberg.
4. Alepis, E., Stathopoulou, I.O., Virvou, M., Tsihrintzis, G.A. and Kabassi, K., 2010, October. Audio-lingual and visual-facial emotion recognition: Towards a bi-modal interaction system. In 2010 22nd IEEE International Conference on Tools with Artificial Intelligence (Vol. 2, pp. 274-281). IEEE.

5. Alepis, E., Virvou, M. and Kabassi, K., 2006, July. Affective student modeling based on microphone and keyboard user actions. In Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06) (pp. 139-141). IEEE.
6. Alogogianni, E. and Virvou, M., 2021. Addressing the issue of undeclared work–Part I: Applying associative classification per the CRISP-DM methodology. *Intelligent Decision Technologies*, 15(4), pp.721-747.
7. Alonistioti, N., Tsihrintzi, E.A., Chrysafiadi, K. and Alepis, E., 2023, July. Requirements for Fuzzy Logic in Personalisation of Fire Emergency Alerts. In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
8. Appari, A. and Johnson, M.E., 2010. Information security and privacy in healthcare: current state of research. *International journal of Internet and enterprise management*, 6(4), pp.279-314.
9. Ayaz, M., Pasha, M.F., Alzahrani, M.Y., Budiarto, R. and Stiawan, D., 2021. The Fast Health Interoperability Resources (FHIR) standard: systematic literature review of implementations, applications, challenges and opportunities. *JMIR medical informatics*, 9(7), p.e21929.
10. Bertino, E., 2003. RBAC models-concepts and trends. *Computers & Security*, 22(6), pp.511-514.
11. Berwick, D.M. and Nolan, T.W., 1998. Physicians as leaders in improving health care: a new series in Annals of Internal Medicine. *Annals of Internal Medicine*, 128(4), pp.289-292.
12. Burde, H., 2011. The HITECH act: an overview. *AMA Journal of Ethics*, 13(3), pp.172-175.
13. Cavoukian, A., 2011. Privacy by design in law, policy and practice. *A white paper for regulators, decision-makers and policy-makers*.
14. Chi, N.C. and Demiris, G., 2015. A systematic review of telehealth tools and interventions to support family caregivers. *Journal of telemedicine and telecare*, 21(1), pp.37-44.
15. Chrysafiadi, K. and Virvou, M., 2013. A knowledge representation approach using fuzzy cognitive maps for better navigation support in an adaptive learning system. *SpringerPlus*, 2, pp.1-13.
16. Chrysafiadi, K. and Virvou, M., 2015. *Advances in personalized web-based education*. Springer International Publishing.
17. Chrysafiadi, K. and Virvou, M., 2013. Dynamically personalized e-training in computer programming and the language C. *IEEE transactions on education*, 56(4), pp.385-392.
18. Chrysafiadi, K., Papadimitriou, S. and Virvou, M., 2022. Cognitive-based adaptive scenarios in educational games using fuzzy reasoning. *Knowledge-Based Systems*, 250, p.109111.
19. Chrysafiadi, K., Virvou, M. and Tsihrintzis, G.A., 2022. A fuzzy-based mechanism for automatic personalized assessment in an e-learning system for computer programming. *Intelligent Decision Technologies*, 16(4), pp.699-714.
20. Chrysafiadi, K., Virvou, M., Tsihrintzis, G.A. and Hatzilygeroudis, I., 2023. An Adaptive Learning Environment for Programming Based on Fuzzy Logic and Machine Learning. *International Journal on Artificial Intelligence Tools*, 32(05), p.2360011.
21. Chrysafiadi, K., Virvou, M., Tsihrintzis, G.A. and Hatzilygeroudis, I., 2023. Evaluating the user's experience, adaptivity and learning outcomes of a fuzzy-based intelligent tutoring system for computer programming for academic students in Greece. *Education and Information Technologies*, 28(6), pp.6453-6483.

22. Cohen, I.G. and Mello, M.M., 2018. HIPAA and protecting health information in the 21st century. *Jama*, 320(3), pp.231-232.
23. Donovan, A.A. and Kernighan, B.W., 2015. *The Go programming language*. Addison-Wesley Professional.
24. Fernández-Alemán, J.L., Señor, I.C., Lozoya, P.Á.O. and Toval, A., 2013. Security and privacy in electronic health records: A systematic literature review. *Journal of biomedical informatics*, 46(3), pp.541-562.
25. Gackenheimer, C., 2015. *Introduction to React*. Apress
26. Goldstein, M.M. and Pewen, W.F., 2013. The HIPAA Omnibus Rule: implications for public health policy and practice. *Public Health Reports*, 128(6), pp.554-558.
27. Green, B. and Seshadri, S., 2013. *AngularJS*. " O'Reilly Media, Inc."
28. Hatzilygeroudis, I. K., Tsihrintzis, G. A., & Jain, L. C. (Eds.). (2023). *Fusion of Machine Learning Paradigms: Theory and Applications* (Vol. 236). Springer Nature.
29. Holden, R.J. and Karsh, B.T., 2010. The technology acceptance model: its past and its future in health care. *Journal of biomedical informatics*, 43(1), pp.159-172.
30. Ismond, K.P., Loch-Ismond, E., Tandon, P. and Spiers, J., 2023. An evaluation of online cirrhosis-related information intended for patients and caregivers. *Clinical Gastroenterology and Hepatology*, 21(3), pp.835-837.
31. Jaar, D. and Zeller, P.E., 2008. Canadian privacy law: The personal information protection and electronic documents act (PIPEDA). *Int'l. In-House Counsel J.*, 2, p.1135.
32. Jones, M., Bradley, J. and Sakimura, N., 2015. *Json web token (jwt)* (No. rfc7519).
33. Kabassi, K. and Virvou, M., 2006. A knowledge-based software life-cycle framework for the incorporation of multicriteria analysis in intelligent user interfaces. *IEEE Transactions on Knowledge and data Engineering*, 18(9), pp.1265-1277.
34. Kadakia, K., Patel, B. and Shah, A., 2020. Advancing digital health: FDA innovation during COVID-19. *Npj Digital Medicine*, 3(1), p.161.
35. Kamitsios, M., Chrysafiadi, K., Virvou, M. and Sakkopoulos, E., 2018, July. A stereotype user model for an educational game: Overcome the difficulties in game playing and focus on the educational goal. In 2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA) (pp. 1-6). IEEE.
36. Katsionis, G. and Virvou, M., 2004, October. A cognitive theory for affective user modelling in a virtual reality educational game. In 2004 IEEE international conference on systems, man and cybernetics (IEEE Cat. No. 04CH37583) (Vol. 2, pp. 1209-1213). IEEE.
37. Katsionis, G. and Virvou, M., 2008. Personalised e-learning through an educational virtual reality game using Web services. *Multimedia Tools and Applications*, 39, pp.47-71.
38. Kim, Y., Xu, X., McDuff, D., Breazeal, C. and Park, H.W., 2024. Health-Ilm: Large language models for health prediction via wearable sensor data. *arXiv preprint arXiv:2401.06866*.
39. Kontogianni, A., Alepis, E., Virvou, M. and Patsakis, C., 2024. "Smart Tourism: The Impact of Artificial Intelligence and Blockchain". *Intelligent Systems Reference Library*, Springer. Electronic ISSN 1868-4408, Print ISSN 1868-4394
40. Kruse, C.S., Krowski, N., Rodriguez, B., Tran, L., Vela, J. and Brooks, M., 2017. Telehealth and patient satisfaction: a systematic review and narrative analysis. *BMJ open*, 7(8), p.e016242.

41. Lampropoulos, A.S. and Tsihrintzis, G.A., 2012, July. Evaluation of MPEG-7 descriptors for speech emotional recognition. In 2012 Eighth international conference on intelligent information hiding and multimedia signal processing (pp. 98-101). IEEE.
42. Lampropoulos, A.S., Lampropoulou, P.S. and Tsihrintzis, G.A., 2005, September. Musical Genre Classification Enhanced by Improved Source Separation Technique. In ISMIR (pp. 576-581).
43. Lampropoulos, A.S., Sotiropoulos, D.N. and Tsihrintzis, G.A., 2004, October. Individualization of music similarity perception via feature subset selection. In 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583) (Vol. 1, pp. 552-556). IEEE.
44. Marcondes, F.S., Gala, A., Magalhães, R., Perez de Britto, F., Durães, D. and Novais, P., 2025. Using ollama. In *Natural Language Analytics with Generative Large-Language Models: A Practical Approach with Ollama and Open-Source LLMs* (pp. 23-35). Cham: Springer Nature Switzerland.
45. Mougiakou, E. and Virvou, M., 2017, August. Based on GDPR privacy in UML: Case of e-learning program. In 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
46. Moundridou, M. and Virvou, M., 2001, August. Authoring and delivering adaptive Web-based textbooks using WEAR. In Proceedings IEEE International Conference on Advanced Learning Technologies (pp. 185-188). IEEE.
47. Norgaard, O., Furstrand, D., Klokke, L., Karnoe, A., Batterham, R., Kayser, L. and Osborne, R.H., 2015. The e-health literacy framework: A conceptual framework for characterizing e-health users and their interaction with e-health systems. *Knowledge Management & E-Learning*, 7(4), pp.522-540.
48. Nurhudatiana, A. and Seo, J.Y., 2020, February. An mHealth Application Redesign based on Nielsen's Usability Heuristics: A Case Study of Halodoc. In *Proceedings of the 2020 The 6th International Conference on E-Business and Applications* (pp. 85-89).
49. Panagoulas, D.P., Sarmas, E., Marinakis, V., Virvou, M., Tsihrintzis, G.A. and Doukas, H., 2023. Intelligent decision support for energy management: A methodology for tailored explainability of artificial intelligence analytics. *Electronics*, 12(21), p.4430.
50. Panagoulas, D.P., Sotiropoulos, D.N. and Tsihrintzis, G.A., 2021. Nutritional biomarkers and machine learning for personalized nutrition applications and health optimization. *Intelligent Decision Technologies*, 15(4), pp.645-653.
51. Panagoulas, D.P., Virvou, M. and Tsihrintzis, G.A., 2024. A novel framework for artificial intelligence explainability via the Technology Acceptance Model and Rapid Estimate of Adult Literacy in Medicine using machine learning. *Expert Systems with Applications*, 248, p.123375.
52. Panagoulas, D.P., Virvou, M. and Tsihrintzis, G.A., 2024. Augmenting large language models with rules for enhanced domain-specific interactions: The case of medical diagnosis. *Electronics*, 13(2), p.320.
53. Panagoulas, D.P., Virvou, M. and Tsihrintzis, G.A., 2022, October. A microservices-based iterative development approach for usable, reliable and explainable AI-infused medical applications using RUP. In 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 1028-1035). IEEE.
54. Papadimitriou, S. and Virvou, M., 2017, August. Adaptivity in scenarios in an educational adventure game. In 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-6). IEEE.

55. Papadimitriou, S., Chrysafiadi, K. and Virvou, M., 2019. FuzzEG: Fuzzy logic for adaptive scenarios in an educational adventure game. *Multimedia Tools and Applications*, 78(22), pp.32023-32053.
56. Papadimitriou, S., Chrysafiadi, K. and Virvou, M., 2023, July. Adaptive quizzes using fuzzy genetic algorithm. In 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
57. Papadimitriou, S., Kamitsios, M., Chrysafiadi, K. and Virvou, M., 2021. Learn-and-play personalised reasoning from point-and-click to virtual reality mobile educational games. *Intelligent Decision Technologies*, 15(2), pp.321-332.
58. Patel, S., Park, H., Bonato, P., Chan, L. and Rodgers, M., 2012. A review of wearable sensors and systems with application in rehabilitation. *Journal of neuroengineering and rehabilitation*, 9(1), p.21.
59. Pavlakis, P., Alepis, E. and Virvou, M., 2012, July. Intelligent mobile multimedia application for the support of the elderly. In 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (pp. 297-300). IEEE.
60. Politou, E.A. Efthimios Alepis, E., Virvou, M., Patsakis, C. : Privacy and Data Protection Challenges in the Distributed Era. Springer 2022, ISBN 978-3-030-85442-3, pp. 1-185, Springer
61. Revesz, P., 2010. Introduction to databases. *Texts in Computer Science*.
62. Rosmala, D. and Rasyidin, I., 2024. Data Storage Database PostgreSQL and JSON File Format in Golang Using Gin-Gonic and Gin-Gonic with GORM. *Industrial Sciencetech Journal*, 1(1), pp.33-44.
63. Savvopoulos, A. and Virvou, M., 2010. Evaluating the Generality of a Life-Cycle Framework for Incorporating Clustering Algorithms in Adaptive Systems. In *Multimedia Services in Intelligent Environments: Software Development Challenges and Solutions* (pp. 5-25). Berlin, Heidelberg: Springer Berlin Heidelberg.
64. Savvopoulos, A., Virvou, M., Sotiropoulos, D.N. and Tsihrintzis, G.A., 2008. Clustering for user modeling in recommender e-commerce application: A rup-based intelligent software life-cycle. In *Knowledge-Based Software Engineering* (pp. 295-304). IOS Press.
65. Seeman, L. and Cooper, M., 2017. Cognitive Accessibility Roadmap and Gap Analysis. W3C First Public Working Draft. *W3C/WAI. Retrieved March, 1*, p.2022.
66. Seng, B.K., Luo, N., Ng, W.Y., Lim, J., Chionh, H.L., Goh, J. and Yap, P., 2010. Validity and reliability of the Zarit Burden Interview in assessing caregiving burden. *Annals of the Academy of Medicine, Singapore*, 39(10), pp.758-763.
67. Sotiropoulos, D.N., Tsihrintzis, G.A., Savvopoulos, A. and Virvou, M., 2006. Artificial immune system-based customer data clustering in an e-shopping application. In *Knowledge-Based Intelligent Information and Engineering Systems: 10th International Conference, KES 2006, Bournemouth, UK, October 9-11, 2006. Proceedings, Part I 10* (pp. 960-967). Springer Berlin Heidelberg.
68. Spina, C., 2019. WCAG 2.1 and the current state of web accessibility in libraries. *Weave: Journal of Library User Experience*, 2(2).
69. Stathopoulou, I.O., Alepis, E., Tsihrintzis, G.A. and Virvou, M., 2010. On assisting a visual-facial affect recognition system with keyboard-stroke pattern information. *Knowledge-Based Systems*, 23(4), pp.350-356.
70. Stathopoulou, I.O. and Tsihrintzis, G.A., 2011. Appearance-based face detection with artificial neural networks. *Intelligent Decision Technologies*, 5(2), pp.101-111.

71. Stathopoulou, I.O. and Tsihrintzis, G.A., 2011, July. Emotion recognition from body movements and gestures. In *Intelligent Interactive Multimedia Systems and Services: Proceedings of the 4th International Conference on Intelligent Interactive Multimedia Systems and Services (IIMSS 2011)* (pp. 295-303). Berlin, Heidelberg: Springer Berlin Heidelberg.
72. Stathopoulou, I.O. and Tsihrintzis, G.A., 2005, November. Detection and expression classification systems for face images (FADECS). In *IEEE Workshop on Signal Processing Systems Design and Implementation*, 2005. (pp. 353-358). IEEE.
73. Stathopoulou, I.O. and Tsihrintzis, G.A., 2010. *Visual affect recognition* (Vol. 214). IOS Press.
74. Theocharis, S.A. and Tsihrintzis, G.A., 2016. Knowledge management systems in the public sector: Critical issues. *Lecture Notes on Software Engineering*, 4(1), p.59.
75. Thomas, K.M., Patel, A.R., Swails, J.L. and Kwak, M.J., 2021. Primary care experience among older adults in the United States: a retrospective cross-sectional study. *Archives of Gerontology and Geriatrics*, 95, p.104396.
76. Triantafyllou, A.M. and Tsihrintzis, G.A., 2018, July. Group affect recognition: evaluation of basic automated sorting. In *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)* (pp. 1-8). IEEE.
77. Triantafyllou, A.M., Tsihrintzis, G.A., Virvou, M. and Alepis, E., 2021. A Bimodal System for Emotion Recognition via Computer of Known or Unknown Persons in Normal or Fatigue Situations. *Advances in Core Computer Science-Based Technologies: Papers in Honor of Professor Nikolaos Alexandris*, pp.9-35.
78. Tsihrintzis, G. A., Toro, C., Rios, S. A., Howlett, R. J., & Jain, L. C. (2023). 27th KES International Conference on Knowledge-Based and Intelligent Information & Engineering Systems KES2023. *Procedia Computer Science*, 225, 1-11.
79. Tsihrintzis, G. A., Virvou, M., & Hatzilygeroudis, I. (2023). Special Issue on Selected Papers from the 33rd Annual IEEE International Conference on Tools with Artificial Intelligence (ICTAI-2021). *INTERNATIONAL JOURNAL ON ARTIFICIAL INTELLIGENCE TOOLS*, 32(05), 2302003.
80. Tsihrintzis, G. A., Virvou, M., & Jain, L. C. (2022). *Advances in Machine Learning/Deep Learning-based Technologies*. Springer International Publishing.
81. Tsihrintzis, G.A. and Virvou, M. eds., 2010. *Multimedia Services in Intelligent Environments: Software Development Challenges and Solutions* (Vol. 2). Springer Science & Business Media.
82. Tsihrintzis, G.A., Virvou, M., Alepis, E. and Stathopoulou, I.O., 2008. Towards improving visual-facial emotion recognition through use of complementary keyboard-stroke pattern information. In *Fifth International Conference on Information Technology: New Generations (itng 2008)* (pp. 32-37). IEEE.
83. Tsihrintzis, G.A., Virvou, M. and Phillips-Wren, G., 2019. Surveys in artificial intelligence-based technologies. *Intelligent Decision Technologies*, 13(4), pp.393-394.
84. Tsihrintzis, G.A., Virvou, M. and Saruwatari, T., 2022. Special Collection of Extended Selected Papers on “Novel Research Results Presented at the 14th International Joint Conference on Knowledge-based Software Engineering (JCKBSE2022), 22–24 August 2022, Larnaca, Cyprus <https://easyconferences.eu/jckbse2022/>”. *Intelligent Decision Technologies*, 16(4), pp.715-716.
85. Tsihrintzis, G.A., Virvou, M., Stathopoulou, I.O. and Alepis, E., 2008, December. On improving visual-facial emotion recognition with audio-lingual and keyboard stroke pattern information. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (Vol. 1, pp. 810-816). IEEE.

86. Tsiriga, V. and Virvou, M., 2004. Evaluating the intelligent features of a web-based intelligent computer assisted language learning system. *International journal on artificial intelligence tools*, 13(02), pp.411-425.
87. Tsiriga, V. and Virvou, M., 2003, July. Initializing student models in web-based ITSs: a generic approach. In *Proceedings 3rd IEEE International Conference on Advanced Technologies* (pp. 42-46). IEEE.
88. Tsiriga, V. and Virvou, M., 2003. Modelling the student to individualise tutoring in a web-based ICALL. *International Journal of Continuing Engineering Education and Life Long Learning*, 13(3-4), pp.350-365.
89. Tsiriga, V. and Virvou, M., 2002, October. Initializing the student model using stereotypes and machine learning. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*.
90. Tsiriga, V. and Virvou, M., 2002, September. Dynamically initializing the student model in a web-based language tutor. In *Proceedings First International IEEE Symposium Intelligent Systems* (Vol. 1, pp. 138-143). IEEE.
91. Usability for Older Adults: Challenges and Change,
<https://www.nngroup.com/articles/usability-for-senior-citizens/>
92. Virvou, M., 2023. Artificial Intelligence and User Experience in reciprocity: Contributions and state of the art. *Intelligent Decision Technologies* 17 (2023) 73–125 73 DOI 10.3233/IDT-230092 IOS Press
93. Virvou, M., 1999. Automatic reasoning and help about human errors in using an operating system. *Interacting with Computers*, 11(5), pp.545-573
94. Virvou, M., 2018, July. A new era towards more engaging and human-like computer-based learning by combining personalisation and artificial intelligence techniques. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (pp. 2-3).
95. Virvou, M., 2022, July. The emerging era of human-AI interaction: Keynote address. In *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1-10). IEEE.
96. Virvou, M., & Nakamura, T. (Eds.). (2008). *Knowledge-based Software Engineering: Proceedings of the Eighth Joint Conference on Knowledge-Based Software Engineering* (Vol. 180). Ios Press.
97. Virvou, M. (2012). On the evaluation of the combined role of virtual reality games and animated agents in edutainment. *Intelligent Computer Graphics 2011*, 79-95.
98. Virvou, M., Alepis, E., Tsihrintzis, G.A. and Jain, L.C., 2020. *Machine learning paradigms: advances in learning analytics* (pp. 1-5). Springer International Publishing.
99. Virvou, M. and Alepis, E., 2005. Mobile educational features in authoring tools for personalised tutoring. *Computers & Education*, 44(1), pp.53-68.
100. Virvou, M. and Du Boulay, B., 1999. Human plausible reasoning for intelligent help. *User modeling and user-adapted interaction*, 9, pp.321-375.
101. Virvou, M. and Kabassi, K., 2004. Adapting the human plausible reasoning theory to a graphical user interface. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 34(4), pp.546-563.
102. Virvou, M. and Kabassi, K., 2004. Evaluating an intelligent graphical user interface by comparison with human experts. *Knowledge-based systems*, 17(1), pp.31-37.

103. Virvou, M. and Kabassi, K., 2002. Reasoning about users' actions in a graphical user interface. *Human-Computer Interaction*, 17(4), pp.369-398.
104. Virvou, M. and Kabassi, K., 2002, September. F-SMILE: An intelligent multi-agent learning environment. In *Proceedings of 2002 IEEE International Conference on Advanced Learning Technologies-ICALT* (pp. 144-149).
105. Virvou, M. and Katerina, K., 2001, August. Evaluation of the advice generator of an intelligent learning environment. In *Proceedings IEEE International Conference on Advanced Learning Technologies* (pp. 339-342). IEEE.
106. Virvou, M. and Moundridou, M., 2001. Student and instructor models: two kinds of user model and their interaction in an ITS authoring tool. In *User Modeling 2001: 8th International Conference, UM 2001 Sonthofen, Germany, July 13–17, 2001 Proceedings 8* (pp. 158-167). Springer Berlin Heidelberg.
107. Virvou, M. and Tourtoglou, K., 2006, July. An adaptive training environment for UML. In *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)* (pp. 147-149). IEEE.
108. Virvou, M. and Tsihrintzis, G.A., 2023, July. A Novel Trust State-Chart Model for Requirements Engineering of Trustful AI-Empowered Software. In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1-6). IEEE.
109. Virvou, M. and Tsihrintzis, G.A., 2023, July. Is ChatGPT Beneficial to Education? A Holistic Evaluation Framework Based on Intelligent Tutoring Systems. In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1-8). IEEE.
110. Virvou, M. and Tsihrintzis, G.A., 2023. Pre-made Empowering Artificial Intelligence and ChatGPT: The Growing Importance of Human AI-Experts. In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1-8). IEEE.
111. Virvou, M. and Tsiriga, V., 2001. An object-oriented software life cycle of an intelligent tutoring system. *Journal of Computer Assisted Learning*, 17(2), pp.200-205.
112. Virvou, M. and Tsiriga, V., 2000. Involving effectively teachers and students in the life cycle of an intelligent tutoring system. *Journal of Educational Technology & Society*, 3(3), pp.511-521.
113. Virvou, M., Katsionis, G. and Manos, K., 2004. On the motivation and attractiveness scope of the virtual reality user interface of an educational game. In *Computational Science-ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part III 4* (pp. 962-969). Springer Berlin Heidelberg.
114. Virvou, M., Manos, C., Katsionis, G. and Tourtoglou, K., 2002, September. VR-ENGAGE: A virtual reality educational game that incorporates intelligence. In *Proceedings of IEEE international conference on advanced learning technologies* (pp. 16-19).
115. Virvou, M., Savvopoulos, A., Tsihrintzis, G.A. and Sotiropoulos, D.N., 2007. Constructing Stereotypes for an Adaptive e-Shop using AIN-based Clustering. In *Adaptive and Natural Computing Algorithms: 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11-14, 2007, Proceedings, Part I 8* (pp. 837-845). Springer Berlin Heidelberg.
116. Virvou, M., Tsihrintzis, G. A., Bourbakis, N. G., & Jain, L. C. (2022). Handbook on Artificial Intelligence-Empowered Applied Software Engineering: VOL. 2: Smart Software Applications in Cyber-Physical Systems (Vol. 3). Springer International Publishing AG.
117. Virvou, M., Tsihrintzis, G. A., Bourbakis, N. G., & Jain, L. C. (2022). Introduction to Handbook on Artificial Intelligence-Empowered Applied Software Engineering-VOL. 1: Novel Methodologies to Engineering Smart Software Systems. In *Handbook on Artificial Intelligence-Empowered Applied Software Engineering: VOL. 1: Novel Methodologies to Engineering Smart Software Systems* (pp. 1-8). Cham: Springer International Publishing.

118. Virvou, M., Tsihrintzis, G. A., Tsoukalas, L. H., & Jain, L. C. (2022). Introduction to Advances in Artificial Intelligence-Based Technologies. *Advances in Artificial Intelligence-based Technologies: Selected Papers in Honour of Professor Nikolaos G. Bourbakis-Vol. 1*, 1-5.
119. Virvou, M., Tsihrintzis, G.A., Alepis, E., Stathopoulou, I.O. and Kabassi, K., 2007. Combining empirical studies of audio-lingual and visual-facial modalities for emotion recognition. In *Knowledge-Based Intelligent Information and Engineering Systems: 11th International Conference, KES 2007, XVII Italian Workshop on Neural Networks, Vietri sul Mare, Italy, September 12-14, 2007. Proceedings, Part II* 11 (pp. 1130-1137). Springer Berlin Heidelberg.
120. Virvou, M., Tsihrintzis, G.A., Alepis, E., Stathopoulou, I.O. and Kabassi, K., 2012. Emotion recognition: empirical studies towards the combination of audio-lingual and visual-facial modalities through multi-attribute decision making. *International Journal on Artificial Intelligence Tools*, 21(02), p.1240001.
121. Virvou, M., Tsihrintzis, G.A. and Jain, L.C., 2022. Introduction to advances in selected artificial intelligence areas. In *Advances in Selected Artificial Intelligence Areas: World Outstanding Women in Artificial Intelligence* (pp. 1-7). Cham: Springer International Publishing.
122. Virvou, M., Tsihrintzis, G.A. and Tsihrintzi, E.A., 2024. VIRTSL: A novel trust dynamics model enhancing Artificial Intelligence collaboration with human users—Insights from a ChatGPT evaluation study. *Information Sciences*, 675, p.120759.
123. Virvou, M., Tsihrintzis, G.A., Sotiropoulos, D.N., Chrysafiadi, K., Sakkopoulos, E. and Tsihrintzi, E.A., 2023, July. ChatGPT in Artificial Intelligence-Empowered E-Learning for Cultural Heritage: The case of Lyrics and Poems. In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 1-9). IEEE.
124. Weng, S., Shieh, C. S., & Tsihrintzis, G. A. (Eds.). (2023). *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the 18th IIH-MSP 2022 Kitakyushu, Japan*. Springer.
125. Wildenbos, G.A., Peute, L. and Jaspers, M., 2018. Aging barriers influencing mobile health usability for older adults: a literature based framework (MOLD-US). *International journal of medical informatics*, 114, pp.66-75.

Διαδικτυακές πηγές

<https://www.apple.com/>

<https://www.garmin.com/en-US/>

<https://www.samsung.com/gr/>

<https://fitbit.google/enterprise/>

<https://www.teladochealth.com/>

<https://business.amwell.com/>

<https://www.mdlive.com/>

<https://doctorondemand.com/>

<https://www.medisafe.com/>

<https://apps.apple.com/us/app/round-health/id1059591124>

<https://builtwith.com/>

<https://www.wappalyzer.com/>

<https://www.mychart.org/>

<https://azure.microsoft.com/en-us/>

Chi, N.C. and Demiris, G., 2015. A systematic review of telehealth tools and interventions to support family caregivers. *Journal of telemedicine and telecare*, 21(1), pp.37-44.

Verma, N., Mamlin, B., Flowers, J., Acharya, S., Labrique, A. and Cullen, T., 2021. OpenMRS as a global good: Impact, opportunities, challenges, and lessons learned from fifteen years of implementation. *International journal of medical informatics*, 149, p.104405.

Shickel, B., Tighe, P.J., Bihorac, A. and Rashidi, P., 2017. Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE journal of biomedical and health informatics*, 22(5), pp.1589-1604.

Παράρτημα Α: Αποσπάσματα κώδικα

Στο παράρτημα αυτό παρατίθενται τα πλήρη αποσπάσματα κώδικα και οι διαμορφώσεις στα οποία παραπέμπει το κυρίως κείμενο της εργασίας. Η αρίθμηση (Α.1, Α.2, ...) ακολουθεί τη σειρά εμφάνισης των παραπομπών στο κείμενο.

Α.1 - Μοντέλο Household και παραγόμενο σχήμα (backend/models/HouseHold.go)

```
package models

import "gorm.io/gorm"

type Household struct {
    gorm.Model
    AdminID uint `json:"admin_id" gorm:"not null"`
    Admin    User  `json:"admin" gorm:"foreignKey:AdminID"`
    Patients []Patient `json:"patients" gorm:"many2many:household_patients;"`
}

-- Σχήμα που παράγεται αυτόματα από το GORM AutoMigrate βάσει του παραπάνω
μοντέλου:
CREATE TABLE households (
    id SERIAL PRIMARY KEY,
    created_at TIMESTAMP WITH TIME ZONE,
    updated_at TIMESTAMP WITH TIME ZONE,
    deleted_at TIMESTAMP WITH TIME ZONE,
    admin_id INTEGER NOT NULL REFERENCES users(id)
);

-- Πίνακας συσχέτισης many-to-many (νοικοκυριά ↔ ασθενείς)
CREATE TABLE household_patients (
    household_id INTEGER NOT NULL REFERENCES households(id),
    patient_id INTEGER NOT NULL REFERENCES patients(id),
    PRIMARY KEY (household_id, patient_id)
);
```

A.2 - Πίνακας chat_messages (παράγεται από το μοντέλο ChatMessage)

```
-- Σχήμα που παράγεται αυτόματα από το GORM AutoMigrate βάσει του μοντέλου
ChatMessage
-- (βλ. Παράρτημα A.13 για το μοντέλο chatMessage.go)
CREATE TABLE chat_messages (
  id BIGSERIAL PRIMARY KEY,
  created_at TIMESTAMP WITH TIME ZONE,
  updated_at TIMESTAMP WITH TIME ZONE,
  deleted_at TIMESTAMP WITH TIME ZONE,
  session_id UUID NOT NULL, -- αναφορά σε
chat_sessions.session_id
  role VARCHAR NOT NULL, -- 'user' ή 'assistant'
  content TEXT NOT NULL,
  tokens_used INTEGER DEFAULT 0,
  response_time INTEGER DEFAULT 0, -- σε ms
  is_encrypted BOOLEAN DEFAULT FALSE,
  message_index INTEGER, -- σειρά στη συνομιλία
  timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

A.3 - Θέμα Material-UI (frontend/src/App.tsx)

```
const theme = createTheme({
  palette: {
    primary: {
      main: '#2ecc71',
    },
    secondary: {
      main: '#e74c3c',
    },
    background: {
      default: '#f0f8ff',
    },
  },
});
```

A.4 - Παράδειγμα στοιχείου διεπαφής (frontend/src/PatientDashboard.tsx, απόσπασμα)

```
<Box sx={{ mb: 4, display: 'flex', justifyContent: 'space-
between', alignItems: 'center' }}>
  <Typography variant="h4" component="h1" gutterBottom>
    Your Dashboard
  </Typography>
  <Button
    variant="outlined"
    color="primary"
    onClick={onLogout}
    startIcon={<LogoutIcon />}
  >
    Logout
  </Button>
</Box>
```

A.5 - Φόρμα στοιχείων ασθενούς (frontend/src/PatientFirstTimeForm.tsx, απόσπασμα)

```
<Box component="form" onSubmit={handleSubmit} sx={{ width:
'100%' }}>
```

```

        <Grid container spacing={3}>
          <Grid item xs={12} sm={6}>
            <TextField
              required
              fullWidth
              id="name"
              label="First Name"
              name="name"
              value={formData.name}
              onChange={handleInputChange}
              variant="outlined"
            />
          </Grid>
          <Grid item xs={12} sm={6}>
            <TextField
              required
              fullWidth
              id="surname"
              label="Last Name"
              name="surname"
              value={formData.surname}
              onChange={handleInputChange}
              variant="outlined"
            />
          </Grid>
          <Grid item xs={12} sm={6}>
            <TextField
              required
              fullWidth
              id="dateOfBirth"
              label="Date of Birth"
              name="dateOfBirth"
              type="date"
              value={formData.dateOfBirth}
              onChange={handleInputChange}
              variant="outlined"
              InputLabelProps={{
                shrink: true,
              }}
              inputProps={{
                min: "1920-01-01",
                max: new
Date().toISOString().split('T')[0]
              }}
            />
          </Grid>
          <Grid item xs={12} sm={6}>
            <FormControl fullWidth required>
              <InputLabel id="gender-
label">Gender</InputLabel>

```

A.6 - Δομή αποθετηρίου my-health

```

my-health/
  AI_AGENT_ROADMAP.md
  AI_CHATBOT_IMPLEMENTATION.md
  README.md

```

```

THEESIS_DRAFT.md
THEESIS_STRUCTURE.txt
convert_mermaid_hq.py
docker-compose.yml
extract_mermaid_diagrams.py
setup-ollama.sh
test-ollama.sh
.github/
  workflows/
    deploy-infrastructure.yml
    setup-instructions.md
backend/
  Dockerfile.dev
  go.sum
  main.go
  controllers/
    authController.go
    chatController.go
    householdController.go
    patientController.go
  initializers/
    database.go
    loadEnvs.go
  middlewares/
    checkAuth.go
  migrate/
    migrate.go
  models/
    Claims.go
    HealthMetrics.go
    HouseHold.go
    Invitation.go
    authInput.go
    chatMessage.go
    chatSession.go
    patient.go
    user.go
  routes/
    route.go
  services/
    ai/
      chatService.go
      contextBuilder.go
      ollamaClient.go
    mockhealth/
      mock_health.go
  utils/
    CompareHashPassword.go
    GenerateHashPassword.go
    ParseToken.go
frontend/
  Dockerfile.dev
  README.md
  package-lock.json
  package.json
  tsconfig.json
  yarn.lock

```

```

public/
  elder-health-logo.png
  favicon.ico
  index.html
  logo192.png
  logo512.png
  manifest.json
  robots.txt
src/
  AdminDashboard.tsx
  AdminProfileEdit.tsx
  App.css
  App.test.tsx
  App.tsx
  Bubbles.tsx
  CreateUserPage.tsx
  HouseHoldMembers.tsx
  LandingPage.tsx
  LoginPage.tsx
  PatientDashboard.tsx
  PatientDetails.tsx
  PatientEditPage.tsx
  PatientFirstTimeForm.tsx
  index.css
  index.tsx
  logo.svg
  react-app-env.d.ts
  reportWebVitals.ts
  setupTests.ts
  suppressResizeObserver.ts
  components/
    PatientStatusModal.tsx
    PendingInvitations.tsx
images/
  UNIPI_Emblem.png
infra/
  README.md
  main.tf
  outputs.tf
  terraform.tfvars.example
  variables.tf

```

A.7 - Αρχείο: backend/main.go

```

package main

import (
    "net/http"

    "github.com/gin-gonic/gin"

    "my-health/initializers"
    "my-health/routes"
)

func init() {
    initializers.LoadEnvs()
}

```

```

    initializers.ConnectDatabase()
    initializers.SyncDatabase()
}

// CORS Middleware
func CORS(c *gin.Context) {
    c.Header("Access-Control-Allow-Origin", "http://localhost:3000") //
    specify frontend origin explicitly
    c.Header("Access-Control-Allow-Methods", "GET, POST, PUT, DELETE,
    OPTIONS")
    c.Header("Access-Control-Allow-Headers", "Authorization, Content-Type")
    c.Header("Access-Control-Allow-Credentials", "true")
    c.Header("Access-Control-Max-Age", "43200") // 12 hours in seconds
    c.Header("Access-Control-Expose-Headers", "Authorization")

    // Handle OPTIONS requests
    if c.Request.Method == "OPTIONS" {
        c.AbortWithStatus(http.StatusOK)
        return
    }

    c.Next()
}

func main() {
    router := gin.Default()
    router.Use(CORS)
    routes.SetupRoutes(router)

    router.Run(":8080")
}

```

A.8 - Αρχείο: backend/models/user.go

```

package models

import (
    "time"

    "gorm.io/gorm"
)

type User struct {
    gorm.Model
    Username    string    `json:"username" gorm:"uniqueIndex"`
    Password    string    `json:"-"" // "-" means this won't be included
in JSON`
    Role        string    `json:"role"`
    IsInHousehold bool      `json:"is_in_household"`
    Households  []Household `json:"households"
gorm:"many2many:household_patients;"`
    Patient     *Patient  `json:"patient,omitEmpty"`
    CreatedAt   time.Time
    UpdatedAt   time.Time
}

```

```
// Add any methods specific to User here
func (u *User) IsPatient() bool {
    return u.Role == "patient"
}
```

A.9 - Αρχείο: backend/models/patient.go

```
package models

import (
    "time"

    "gorm.io/gorm"
)

type EmergencyContact struct {
    Name          string `json:"name" gorm:"column:emergency_contact_name"`
    Relationship   string `json:"relationship" gorm:"column:emergency_contact_relationship"`
    PhoneNumber    string `json:"phone_number" gorm:"column:emergency_contact_phone_number"`
}

type Patient struct {
    gorm.Model
    UserID      uint      `json:"user_id" gorm:"uniqueIndex"`
    User        User      `json:"user" gorm:"foreignKey:UserID"`
    Name        string    `json:"name"`
    Surname     string    `json:"surname"`
    Address     string    `json:"address"`
    MedicalRecord string    `json:"medical_record"`
    DateOfBirth time.Time `json:"date_of_birth"`

    Gender      string    `json:"gender"`
    BloodType   string    `json:"blood_type"`
    Height      float64   `json:"height"`
    MedicalHistory string    `json:"medical_history"`
    Allergies   string    `json:"allergies"`
    Medications string    `json:"medications"`
    EmergencyContact EmergencyContact `json:"emergency_contact" gorm:"embedded;embeddedPrefix:emergency_contact_"`
    Households []Household `json:"households" gorm:"many2many:household_patients;"`
    HealthMetrics []HealthMetrics `json:"health_metrics" gorm:"foreignKey:PatientID"`
}

func (p *Patient) Age() int {
    return int(time.Since(p.DateOfBirth).Hours() / 24 / 365)
}
```

A.10 - Αρχείο: backend/models/HealthMetrics.go

```
package models

import (
    "time"

```

```

    "gorm.io/gorm"
)

type SleepStages struct {
    Light    float64 `json:"light"`
    Deep     float64 `json:"deep"`
    REM      float64 `json:"rem"`
    AwakeTime int    `json:"awake_time" // in minutes
}

type HealthMetrics struct {
    gorm.Model
    PatientID uint    `json:"patient_id"`
    Date      time.Time `json:"date" gorm:"not
null;default:CURRENT_TIMESTAMP"`
    Weight     float64 `json:"weight"`
    HeartRate  int     `json:"heart_rate"`
    SystolicBP int     `json:"systolic_bp"`
    DiastolicBP int    `json:"diastolic_bp"`
    OxygenSaturation float64 `json:"oxygen_saturation"`
    BloodPressure string  `json:"blood_pressure"`
    StepsCount int     `json:"steps_count"`
    Sleep      SleepStages `json:"sleep" gorm:"embedded"`
    SleepDuration float64 `json:"sleep_duration" // in hours
    IrregularRhythm bool   `json:"irregular_rhythm"`
    FallDetected bool   `json:"fall_detected"`
}

// BeforeCreate will set the Date field to current time if not set
func (h *HealthMetrics) BeforeCreate(tx *gorm.DB) error {
    if h.Date.IsZero() {
        h.Date = time.Now()
    }
    return nil
}

```

A.11 - Αρχείο: backend/models/Invitation.go

```

package models

import (
    "gorm.io/gorm"
)

type Invitation struct {
    gorm.Model
    AdminID    uint    `gorm:"not null"`
    Admin      User    `gorm:"foreignKey:AdminID"`
    PatientID  uint    `gorm:"not null"`
    Patient    User    `gorm:"foreignKey:PatientID"`
    HouseholdID uint    `gorm:"not null"`
    Household  Household `json:"household" gorm:"foreignKey:HouseholdID"`
    Status     string
}

```

A.12 - Αρχείο: backend/models/chatSession.go

```

package models

import (
    "time"

    "github.com/google/uuid"
    "gorm.io/gorm"
)

type ChatSession struct {
    gorm.Model
    UserID      uint      `json:"user_id" gorm:"not null"`
    User        User      `json:"user" gorm:"foreignKey:UserID"`
    SessionID   uuid.UUID `json:"session_id"
gorm:"type:uuid;default:gen_random_uuid();uniqueIndex"`
    SessionType string    `json:"session_type" gorm:"default:'general'"` //
    'patient' or 'admin'
    ContextData string    `json:"context_data" gorm:"type:text"` //
    JSON string of user's health context
    IsActive    bool    `json:"is_active" gorm:"default:true"`
    LastUsedAt  time.Time `json:"last_used_at"`
    Messages    []ChatMessage `json:"messages"
gorm:"foreignKey:SessionID;references:SessionID"`
}

// BeforeCreate will set the SessionID and LastUsedAt if not set
func (cs *ChatSession) BeforeCreate(tx *gorm.DB) error {
    if cs.SessionID == uuid.Nil {
        cs.SessionID = uuid.New()
    }
    if cs.LastUsedAt.IsZero() {
        cs.LastUsedAt = time.Now()
    }
    // Set session type based on user role
    if cs.User.Role == "patient" {
        cs.SessionType = "patient"
    } else if cs.User.Role == "admin" {
        cs.SessionType = "admin"
    }
    return nil
}

// UpdateLastUsed updates the last used timestamp
func (cs *ChatSession) UpdateLastUsed(db *gorm.DB) error {
    cs.LastUsedAt = time.Now()
    return db.Save(cs).Error
}

```

A.13 - Αρχείο: backend/models/chatMessage.go

```

package models

import (
    "time"

    "github.com/google/uuid"
    "gorm.io/gorm"

```

```

    }

type ChatMessage struct {
    gorm.Model
    SessionID    uuid.UUID `json:"session_id" gorm:"type:uuid;not null"`
    Role         string    `json:"role" gorm:"not null" // 'user'
    // or 'assistant'
    Content       string    `json:"content" gorm:"type:text;not null"`
    TokensUsed    int       `json:"tokens_used" gorm:"default:0"`
    ResponseTime  int       `json:"response_time" gorm:"default:0" // in
    milliseconds
    IsEncrypted   bool      `json:"is_encrypted" gorm:"default:false"`
    MessageIndex int       `json:"message_index" // Order
    // in conversation
    Timestamp     time.Time `json:"timestamp"
    gorm:"default:CURRENT_TIMESTAMP"`
}

// BeforeCreate will set the timestamp and message index
func (cm *ChatMessage) BeforeCreate(tx *gorm.DB) error {
    if cm.Timestamp.IsZero() {
        cm.Timestamp = time.Now()
    }

    // Set message index based on existing messages in session
    var count int64
    tx.Model(&ChatMessage{}).Where("session_id = ?",
    cm.SessionID).Count(&count)
    cm.MessageIndex = int(count) + 1

    return nil
}

// IsUserMessage checks if the message is from user
func (cm *ChatMessage) IsUserMessage() bool {
    return cm.Role == "user"
}

// IsAssistantMessage checks if the message is from AI assistant
func (cm *ChatMessage) IsAssistantMessage() bool {
    return cm.Role == "assistant"
}

```

A.14 - Αρχείο: backend/controllers/authController.go (λειτουργίες CreateUser και Login)

```

func CreateUser(c *gin.Context) {
    var authInput models.AuthInput

    if err := c.ShouldBindJSON(&authInput); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    var userFound models.User
    initializers.DB.Where("username=?", authInput.Username).Find(&userFound)

```

```

        if userFound.ID != 0 {
            c.JSON(http.StatusBadRequest, gin.H{"error": "username already
used"})
            return
        }

        passwordHash, err :=
bcrypt.GenerateFromPassword([]byte(authInput.Password), bcrypt.DefaultCost)
        if err != nil {
            c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
            return
        }

        user := models.User{
            Username: authInput.Username,
            Password: string(passwordHash),
            Role:     authInput.Role,
        }

        initializers.DB.Create(&user)

        // If the user role is "patient", also create a corresponding patient
record
        if authInput.Role == "patient" {
            patient := models.Patient{
                UserID: user.ID,
                // Initialize with empty/default values that can be filled later
                Name:     "",
                Surname: "",
                // Other fields will be nil/empty and can be updated later
            }
            initializers.DB.Create(&patient)
        }

        c.JSON(http.StatusOK, gin.H{"data": user})
    }

}

func Login(c *gin.Context) {
    var authInput models.AuthInput

    if err := c.ShouldBindJSON(&authInput); err != nil {
        log.Printf("Failed to bind JSON: %v", err)
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    var userFound models.User
    if err := initializers.DB.Where("username = ?",
authInput.Username).First(&userFound).Error; err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "user not found"})
        return
    }

    if err := bcrypt.CompareHashAndPassword([]byte(userFound.Password),
[]byte(authInput.Password)); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "invalid password"})
    }
}

```

```

    }
    return

    claims := jwt.MapClaims{
        "user_id": userFound.ID,
        "role":    userFound.Role,
        "exp":     time.Now().Add(time.Hour * 24).Unix(),
    }

    // Log token claims for debugging
    log.Printf("Creating token with claims: %+v", claims)

    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    signedToken, err := token.SignedString([]byte(os.Getenv("SECRET")))
    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": "failed to
generate token"})
        return
    }

    var patientDetails models.Patient
    hasDetails := true
    if userFound.Role == "patient" {
        if err := initializers.DB.Where("user_id = ?",
userFound.ID).First(&patientDetails).Error; err != nil {
            hasDetails = false
        }
    }

    c.JSON(http.StatusOK, gin.H{
        "token":    signedToken,
        "role":     userFound.Role,
        "userId":   userFound.ID,
        "hasDetails": hasDetails,
    })
}

```

A.15 - Αρχείο: backend/services/ai/chatService.go (λειτουργία ProcessMessage)

```

func (cs *ChatService) ProcessMessage(req ChatRequest) (*ChatResponse, error)
{
    startTime := time.Now()

    // Check if Ollama is healthy
    if err := cs.ollamaClient.IsHealthy(); err != nil {
        return &ChatResponse{
            Success: false,
            Error:   "AI service is currently unavailable. Please try again
later.",
        }, err
    }

    // Get or create chat session
    session, err := cs.getOrCreateSession(req.UserID)
    if err != nil {
        return &ChatResponse{
            Success: false,

```

```

        Error: "Failed to initialize chat session.",
    }, err
}

// Build user context
context, err := BuildUserContext(req.UserID)
if err != nil {
    return &ChatResponse{
        Success: false,
        Error: "Failed to load user context.",
    }, err
}

// Generate healthcare-specific prompt
prompt := GenerateHealthcarePrompt(context, req.Message)

// Save user message
userMessage := models.ChatMessage{
    SessionID: session.SessionID,
    Role: "user",
    Content: req.Message,
}
if err := initializers.DB.Create(&userMessage).Error; err != nil {
    return &ChatResponse{
        Success: false,
        Error: "Failed to save your message.",
    }, err
}

// Call Ollama API
ollamaResp, err := cs.ollamaClient.GenerateResponse("healthbot-v2",
prompt)
if err != nil {
    // Save error message
    errorMessage := models.ChatMessage{
        SessionID: session.SessionID,
        Role: "assistant",
        Content: "I'm having trouble processing your request right now.
Please try again in a few moments.",
    }
    initializers.DB.Create(&errorMessage)

    return &ChatResponse{
        Success: false,
        Error: "AI service is temporarily unavailable.",
    }, err
}

responseTime := int(time.Since(startTime).Milliseconds())

// Save assistant response
assistantMessage := models.ChatMessage{
    SessionID: session.SessionID,
    Role: "assistant",
    Content: ollamaResp.Response,
    ResponseTime: responseTime,
    TokensUsed: len(ollamaResp.Response) / 4, // Rough token estimation
}

```

```

    }
    if err := initializers.DB.Create(&assistantMessage).Error; err != nil {
        return &ChatResponse{
            Success: false,
            Error:    "Failed to save AI response.",
        }, err
    }

    // Update session last used time
    session.UpdateLastUsed(initializers.DB)

    return &ChatResponse{
        Response:    ollamaResp.Response,
        SessionID:   session.SessionID,
        ResponseTime: responseTime,
        TokensUsed:  assistantMessage.TokensUsed,
        Success:     true,
    }, nil
}

```

A.16 - Αρχείο: backend/services/ai/contextBuilder.go (BuildUserContext)

```

func BuildUserContext(userID uint) (*UserContext, error) {
    var user models.User
    if err := initializers.DB.Preload("Patient").First(&user, userID).Error;
err != nil {
        return nil, fmt.Errorf("failed to get user: %v", err)
    }

    context := &UserContext{
        UserID:    user.ID,
        UserRole:  user.Role,
        Username:  user.Username,
        IsPatient: user.IsPatient(),
    }

    if user.IsPatient() && user.Patient != nil {
        patientContext, err := buildPatientContext(user.Patient)
        if err != nil {
            return nil, err
        }
        context.PatientData = patientContext
    } else {
        adminContext, err := buildAdminContext(userID)
        if err != nil {
            return nil, err
        }
        context.AdminData = adminContext
    }

    return context, nil
}

func buildPatientContext(patient *models.Patient) (*PatientContextData,
error) {
    // Get latest health metrics
    var latestMetrics models.HealthMetrics

```

```

err := initializers.DB.Where("patient_id = ?", patient.ID).
    Order("date DESC").
    First(&latestMetrics).Error

var metricsPtr *models.HealthMetrics
if err == nil {
    metricsPtr = &latestMetrics
}

return &PatientContextData{
    Name:      fmt.Sprintf("%s %s", patient.Name,
patient.Surname),
    Age:       patient.Age(),
    Gender:    patient.Gender,
    BloodType: patient.BloodType,
    Height:    patient.Height,
    Medications: patient.Medications,
    Allergies: patient.Allergies,
    MedicalHistory: patient.MedicalHistory,
    EmergencyContact: patient.EmergencyContact,
    LatestMetrics: metricsPtr,
}, nil
}

```

A.17 - Αρχείο: backend/services/ai/ollamaClient.go

```

package ai

import (
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "net/http"
    "os"
    "time"
)

type OllamaClient struct {
    BaseURL string
    Client *http.Client
}

type OllamaRequest struct {
    Model string `json:"model"`
    Prompt string `json:"prompt"`
    Stream bool `json:"stream"`
}

type OllamaResponse struct {
    Response string `json:"response"`
    Done bool `json:"done"`
    Model string `json:"model"`
}

func NewOllamaClient() *OllamaClient {
    // Get Ollama URL from environment variable, fallback to localhost for

```

```

development
ollamaURL := os.Getenv("OLLAMA_URL")
if ollamaURL == "" {
    ollamaURL = "http://localhost:11434" // Default for local development
}

return &OllamaClient{
    BaseURL: ollamaURL,
    Client: &http.Client{
        Timeout: 60 * time.Second, // 60 second timeout for AI responses
    },
}

func (oc *OllamaClient) GenerateResponse(model, prompt string)
(*OllamaResponse, error) {
    reqBody := OllamaRequest{
        Model: model,
        Prompt: prompt,
        Stream: false,
    }

    jsonData, err := json.Marshal(reqBody)
    if err != nil {
        return nil, fmt.Errorf("failed to marshal request: %v", err)
    }

    resp, err := oc.Client.Post(
        fmt.Sprintf("%s/api/generate", oc.BaseURL),
        "application/json",
        bytes.NewBuffer(jsonData),
    )
    if err != nil {
        return nil, fmt.Errorf("failed to call Ollama API: %v", err)
    }
    defer resp.Body.Close()

    if resp.StatusCode != http.StatusOK {
        body, _ := io.ReadAll(resp.Body)
        return nil, fmt.Errorf("Ollama API error (status %d): %s",
resp.StatusCode, string(body))
    }

    var ollamaResp OllamaResponse
    if err := json.NewDecoder(resp.Body).Decode(&ollamaResp); err != nil {
        return nil, fmt.Errorf("failed to decode Ollama response: %v", err)
    }

    return &ollamaResp, nil
}

func (oc *OllamaClient) IsHealthy() error {
    resp, err := oc.Client.Get(fmt.Sprintf("%s/api/tags", oc.BaseURL))
    if err != nil {
        return fmt.Errorf("Ollama service unreachable: %v", err)
    }
    defer resp.Body.Close()

```

```

    if resp.StatusCode != http.StatusOK {
        return fmt.Errorf("Ollama service unhealthy (status %d)",
resp.StatusCode)
    }

    return nil
}

```

A.18 - Αρχείο: backend/routes/route.go

```

package routes

import (
    "github.com/gin-gonic/gin"

    "my-health/controllers"
    "my-health/middlewares"
)

func SetupRoutes(r *gin.Engine) {
    // Public routes
    r.POST("/login", controllers.Login)
    r.POST("/create-user", controllers.CreateUser)
    r.GET("/logout", controllers.Logout)

    // Protected routes
    protected := r.Group("/")
    protected.Use(middlewares.CheckAuth)
    {
        // User routes
        protected.GET("/user", controllers.GetUserDetails)

        // Admin routes
        protected.GET("/admin/profile", controllers.GetAdminProfile)
        protected.POST("/admin/profile", controllers.UpdateAdminProfile)

        // Patient routes
        protected.GET("/patient/:id", controllers.GetPatientDetails)
        protected.POST("/patient/edit/:id", controllers.UpdatePatient)
        protected.GET("/patient/check-details/:userId",
controllers.CheckPatientDetails)

        // Health metrics routes
        protected.GET("/api/health-metrics/:patientId",
controllers.GetPatientHealthMetrics)

        // Household routes
        protected.GET("/household/patients",
controllers.GetHouseholdPatients)
        protected.GET("/household/patient-statuses",
controllers.GetPatientStatuses)
        protected.POST("/create-invitation", controllers.CreateInvitation)
        protected.POST("/respond-invitation",
controllers.RespondToInvitation)
        protected.GET("/invitations", controllers.GetInvitations)
    }
}

```

```

// AI Chat routes
chatController := controllers.NewChatController()
protected.POST("/api/ai/chat", chatController.SendMessage)
protected.GET("/api/ai/history", chatController.GetHistory)
protected.DELETE("/api/ai/clear", chatController.ClearHistory)
protected.GET("/api/ai/status", chatController.GetStatus)
}
}

```

A.19 - Middleware: backend/middlewares/checkAuth.go

```

package middlewares

import (
    "fmt"
    "log"
    "net/http"
    "os"
    "strings"
    "time"

    "github.com/gin-gonic/gin"
    "github.com/golang-jwt/jwt/v4"

    "my-health/initializers"
    "my-health/models"
)

func CheckAuth(c *gin.Context) {

    authHeader := c.GetHeader("Authorization")

    if authHeader == "" {
        c.JSON(http.StatusUnauthorized, gin.H{"error": "Authorization header is missing"})
        c.AbortWithStatus(http.StatusUnauthorized)
        return
    }

    authToken := strings.Split(authHeader, " ")
    if len(authToken) != 2 || authToken[0] != "Bearer" {
        c.JSON(http.StatusUnauthorized, gin.H{"error": "Invalid token format"})
        c.AbortWithStatus(http.StatusUnauthorized)
        return
    }

    tokenString := authToken[1]
    token, err := jwt.Parse(tokenString, func(token *jwt.Token) (interface{}, error) {
        if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
            return nil, fmt.Errorf("Unexpected signing method: %v", token.Header["alg"])
        }
        return []byte(os.Getenv("SECRET")), nil
    })
    if err != nil || !token.Valid {

```

```

        c.JSON(http.StatusUnauthorized, gin.H{"error": "Invalid or expired
token"})
        c.AbortWithStatus(http.StatusUnauthorized)
        return
    }

    claims, ok := token.Claims.(jwt.MapClaims)
    if !ok {
        c.JSON(http.StatusUnauthorized, gin.H{"error": "Invalid token
claims"})
        c.Abort()
        return
    }

    // Log claims for debugging
    log.Printf("Token claims: %+v", claims)

    if float64(time.Now().Unix()) > claims["exp"].(float64) {
        c.JSON(http.StatusUnauthorized, gin.H{"error": "Token expired"})
        c.AbortWithStatus(http.StatusUnauthorized)
        return
    }

    var user models.User
    userIDInterface := claims["user_id"]
    if userIDInterface == nil {
        c.JSON(http.StatusUnauthorized, gin.H{"error": "No user_id in
token"})
        c.AbortWithStatus(http.StatusUnauthorized)
        return
    }

    var userID uint
    switch v := userIDInterface.(type) {
    case float64:
        userID = uint(v)
    case float32:
        userID = uint(v)
    case int:
        userID = uint(v)
    case int32:
        userID = uint(v)
    case int64:
        userID = uint(v)
    case uint:
        userID = v
    case uint32:
        userID = uint(v)
    case uint64:
        userID = uint(v)
    default:
        log.Printf("Invalid user_id type in token: %T", userIDInterface)
        c.JSON(http.StatusUnauthorized, gin.H{"error": "Invalid user_id type
in token"})
        c.AbortWithStatus(http.StatusUnauthorized)
        return
    }

```

```

    if err := initializers.DB.First(&user, userID).Error; err != nil {
        log.Printf("Failed to find user with ID %v: %v", userID, err)
        c.JSON(http.StatusUnauthorized, gin.H{"error": "User not found"})
        c.AbortWithStatus(http.StatusUnauthorized)
        return
    }

    log.Printf("Successfully authenticated user: ID=%v, Role=%v", user.ID,
user.Role)
    c.Set("currentUser", user)

    c.Next()
}

```

A.20 - Χρήση του αυθεντικοποιημένου χρήστη στους ελεγκτές (patientController.go, GetPatientDetails, απόσπασμα)

```

func GetPatientDetails(c *gin.Context) {
    patientID := c.Param("id")

    var patient models.Patient
    log.Printf("Attempting to fetch patient with ID: %s", patientID)

    log.Printf("Attempting to fetch patient details for ID: %s", patientID)

    // Try to find by ID first
    query := initializers.DB.Model(&models.Patient{}).
        Preload("User").
        Where("id = ?", patientID)
    log.Printf("Executing query for ID: %s", patientID)
    err := query.First(&patient).Error

    if err != nil {
        log.Printf("Not found by ID, trying user_id")
        query = initializers.DB.Model(&models.Patient{}).
            Preload("User").
            Where("user_id = ?", patientID)
        log.Printf("Executing query for user_id: %s", patientID)
        if err := query.First(&patient).Error; err != nil {
            c.JSON(http.StatusNotFound, gin.H{"error": "Patient not found"})
            return
        }
    }

    // Debug log the raw patient data right after fetching
    log.Printf("Raw patient data from DB: %v", patient)
    log.Printf("Patient.Address value: %q", patient.Address)

    // Double check with a fresh database query

```

A.21 - Αρχείο: backend/initializers/database.go

```

package initializers

import (
    "log"

```

```

"os"

"gorm.io/driver/postgres"
"gorm.io/gorm"

"my-health/models"
)

var DB *gorm.DB

func ConnectDatabase() {
    // Get environment variables
    dbHost := os.Getenv("DB_HOST")
    dbUser := os.Getenv("DB_USER")
    dbPassword := os.Getenv("DB_PASSWORD")
    dbName := os.Getenv("DB_NAME")

    dsn := "host=" + dbHost + " user=" + dbUser + " password=" + dbPassword +
    " dbname=" + dbName + " port=5432 sslmode=disable"
    database, err := gorm.Open(postgres.Open(dsn), &gorm.Config{})
    if err != nil {
        log.Fatalf("Failed to connect to DB: %v", err)
    }
    if err := database.AutoMigrate(&models.User{}, &models.Patient{},
    &models.Household{},
    &models.Invitation{}, &models.HealthMetrics{}, &models.ChatSession{},
    &models.ChatMessage{}); err != nil {
        panic(err)
    }

    DB = database
    log.Println("Database connection established")
}

func SyncDatabase() {
    err := DB.AutoMigrate(
        &models.User{},
        &models.Patient{},
        &models.Household{},
        &models.HealthMetrics{},
        &models.Invitation{},
        &models.ChatSession{},
        &models.ChatMessage{},
    )
    if err != nil {
        log.Fatal("Failed to sync database:", err)
    }
}

```

A.22 - Αρχείο: frontend/src/LoginPage.tsx (απόσπασμα)

```

const LoginPage: React.FC<LoginPageProps> = ({ onCreateUserClick,
onLoginSuccess }) => {
    const navigate = useNavigate();
    const [username, setUsername] = useState('');
    const [password, setPassword] = useState('');
    const [error, setError] = useState('');

```

```

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  if (!username || !password) {
    setError('Username and password are required');
    return;
  }

  try {
    const response = await axios.post('http://localhost:8080/login',
    {
      username,
      password,
    });

    console.log(response.data);
    const { role, token, userId, hasDetails } = response.data;

    if (role === "admin" || role === "patient") {
      onLoginSuccess(role, token, userId);

      // If it's a patient, check if they need to fill in details
      if (role === "patient") {
        if (!hasDetails) {
          // Redirect to first-time patient form
          navigate('/patient-first-time');
        } else {
          // Redirect to dashboard
          navigate('/dashboard');
        }
      } else {
        // For admin, always go to dashboard
        navigate('/dashboard');
      }
    } else {
      setError('Insufficient permissions');
    }
  } catch (err) {
    console.error(err);
    if (axios.isAxiosError(err) && err.response) {
      setError(err.response.data.message || 'Invalid credentials');
    } else {
      setError('An error occurred. Please try again later.');
```

A.23 - Αρχείο: frontend/src/PatientDetails.tsx (απόσπασμα: ανάκτηση μετρήσεων υγείας)

```

export default function PatientDetails() {
  const { patientId } = useParams<{ patientId: string }>();
  const navigate = useNavigate();
  const [patientData, setPatientData] = useState<PatientData | null>(null);
  const [healthMetrics, setHealthMetrics] = useState<HealthMetric[]>([]);
  const [chartData, setChartData] = useState<HealthMetric[]>([]);
  const [loading, setLoading] = useState(true);
  const [refreshing, setRefreshing] = useState(false);
```

```

const [error, setError] = useState<string | null>(null);
const [tabValue, setTabValue] = useState(0);

// Get current user role and ID from localStorage for AI chat
const userRole = localStorage.getItem('userRole') as 'patient' | 'admin'
| null;
const currentUserId = localStorage.getItem('userId') || '';

const handleTabChange = (event: React.SyntheticEvent, newValue: number)
=> {
  setTabValue(newValue);
};

const fetchHealthMetrics = useCallback(async () => {
  try {
    setRefreshing(true);
    const token = localStorage.getItem('token');
    if (!token || !patientId) {
      throw new Error('No authentication token or patient ID
found');
    }

    const response = await
axios.get(`http://localhost:8080/api/health-metrics/${patientId}`, {
  headers: { Authorization: `Bearer ${token}` }
});

    if (response.data) {
      // Sort metrics by date descending for latest values (newest
first)
      const sortedMetrics = [...response.data].sort((a, b) =>
        new Date(b.date).getTime() - new Date(a.date).getTime()
      );
      setHealthMetrics(sortedMetrics);

      // Sort metrics by date ascending for charts (chronological:
past → today)
      const chartMetrics = [...response.data].sort((a, b) =>
        new Date(a.date).getTime() - new Date(b.date).getTime()
      );
      setChartData(chartMetrics);
    }
  } catch (error: any) {
    console.error('Failed to fetch health metrics:', error);
    setError(error.response?.data?.error || error.message || 'Failed
to fetch health metrics');
  } finally {
    setRefreshing(false);
  }
}, [patientId]);

useEffect(() => {

```

A.24 - Αρχείο: frontend/src/components/AIChat/AIChatBot.tsx (απόσπασμα)

```

const sendMessage = async (messageText: string) => {
  if (!messageText.trim()) return;

```

```

const userMessage: ChatMessage = {
  id: Date.now(),
  role: 'user',
  content: messageText,
  timestamp: new Date().toISOString(),
};

setMessages(prev => [...prev, userMessage]);
setInput('');
setIsTyping(true);
setError(null);

try {
  const token = localStorage.getItem('token');
  const response = await axios.post('http://localhost:8080/api/ai/chat',
    { message: messageText },
    { headers: { Authorization: `Bearer ${token}` } }
  );

  if (response.data.success) {
    const aiMessage: ChatMessage = {
      id: Date.now() + 1,
      role: 'assistant',
      content: response.data.response,
      timestamp: new Date().toISOString(),
      response_time: response.data.response_time,
      tokens_used: response.data.tokens_used,
    };

    setMessages(prev => [...prev, aiMessage]);
  } else {
    throw new Error(response.data.error || 'Failed to get AI response');
  }
} catch (error: any) {
  console.error('Chat error:', error);
  setError(error.response?.data?.error || error.message || 'Failed to send message');

  const errorMessage: ChatMessage = {
    id: Date.now() + 1,
    role: 'assistant',
    content: "I'm having trouble processing your request right now. Please try again in a few moments.",
    timestamp: new Date().toISOString(),
  };

  setMessages(prev => [...prev, errorMessage]);
} finally {
  setIsTyping(false);
}
};

```

A.25 - Αρχείο: infra/main.tf (ανάπτυξη Azure)

```

terraform {
  required_version = ">= 1.0"
}

```

```

    required_providers {
      azurerm = {
        source = "hashicorp/azurerm"
        version = "~> 3.0"
      }
      random = {
        source = "hashicorp/random"
        version = "~> 3.1"
      }
    }

    backend "azurerm" {
      resource_group_name = "rg-dk"
      storage_account_name = "myhealthterraformstate001"
      container_name      = "tfstate"
      key                  = "my-health-app.terraform.tfstate"
    }

    provider "azurerm" {
      features {}
    }

    # Generate random password for PostgreSQL
    resource "random_password" "postgres_password" {
      length = 16
      special = true
    }

    # Existing Resource Group (pre-created) - removed from Terraform management
    data "azurerm_resource_group" "existing" {
      name = var.existing_resource_group_name
    }

    locals {
      common_tags = {
        Environment = var.environment
        Project      = "my-health"
        ManagedBy    = "terraform"
      }
    }

    # Azure Container Registry
    resource "azurerm_container_registry" "main" {
      name                = var.acr_name
      resource_group_name = data.azurerm_resource_group.existing.name
      location            = data.azurerm_resource_group.existing.location
      sku                 = "Basic"
      admin_enabled       = true

      tags = local.common_tags
    }

    # Log Analytics Workspace for Container Apps
    resource "azurerm_log_analytics_workspace" "main" {
      name                = "${var.app_name}-logs"
      location            = data.azurerm_resource_group.existing.location

```

```

resource_group_name = data.azure_rm_resource_group.existing.name
sku                 = "PerGB2018"
retention_in_days   = 30

tags = local.common_tags
}

# Container Apps Environment
resource "azurerm_container_app_environment" "main" {
  name           = "${var.app_name}-env"
  location       = data.azure_rm_resource_group.existing.location
  resource_group_name = data.azure_rm_resource_group.existing.name
  log_analytics_workspace_id = azurerm_log_analytics_workspace.main.id

  tags = local.common_tags
}

# PostgreSQL Flexible Server
resource "azurerm_postgresql_flexible_server" "main" {
  name           = "${var.app_name}-postgres"
  resource_group_name = data.azure_rm_resource_group.existing.name
  location       = data.azure_rm_resource_group.existing.location
  version        = "15"
  delegated_subnet_id = null
  private_dns_zone_id = null
  public_network_access_enabled = true

  administrator_login     = var.postgres_admin_login
  administrator_password = random_password.postgres_password.result

  zone = "1"

  storage_mb      = 32768
  storage_tier    = "P4"
  sku_name        = "B_Standard_B1ms"
  backup_retention_days = 7

  tags = local.common_tags
}

# PostgreSQL Database
resource "azurerm_postgresql_flexible_server_database" "main" {
  name       = var.database_name
  server_id  = azurerm_postgresql_flexible_server.main.id
  collation = "en_US.utf8"
  charset   = "utf8"
}

# PostgreSQL Firewall Rule - Allow all (for demonstration purposes)
resource "azurerm_postgresql_flexible_server_firewall_rule" "allow_all" {
  name           = "allow-all-ips"
  server_id      = azurerm_postgresql_flexible_server.main.id
  start_ip_address = "0.0.0.0"
  end_ip_address   = "255.255.255.255"
}

```

```

# Backend Container App
resource "azurerm_container_app" "backend" {
  name           = "${var.app_name}-backend"
  container_app_environment_id = azurerm_container_app_environment.main.id
  resource_group_name = data.azurerm_resource_group.existing.name
  revision_mode     = "Single"

  template {
    container {
      name     = "${var.app_name}-backend"
      image    = var.backend_image
      cpu      = 0.25
      memory   = "0.5Gi"

      env {
        name = "DATABASE_URL"
        value = "postgres://${var.postgres_admin_login}:${random_password.postgres_password.result}@${azurerm_postgresql_flexible_server.main.fqdn}:5432/${var.database_name}?sslmode=require"
      }

      env {
        name = "POSTGRES_HOST"
        value = azurerm_postgresql_flexible_server.main.fqdn
      }

      env {
        name = "POSTGRES_PORT"
        value = "5432"
      }

      env {
        name = "POSTGRES_DB"
        value = var.database_name
      }

      env {
        name = "POSTGRES_USER"
        value = var.postgres_admin_login
      }

      env {
        name       = "POSTGRES_PASSWORD"
        secret_name = "postgres-password"
      }

      env {
        name = "NODE_ENV"
        value = var.environment
      }

      env {
        name = "PORT"
        value = toString(var.backend_port)
      }
    }
  }
}

```

```

    }

    min_replicas = 1
    max_replicas = 3
  }

  secret {
    name = "postgres-password"
    value = random_password.postgres_password.result
  }

  secret {
    name = "acr-password"
    value = azurerm_container_registry.main.admin_password
  }

  ingress {
    allow_insecure_connections = false
    external_enabled          = true
    target_port                = var.backend_port
  }

  traffic_weight {
    percentage = 100
    latest_revision = true
  }
}

tags = local.common_tags
}

# Frontend Container App
resource "azurerm_container_app" "frontend" {
  name = "${var.app_name}-frontend"
  container_app_environment_id = azurerm_container_app_environment.main.id
  resource_group_name = data.azurerm_resource_group.existing.name
  revision_mode = "Single"

  template {
    container {
      name = "${var.app_name}-frontend"
      image = var.frontend_image
      cpu = 0.25
      memory = "0.5Gi"

      env {
        name = "REACT_APP_API_URL"
        value = "https://${azurerm_container_app.backend.latest_revision_fqdn}"
      }

      env {
        name = "NODE_ENV"
        value = var.environment
      }

      env {

```

```

        name = "PORT"
        value = toString(var.frontend_port)
    }
}

min_replicas = 1
max_replicas = 3
}

secret {
    name = "acr-password"
    value = azurerm_container_registry.main.admin_password
}

ingress {
    allow_insecure_connections = false
    external_enabled          = true
    target_port                = var.frontend_port
}

traffic_weight {
    percentage = 100
    latest_revision = true
}
}

tags = local.common_tags
}

```

A.26 - Αρχείο: docker-compose.yml (ενορχήστρωση υπηρεσιών)

```

services:
  postgres-gorm:
    image: postgres:13
    environment:
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mysecretpassword
      POSTGRES_DB: mydatabase
      POSTGRES_HOST: postgres-gorm
      POSTGRES_PORT: 5432
      POSTGRES_SSLMODE: disable
    volumes:
      - db_data:/var/lib/postgresql/data

  backend:
    container_name: my-health-backend
    build:
      context: ./backend
      dockerfile: Dockerfile.dev
    ports:
      - '8080:8080'
    command: ['go', 'run', 'main.go']
    depends_on:
      - postgres-gorm
    environment:
      SECRET: vNrSf+JwRpc/11gHFu6Edf02p0SXEfLNlapBkShMBVQ=
      DB_HOST: postgres-gorm
      DB_USER: myuser

```

```

    DB_PASSWORD: mysecretpassword
    DB_NAME: mydatabase
    OLLAMA_URL: http://ollama:11434
    volumes:
      - ./backend:/my-health
    working_dir: /my-health

frontend:
  container_name: my-health-frontend
  build:
    context: ./frontend
    dockerfile: Dockerfile.dev
  volumes:
    - './frontend:/app' # Mount the frontend source code
    - '/app/node_modules' # Prevent node_modules from being overridden by
the container
  ports:
    - '3000:3000'
  environment:
    - NODE_ENV=development
  command: sh -c "yarn install && yarn start"
  working_dir: /app

ollama:
  container_name: my-health-ollama
  image: ollama/ollama:latest
  ports:
    - '11434:11434'
  volumes:
    - ollama_models:/root/.ollama
  environment:
    - OLLAMA_ORIGINS=*
    - OLLAMA_HOST=0.0.0.0:11434
  restart: unless-stopped
  healthcheck:
    test: ['CMD', 'curl', '-f', 'http://localhost:11434/api/tags']
    interval: 30s
    timeout: 10s
    retries: 3
    start_period: 5s

volumes:
  db_data:
  ollama_models:

```

A.27 - Αρχείο: backend/Dockerfile.dev (Ανάπτυξη με Hot-Reload)

```

# Stage 1: Build the Go application
FROM golang:1.23-alpine AS build

# Environment variables which CompileDaemon requires to run
ENV PROJECT_DIR=/my-health \
    GO111MODULE=on \
    CGO_ENABLED=0

# Set the Current Working Directory inside the container
WORKDIR /my-health

```

```
# Copy go.mod and go.sum files from the project root
COPY go.mod go.sum ./

# Download all dependencies and tidy modules
RUN go mod download && go mod tidy

# Copy the entire source code from your local directory into the container
COPY . .

# Run go mod tidy again after copying source code to ensure consistency
RUN go mod tidy

# Expose port 8080 (or the port your service runs on)
EXPOSE 8080

# For development, use CompileDaemon for hot-reloading
RUN go install github.com/githubnemo/CompileDaemon@latest

# Command for hot-reloading during development (use ENTRYPOINT for dev)
ENTRYPOINT [ "CompileDaemon", "--build=go mod tidy && go build main.go", "--command=./main" ]

# # Build the Go app for production
# RUN go build -o main cmd/auth-service/main.go

# For production (uncomment below if you're targeting production and remove the ENTRYPOINT line)
# CMD [ "./main" ]
```

A.28 - Dockerfile παραγωγής (κατασκευή πολλαπλών σταδίων):

```
# Παραγωγή: μικρότερη εικόνα, χωρίς hot-reload (πρόταση για μελλοντική χρήση)
FROM golang:1.23-alpine AS build
WORKDIR /app
COPY go.mod go.sum ./
RUN go mod download
COPY . .
RUN CGO_ENABLED=0 GOOS=linux go build -o main .

# Τελικό στάδιο: ελάχιστη εικόνα Alpine Linux
FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=build /app/main .
EXPOSE 8080
CMD [ "./main" ]
```

A.29 - Παράδειγμα απόκρισης δημιουργίας χρήστη (200 OK)

```
HTTP/1.1 200 OK
```

```
{
  "data": {
    "ID": 42,
    "CreatedAt": "2026-05-01T10:15:00Z",
    "UpdatedAt": "2026-05-01T10:15:00Z",
```

```

      "DeletedAt": null,
      "username": "johndoe72",
      "role": "patient",
      "is_in_household": false,
      "households": null
    }
  }
}

```

// Ο κωδικός πρόσβασης δεν περιλαμβάνεται ποτέ στην απόκριση (ετικέτα json:"- " στο μοντέλο User).

A.30 - Το Frontend App.tsx διατηρεί την κατάσταση πιστοποίησης:

```

const App = () => {
  const [userRole, setUserRole] = useState<string | null>(null);
  const [token, setToken] = useState<string | null>(null);
  const [userId, setUserId] = useState<string | null>(null);
  const navigate = useNavigate();
  const location = useLocation();

  useEffect(() => {
    const storedRole = localStorage.getItem('userRole');
    const storedToken = localStorage.getItem('token');
    const storedUserId = localStorage.getItem('userId');
    if (!storedRole || !storedToken || !storedUserId) {
      if (location.pathname !== '/create-user') {
        localStorage.removeItem('userRole');
        localStorage.removeItem('token');
        localStorage.removeItem('userId');
        navigate('/login');
      }
    } else {
      setUserRole(storedRole);
      setToken(storedToken);
      setUserId(storedUserId);
    }
  }, [navigate, location.pathname]);

  const handleLoginSuccess = (role: string, newToken: string, newUserId: string) => {
    setUserRole(role);
    setToken(newToken);
    setUserId(newUserId);
    localStorage.setItem('userRole', role);
    localStorage.setItem('token', newToken);
    localStorage.setItem('userId', newUserId);
  };

  const handleLogout = () => {
    setUserRole(null);
    setToken(null);
    setUserId(null);
    localStorage.removeItem('userRole');
    localStorage.removeItem('token');
    localStorage.removeItem('userId');
    navigate('/login');
  };
}

```

A.31 - Φόρμα Material-UI (frontend/src/PatientEditPage.tsx, απόσπασμα)

```

<Grid container spacing={2}>
  <Grid item xs={12} sm={6}>
    <TextField
      fullWidth
      label="First Name"
      name="name"

value={patientData.name}

onChange={handleInputChange}
      required
    />
  </Grid>
  <Grid item xs={12} sm={6}>
    <TextField
      fullWidth
      label="Last Name"
      name="surname"

value={patientData.surname}

onChange={handleInputChange}
      required
    />
  </Grid>
  <Grid item xs={12} sm={6}>
    <TextField
      fullWidth
      label="Date of Birth"
      name="dateOfBirth"
      type="date"

value={patientData.dateOfBirth}

onChange={handleInputChange}
      InputLabelProps={{
shrink: true }}
      inputProps={{
        min: "1920-01-01",
        max: new
Date().toISOString().split('T')[0]
      }}
      required
    />
  </Grid>
  <Grid item xs={12} sm={6}>
    <FormControl fullWidth
required>
      <InputLabel
id="gender-label">Gender</InputLabel>
      <Select
        labelId="gender-
label"
        name="gender"

```

```
value={patientData.gender}
label="Gender"

onChange={handleInputChange}
>
```

A.32 - Παράδειγμα αιτήματος ενημέρωσης ασθενούς (POST /patient/edit/:id)

```
POST /patient/edit/42
Authorization: Bearer eyJhbGc...

{
  "name": "John",
  "surname": "Doe",
  "dateOfBirth": "1950-03-15",
  "address": "Athens, Greece",
  "gender": "male",
  "bloodType": "A+",
  "height": 175,
  "medicalHistory": "Hypertension",
  "allergies": "Penicillin",
  "medications": "Lisinopril 10mg",
  "emergencyContact": {
    "name": "Jane Doe",
    "relationship": "Daughter",
    "phoneNumber": "+30 691 234 5678"
  }
}

// Τα ονόματα των πεδίων αντιστοιχούν στη δομή requestBody του
UpdatePatient (βλ. Παράρτημα A.33).
```

A.33 - Αρχείο: backend/controllers/patientController.go (UpdatePatient, απόσπασμα)

```
func UpdatePatient(c *gin.Context) {
    patientID := c.Param("id")

    var requestBody struct {
        Name          string `json:"name"`
        Surname       string `json:"surname"`
        DateOfBirth   string `json:"dateOfBirth"`
        Address       string `json:"address"`
        MedicalRecord string `json:"medicalRecord"`
        Gender        string `json:"gender"`
        BloodType     string `json:"bloodType"`
        Height        float64 `json:"height"`
        MedicalHistory string `json:"medicalHistory"`
        Allergies     string `json:"allergies"`
        Medications   string `json:"medications"`
        EmergencyContact EmergencyContactRequest `json:"emergencyContact"`
        HealthMetrics HealthMetricsRequest `json:"healthMetrics"`
    }
}
```

```

    if err := c.ShouldBindJSON(&requestBody); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    // Validate blood pressure if provided
    if requestBody.HealthMetrics.SystolicBP != 0 ||
requestBody.HealthMetrics.DiastolicBP != 0 {
        if !validateBloodPressure(requestBody.HealthMetrics.SystolicBP,
requestBody.HealthMetrics.DiastolicBP) {
            c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid
blood pressure values"})
            return
        }
    }

    var patient models.Patient
    // First try to find by ID
    result := initializers.DB.Where("id = ?",
patientID).First(&patient)

    if result.Error != nil {
        // If not found by ID, specifically check if this is a user_id
case
        var checkPatient models.Patient
        checkResult := initializers.DB.Where("user_id = ?",
patientID).First(&checkPatient)

        if checkResult.Error != nil {
            // Patient doesn't exist, check if user exists and create
patient record
            var user models.User
            userResult := initializers.DB.Where("id = ?",
patientID).First(&user)

            if userResult.Error != nil {
                c.JSON(http.StatusNotFound, gin.H{"error": "User not
found"})
                return
            }

            if user.Role != "patient" {
                c.JSON(http.StatusBadRequest, gin.H{"error": "User is
not a patient"})
                return
            }

            // Create new patient record for existing user
            patient = models.Patient{
                UserID: user.ID,
                Name: "", // Will be filled from request
                Surname: "", // Will be filled from request
            }

            log.Printf("Creating new patient record for user ID: %d",
user.ID)

```

```
// ... (συνέχεια: ενημέρωση πεδίων και αποθήκευση - βλ. πηγαίο
αρχείο)
```

A.34 - Διάγραμμα τάσεων αρτηριακής πίεσης (frontend/src/PatientDetails.tsx, απόσπασμα)

```
<TabPanel value={tabValue} index={0}>
  <ResponsiveContainer width="100%"
height={400}>
    <LineChart data={chartData}>
      <CartesianGrid strokeDasharray="3
3" />
      <XAxis
        dataKey="date"
        tickFormatter={formatChartDate}
        interval="preserveStartEnd"
        minTickGap={50}
      />
      <YAxis
        domain={['dataMin - 10',
'dataMax + 10']}
        label={{ value: 'mmHg',
angle: -90, position: 'insideLeft' }}
      />
      <Tooltip
        labelFormatter={formatTooltipDate}
        contentStyle={{ background:
'#fff', border: '1px solid #ccc' }}
      />
    <Legend verticalAlign="top"
height={36} />
    <Line
      type="monotone"
      dataKey="systolic_bp"
      stroke="#e74c3c"
      name="Systolic BP"
      strokeWidth={2}
      dot={false}
      activeDot={{ r: 6 }}
    />
    <Line
      type="monotone"
      dataKey="diastolic_bp"
      stroke="#3498db"
      name="Diastolic BP"
      strokeWidth={2}
      dot={false}
      activeDot={{ r: 6 }}
    />
  </LineChart>
</ResponsiveContainer>
</TabPanel>

<TabPanel value={tabValue} index={1}>
  <ResponsiveContainer width="100%"
height={400}>
```

A.35 - Επικύρωση από την πλευρά του πελάτη (frontend/src/PatientFirstTimeForm.tsx, απόσπασμα)

```
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  setError('');

  if (!formData.name || !formData.surname ||
    !formData.dateOfBirth || !formData.gender) {
    setError('Please fill in all required fields');
    return;
  }

  try {
    const token = localStorage.getItem('token');
    const userId = localStorage.getItem('userId');

    if (!token || !userId) {
      setError('Authentication error');
      navigate('/login');
    }
  }
}
```

A.36 - Παράδειγμα κλήσης API μετρήσεων υγείας (GET /api/health-metrics/:patientId)

```
GET /api/health-metrics/42
Authorization: Bearer eyJhbGc...

// Απόκριση 200 OK (απόσπασμα - πλήρης λίστα εγγραφών HealthMetrics)
[
  {
    "patient_id": 42,
    "date": "2026-05-01T08:30:00Z",
    "systolic_bp": 128,
    "diastolic_bp": 82,
    "heart_rate": 76,
    "weight": 82.5,
    "oxygen_saturation": 97.2,
    "steps_count": 5400,
    "sleep_duration": 7.2,
    "irregular_rhythm": false,
    "fall_detected": false
  }
]

// Τα ονόματα των πεδίων αντιστοιχούν στις ετικέτες json του μοντέλου
HealthMetrics (βλ. Παράρτημα A.10).
```

A.37 - Επικύρωση και αρχικοποίηση μετρήσεων στο backend (patientController.go, απόσπασμα)

```
func ensurePatientMetrics(patientID uint) error {
  // Check for data in the last 30 days
  thirtyDaysAgo := time.Now().AddDate(0, 0, -30)
  var metrics []models.HealthMetrics

  if err := initializers.DB.Where("patient_id = ? AND date >= ?",
    patientID, thirtyDaysAgo).
    Order("date ASC").
    Find(&metrics).Error; err != nil {
```

```

        return err
    }

    // If we have no data at all, generate full history
    if len(metrics) == 0 {
        if err := mockhealth.GenerateHistoricalData(patientID); err !=
nil {
            return err
        }
        mockhealth.StartPatientDataWorker(patientID)
        return nil
    }

    // Generate mock data for missing dates
    existingDates := make(map[string]bool)
    for _, metric := range metrics {
        existingDates[metric.Date.Format("2006-01-02")] = true
    }

    now := time.Now()
    for d := thirtyDaysAgo; d.Before(now) || d.Equal(now); d =
d.AddDate(0, 0, 1) {
        dateStr := d.Format("2006-01-02")
        if !existingDates[dateStr] {
            if _, err :=
mockhealth.GenerateMockDataForPatient(patientID, d); err != nil {
                return err
            }
        }
    }

    mockhealth.StartPatientDataWorker(patientID)
    return nil
}

// validateBloodPressure checks if blood pressure values are within
normal range
func validateBloodPressure(systolic, diastolic int) bool {
    return systolic >= 70 && systolic <= 190 && diastolic >= 40 &&
diastolic <= 130
}

type EmergencyContactRequest struct {
    Name          string `json:"name"`
    Relationship   string `json:"relationship"`
    PhoneNumber    string `json:"phoneNumber"`
}

type HealthMetricsRequest struct {
    Weight          float64 `json:"weight"`
    HeartRate       int      `json:"heartRate"`
    SystolicBP      int      `json:"systolicBP"`
    DiastolicBP     int      `json:"diastolicBP"`
    OxygenSaturation float64 `json:"oxygenSaturation"`
    BloodPressure   string  `json:"bloodPressure"`
    LastCheckup     string  `json:"lastCheckup"`
}

```

A.38 - Το frontend επισημαίνει τιμές που ενδέχεται να προκαλούν ανησυχία:

```
const getBloodPressureColor = (systolic: number, diastolic: number) => {
  if (systolic >= 180 || diastolic >= 120) {
    return 'error'; // Κόκκινο: υπερτασική κρίση
  } else if (systolic >= 140 || diastolic >= 90) {
    return 'warning'; // Πορτοκαλί: υπέρταση
  } else if (systolic < 90 || diastolic < 60) {
    return 'warning'; // Πορτοκαλί: υπόταση
  }
  return 'success'; // Πράσινο: φυσιολογικό
};
```

// Χρήση στο UI

```
<Chip
  label={` ${systolicBP}/${diastolicBP} mmHg`}
  color={getBloodPressureColor(systolicBP, diastolicBP)}
/>
```

A.39 - Λειτουργία εξαγωγής δεδομένων:

```
const exportToCSV = () => {
  const csvHeaders =
    'Date,SystolicBP,DiastolicBP,HeartRate,Weight,OxygenSaturation,Steps\n';
  const csvRows = metrics.map(m =>

`${m.date},${m.systolic_bp},${m.diastolic_bp},${m.heart_rate},${m.weight},${m
.oxygen_saturation},${m.steps_count}`
).join('\n');

  const csvContent = csvHeaders + csvRows;
  const blob = new Blob([csvContent], { type: 'text/csv' });
  const url = URL.createObjectURL(blob);

  const link = document.createElement('a');
  link.href = url;
  link.download = `health-metrics-${new Date().toISOString().slice(0,
10)}.csv`;
  link.click();
  URL.revokeObjectURL(url);
};
```

A.40 - Χειριστής μηνυμάτων συνομιλίας (backend/controllers/chatController.go, SendMessage)

```
// SendMessage handles POST /api/ai/chat
func (cc *ChatController) SendMessage(c *gin.Context) {
  // Get user from JWT token (set by auth middleware)
  userInterface, exists := c.Get("currentUser")
  if !exists {
    c.JSON(http.StatusUnauthorized, gin.H{"error": "Authentication
required"})
    return
  }

  user, ok := userInterface.(models.User)
  if !ok {
    c.JSON(http.StatusInternalServerError, gin.H{"error": "Invalid
user data"})
  }
}
```

```

    }
    return

    var req ChatMessageRequest
    if err := c.ShouldBindJSON(&req); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid request format", "details": err.Error()})
        return
    }

    // Validate message length
    if len(req.Message) > 1000 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Message too long. Maximum 1000 characters allowed."})
        return
    }

    if len(req.Message) < 1 {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Message cannot be empty."})
        return
    }

    // Create chat request
    chatReq := ai.ChatRequest{
        UserID: user.ID,
        Message: req.Message,
    }

    // Process message through AI service
    response, err := cc.chatService.ProcessMessage(chatReq)
    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{
            "error": "Failed to process message",
            "details": err.Error(),
        })
        return
    }

    if !response.Success {
        c.JSON(http.StatusServiceUnavailable, gin.H{
            "error": response.Error,
        })
        return
    }

    c.JSON(http.StatusOK, gin.H{
        "response": response.Response,
        "session_id": response.SessionID,
        "response_time": response.ResponseTime,
        "tokens_used": response.TokensUsed,
        "success": true,
    })
}

```

A.41 - Δημιουργία προτροπής με βάση το πλαίσιο (contextBuilder.go, GenerateHealthcarePrompt)

```

func GenerateHealthcarePrompt(context *UserContext, userMessage string)
string {
    var prompt strings.Builder

    prompt.WriteString("You are a helpful AI assistant for an elderly
health monitoring system.\n\n")

    if context.IsPatient && context.PatientData != nil {
        prompt.WriteString("=== ACCESS LEVEL: PATIENT ===\n")
        prompt.WriteString(fmt.Sprintf("Current User: %s (Patient)\n",
context.PatientData.Name))
        prompt.WriteString("You can ONLY answer questions about THIS
PATIENT'S own health data below.\n\n")
        prompt.WriteString("Patient's Health Information:\n")
        prompt.WriteString(fmt.Sprintf("- Age: %d years old\n",
context.PatientData.Age))
        prompt.WriteString(fmt.Sprintf("- Gender: %s\n",
context.PatientData.Gender))
        prompt.WriteString(fmt.Sprintf("- Blood Type: %s\n",
context.PatientData.BloodType))
        prompt.WriteString(fmt.Sprintf("- Height: %.1f cm\n",
context.PatientData.Height))

        if context.PatientData.Medications != "" {
            prompt.WriteString(fmt.Sprintf("- Current Medications:
%s\n", context.PatientData.Medications))
        }

        if context.PatientData.Allergies != "" {
            prompt.WriteString(fmt.Sprintf("- Known Allergies: %s\n",
context.PatientData.Allergies))
        }

        if context.PatientData.LatestMetrics != nil {
            metrics := context.PatientData.LatestMetrics
            prompt.WriteString("- Latest Vital Signs:\n")
            prompt.WriteString(fmt.Sprintf(" • Blood Pressure: %d/%d
mmHg\n", metrics.SystolicBP, metrics.DiastolicBP))
            prompt.WriteString(fmt.Sprintf(" • Heart Rate: %d bpm\n",
metrics.HeartRate))
            prompt.WriteString(fmt.Sprintf(" • Weight: %.1f kg\n",
metrics.Weight))
            prompt.WriteString(fmt.Sprintf(" • Oxygen Saturation:
%.1f%\n", metrics.OxygenSaturation))
            if metrics.StepsCount > 0 {
                prompt.WriteString(fmt.Sprintf(" • Daily Steps:
%d\n", metrics.StepsCount))
            }
        }

        prompt.WriteString("\n=== INSTRUCTIONS ===\n")
        prompt.WriteString("- Answer questions about this patient's
health data\n")
        prompt.WriteString("- Explain vital signs in simple terms\n")
        prompt.WriteString("- For medical concerns, suggest consulting
healthcare providers\n\n")
    } else if context.AdminData != nil {

```

```

    prompt.WriteString("=== ACCESS LEVEL: ADMIN/CAREGIVER ===\n")
    prompt.WriteString(fmt.Sprintf("Current User: %s\n", context.UserName))
    prompt.WriteString("You have FULL ACCESS to answer questions about ALL patients listed below.\n")
    prompt.WriteString(fmt.Sprintf("Household Information: %s\n\n", context.AdminData.HouseholdInfo))

```

```

    if len(context.AdminData.Patients) > 0 {
        prompt.WriteString("Patients under your care:\n")
        for _, patient := range context.AdminData.Patients {
            prompt.WriteString(fmt.Sprintf("\n• %s (Age: %d, Gender: %s)\n", patient.Name, patient.Age, patient.Gender))
            prompt.WriteString(fmt.Sprintf("    - Blood Type: %s, Height: %.1f cm\n", patient.BloodType, patient.Height))

```

```

            if patient.Medications != "" {
                prompt.WriteString(fmt.Sprintf("    - Medications: %s\n", patient.Medications))
            }

```

```

            if patient.Allergies != "" {
                prompt.WriteString(fmt.Sprintf("    - Allergies: %s\n", patient.Allergies))
            }

```

```

            if patient.LatestMetrics != nil {
                metrics := patient.LatestMetrics
                prompt.WriteString("    - Latest Vital Signs:\n")
                prompt.WriteString(fmt.Sprintf("        • Blood Pressure: %d/%d mmHg\n", metrics.SystolicBP, metrics.DiastolicBP))
                prompt.WriteString(fmt.Sprintf("        • Heart Rate: %d bpm\n", metrics.HeartRate))
                prompt.WriteString(fmt.Sprintf("        • Weight: %.1f kg\n", metrics.Weight))
                prompt.WriteString(fmt.Sprintf("        • Oxygen Saturation: %.1f%\n", metrics.OxygenSaturation))
                if metrics.StepsCount > 0 {
                    prompt.WriteString(fmt.Sprintf("        • Daily Steps: %d\n", metrics.StepsCount))
                }
            }
        }
    }

```

```

    prompt.WriteString("\n=== INSTRUCTIONS ===\n")
    prompt.WriteString("As an admin/caregiver, you can:\n")
    prompt.WriteString("- Answer questions about ANY patient listed above\n")
    prompt.WriteString("- Compare patient data and identify trends\n")
    prompt.WriteString("- Provide health information and explanations\n")
    prompt.WriteString("- Alert on concerning vital signs\n")
    prompt.WriteString("When answering, be specific and use patient names from the data above.\n\n")
}

```

```

    prompt.WriteString(fmt.Sprintf("User Question: %s\n\n",
userMessage))
    prompt.WriteString("Provide a clear, helpful response based on the
data above:")

    return prompt.String()
}

```

A.42 - Κλήση του Ollama API (ollamaClient.go, GenerateResponse)

```

func (oc *OllamaClient) GenerateResponse(model, prompt string)
(*OllamaResponse, error) {
    reqBody := OllamaRequest{
        Model: model,
        Prompt: prompt,
        Stream: false,
    }

    jsonData, err := json.Marshal(reqBody)
    if err != nil {
        return nil, fmt.Errorf("failed to marshal request: %v", err)
    }

    resp, err := oc.Client.Post(
        fmt.Sprintf("%s/api/generate", oc.BaseURL),
        "application/json",
        bytes.NewBuffer(jsonData),
    )
    if err != nil {
        return nil, fmt.Errorf("failed to call Ollama API: %v", err)
    }
    defer resp.Body.Close()

    if resp.StatusCode != http.StatusOK {
        body, _ := io.ReadAll(resp.Body)
        return nil, fmt.Errorf("Ollama API error (status %d): %s",
resp.StatusCode, string(body))
    }

    var ollamaResp OllamaResponse
    if err := json.NewDecoder(resp.Body).Decode(&ollamaResp); err !=
nil {
        return nil, fmt.Errorf("failed to decode Ollama response: %v",
err)
    }

    return &ollamaResp, nil
}

```

A.43 - Αποθήκευση μηνυμάτων στη βάση δεδομένων (chatService.go, απόσπασμα)

```

userMessage := models.ChatMessage{
    SessionID: session.SessionID,
    Role:      "user",
    Content:   req.Message,
}

```

```

    if err := initializers.DB.Create(&userMessage).Error; err != nil {
        return &ChatResponse{
            Success: false,
            Error:    "Failed to save your message.",
        }, err
    }

    // Call Ollama API
    ollamaResp, err := cs.ollamaClient.GenerateResponse("healthbot-
v2", prompt)
    if err != nil {
        // Save error message
        errorMessage := models.ChatMessage{
            SessionID: session.SessionID,
            Role:       "assistant",
            Content:    "I'm having trouble processing your request
right now. Please try again in a few moments.",
        }
        initializers.DB.Create(&errorMessage)

        return &ChatResponse{
            Success: false,
            Error:    "AI service is temporarily unavailable.",
        }, err
    }

    responseTime := int(time.Since(startTime).Milliseconds())

    // Save assistant response
    assistantMessage := models.ChatMessage{
        SessionID: session.SessionID,
        Role:       "assistant",
        Content:    ollamaResp.Response,
        ResponseTime: responseTime,
        TokensUsed: len(ollamaResp.Response) / 4, // Rough token
estimation
    }
    if err := initializers.DB.Create(&assistantMessage).Error; err !=
nil {
        return &ChatResponse{
            Success: false,
            Error:    "Failed to save AI response.",
        }, err
    }

    // Update session last used time
    session.UpdateLastUsed(initializers.DB)

    return &ChatResponse{
        Response:    ollamaResp.Response,
        SessionID:    session.SessionID,
        ResponseTime: responseTime,
        TokensUsed:   assistantMessage.TokensUsed,
        Success:      true,
    }

```

A.44 - Αρχείο: frontend/src/components/PendingInvitations.tsx (απόσπασμα)

```

const [invitations, setInvitations] =
  useState<Invitation[]>([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState<string | null>(null);

const fetchInvitations = async () => {
  try {
    const token = localStorage.getItem('token');
    const response = await
    axios.get('http://localhost:8080/invitations', {
      headers: { Authorization: `Bearer ${token}` }
    });
    setInvitations(response.data.invitations || []);
    setError(null);
  } catch (err: any) {
    setError(err.response?.data?.error || 'Failed to
    fetch invitations');
  } finally {
    setLoading(false);
  }
};

useEffect(() => {
  fetchInvitations();
}, []);

const handleInvitation = async (invitationId: number,
  accept: boolean) => {
  try {
    setLoading(true);
    setError(null);
    const token = localStorage.getItem('token');

    await axios.post('http://localhost:8080/respond-
    invitation', {
      invitation_id: invitationId,
      response: accept ? "accept" : "reject"
    }, {
      headers: { Authorization: `Bearer ${token}` }
    });

    // Wait a bit before refreshing to ensure backend
    has processed the change
    setTimeout(() => {
      fetchInvitations(); // Refresh the list
    }, 500);

  } catch (err: any) {
    setError(err.response?.data?.error || 'Failed to
    respond to invitation');
    setLoading(false);
  }
};

```

A.45 - Προβολή ασθενών στον πίνακα διαχειριστή (frontend/src/AdminDashboard.tsx, απόσπασμα)

```

        <Grid container spacing={3}>
          {patients.map((patient) => (
            <Grid item xs={12} key={patient.id}>
              <Paper elevation={3} sx={{ p: 2 }}>
                <Accordion>
                  <AccordionSummary expandIcon={<ExpandMoreIcon
/>}>
                    <Grid container alignItems="center"
spacing={2}>
                      <Grid item xs={3}>
                        <Typography variant="h6">
                          {patient.name}
                        </Typography>
                      </Grid>
                      <Grid item xs={3}>
                        <Chip
                          label={`Blood Type:
${patient.bloodType || 'N/A'}}`
                          color="primary"
                          size="small"
                        />
                      </Grid>
                      <Grid item xs={3}>
                        <Typography variant="body2"
color="textSecondary">
                          ID: {patient.id}
                        </Typography>
                      </Grid>
                      <Grid item xs={3}>
                        <Box sx={{ display: 'flex', gap:
1 }}>
                          <IconButton
                            color="primary"
                            onClick={() =>
handlePatientDetails(patient.id)}
                          >
                            <VisibilityIcon />
                          </IconButton>
                          <IconButton
                            color="error"
                            onClick={() =>
handleEmergencyContact(patient)}
                          >
                            <EmergencyShareIcon />
                          </IconButton>
                        </Box>
                      </Grid>
                    </Grid>
                  </AccordionSummary>
                <AccordionDetails>

```

A.46 - Κατάσταση ασθενών με χρωματικούς δείκτες (PatientStatusModal.tsx, απόσπασμα)

```

interface PatientStatus {
  id: number;
  username: string;

```

```

    name: string;
    surname: string;
    status: string;
    joined_at?: string;
  }

interface PatientStatusData {
  pending: PatientStatus[];
  approved: PatientStatus[];
  canceled: PatientStatus[];
}

interface PatientStatusModalProps {
  open: boolean;
  onClose: () => void;
}

const PatientStatusModal: React.FC<PatientStatusModalProps> = ({ open,
onClose }) => {
  const [tabValue, setTabValue] = useState(0);
  const [statusData, setStatusData] = useState<PatientStatusData>({
    pending: [],
    approved: [],
    canceled: []
  });
  const [loading, setLoading] = useState(false);
  const theme = useTheme();
  const fullScreen = useMediaQuery(theme.breakpoints.down('md'));

  useEffect(() => {
    if (open) {
      fetchPatientStatuses();
    }
  }, [open]);

  const fetchPatientStatuses = async () => {
    setLoading(true);
    try {
      const response = await
    axios.get('http://localhost:8080/household/patient-statuses', {
        headers: {
          Authorization: `Bearer ${localStorage.getItem('token')}`
        },
      },
    );
    const data = response.data || {};
    setStatusData({

```

A.47 - Παράδειγμα εξόδου (μηνιαία συγκέντρωση):

```

[
  {
    "period_key": "2024-01",
    "avg_systolic_bp": 126.5,
    "avg_diastolic_bp": 81.2,
    "avg_heart_rate": 74.8,
    "avg_weight": 82.3,
    "min_systolic_bp": 118,

```

```

    "max_systolic_bp": 135,
    "total_steps": 95000,
    "measurement_count": 28
  },
  {
    "period_key": "2024-02",
    "avg_systolic_bp": 124.1,
    "avg_diastolic_bp": 79.5,
    "avg_heart_rate": 73.2,
    "avg_weight": 81.8,
    "min_systolic_bp": 115,
    "max_systolic_bp": 132,
    "total_steps": 101200,
    "measurement_count": 29
  }
]

```

A.48 - Χρήση Preload του GORM (householdController.go, GetHouseholdPatients, απόσπασμα)

```

func GetHouseholdPatients(c *gin.Context) {
    currentUser, exists := c.Get("currentUser")
    if !exists {
        c.JSON(http.StatusUnauthorized, gin.H{"error": "User not found in context"})
        return
    }
    admin := currentUser.(models.User)

    if admin.Role != "admin" {
        c.JSON(http.StatusForbidden, gin.H{"error": "user is not an admin"})
        return
    }

    var household models.Household
    result := initializers.DB.
        Preload("Patients").
        Where("admin_id = ?", admin.ID).
        First(&household)

    if result.Error != nil {
        if result.Error == gorm.ErrRecordNotFound {
            household = models.Household{
                AdminID: admin.ID,
            }
            if err := initializers.DB.Create(&household).Error; err != nil {
                c.JSON(http.StatusInternalServerError, gin.H{"error": "failed to create household"})
                return
            }
            c.JSON(http.StatusOK, gin.H{
                "household_id": household.ID,
                "patients":      []interface{}{},
            })
            return
        }
    }
}

```

A.49 — Πλήρες λογικό σχήμα της βάσης δεδομένων (DDL, PostgreSQL)

```

-- =====
-- Λογικό σχήμα της βάσης δεδομένων (PostgreSQL)
-- Παράγεται από τα μοντέλα GORM του καταλόγου backend/models/
-- =====

-- Πίνακας users (backend/models/user.go)
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    password TEXT NOT NULL, -- bcrypt hash
    role VARCHAR(50) NOT NULL CHECK (role IN ('patient',
'admin')),
    is_in_household BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE -- soft delete
(GORM)
);

CREATE INDEX idx_users_username ON users(username);
CREATE INDEX idx_users_role ON users(role);

-- Πίνακας patients (backend/models/patient.go)
CREATE TABLE patients (
    id SERIAL PRIMARY KEY,
    user_id INTEGER UNIQUE NOT NULL
REFERENCES users(id) ON DELETE
CASCADE,
    name VARCHAR(255),
    surname VARCHAR(255),
    address TEXT,
    medical_record TEXT,
    date_of_birth DATE,
    gender VARCHAR(50),
    blood_type VARCHAR(10),
    height DECIMAL(5,2), -- cm
    medical_history TEXT,
    allergies TEXT,
    medications TEXT,
    emergency_contact_name VARCHAR(255),
    emergency_contact_relationship VARCHAR(255),
    emergency_contact_phone_number VARCHAR(50),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_patients_user_id ON patients(user_id);
CREATE INDEX idx_patients_dob ON patients(date_of_birth);

```

```
-- Πίνακας health_metrics (backend/models/HealthMetrics.go)
CREATE TABLE health_metrics (
  id BIGSERIAL PRIMARY KEY,
  patient_id INTEGER NOT NULL REFERENCES patients(id) ON
DELETE CASCADE,
  date TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT
CURRENT_TIMESTAMP,
  weight DECIMAL(5,2), -- kg
  heart_rate INTEGER, -- bpm
  systolic_bp INTEGER, -- mmHg
  diastolic_bp INTEGER, -- mmHg
  oxygen_saturation DECIMAL(5,2), -- %
  blood_pressure VARCHAR(50), -- μορφοποιημένη συμβολοσειρά
  "120/80"
  steps_count INTEGER,
  -- Στάδια ύπνου (ενσωματωμένη δομή SleepStages, ισοπεδώνεται από
  to GORM)
  light DECIMAL(5,2), -- ώρες
  deep DECIMAL(5,2),
  rem DECIMAL(5,2),
  awake_time INTEGER, -- λεπτά
  sleep_duration DECIMAL(5,2), -- συνολικές ώρες
  irregular_rhythm BOOLEAN DEFAULT FALSE,
  fall_detected BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT
CURRENT_TIMESTAMP,
  deleted_at TIMESTAMP WITH TIME ZONE
);

-- Κρίσιμο ευρετήριο για τα ερωτήματα χρονοσειρών
CREATE INDEX idx_health_metrics_patient_date ON
health_metrics(patient_id, date DESC);
CREATE INDEX idx_health_metrics_date ON health_metrics(date
DESC);

-- Πίνακας invitations (backend/models/Invitation.go)
CREATE TABLE invitations (
  id SERIAL PRIMARY KEY,
  admin_id INTEGER NOT NULL REFERENCES users(id) ON DELETE
CASCADE,
  patient_id INTEGER NOT NULL REFERENCES users(id) ON DELETE
CASCADE,
  household_id INTEGER NOT NULL REFERENCES households(id) ON DELETE
CASCADE,
  status VARCHAR(50) NOT NULL
CHECK (status IN ('pending', 'accepted',
'declined')),
  created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  deleted_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_invitations_patient_status ON invitations(patient_id,
status);
CREATE INDEX idx_invitations_admin ON invitations(admin_id);
```

```
-- Πίνακας chat_sessions (backend/models/chatSession.go)
CREATE TABLE chat_sessions (
  id SERIAL PRIMARY KEY,
  user_id INTEGER NOT NULL REFERENCES users(id) ON DELETE
CASCADE,
  session_id UUID UNIQUE NOT NULL DEFAULT gen_random_uuid(),
  session_type VARCHAR(50) DEFAULT 'general', -- 'patient' ή
'admin'
  context_data TEXT, -- JSON serialized
  UserContext
  is_active BOOLEAN DEFAULT TRUE,
  last_used_at TIMESTAMP WITH TIME ZONE NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
  deleted_at TIMESTAMP WITH TIME ZONE
);

CREATE INDEX idx_chat_sessions_user_active ON
chat_sessions(user_id, is_active);
CREATE UNIQUE INDEX idx_chat_sessions_session_id ON
chat_sessions(session_id);
```

A.50 — Ψευδοκώδικας: BuildUserContext (backend/services/ai/contextBuilder.go)

```
ALGORITHM BuildUserContext(userID)
  INPUT : userID - αναγνωριστικό του χρήστη που καλεί τον βοηθό ΑΙ
  OUTPUT: UserContext - δεδομένα υγείας προσαρμοσμένα στον ρόλο του
  χρήστη

1. user <- SELECT * FROM users WHERE id = userID

2. IF user NOT FOUND THEN
  RETURN ERROR "user not found"

3. context <- {
  UserID : user.ID,
  UserRole : user.Role,
  Username : user.Username,
  IsPatient: (user.Role = 'patient')
}

4. IF user.Role = 'patient' THEN
  4.1 patient <- SELECT * FROM patients WHERE user_id = userID

  4.2 latestMetrics <- SELECT * FROM health_metrics
  WHERE patient_id = patient.ID
  ORDER BY date DESC
  LIMIT 1

  4.3 context.PatientData <- {
  Name : patient.Name + ' ' +
patient.Surname,
  Age : CALCULATE_AGE(patient.DateOfBirth),
  Gender : patient.Gender,
  BloodType : patient.BloodType,
  Height : patient.Height,
```

```

        Medications      : patient.Medications,
        Allergies         : patient.Allergies,
        MedicalHistory    : patient.MedicalHistory,
        EmergencyContact  : patient.EmergencyContact,
        LatestMetrics     : latestMetrics -- εφόσον υπάρχουν
    }

5. ELSE IF user.Role = 'admin' THEN
    5.1 households <- SELECT * FROM households WHERE admin_id =
        userID

    5.2 patientsList <- []

    5.3 FOR EACH household IN households DO
        5.3.1 householdPatients <-
            SELECT patients.*
            FROM patients
            JOIN household_patients
            ON patients.id =
                household_patients.patient_id
            WHERE household_patients.household_id =
                household.ID

        5.3.2 FOR EACH patient IN householdPatients DO
            latestMetrics <- SELECT * FROM
                health_metrics
            WHERE patient_id =
                patient.ID
            ORDER BY date DESC
            LIMIT 1

            patientSummary <- {
                Name      : patient.Name + ' ' +
                    patient.Surname,
                Age       :
                    CALCULATE_AGE(patient.DateOfBirth),
                Gender    : patient.Gender,
                BloodType : patient.BloodType,
                Height    : patient.Height,
                Medications : patient.Medications,
                Allergies  : patient.Allergies,
                LatestMetrics: latestMetrics
            }

            APPEND patientSummary TO patientsList

    5.4 context.AdminData <- {
        PatientCount : LENGTH(patientsList),
        Patients      : patientsList,
        HouseholdInfo: 'Managing ' + LENGTH(households) +
            ' household(s) with ' +
            LENGTH(patientsList) + ' patient(s)'
    }

6. RETURN context

```

```

FUNCTION CALCULATE_AGE(dateOfBirth)
  years <- (CURRENT_DATE - dateOfBirth) / 365.25
  RETURN FLOOR(years)

```

A.51 — Ψευδοκώδικας: AggregateHealthMetrics (συγκέντρωση μετρήσεων υγείας)

```

ALGORITHM AggregateHealthMetrics(patientID, startDate, endDate,
  aggregationType)
  INPUT : patientID      - αναγνωριστικό ασθενούς
          startDate      - αρχή του διαστήματος (π.χ. '2024-01-01')
          endDate        - τέλος του διαστήματος (π.χ. '2024-01-31')
          aggregationType - {DAILY, WEEKLY, MONTHLY}
  OUTPUT: πίνακας συγκεντρωτικών περιλήψεων μετρήσεων

1. rawMetrics <- SELECT * FROM health_metrics
   WHERE patient_id = patientID
   AND date >= startDate
   AND date <= endDate
   ORDER BY date ASC

2. IF rawMetrics IS EMPTY THEN
   RETURN []

3. buckets <- MAP (bucketKey -> λίστα μετρήσεων)

4. FOR EACH metric IN rawMetrics DO
   4.1 IF aggregationType = DAILY THEN
       bucketKey <- FORMAT_DATE(metric.Date, 'YYYY-MM-DD')
     ELSE IF aggregationType = WEEKLY THEN
       weekNumber <- GET_ISO_WEEK_NUMBER(metric.Date)
       bucketKey <- FORMAT_DATE(metric.Date, 'YYYY') + '-W'
       + weekNumber
     ELSE IF aggregationType = MONTHLY THEN
       bucketKey <- FORMAT_DATE(metric.Date, 'YYYY-MM')

   4.2 IF bucketKey NOT IN buckets THEN
       buckets[bucketKey] <- []

   4.3 APPEND metric TO buckets[bucketKey]

5. aggregatedResults <- []

6. FOR EACH (bucketKey, metricsInBucket) IN buckets DO
   6.1 avgSystolicBP <- AVERAGE(m.SystolicBP FOR m IN
   metricsInBucket)
       avgDiastolicBP <- AVERAGE(m.DiastolicBP FOR m IN
   metricsInBucket)
       avgHeartRate <- AVERAGE(m.HeartRate FOR m IN
   metricsInBucket)
       avgWeight <- AVERAGE(m.Weight FOR m IN
   metricsInBucket)

       minSystolicBP <- MIN(m.SystolicBP FOR m IN
   metricsInBucket)
       maxSystolicBP <- MAX(m.SystolicBP FOR m IN
   metricsInBucket)

```

```

        totalSteps      <- SUM(m.StepsCount FOR m IN
metricsInBucket)
        measurementCount <- LENGTH(metricsInBucket)

```

```

6.2 summary <- {
    PeriodKey      : bucketKey,
    AverageSystolicBP : ROUND(avgSystolicBP, 1),
    AverageDiastolicBP: ROUND(avgDiastolicBP, 1),
    AverageHeartRate  : ROUND(avgHeartRate, 1),
    AverageWeight     : ROUND(avgWeight, 2),
    MinSystolicBP     : minSystolicBP,
    MaxSystolicBP     : maxSystolicBP,
    TotalSteps       : totalSteps,
    MeasurementCount  : measurementCount
}

```

```

6.3 APPEND summary TO aggregatedResults

```

```

7. SORT aggregatedResults BY PeriodKey ASC

```

```

8. RETURN aggregatedResults

```