ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	Βελτιστοποίηση Διαχείρισης Εργασιών με Χρήση Αλγορίθμων Χρονισμού Efficient Task Scheduling and Management via Timing Algorithms
Ονοματεπώνυμο Φοιτητή	**ΔΗΜΗΤΡΙΟΣ ΤΖΕΛΕΠΗΣ**
Πατρώνυμο	**ΒΑΣΙΛΕΙΟΣ**
Αριθμός Μητρώου	**mpsp2350**
Επιβλέπων	**Μαρία Βίρβου, Καθηγήτρια**

Ημερομηνία Παράδοσης **Απρίλιος2025**

Τριμελής Εξεταστική Επιτροπή

Μαρία Βίρβου	Ευάγγελος Σακκόπουλος	Κωνσταντίνα Χρυσαφιάδη
Καθηγήτρια	Αναπληρωτής Καθηγητής	Επίκουρη Καθηγήτρια

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΔΙΑΧΕΙΡΙΣΗΣ ΕΡΓΑΣΙΩΝ ΜΕ ΧΡΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΧΡΟΝΙΣΜΟΥ

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας ολοκληρωμένης desktop εφαρμογής διαχείρισης εργασιών, η οποία αξιοποιεί προηγμένους αλγορίθμους χρονοπρογραμματισμού για την οργάνωση και την προτεραιοποίηση καθημερινών δραστηριοτήτων. Ειδικότερα, υλοποιούνται τρεις αλγόριθμοι χρονολόγησης: Shortest Job Next, Least Laxity First και Round-Robin, οι οποίοι συγκρίνονται ως προς την απόδοση και την καταλληλότητά τους σε διαφορετικά σενάρια χρήσης. Η εφαρμογή σχεδιάστηκε χρησιμοποιώντας τη γλώσσα Java και το framework JavaFX, με τη χρήση της μεθοδολογίας ανάπτυξης Rational Unified Process και τεκμηρίωση μέσω UML διαγραμμάτων. Βασικός στόχος της εργασίας ήταν η δημιουργία ενός διαισθητικού και φιλικού προς τον χρήστη περιβάλλοντος, το οποίο επιτρέπει τη δημιουργία, επεξεργασία και διαγραφή εργασιών, καθώς και την αυτόματη ταξινόμησή τους μέσω των ενσωματωμένων αλγορίθμων. Επιπλέον, η εργασία εξετάζει τη δυνατότητα ενσωμάτωσης μελλοντικών λειτουργιών, όπως η ανάλυση δεδομένων μέσω τεχνητής νοημοσύνης, η υποστήριξη πολλαπλών πλατφορμών και η διασύνδεση με εξωτερικές εφαρμογές. Τα αποτελέσματα της έρευνας και της εφαρμογής καταδεικνύουν τη χρησιμότητα των αλγορίθμων στην αποτελεσματική διαχείριση χρόνου και προτείνουν κατευθύνσεις για περαιτέρω ανάπτυξη.

Λέξεις-κλειδιά: Διαχείριση εργασιών, Αλγόριθμοι χρονοπρογραμματισμού, Shortest Job Next, Least Laxity First, Round-Robin, Java, JavaFX, UML, Rational Unified Process

EFFICIENT TASK SCHEDULING AND MANAGEMENT VIA TIMING ALGORITHMS

ABSTRACT

This thesis focuses on the development of a comprehensive desktop task management application utilizing advanced scheduling algorithms to organize and prioritize daily activities. Specifically, three scheduling algorithms are implemented: Shortest Job Next, Least Laxity First, and Round-Robin, which are compared in terms of performance and suitability across different use cases. The application was developed using Java and the JavaFX framework, following the Rational Unified Process development methodology and documented with UML diagrams. The primary objective of this study was to create an intuitive and user-friendly environment that allows users to create, edit, and delete tasks while automatically organizing them using the embedded algorithms. Furthermore, the study explores potential future functionalities, such as data analysis powered by artificial intelligence, multi-platform support, and integration with external applications. The results demonstrate the utility of the algorithms in effective time management and provide directions for further development.

Keywords: Task management, Scheduling algorithms, Shortest Job Next, Least Laxity First, Round-Robin, Java, JavaFX, UML, Rational Unified Process

ΠΕΡΙΕΧΟΜΕΝΑ

ΤΖΕΛΕΠΗΣ ΔΗΜΗΤΡΙΟΣ

ΕΠΙΒΛΕΠΟΥΣΑ ΚΑΘΗΓΗΤΡΙΑ:

Μ.ΒΙΡΒΟΥ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΣΥΝΤΟΜΕΥΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

SJN	Shortest Job Next
LLF	Least Laxity First
RR	Round-Robin
UML	Unified Modeling Language
RUP	Rational Unified Process
IWC	Improved Weighted Circle
HS	Hybrid Scheduling Algorithm
GA	Genetic Algorithm
FPA	Flower Pollination based Algorithm
SJF	Shortest Job First
MM-BF	Min-Min Best Fit
QMW	Queue-length-based MaxWeight
MLLF	Modified Least-Laxity-First
ELLF	Enhanced Least-Laxity-First
LLLP	Less Laxity and Longer remaining Processing time
SAD	Single Administrative Document
HHG	Hybrid Heuristic and Genetic-based Task Scheduling Algorithm
HTDG	Hybrid Task Duplication and Genetic-based Task Scheduling Algorithm
HRC	Human-Robot Collaboration
MVC	Model-View-Controller

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	Ελληνικός Όρος
Human-Computer Interaction	Αλληλεπίδραση Ανθρώπου-Υπολογιστή
Improved Weighted Circle	Αλγόριθμος Βελτιστοποίησης Φάλαινας
Hybrid Scheduling Algorithm	Υβριδικός Αλγόριθμος Χρονοπρογραμματισμού
Genetic Algorithm	Γενετικός Αλγόριθμος
Flower Pollination based Algorithm	Αλγόριθμος Επικονίασης Λουλουδιών
Laxity	Χαλαρότητα
Time Quantum	Χρονικό Κβάντο
Incremental Build	Σταδιακή Κατασκευή
Human-Robot Collaboration	Συνεργασία Ανθρώπου-Ρομπότ
Trace-driven Simulation	Προσομοίωση βασισμένη σε ίχνη

ΕΙΣΑΓΩΓΗ

Η διαχείριση εργασιών και ο αποτελεσματικός χρονοπρογραμματισμός αποτελούν κρίσιμους παράγοντες επιτυχίας σε κάθε επαγγελματικό και προσωπικό πλαίσιο στη σύγχρονη εποχή. Η συνεχής αύξηση της πολυπλοκότητας των εργασιακών απαιτήσεων, σε συνδυασμό με την ανάγκη για βέλτιστη αξιοποίηση του διαθέσιμου χρόνου, έχει οδηγήσει στην ανάπτυξη εξειδικευμένων συστημάτων διαχείρισης εργασιών. Τα συστήματα αυτά, αξιοποιώντας προηγμένους αλγόριθμους χρονολόγησης και σύγχρονες τεχνολογίες πληροφορικής, επιχειρούν να προσφέρουν λύσεις για την αποδοτική οργάνωση και προτεραιοποίηση των καθημερινών δραστηριοτήτων. Ο τομέας της διαχείρισης εργασιών έχει εξελιχθεί σημαντικά τις τελευταίες δεκαετίες, ενσωματώνοντας θεωρίες και πρακτικές από διάφορους επιστημονικούς κλάδους, όπως η επιχειρησιακή έρευνα, η τεχνητή νοημοσύνη και η επιστήμη των υπολογιστών. Ιδιαίτερο ενδιαφέρον παρουσιάζει η εφαρμογή αλγορίθμων χρονολόγησης, οι οποίοι, αν και αρχικά αναπτύχθηκαν για τη διαχείριση διεργασιών σε λειτουργικά συστήματα, έχουν αποδειχθεί εξαιρετικά αποτελεσματικοί στην επίλυση προβλημάτων χρονοπρογραμματισμού σε ένα ευρύ φάσμα εφαρμογών.

Σκοπός και Στόχοι της Εργασίας

Κύριος σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης desktop εφαρμογής διαχείρισης εργασιών, η οποία θα αξιοποιεί προηγμένους αλγόριθμους χρονολόγησης για την αποτελεσματική οργάνωση και προτεραιοποίηση των καθημερινών εργασιών των χρηστών. Η εφαρμογή στοχεύει στη βελτιστοποίηση της διαχείρισης χρόνου μέσω της εφαρμογής τριών διαφορετικών αλγορίθμων χρονολόγησης: Shortest Job Next (SJN), Least Laxity First (LLF) και Round-Robin (RR), προσφέροντας στους χρήστες την ευελιξία να επιλέξουν τον καταλληλότερο αλγόριθμο για τις ανάγκες τους.

Ένας βασικός στόχος της εργασίας είναι η δημιουργία ενός διαισθητικού και φιλικού προς το χρήστη περιβάλλοντος με τη χρήση της τεχνολογίας JavaFX. Το περιβάλλον αυτό θα επιτρέπει την εύκολη προσθήκη, επεξεργασία και διαγραφή εργασιών, καθώς και την παρακολούθηση τους μέσω ενός ολοκληρωμένου ημερολογίου. Ιδιαίτερη έμφαση δίνεται στην υποστήριξη του οκταώρου εργασίας, επιτρέποντας στους χρήστες να οργανώνουν αποτελεσματικά τις καθημερινές τους δραστηριότητες εντός του προκαθορισμένου χρονικού πλαισίου.

Επιπλέον, η εργασία στοχεύει στην πρακτική εφαρμογή και σύγκριση των τριών αλγορίθμων χρονολόγησης, αναλύοντας τη συμπεριφορά και την αποτελεσματικότητά τους σε πραγματικές συνθήκες χρήσης. Μέσω της υλοποίησης αυτής, επιδιώκεται η σε βάθος κατανόηση των πλεονεκτημάτων και των περιορισμών κάθε αλγορίθμου, καθώς και η ανάδειξη των καταλληλότερων σεναρίων χρήσης για τον καθένα.

Τέλος, η εργασία αποσκοπεί στην ανάπτυξη μιας εφαρμογής που θα μπορεί να αποτελέσει πρακτικό εργαλείο για επαγγελματίες και οργανισμούς, προσφέροντας μια ολοκληρωμένη λύση διαχείρισης εργασιών. Παράλληλα, μέσω της λεπτομερούς τεκμηρίωσης και των Unified Modeling Language (UML) διαγραμμάτων, στοχεύει στη δημιουργία ενός ολοκληρωμένου παραδείγματος εφαρμογής σύγχρονων μεθοδολογιών ανάπτυξης λογισμικού, το οποίο μπορεί να αξιοποιηθεί ως αναφορά για μελλοντικά έργα.

Περιγραφή του Προβλήματος

Στο σύγχρονο επαγγελματικό περιβάλλον, η διαχείριση του καθημερινού φόρτου εργασίας αποτελεί μια σύνθετη πρόκληση που απαιτεί αποτελεσματικά εργαλεία και μεθοδολογίες. Η ανάγκη για βέλτιστη αξιοποίηση του εργασιακού οκταώρου, σε συνδυασμό με την πολυπλοκότητα των εργασιών και τις διαφορετικές προτεραιότητες, δημιουργεί ένα πολυδιάστατο πρόβλημα χρονοπρογραμματισμού. Παρά την ύπαρξη διαφόρων εφαρμογών διαχείρισης εργασιών, παρατηρείται έλλειψη λύσεων που να συνδυάζουν την ευελιξία στην επιλογή αλγορίθμων χρονολόγησης με ένα εύχρηστο περιβάλλον διεπαφής.

Ένα βασικό ζήτημα που αντιμετωπίζουν οι χρήστες είναι η δυσκολία στην αποτελεσματική κατανομή των εργασιών τους μέσα στο διαθέσιμο χρόνο. Οι υπάρχουσες εφαρμογές συχνά προσφέρουν στατικές λύσεις χρονοπρογραμματισμού, χωρίς να λαμβάνουν υπόψη τις διαφορετικές ανάγκες και προτιμήσεις των χρηστών ως προς τον τρόπο οργάνωσης των εργασιών τους. Η έλλειψη ευελιξίας στην επιλογή της μεθόδου χρονολόγησης περιορίζει τη δυνατότητα προσαρμογής του συστήματος στις εκάστοτε απαιτήσεις και συνθήκες εργασίας.

Επιπλέον, η πλειονότητα των διαθέσιμων εφαρμογών δεν αξιοποιεί προηγμένους αλγόριθμους χρονολόγησης για τη βελτιστοποίηση της κατανομής των εργασιών. Αυτό έχει ως αποτέλεσμα τη μη βέλτιστη διαχείριση του χρόνου και των πόρων, ιδιαίτερα σε περιπτώσεις όπου υπάρχουν πολλαπλές εργασίες με διαφορετικές προτεραιότητες και προθεσμίες. Η έλλειψη επιστημονικά τεκμηριωμένων μεθόδων χρονολόγησης οδηγεί συχνά σε αναποτελεσματική διαχείριση του εργασιακού φόρτου και σε μειωμένη παραγωγικότητα.

Σημασία της Εργασίας

Η ανάπτυξη εξειδικευμένων εφαρμογών διαχείρισης εργασιών με χρήση προηγμένων αλγορίθμων χρονολόγησης αποτελεί ένα ιδιαίτερα σημαντικό πεδίο έρευνας και ανάπτυξης στην επιστήμη των υπολογιστών. Η συγκεκριμένη πτυχιακή εργασία συμβάλλει ουσιαστικά στην κατανόηση και πρακτική εφαρμογή αλγορίθμων που παραδοσιακά χρησιμοποιούνται στα λειτουργικά συστήματα, επεκτείνοντας τη χρήση τους στο πεδίο της διαχείρισης ανθρώπινων

εργασιών. Η μελέτη και υλοποίηση τέτοιων συστημάτων είναι καίριας σημασίας για την εξέλιξη των εργαλείων παραγωγικότητας και την βελτιστοποίηση της διαχείρισης χρόνου στο σύγχρονο επαγγελματικό περιβάλλον.

Επιπρόσθετα, η εργασία αυτή αποτελεί σημαντική συνεισφορά στον τομέα της αλληλεπίδρασης ανθρώπου-υπολογιστή (Human-Computer Interaction), καθώς διερευνά τρόπους αποτελεσματικής παρουσίασης και διαχείρισης πολύπλοκων αλγοριθμικών αποφάσεων μέσω ενός διαισθητικού περιβάλλοντος χρήστη. Η μελέτη της σχέσης μεταξύ των τεχνικών χαρακτηριστικών των αλγορίθμων χρονολόγησης και της εμπειρίας χρήστη είναι ζωτικής σημασίας για την ανάπτυξη αποτελεσματικών εργαλείων διαχείρισης εργασιών που θα γίνουν ευρέως αποδεκτά από τους τελικούς χρήστες.

Η συστηματική μελέτη και σύγκριση διαφορετικών αλγορίθμων χρονολόγησης στο πλαίσιο της διαχείρισης ανθρώπινων εργασιών παρέχει πολύτιμα συμπεράσματα για την καταλληλότητα και αποτελεσματικότητα κάθε προσέγγισης. Μέσω της πρακτικής εφαρμογής των αλγορίθμων SJN, LLF και Round-Robin σε πραγματικές συνθήκες χρήσης, η εργασία προσφέρει εμπειρικά δεδομένα που μπορούν να αξιοποιηθούν τόσο στην ακαδημαϊκή έρευνα όσο και στην ανάπτυξη μελλοντικών εφαρμογών διαχείρισης εργασιών.

Από τεχνολογική σκοπιά, η εργασία αυτή αναδεικνύει τη σημασία της χρήσης σύγχρονων εργαλείων ανάπτυξης όπως το JavaFX και το Gradle στη δημιουργία επαγγελματικών εφαρμογών desktop. Η λεπτομερής τεκμηρίωση της διαδικασίας ανάπτυξης μέσω UML διαγραμμάτων και η υιοθέτηση της μεθοδολογίας Rational unified process (RUP) προσφέρει ένα ολοκληρωμένο παράδειγμα εφαρμογής σύγχρονων πρακτικών σχεδίασης και υλοποίησης λογισμικού. Αυτή η προσέγγιση αποτελεί πολύτιμο εκπαιδευτικό υλικό για μελλοντικούς προγραμματιστές και μηχανικούς λογισμικού που επιθυμούν να εμβαθύνουν στην ανάπτυξη παρόμοιων συστημάτων.

Τέλος, η σημασία της εργασίας ενισχύεται από την πρακτική της συνεισφορά στην επίλυση ενός διαχρονικού προβλήματος στον επαγγελματικό χώρο. Η ανάπτυξη ενός συστήματος που συνδυάζει την επιστημονική ακρίβεια των αλγορίθμων χρονολόγησης με την πρακτική χρησιμότητα ενός εργαλείου διαχείρισης εργασιών καλύπτει ένα σημαντικό κενό στην αγορά εφαρμογών παραγωγικότητας. Η δυνατότητα προσαρμογής του συστήματος στις ιδιαίτερες ανάγκες κάθε χρήστη μέσω της επιλογής διαφορετικών αλγορίθμων χρονολόγησης αποτελεί καινοτόμο προσέγγιση που μπορεί να επηρεάσει σημαντικά τον τρόπο ανάπτυξης μελλοντικών εφαρμογών στον τομέα αυτό.

Σύγχρονες Εργασίες στο Πεδίο

Η έρευνα στον τομέα των εφαρμογών διαχείρισης εργασιών με έμφαση στους αλγόριθμους χρονολόγησης έχει σημειώσει σημαντική πρόοδο τα τελευταία χρόνια.

Χαρακτηριστικό παράδειγμα αποτελεί η μελέτη των Wan και Bard (2007), η οποία εστίασε στην επίλυση του προβλήματος της εβδομαδιαίας προσαρμογής του προγραμματισμού προσωπικού με περιορισμούς ομάδων εργασίας. Η προτεινόμενη προσέγγιση ενσωμάτωσε τους περιορισμούς αυτούς με τον προγραμματισμό βαρδιών και την ανάθεση εργασιών, χρησιμοποιώντας ευρετικές μεθόδους αποσύνθεσης για την επίλυση του προβλήματος. Τα αποτελέσματα των δοκιμών με δεδομένα από την Ταχυδρομική Υπηρεσία των ΗΠΑ έδειξαν ότι εφικτές λύσεις μπορούσαν να επιτευχθούν σε λιγότερο από μία ώρα.

Στο πλαίσιο της βελτιστοποίησης των αλγορίθμων χρονολόγησης, οι Paprocka και Skolud (2017) ανέπτυξαν έναν υβριδικό πολλών αντικειμένων αλγόριθμο ανοσοποίησης (H-MOIA) για προβλεπτικό και αντιδραστικό προγραμματισμό. Η μελέτη τους συνέκρινε τον H-MOIA με διάφορους άλλους αλγόριθμους, συμπεριλαμβανομένων αλγορίθμων βασισμένων σε κανόνες προτεραιότητας, όπως ο Least Flexible Job First και ο Longest Processing Time. Τα αποτελέσματα έδειξαν ότι ο H-MOIA, σε συνδυασμό με ευρετικές μεθόδους ελάχιστης επίπτωσης, ξεπέρασε τις άλλες μεθόδους όσον αφορά τη διατήρηση της σταθερότητας και της ευρωστίας του προγράμματος, βελτιώνοντας έτσι τη συνολική παραγωγικότητα.

Σημαντική πρόοδος έχει επίσης σημειωθεί στον τομέα των συστημάτων υπολογιστικού νέφους, όπου οι Jia et al. (2021) πρότειναν έναν βελτιωμένο αλγόριθμο βελτιστοποίησης φάλαινας (Improved Weighted Circle – IWC) για την αντιμετώπιση ζητημάτων όπως οι μεγάλοι χρόνοι προγραμματισμού και τα υψηλά κόστη. Σε σύγκριση με παραδοσιακούς αλγόριθμους όπως ο αλγόριθμος αποικίας μυρμηγκιών και ο αλγόριθμος σμήνους σωματιδίων, ο IWC παρουσίασε σημαντική βελτίωση στο χρόνο προγραμματισμού εργασιών και στην εξισορρόπηση φορτίου των εικονικών μηχανών, προσφέροντας έτσι πολύτιμες ιδέες για τη βελτιστοποίηση συστημάτων διαχείρισης εργασιών.

Τέλος, ιδιαίτερο ενδιαφέρον παρουσιάζει η έρευνα των Walia et al. (2021), οι οποίοι ανέπτυξαν έναν υβριδικό αλγόριθμο χρονοπρογραμματισμού (Hybrid Scheduling Algorithm – HS) που συνδυάζει τον Γενετικό Αλγόριθμο (Genetic Algorithm – GA) με τον Αλγόριθμο Επικονίασης Λουλουδιών (Flower Pollination based Algorithm – FPA). Ο προτεινόμενος αλγόριθμος επέδειξε εξαιρετική απόδοση σε διάφορες παραμέτρους, συμπεριλαμβανομένου του χρόνου ολοκλήρωσης και της χρήσης πόρων, με βελτίωση 36% στη χρήση πόρων σε ομογενή περιβάλλοντα και μείωση του χρόνου ολοκλήρωσης κατά 33.7% σε ετερογενή περιβάλλοντα, σε σύγκριση με τον GA. Τα αποτελέσματα αυτά υποδεικνύουν τη σημασία της υβριδικής προσέγγισης στην ανάπτυξη αποτελεσματικών συστημάτων διαχείρισης εργασιών.

Δομή της Εργασίας

Η παρούσα εργασία οργανώνεται σε πέντε κεφάλαια, τα οποία καλύπτουν τη θεωρητική βάση, την ανάλυση, τον σχεδιασμό, την υλοποίηση και την αξιολόγηση του συστήματος διαχείρισης εργασιών. Κάθε κεφάλαιο αποτελεί ένα ξεχωριστό αλλά αλληλένδετο μέρος της

συνολικής μελέτης, με σκοπό να διασφαλιστεί η σαφής παρουσίαση του έργου και των αποτελεσμάτων του.

1. Κεφάλαιο 1: Θεωρητικό Πλαίσιο και Βιβλιογραφική Ανασκόπηση

Το πρώτο κεφάλαιο εισάγει τις βασικές αρχές της διαχείρισης εργασιών και των αλγορίθμων χρονοπρογραμματισμού. Παρουσιάζονται οι τρεις βασικοί αλγόριθμοι που υλοποιήθηκαν στην εφαρμογή: Shortest Job Next (SJN), Least Laxity First (LLF) και Round-Robin (RR). Εξετάζονται οι θεωρητικές βάσεις και οι σύγχρονες εφαρμογές τους, ενώ αναλύονται οι περιορισμοί και οι δυνατότητές τους. Παράλληλα, γίνεται ανασκόπηση σχετικών εργασιών και εφαρμογών για την ανάδειξη της καινοτομίας της παρούσας προσέγγισης.

2. Κεφάλαιο 2: Ανάλυση και Σχεδιασμός

Στο δεύτερο κεφάλαιο παρουσιάζονται οι απαιτήσεις του συστήματος και οι βασικές σχεδιαστικές αποφάσεις. Αναλύεται η αρχιτεκτονική της εφαρμογής, η οποία βασίζεται στο μοντέλο Model-View-Controller (MVC), και τεκμηριώνεται με UML διαγράμματα. Περιγράφεται η λειτουργικότητα της εφαρμογής, όπως η δημιουργία, η επεξεργασία και η ταξινόμηση εργασιών, καθώς και οι τεχνολογίες που χρησιμοποιήθηκαν, όπως η Java και το JavaFX.

3. Κεφάλαιο 3: Υλοποίηση

Το τρίτο κεφάλαιο περιγράφει τη διαδικασία υλοποίησης της εφαρμογής. Εστιάζει στην ανάπτυξη των βασικών λειτουργιών, συμπεριλαμβανομένης της ενσωμάτωσης των αλγορίθμων χρονοπρογραμματισμού και της διεπαφής χρήστη. Επιπλέον, αναλύονται οι τεχνικές προκλήσεις που προέκυψαν κατά την ανάπτυξη και οι λύσεις που δόθηκαν για την αντιμετώπισή τους. Παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν, όπως το Gradle για την αυτοματοποίηση των διαδικασιών ανάπτυξης.

4. Κεφάλαιο 4: Έλεγχος και Αποτελέσματα

Στο τέταρτο κεφάλαιο γίνεται παρουσίαση της μεθοδολογίας ελέγχου που εφαρμόστηκε για την αξιολόγηση της εφαρμογής. Περιγράφονται τα σενάρια ελέγχου και τα αποτελέσματα των δοκιμών, τα οποία αξιολογούν την αποτελεσματικότητα των αλγορίθμων και τη χρηστικότητα του συστήματος. Επιπλέον, παρέχεται μια ανάλυση συγκριτικών δεδομένων για τους τρεις αλγορίθμους που εφαρμόστηκαν.

5. Κεφάλαιο 5: Συμπεράσματα και Μελλοντικές Επεκτάσεις

Το πέμπτο και τελευταίο κεφάλαιο συνοψίζει τα κύρια ευρήματα της εργασίας και περιγράφει τις δυνατότητες βελτίωσης και επέκτασης του συστήματος. Ιδιαίτερη έμφαση δίνεται στη μελλοντική ενσωμάτωση περισσότερων αλγορίθμων, την υποστήριξη

πολλαπλών πλατφορμών, τη χρήση τεχνητής νοημοσύνης και τη διασύνδεση με εξωτερικές εφαρμογές.

Η συνολική δομή της εργασίας αποσκοπεί στη συστηματική παρουσίαση των βημάτων που ακολουθήθηκαν για την ανάπτυξη της εφαρμογής και στην τεκμηρίωση της αποτελεσματικότητας της προτεινόμενης προσέγγισης.

ΚΕΦΑΛΑΙΟ 1. ΘΕΩΡΗΤΙΚΟ ΠΛΑΙΣΙΟ ΚΑΙ ΒΙΒΛΙΟΓΡΑΦΙΚΗ ΑΝΑΣΚΟΠΗΣΗ

Το παρόν κεφάλαιο εμβαθύνει στο θεωρητικό υπόβαθρο και τις τεχνολογίες που αξιοποιούνται στην ανάπτυξη του συστήματος διαχείρισης εργασιών. Αρχικά, αναλύονται οι βασικές αρχές και μεθοδολογίες της διαχείρισης εργασιών και του χρονοπρογραμματισμού, με έμφαση στις σύγχρονες προσεγγίσεις και τεχνικές. Στη συνέχεια, παρουσιάζεται λεπτομερής ανάλυση των τριών αλγορίθμων χρονολόγησης που επιλέχθηκαν για την υλοποίηση του συστήματος, εξετάζοντας τα ιδιαίτερα χαρακτηριστικά, τα πλεονεκτήματα και τους περιορισμούς τους. Ακολουθεί η παρουσίαση των τεχνολογιών ανάπτυξης που χρησιμοποιήθηκαν, με έμφαση στο περιβάλλον Java και τα εργαλεία JavaFX και Gradle. Το κεφάλαιο ολοκληρώνεται με μια επισκόπηση σχετικών εργασιών και εφαρμογών, αναδεικνύοντας τις τρέχουσες τάσεις και καινοτομίες στον τομέα της διαχείρισης εργασιών.

1.1 Διαχείριση Εργασιών και Χρονοπρογραμματισμός

Η διαχείριση εργασιών αποτελεί θεμελιώδη πυλώνα της σύγχρονης οργάνωσης εργασίας, καθώς επιτρέπει τη συστηματική παρακολούθηση, ιεράρχηση και εκτέλεση δραστηριοτήτων με στόχο τη μεγιστοποίηση της παραγωγικότητας. Στον πυρήνα της αποτελεσματικής διαχείρισης εργασιών βρίσκεται η βελτιστοποίηση της χρήσης των διαθέσιμων πόρων και η ελαχιστοποίηση του χρόνου ολοκλήρωσης των εργασιών. Σύμφωνα με τους Malakooti (2013) και Shyalika et al. (2020), η αποτελεσματική διαχείριση εργασιών οδηγεί σε σημαντική βελτίωση της συνολικής απόδοσης του συστήματος, μειώνοντας παράλληλα το λειτουργικό κόστος και τη σπατάλη πόρων.

Ο χρονοπρογραμματισμός, ως βασικό συστατικό της διαχείρισης εργασιών, διαδραματίζει καθοριστικό ρόλο στην επίτευξη των επιχειρησιακών στόχων στο σύγχρονο εργασιακό περιβάλλον. Η σημασία του έγκειται στην ικανότητά του να ελαχιστοποιεί τον συνολικό χρόνο ολοκλήρωσης των εργασιών, να διασφαλίζει την τήρηση των προθεσμιών και να μειώνει τον χρόνο αναμονής των εργασιών στο σύστημα (Malakooti, 2013). Σε δυναμικά περιβάλλοντα, όπως το cloud computing και οι υπηρεσίες πεδίου, η εφαρμογή προηγμένων τεχνικών χρονοπρογραμματισμού καθίσταται απαραίτητη για την αποτελεσματική διαχείριση της πολυπλοκότητας και της μεταβλητότητας των εργασιών (Sasmita Kumari Nayak, 2023; Shyalika et al., 2020).

Η εξέλιξη των συστημάτων διαχείρισης εργασιών έχει οδηγήσει στην ανάπτυξη προηγμένων τεχνικών χρονοπρογραμματισμού που αξιοποιούν σύγχρονες τεχνολογίες, όπως η ενισχυτική μάθηση και οι αλγόριθμοι εμπνευσμένοι από τη φύση. Οι τεχνικές αυτές επιτρέπουν τη δυναμική επίλυση προβλημάτων βελτιστοποίησης εργασιών και πόρων, βελτιώνοντας τη λήψη αποφάσεων και την αποδοτικότητα του συστήματος σε πραγματικό χρόνο (Sasmita Kumari

Nayak, 2023; Shyalika et al., 2020). Επιπλέον, η χρήση τεχνικών πολυκριτηριακής βελτιστοποίησης, όπως οι γενετικοί αλγόριθμοι και οι υβριδικές ευρετικές μέθοδοι, επιτρέπει τη βελτιστοποίηση πολλαπλών αντικρουόμενων στόχων στον χρονοπρογραμματισμό (Hosseinzadeh et al., 2020; Sulaiman et al., 2021).

Η υλοποίηση συστημάτων διαχείρισης εργασιών αντιμετωπίζει σημαντικές προκλήσεις που επηρεάζουν καθοριστικά την επιλογή των αλγορίθμων χρονοπρογραμματισμού. Μία από τις βασικότερες προκλήσεις είναι η ετερογένεια των συστημάτων, καθώς τα σύγχρονα υπολογιστικά περιβάλλοντα αποτελούνται από πόρους με διαφορετικές υπολογιστικές και επικοινωνιακές δυνατότητες. Αυτή η ετερογένεια περιπλέκει τη διαδικασία χρονοπρογραμματισμού, καθώς κάθε εργασία μπορεί να έχει διαφορετικούς χρόνους εκτέλεσης σε διαφορετικούς επεξεργαστές (Kaur et al., 2021; Khan et al., 2023).

Επιπρόσθετα, ο ανταγωνισμός για πόρους επικοινωνίας και η εμπλοκή των επεξεργαστών στην επικοινωνία μπορούν να οδηγήσουν σε ανεπάρκειες στον χρονοπρογραμματισμό των εργασιών. Η ανάπτυξη ακριβών μοντέλων που λαμβάνουν υπόψη αυτούς τους παράγοντες είναι απαραίτητη για τη βελτίωση της ακρίβειας και της αποδοτικότητας του χρονοπρογραμματισμού (Sinnen et al., 2006). Παράλληλα, πολλές εφαρμογές έχουν αυστηρές απαιτήσεις καθυστέρησης και προθεσμιών, γεγονός που προσθέτει επιπλέον πολυπλοκότητα στη διαδικασία χρονοπρογραμματισμού (Khan et al., 2023).

Στο πλαίσιο της διαχείρισης του οκταώρου εργασίας, η ανάπτυξη αποτελεσματικών στρατηγικών χρονοπρογραμματισμού αποκτά ιδιαίτερη σημασία. Σύμφωνα με έρευνες, η χρήση ευέλικτων μοντέλων εργασίας που επιτρέπουν στους εργαζόμενους να επιλέγουν τις ώρες έναρξης, τις ημέρες αδείας και να δημιουργούν μη τυποποιημένες εβδομάδες εργασίας μπορεί να βελτιώσει σημαντικά την ικανοποίηση από την εργασία και την αποδοτικότητα (Bailey & Field, 1985). Η διαχείριση αυτών των ευέλικτων προγραμμάτων απαιτεί εξελιγμένα συστήματα χρονοπρογραμματισμού που μπορούν να προσαρμόζονται στις μεταβαλλόμενες απαιτήσεις και προτιμήσεις των εργαζομένων.

Η προτεραιοποίηση των εργασιών αποτελεί καθοριστικό παράγοντα στη διαδικασία χρονοπρογραμματισμού, επηρεάζοντας σημαντικά τη συνολική απόδοση του συστήματος. Σε περιβάλλοντα ετερογενών συστημάτων, η προτεραιοποίηση μπορεί να οδηγήσει σε αναποτελεσματική χρήση πόρων λόγω της εκτόπισης εργασιών, ωστόσο είναι απαραίτητη για την επίτευξη διαφορετικών στόχων απόδοσης (Çavdar et al., 2015). Η χρήση δυναμικών αλγορίθμων χρονοπρογραμματισμού που ενσωματώνουν μηχανισμούς γήρανσης μπορεί να αποτρέψει την ασιτία των εργασιών, αυξάνοντας την προτεραιότητα των εργασιών που περιμένουν για μεγάλο χρονικό διάστημα και εξισορροπώντας έτσι την εκτέλεση εργασιών υψηλής και χαμηλής προτεραιότητας (Becker & Schüle, 2020).

Η σύγχρονη τάση στη διαχείριση εργασιών κατευθύνεται προς την ανάπτυξη εξελιγμένων συστημάτων που συνδυάζουν πολλαπλές τεχνικές και προσεγγίσεις. Για παράδειγμα, η χρήση ασαφούς λογικής στην αξιολόγηση της σχετικής σημαντικότητας των εργασιών μπορεί να μιμηθεί

αποτελεσματικά την ανθρώπινη διαδικασία λήψης αποφάσεων και να βελτιώσει την κατανομή των πόρων (Miranda et al., 2006). Επιπλέον, η ενσωμάτωση δυναμικού κβάντου χρόνου και προτεραιότητας στον αλγόριθμο Round Robin μπορεί να βελτιώσει σημαντικά τον χρόνο εκτέλεσης των εργασιών και την αξιοποίηση των πόρων, προσαρμόζοντας τα χρονικά διαστήματα με βάση την προτεραιότητα και τον χρόνο εκτέλεσης κάθε εργασίας (Iqbal et al., 2023).

Η αυξανόμενη πολυπλοκότητα στα περιβάλλοντα διαχείρισης εργασιών απαιτεί ολοένα και πιο προσαρμοστικές και ευφυείς τεχνικές. Σύμφωνα με τους Walia et al. (2021), η χρήση υβριδικών αλγορίθμων προγραμματισμού που συνδυάζουν ενεργειακά αποδοτικές στρατηγικές με εξελικτικούς αλγόριθμους μπορεί να μειώσει τη συνολική κατανάλωση ενέργειας σε περιβάλλοντα cloud computing. Αυτό έχει ιδιαίτερη σημασία για σύγχρονα επιχειρησιακά περιβάλλοντα, όπου η αποδοτική χρήση των πόρων αποτελεί κρίσιμο παράγοντα.

Επιπλέον, οι Wan & Bard (2007) μελέτησαν μοντέλα εβδομαδιαίου προγραμματισμού προσωπικού με περιορισμούς ομάδων εργασίας, προτείνοντας τεχνικές που μπορούν να εφαρμοστούν και στη διαχείριση εργασιών σε ευρύτερα συστήματα προγραμματισμού. Αυτή η προσέγγιση μπορεί να ενσωματωθεί σε πλαίσια κατανομής εργασιών για τη βελτιστοποίηση της χρήσης ανθρώπινου δυναμικού και υπολογιστικών πόρων.

Η εξέλιξη των ευφυών αλγορίθμων διαχείρισης εργασιών επικεντρώνεται στην προσαρμογή στις μεταβαλλόμενες απαιτήσεις των χρηστών και των συστημάτων. Οι Triantafyllou & Tsihrintzis (2018) παρουσιάζουν έναν μηχανισμό ομαδικής αναγνώρισης συναισθημάτων που μπορεί να εφαρμοστεί στην προσαρμογή των στρατηγικών διαχείρισης εργασιών, επιτρέποντας τη δυναμική προσαρμογή του περιβάλλοντος εργασίας ανάλογα με τις ανάγκες των χρηστών.

Επιπλέον, οι Theocharis & Tsihrintzis (2016) εξετάζουν τη διαχείριση της γνώσης στον δημόσιο τομέα, αναδεικνύοντας τη σημασία της χρήσης έξυπνων συστημάτων για τη βελτιστοποίηση της διαχείρισης εργασιών και την αποτελεσματική εκμετάλλευση των διαθέσιμων πόρων.

Η σύγχρονη έρευνα στον χρονοπρογραμματισμό εργασιών επικεντρώνεται στη βελτίωση της απόδοσης μέσω ευφυών μεθόδων διαχείρισης φορτίου. Οι Sotiropoulos et al. (2007) εξετάζουν τη χρήση τεχνητών ανοσολογικών συστημάτων για την ομαδοποίηση δεδομένων σε προσαρμοστικά περιβάλλοντα, γεγονός που μπορεί να βελτιώσει τις στρατηγικές κατανομής εργασιών.

Επιπλέον, η ενσωμάτωση μεθόδων τεχνητής νοημοσύνης στην προτεραιοποίηση των εργασιών επιτρέπει την ανάπτυξη πιο αποδοτικών και ευέλικτων στρατηγικών. Σύμφωνα με τους Lampropoulos & Tsihrintzis (2012), η χρήση περιγραφικών χαρακτηριστικών MPEG-7 μπορεί να ενισχύσει τη δυναμική αξιολόγηση των εργασιών και τη διαμόρφωση προσαρμοστικών στρατηγικών κατανομής πόρων.

Η χρήση εξατομικευμένων προσεγγίσεων στη διαχείριση εργασιών μπορεί να βελτιώσει την αποδοτικότητα των συστημάτων. Σύμφωνα με τους Panagoulias et al. (2022), η υιοθέτηση μικρο-υπηρεσιών (microservices) σε συνδυασμό με τη μεθοδολογία RUP επιτρέπει την ανάπτυξη

αξιόπιστων και εξηγήσιμων συστημάτων διαχείρισης εργασιών. Αυτή η προσέγγιση συμβάλλει στη βελτίωση της διαφάνειας και της δυνατότητας προσαρμογής των εφαρμογών.

Η ανάπτυξη προσαρμοστικών συστημάτων διαχείρισης εργασιών βασίζεται σε ευφυή μοντέλα χρηστών που επιτρέπουν τη δυναμική προσαρμογή των διεπαφών και των στρατηγικών χρονοπρογραμματισμού. Οι Virvou & Moundridou (2001) προτείνουν τη διάκριση μεταξύ μοντέλων φοιτητών και εκπαιδευτών σε έξυπνα συστήματα διδασκαλίας, μια προσέγγιση που μπορεί να προσαρμοστεί και στη διαχείριση εργασιών για τη βελτίωση της εξατομίκευσης και της διαδραστικότητας.

1.2 Αλγόριθμοι Χρονολόγησης

Οι αλγόριθμοι χρονολόγησης αποτελούν τον πυρήνα ενός αποτελεσματικού συστήματος διαχείρισης εργασιών, καθορίζοντας τον τρόπο με τον οποίο κατανέμονται οι διαθέσιμοι πόροι και προγραμματίζεται η εκτέλεση των εργασιών. Στο πλαίσιο της παρούσας εργασίας, εξετάζονται τρεις θεμελιώδεις αλγόριθμοι χρονολόγησης: ο Shortest Job Next, ο Least Laxity First και ο Round Robin. Κάθε αλγόριθμος προσεγγίζει το πρόβλημα του χρονοπρογραμματισμού με διαφορετικό τρόπο, προσφέροντας μοναδικά πλεονεκτήματα και παρουσιάζοντας συγκεκριμένους περιορισμούς. Σε αυτή την ενότητα, αναλύεται λεπτομερώς η λειτουργία κάθε αλγορίθμου, εξετάζονται οι μηχανισμοί τους και αξιολογείται η αποτελεσματικότητά τους στο πλαίσιο ενός συστήματος διαχείρισης εργασιών με περιορισμό οκταώρου. Τέλος, πραγματοποιείται μια συγκριτική ανάλυση των τριών αλγορίθμων, αναδεικνύοντας τις περιπτώσεις στις οποίες ο καθένας αποδεικνύεται καταλληλότερος.

Οι πρόσφατες μελέτες στον τομέα της δυναμικής κατανομής εργασιών έχουν δείξει ότι η ευελιξία στους αλγόριθμους χρονοπρογραμματισμού βελτιώνει σημαντικά τη συνολική απόδοση του συστήματος. Οι Xu et al. (2016) προτείνουν ένα δυναμικό μοντέλο προτεραιότητας που επιτρέπει την προσαρμογή των πολιτικών προγραμματισμού με βάση τις μεταβαλλόμενες απαιτήσεις εργασιών, εξασφαλίζοντας έτσι καλύτερη αξιοποίηση των διαθέσιμων πόρων.

Επιπλέον, οι Kontogianni et al. (2024) εισάγουν τεχνικές μηχανικής μάθησης και blockchain για τη διαχείριση εργασιών στον έξυπνο τουρισμό, αναδεικνύοντας τη σημασία της τεχνητής νοημοσύνης και της διαφάνειας στις στρατηγικές προγραμματισμού εργασιών.

Οι σύγχρονες τάσεις στον χρονοπρογραμματισμό εργασιών περιλαμβάνουν τη χρήση τεχνικών νευρωνικών δικτύων και γενετικών αλγορίθμων για την πρόβλεψη των βέλτιστων πολιτικών κατανομής εργασιών. Σύμφωνα με τους Stathopoulou & Tsihrintzis (2011), η αναγνώριση συναισθημάτων μέσω ανάλυσης κινήσεων σώματος και χειρονομιών μπορεί να ενισχύσει την εξατομίκευση των στρατηγικών προγραμματισμού.

Επιπλέον, οι Virvou & Tsihrintzis (2023) προτείνουν ένα καινοτόμο μοντέλο εμπιστοσύνης βασισμένο σε καταστάσεις (Trust State-Chart Model) για την ενσωμάτωση της αξιοπιστίας στη διαχείριση εργασιών και την αλληλεπίδραση ανθρώπου-μηχανής.

1.2.1 Shortest Job Next

Ο αλγόριθμος Shortest Job Next, γνωστός και ως Shortest Job First (SJF), αποτελεί έναν από τους θεμελιώδεις αλγόριθμους χρονοπρογραμματισμού που βασίζεται στην αρχή της προτεραιοποίησης των εργασιών με βάση τον εκτιμώμενο χρόνο εκτέλεσής τους. Ο αλγόριθμος επιλέγει για εκτέλεση την εργασία με τον μικρότερο χρόνο ολοκλήρωσης, προσφέροντας σημαντικά πλεονεκτήματα στη διαχείριση των πόρων του συστήματος. Σύμφωνα με τους Praveen et al. (2018), ο SJN μπορεί να μειώσει σημαντικά τον μέσο χρόνο αναμονής των εργασιών, καθώς οι σύντομες εργασίες ολοκληρώνονται γρήγορα, επιτρέποντας την επεξεργασία περισσότερων εργασιών σε μικρότερο χρονικό διάστημα.

Η εφαρμογή του SJN σε ένα σύστημα διαχείρισης εργασιών παρουσιάζει τόσο πλεονεκτήματα όσο και προκλήσεις. Ο αλγόριθμος μπορεί να οδηγήσει σε αυξημένη διεκπεραιωτική ικανότητα και αποτελεσματικότερη χρήση των πόρων του συστήματος, καθώς οι μικρότερες εργασίες είναι λιγότερο πιθανό να προκαλέσουν συμφόρηση (Praveen et al., 2018). Ωστόσο, όπως επισημαίνουν οι Ibanez et al. (2015), η συνεχής εναλλαγή μεταξύ εργασιών μπορεί να επιφέρει χρονικό κόστος και να οδηγήσει σε απώλεια των πλεονεκτημάτων που προσφέρει η ομαδοποίηση παρόμοιων εργασιών.

Για την αποτελεσματική διαχείριση νέων εργασιών κατά το χρόνο εκτέλεσης, ο SJN μπορεί να υλοποιηθεί χρησιμοποιώντας διάφορες στρατηγικές. Μία προσέγγιση είναι η εφαρμογή ενός μοντέλου ουράς που ομαδοποιεί εργασίες του ίδιου τύπου σε μία εικονική ουρά, χρησιμοποιώντας την πολιτική Min-Min Best Fit (MM-BF) για τον προγραμματισμό εργασιών μεταξύ διαφορετικών ουρών. Για την αποφυγή του φαινομένου της ασιτίας των μεγαλύτερων εργασιών, μπορεί να εφαρμοστεί η πολιτική MaxWeight βασισμένη στο μήκος της ουράς (Queue-length-based MaxWeight – QMW), επιτυγχάνοντας έτσι χαμηλή καθυστέρηση και υψηλή απόδοση (Guo et al., 2022).

1.2.2 Least Laxity First

Ο αλγόριθμος Least Laxity First αποτελεί έναν προηγμένο αλγόριθμο χρονοπρογραμματισμού που λαμβάνει υπόψη τόσο την προθεσμία όσο και τον χρόνο εκτέλεσης των εργασιών για τη λήψη αποφάσεων προγραμματισμού. Η βασική αρχή του αλγορίθμου βασίζεται στην έννοια της χαλαρότητας (laxity), η οποία ορίζεται ως η διαφορά μεταξύ της προθεσμίας και του υπολειπόμενου χρόνου εκτέλεσης μιας εργασίας. Σύμφωνα με τους Perret et al. (2013), ο LLF προσφέρει σημαντικά πλεονεκτήματα στη διαχείριση προσωπικών εργασιών,

καθώς εξασφαλίζει ότι οι εργασίες με τον λιγότερο διαθέσιμο χρόνο αντιμετωπίζονται πρώτες, μειώνοντας τον κίνδυνο αθέτησης προθεσμιών.

Ωστόσο, η υλοποίηση του LLF σε ένα σύστημα με περιορισμό οκταώρου παρουσιάζει σημαντικές προκλήσεις. Όπως επισημαίνουν οι Oh & Yang (1998), ο αλγόριθμος μπορεί να οδηγήσει σε συχνές εναλλαγές περιεχομένου (context switches) όταν πολλαπλές εργασίες έχουν παρόμοιες ή ταυτόσημες τιμές χαλαρότητας. Αυτές οι συχνές εναλλαγές μπορούν να υποβαθμίσουν την απόδοση του συστήματος και να αυξήσουν την επιβάρυνση, καθιστώντας τον αλγόριθμο λιγότερο πρακτικό για εφαρμογές πραγματικού κόσμου. Για την αντιμετώπιση αυτών των προκλήσεων, έχουν προταθεί διάφορες βελτιώσεις του αλγορίθμου, όπως ο Modified Least-Laxity-First (MLLF), ο οποίος μειώνει σημαντικά τον αριθμό των εναλλαγών περιεχομένου.

Η βελτιστοποίηση του LLF για τη διαχείριση δυναμικών ενημερώσεων στις προθεσμίες και τις διάρκειες των εργασιών περιλαμβάνει διάφορες στρατηγικές. Μία σημαντική προσέγγιση είναι ο Enhanced Least-Laxity-First (ELLF), ο οποίος, σύμφωνα με τους Hildebrandt et al. (1999), χρησιμοποιεί έναν επεξεργαστή χρονοπρογραμματισμού για τη διαχείριση της υπολογιστικής επιβάρυνσης. Επιπλέον, η εφαρμογή δυναμικού προγραμματισμού και κανόνων προτεραιότητας, όπως η αρχή Less Laxity and Longer remaining Processing time (LLLP), μπορεί να βελτιστοποιήσει τον χρονοπρογραμματισμό, ελαχιστοποιώντας το συνολικό κόστος και τις ποινές για αθέτηση προθεσμιών (Xu et al., 2016).

1.2.3 Round Robin

Ο αλγόριθμος Round Robin αποτελεί έναν από τους πιο διαδεδομένους αλγόριθμους χρονοπρογραμματισμού, που διακρίνεται για την απλότητα και τη δικαιοσύνη στην κατανομή των πόρων. Η βασική αρχή του αλγορίθμου είναι η ισότιμη κατανομή του χρόνου εκτέλεσης μεταξύ των εργασιών, αποτρέποντας τη μονοπώληση του συστήματος από μεμονωμένες εργασίες (Hwang et al., 2021). Ο αλγόριθμος χρησιμοποιεί ένα χρονικό κβάντο (time quantum), που καθορίζει το μέγιστο χρόνο που μπορεί να εκτελεστεί μια εργασία πριν παραχωρήσει τη σειρά της στην επόμενη.

Για την αποτελεσματική διαχείριση εργασιών διαφορετικής προτεραιότητας και προθεσμιών, έχουν προταθεί διάφορες προσαρμογές του αλγορίθμου RR. Μία σημαντική βελτίωση είναι η εισαγωγή δυναμικού χρονικού κβάντου, το οποίο προσαρμόζεται με βάση τα χαρακτηριστικά κάθε εργασίας, όπως ο χρόνος εκτέλεσης και η προτεραιότητα. Σύμφωνα με τους Iqbal et al. (2023) και Datta (2015), αυτή η προσέγγιση μπορεί να βελτιώσει σημαντικά την αξιοποίηση των πόρων και το χρόνο εκτέλεσης των εργασιών, διασφαλίζοντας ότι οι εργασίες υψηλής προτεραιότητας λαμβάνουν περισσότερο χρόνο CPU και ολοκληρώνονται έγκαιρα.

Η αποτελεσματικότητα του αλγορίθμου RR μπορεί να ενισχυθεί περαιτέρω μέσω της υιοθέτησης υβριδικών προσεγγίσεων και τεχνικών βελτιστοποίησης. Για παράδειγμα, η χρήση νευρωνικών δικτύων για την πρόβλεψη του βέλτιστου μήκους κβάντου με βάση τους χρόνους

εξυπηρέτησης των εργασιών στην ουρά αναμονής μπορεί να ελαχιστοποιήσει το μέσο χρόνο διεκπεραίωσης και να βελτιώσει τη συνολική αποδοτικότητα (AlHeyasat & Herzallah, 2010; Zouaoui et al., 2019). Επιπλέον, η ενσωμάτωση τεχνικών πολλαπλών κριτηρίων στον αλγόριθμο επιτρέπει την καλύτερη εξισορρόπηση μεταξύ διαφορετικών στόχων απόδοσης, όπως η ελαχιστοποίηση του χρόνου αναμονής και η βελτιστοποίηση της χρήσης των πόρων.

1.2.4 Σύγκριση Αλγορίθμων

Οι τρεις αλγόριθμοι χρονοπρογραμματισμού - SJN, LLF και RR - παρουσιάζουν διακριτά χαρακτηριστικά και συμπεριφορές στη διαχείριση των εργασιών. Σύμφωνα με τους Damuut & Dung (2019), ο SJN προσφέρει τους χαμηλότερους μέσους χρόνους απόκρισης καθώς δίνει προτεραιότητα στις συντομότερες εργασίες, ωστόσο μπορεί να οδηγήσει σε φαινόμενα ασιτίας για τις μεγαλύτερες εργασίες. Αντίθετα, ο αλγόριθμος LLF επιτυγχάνει καλύτερη εξισορρόπηση μεταξύ χρόνου απόκρισης και δικαιοσύνης, καθώς λαμβάνει υπόψη τόσο τις προθεσμίες όσο και τους χρόνους εκτέλεσης των εργασιών (Sahkhar et al., 2022), ενώ ο RR εξασφαλίζει τη μεγαλύτερη δικαιοσύνη μέσω της ισότιμης κατανομής του χρόνου επεξεργασίας.

Όσον αφορά την αξιοποίηση των πόρων του συστήματος, κάθε αλγόριθμος παρουσιάζει διαφορετικά πλεονεκτήματα και περιορισμούς. Ο SJN μπορεί να οδηγήσει σε υψηλή αξιοποίηση της CPU καθώς ελαχιστοποιεί τον ανενεργό χρόνο επεξεργάζοντας γρήγορα τις μικρότερες εργασίες, αλλά ενδέχεται να μην είναι βέλτιστος για συστήματα με μείγμα μικρών και μεγάλων εργασιών (Damuut & Dung, 2019). Ο LLF επιτυγχάνει αποτελεσματική χρήση των πόρων εστιάζοντας στις εργασίες που πλησιάζουν στις προθεσμίες τους, αλλά απαιτεί πιο πολύπλοκους υπολογισμούς και παρακολούθηση, ενώ ο RR προσφέρει μέτρια αξιοποίηση των πόρων λόγω της αυξημένης επιβάρυνσης από τις εναλλαγές περιεχομένου (Sahkhar et al., 2022).

Η επιλογή του καταλληλότερου αλγορίθμου εξαρτάται από τις συγκεκριμένες απαιτήσεις του συστήματος και τα χαρακτηριστικά του φόρτου εργασίας. Σε περιπτώσεις όπου κυριαρχούν οι σύντομες εργασίες, ο SJN αποτελεί την καλύτερη επιλογή για την ελαχιστοποίηση του χρόνου αναμονής και τη βελτίωση της αξιοποίησης της CPU. Για συστήματα με αυστηρές προθεσμίες και απαιτήσεις πραγματικού χρόνου, ο LLF προσφέρει τη βέλτιστη λύση, ενώ σε περιβάλλοντα όπου η δικαιοσύνη είναι πρωταρχικής σημασίας, ο RR αποτελεί την πιο κατάλληλη επιλογή (Caranto et al., 2020). Σε πολλές περιπτώσεις, ένας συνδυασμός των αλγορίθμων μπορεί να προσφέρει την καλύτερη ισορροπία μεταξύ αποδοτικότητας, δικαιοσύνης και τήρησης προθεσμιών.

Οι αλγόριθμοι χρονοπρογραμματισμού συνεχίζουν να εξελίσσονται με την εισαγωγή ευφυών τεχνικών που βασίζονται στη συμπεριφορά των χρηστών. Σύμφωνα με τους Papadimitriou et al. (2021), η δυναμική εξατομίκευση των χρονοδιαγραμμάτων με χρήση μεθόδων μηχανικής μάθησης μπορεί να οδηγήσει σε πιο αποδοτικές και δίκαιες πολιτικές προγραμματισμού.

Επιπλέον, οι Alepis et al. (2010) αναλύουν τη σημασία της αναγνώρισης συναισθημάτων μέσω φωνητικών και οπτικών σημάτων για την προσαρμογή των στρατηγικών διαχείρισης εργασιών, προτείνοντας ένα δίδυμο μοντέλο αλληλεπίδρασης που λαμβάνει υπόψη τις συναισθηματικές αντιδράσεις των χρηστών.

Οι σύγχρονες τεχνικές προγραμματισμού εργασιών ενσωματώνουν πλέον εξελιγμένα μοντέλα πρόβλεψης για τη διαχείριση των πόρων. Οι Stathopoulou & Tsihrintzis (2005) εξετάζουν τη χρήση τεχνητών νευρωνικών δικτύων για την ανίχνευση και ταξινόμηση συναισθημάτων, τα οποία μπορούν να αξιοποιηθούν για τη δυναμική προσαρμογή των αλγορίθμων χρονοπρογραμματισμού, λαμβάνοντας υπόψη την ανθρώπινη συμπεριφορά.

Η προσαρμογή των αλγορίθμων χρονοπρογραμματισμού στις ατομικές ανάγκες των χρηστών αποτελεί σημαντική πρόκληση. Οι Zouaoui et al. (2019) προτείνουν τη χρήση νευρωνικών δικτύων για την εκτίμηση του βέλτιστου χρονικού κβάντου στον αλγόριθμο Round Robin, βελτιώνοντας τη δυναμική διαχείριση των εργασιών και την αποδοτικότητα του συστήματος.

1.3 Τεχνολογίες Ανάπτυξης

Για την ανάπτυξη του συστήματος διαχείρισης εργασιών επιλέχθηκαν σύγχρονες και ώριμες τεχνολογίες που εξασφαλίζουν την αποτελεσματική υλοποίηση, συντήρηση και επεκτασιμότητα της εφαρμογής. Συγκεκριμένα, χρησιμοποιήθηκε ο συνδυασμός Java και JavaFX για την ανάπτυξη του κώδικα και της διεπαφής χρήστη, ενώ για τη διαχείριση του έργου και την αυτοματοποίηση της διαδικασίας κατασκευής επιλέχθηκε το εργαλείο Gradle.

Η Java και το JavaFX επιλέχθηκαν λόγω της ωριμότητας και της αξιοπιστίας τους στην ανάπτυξη εφαρμογών ελέγχου, προσφέροντας ένα σταθερό περιβάλλον ανάπτυξης. Σύμφωνα με τους Kruk et al. (2017), το JavaFX υποστηρίζει τον σαφή διαχωρισμό μεταξύ της λογικής (Controller) και του οπτικού μέρους (FXML), το οποίο μπορεί να σχεδιαστεί χρησιμοποιώντας εργαλεία όπως το Scene Builder. Αυτός ο διαχωρισμός ενισχύει τη συντηρησιμότητα και τη δυνατότητα ελέγχου των εφαρμογών, ενώ η ενσωμάτωση χαρακτηριστικών της Java 8, όπως οι lambda εκφράσεις και οι αναφορές μεθόδων, απλοποιεί τη διαδικασία ανάπτυξης.

Το JavaFX προσφέρει επίσης ένα δηλωτικό στυλ προγραμματισμού για τη δημιουργία διεπαφών χρήστη, μειώνοντας την πολυπλοκότητα σε σύγκριση με παλαιότερα frameworks όπως το Java Swing και το Java 2D (Vos et al., 2014). Επιπλέον, χαρακτηριστικά όπως τα property bindings και οι change listeners μειώνουν την ανάγκη για boilerplate κώδικα, όπως οι μέθοδοι setter, καθιστώντας τη διαδικασία ανάπτυξης πιο αποδοτική.

Το Gradle επιλέχθηκε ως εργαλείο αυτοματοποίησης και διαχείρισης της διαδικασίας κατασκευής (build automation) λόγω της ευελιξίας και της αποτελεσματικότητάς του. Σύμφωνα με τον Varanasi (2015), το Gradle χρησιμοποιεί μια προσέγγιση build-by-convention, προτείνοντας τυποποιημένες δομές έργου, όπως την τοποθέτηση του πηγαίου κώδικα Java στο src/main/java και του κώδικα δοκιμών στο src/test/java. Αυτή η τυποποίηση απλοποιεί τη ρύθμιση του έργου και διευκολύνει τους προγραμματιστές στη μετάβαση μεταξύ έργων.

Ένα σημαντικό πλεονέκτημα του Gradle είναι η ισχυρή υποστήριξη για τη διαχείριση εξαρτήσεων και την επεκτασιμότητα μέσω plugins. Όπως επισημαίνει ο Prakash (2022), το Gradle είναι γνωστό για την απόδοση και την αποτελεσματικότητά του στην αυτοματοποίηση της κατασκευής και της ανάπτυξης, ενώ είναι σχεδιασμένο να χειρίζεται πολύπλοκες διαδικασίες κατασκευής πιο αποτελεσματικά από εναλλακτικές λύσεις όπως το Maven και το Ant. Επιπλέον, η χρήση της γλώσσας Groovy για τη διαμόρφωση, σε αντίθεση με την XML του Maven, προσφέρει μεγαλύτερη ευελιξία και λιγότερη πολυπλοκότητα στα build scripts.

Η υποστήριξη για σταδιακές κατασκευές (incremental builds) αποτελεί ένα ακόμη σημαντικό χαρακτηριστικό του Gradle. Αυτό σημαίνει ότι επανακατασκευάζει μόνο τα απαραίτητα τμήματα του έργου, μειώνοντας σημαντικά τους χρόνους κατασκευής σε σύγκριση με το Maven και το Ant (Prakash, 2022). Επιπρόσθετα, σύμφωνα με τους Lou et al. (2019), το Gradle έχει γίνει το πιο ευρέως χρησιμοποιούμενο σύστημα κατασκευής για έργα Java στην κοινότητα ανοιχτού κώδικα, γεγονός που υποδηλώνει ισχυρή υποστήριξη από την κοινότητα και πλούσιους διαθέσιμους πόρους.

Η διαδικασία ανάπτυξης RUP επιλέχθηκε ως μεθοδολογία για τη συστηματική ανάπτυξη της εφαρμογής, καθώς παρέχει ένα δομημένο πλαίσιο με καθορισμένες φάσεις και ροές εργασίας. Η RUP διαιρεί τη διαδικασία ανάπτυξης σε επαναλήψεις, επιτρέποντας τη σταδιακή βελτίωση και τη συνεχή ενσωμάτωση του συστήματος. Αυτή η προσέγγιση είναι ιδιαίτερα χρήσιμη για τη διαχείριση αλλαγών και τη μείωση των κινδύνων κατά την ανάπτυξη πολύπλοκων συστημάτων όπως οι αλγόριθμοι χρονοπρογραμματισμού.

Η RUP υποστηρίζει επίσης την ανάπτυξη σύνθετων αλγορίθμων χρονοπρογραμματισμού και στοιχείων διεπαφής χρήστη μέσω της επαναληπτικής και σταδιακής προσέγγισής της. Όπως αναφέρουν οι Prihartono & Utami (2023), η μεθοδολογία δίνει έμφαση στη συνεργασία μεταξύ της ομάδας ανάπτυξης και των ενδιαφερομένων, καθώς και στην ολοκληρωμένη τεκμηρίωση σε κάθε στάδιο της ανάπτυξης. Αυτή η συνεργατική προσέγγιση διασφαλίζει ότι όλες οι απαιτήσεις και οι προσδοκίες κατανοούνται σαφώς και ικανοποιούνται, ενώ η λεπτομερής τεκμηρίωση παρέχει μια σαφή αναφορά για μελλοντικές επαναλήψεις και συντήρηση.

Τέλος, η RUP εξασφαλίζει την ποιότητα και τη συντηρησιμότητα του λογισμικού μέσω συγκεκριμένων τεχνουργημάτων (artifacts) που παράγονται σε κάθε φάση ανάπτυξης. Σύμφωνα με τους Pareto & Boquist (2006) και Monteiro et al. (2013), αυτά τα τεχνουργήματα περιλαμβάνουν το έγγραφο αρχιτεκτονικής λογισμικού (Single Administrative Document – SAD),

λεπτομερείς προδιαγραφές απαιτήσεων, μοντέλα σχεδιασμού και σχέδια ελέγχου, τα οποία συμβάλλουν στη διασφάλιση ενός ισχυρού και κλιμακώσιμου σχεδιασμού που υποστηρίζει τη συντηρησιμότητα.

Οι τεχνολογικές εξελίξεις στον προγραμματισμό και τη βελτιστοποίηση έχουν οδηγήσει στη χρήση καινοτόμων μεθόδων ανάπτυξης εφαρμογών διαχείρισης εργασιών. Οι Chrysafiadi et al. (2023) εξετάζουν τη χρήση ασαφούς λογικής και μηχανικής μάθησης για τη δημιουργία προσαρμοστικών εκπαιδευτικών συστημάτων, μία προσέγγιση που μπορεί να επεκταθεί και στη διαχείριση εργασιών.

Επιπλέον, οι Kabassi & Virvou (2006) προτείνουν ένα πολυκριτηριακό πλαίσιο λήψης αποφάσεων για διεπαφές χρήστη, το οποίο μπορεί να αξιοποιηθεί στη δημιουργία έξυπνων διεπαφών χρήστη για συστήματα διαχείρισης εργασιών.

Η εφαρμογή προηγμένων τεχνικών μηχανικής μάθησης και συστημάτων βασισμένων σε γνώσεις αποτελεί κεντρικό σημείο στις σύγχρονες τεχνολογίες ανάπτυξης εφαρμογών διαχείρισης εργασιών. Οι Savvopoulos & Virvou (2010) εξετάζουν τη χρήση ενός πλαισίου κύκλου ζωής λογισμικού για την ενσωμάτωση αλγορίθμων ομαδοποίησης δεδομένων σε προσαρμοστικά περιβάλλοντα εργασίας.

Επιπλέον, οι Tsihrintzis & Virvou (2010) αναλύουν τις προκλήσεις και τις λύσεις που σχετίζονται με την ανάπτυξη πολυμεσικών υπηρεσιών σε έξυπνα περιβάλλοντα, υπογραμμίζοντας τη σημασία των διεπαφών χρήστη που βασίζονται σε ευφυή αλγορίθμους.

Η τεχνολογική πρόοδος έχει επιτρέψει την ανάπτυξη πιο ευφυών και προσαρμοστικών συστημάτων διαχείρισης εργασιών. Οι AlHeyasat & Herzallah (2010) προτείνουν μια μέθοδο εκτίμησης του ιδανικού χρονικού κβάντου για τον αλγόριθμο Round Robin, χρησιμοποιώντας νευρωνικά δίκτυα. Αυτή η προσέγγιση επιτρέπει την αυτοματοποιημένη προσαρμογή του χρονοπρογραμματισμού με βάση το εκάστοτε φορτίο εργασίας.

Επιπλέον, οι Tsiriga & Virvou (2004) παρουσιάζουν μια μεθοδολογία για την ενσωμάτωση λογισμικού βασισμένου σε ανθρώπινη λογική στα περιβάλλοντα διαχείρισης εργασιών, καθιστώντας τις εφαρμογές πιο προσαρμοστικές στις ανάγκες των χρηστών.

Η ανάγκη για προσαρμοστικές εφαρμογές διαχείρισης εργασιών έχει οδηγήσει στην ανάπτυξη μεθόδων που βασίζονται στη μοντελοποίηση της ανθρώπινης συμπεριφοράς. Σύμφωνα με τους Papadimitriou & Virvou (2017), η προσαρμογή των σεναρίων σε εκπαιδευτικά και διαχειριστικά περιβάλλοντα μέσω ασαφούς λογικής μπορεί να εφαρμοστεί και σε συστήματα διαχείρισης εργασιών, ενισχύοντας την εξατομίκευση και την ευελιξία.

Η ανάπτυξη σύγχρονων εφαρμογών διαχείρισης εργασιών απαιτεί τη χρήση αποδοτικών εργαλείων και πλαισίων εργασίας. Οι Vos et al. (2014) παρουσιάζουν το JavaFX ως μια εξελιγμένη πλατφόρμα για τη δημιουργία διαδραστικών εφαρμογών, επιτρέποντας την ανάπτυξη πιο ευέλικτων και δυναμικών συστημάτων διαχείρισης εργασιών.

Επιπλέον, η πρόοδος στην τεχνητή νοημοσύνη έχει επηρεάσει την ανάπτυξη εφαρμογών με επίκεντρο τον χρήστη. Οι Virvou & Nakamura (2008) αναλύουν πώς η γνώση-βασισμένη μηχανική λογισμικού μπορεί να ενισχύσει την προσαρμοστικότητα και την απόδοση των εφαρμογών διαχείρισης εργασιών.

1.4 Σχετικές Εργασίες και Εφαρμογές

Η βιβλιογραφική ανασκόπηση στον τομέα των εφαρμογών διαχείρισης εργασιών αναδεικνύει μια σειρά από καινοτόμες προσεγγίσεις και εφαρμογές που σχετίζονται με τον χρονοπρογραμματισμό εργασιών. Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι εξελίξεις στην ανάπτυξη υβριδικών αλγορίθμων και τεχνικών προσαρμοστικού χρονοπρογραμματισμού που στοχεύουν στη βελτιστοποίηση της διαχείρισης εργασιών σε διάφορα περιβάλλοντα.

Μια σημαντική καινοτομία στον τομέα είναι η ανάπτυξη υβριδικών ευρετικών και γενετικών αλγορίθμων για τον χρονοπρογραμματισμό εργασιών. Οι Sulaiman et al. (2021) προτείνουν τον Hybrid Heuristic and Genetic-based Task Scheduling Algorithm (HHG) και τον Hybrid Task Duplication and Genetic-based Task Scheduling Algorithm (HTDG), οι οποίοι συνδυάζουν γενετικούς αλγορίθμους με προσεγγίσεις βασισμένες σε λίστες για τη βελτίωση της ποιότητας των αρχικών πληθυσμών. Αυτοί οι αλγόριθμοι έχουν επιδείξει σημαντικά καλύτερα αποτελέσματα σε όρους χρόνου ολοκλήρωσης, αναλογίας μήκους χρονοδιαγράμματος και ταχύτητας εκτέλεσης σε σύγκριση με τις υπάρχουσες μεθόδους.

Στον τομέα του προγραμματισμού πραγματικού χρόνου και της ανθεκτικότητας στις αλλαγές, οι Pupa et al. (2022) έχουν αναπτύξει ένα διαδικτυακό πλαίσιο για τη συνεργασία ανθρώπου-ρομπότ (Human-Robot Collaboration – HRC) που προσαρμόζεται σε αποκλίσεις κατά τη λειτουργία, διαφορετικές δεξιότητες χειριστών και σφάλματα. Το πλαίσιο αυτό διασφαλίζει ένα ανθεκτικό και αξιόπιστο χρονοδιάγραμμα εργασιών, ενώ παράλληλα χρησιμοποιεί δίκτυα προσοχής γράφων για τον αυτόματο προσδιορισμό χαρακτηριστικών χρονοπρογραμματισμού, παρέχοντας λύσεις υψηλής ποιότητας με σημαντικά μειωμένο χρόνο υπολογισμού.

Στον τομέα της νοημοσύνης σμήνους και της ενεργειακής αποδοτικότητας, οι ερευνητές έχουν προτείνει καινοτόμες προσεγγίσεις για τη βελτιστοποίηση του χρονοπρογραμματισμού εργασιών. Συγκεκριμένα, οι Attiya et al. (2022) ανέπτυξαν τον αλγόριθμο MRFOSSA, ο οποίος συνδυάζει τη βελτιστοποίηση αναζήτησης τροφής σαλαχιών Manta Ray με τον αλγόριθμο Salp Swarm για την ενίσχυση της τοπικής αναζήτησης και των ρυθμών σύγκλισης. Αυτή η προσέγγιση

έχει αποδειχθεί ανώτερη από άλλες μεταευρετικές τεχνικές όσον αφορά τον χρόνο ολοκλήρωσης και τη διεκπεραιωτική ικανότητα του συστήματος.

Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι προσεγγίσεις που εστιάζουν στον ανθρωποκεντρικό και δυναμικό χρονοπρογραμματισμό. Οι Pupa et al. (2021) προτείνουν μια αρχιτεκτονική δυναμικού χρονοπρογραμματισμού με επίκεντρο τον άνθρωπο, η οποία λαμβάνει υπόψη τους περιορισμούς εκτέλεσης εργασιών, την ανθρώπινη μεταβλητότητα και την ποιότητα εργασίας. Επιπλέον, οι Zhang et al. (2022) έχουν ενσωματώσει μικρο-διαλείμματα στους κύκλους εργασίας για την πρόληψη της ανθρώπινης κόπωσης, βελτιστοποιώντας τον χρόνο κύκλου εργασίας και διατηρώντας την ανθρώπινη υγεία.

Οι Richards & Stottler (2019) έχουν αναπτύξει το πλαίσιο Aurora, το οποίο αποτυπώνει τη συλλογιστική των ανθρώπων χρονοπρογραμματιστών, μοντελοποιεί λεπτομερώς τους ανθρώπινους πόρους και διαχειρίζεται ατελή δεδομένα. Το πλαίσιο αυτό παρέχει ισχυρή υποστήριξη περιορισμών και εργαλεία οπτικοποίησης, και έχει εφαρμοστεί με επιτυχία σε πολύπλοκα σενάρια χρονοπρογραμματισμού από οργανισμούς όπως η NASA και η Boeing, αποδεικνύοντας τη σημασία της ενσωμάτωσης της ανθρώπινης εμπειρίας στα συστήματα χρονοπρογραμματισμού.

Σημαντικές εξελίξεις έχουν σημειωθεί και στον τομέα της οπτικοποίησης και διαχείρισης των χρονοδιαγραμμάτων εργασιών μέσω διεπαφών ημερολογίου. Οι Mackinlay et al. (1994) έχουν αναπτύξει καινοτόμες προσεγγίσεις όπως το Spiral Calendar και το Time Lattice, τα οποία χρησιμοποιούν τρισδιάστατα γραφικά και διαδραστικά κινούμενα σχέδια για την ενίσχυση της οπτικοποίησης των χρονοδιαγραμμάτων εργασιών. Επιπλέον, οι Faulring & Myers (2006) εισήγαγαν την έννοια των "availability bars", μια καινοτόμο τεχνική αλληλεπίδρασης και οπτικοποίησης σχεδιασμένη για τη διαχείριση πολύπλοκων περιορισμών χρονοπρογραμματισμού, επιτρέποντας στους χρήστες να καθορίζουν την προτίμησή τους για διαθέσιμες χρονικές θυρίδες.

Τέλος, στον τομέα της αξιολόγησης της απόδοσης των αλγορίθμων χρονοπρογραμματισμού, έχουν αναπτυχθεί διάφορες μεθοδολογίες για τη σύγκριση και βελτιστοποίηση των διαφορετικών προσεγγίσεων. Σύμφωνα με τους Kondo et al. (2007), η χρήση προσομοιώσεων βασισμένων σε ίχνη (trace-driven simulations) επιτρέπει την αξιολόγηση των πολιτικών χρονοπρογραμματισμού σε περιβάλλοντα desktop grid, εστιάζοντας στην προτεραιοποίηση πόρων, τον αποκλεισμό και την αναπαραγωγή εργασιών. Επιπλέον, οι Mishra (2017) και Stan et al. (2021) έχουν προτείνει τη χρήση στατιστικών μεθόδων και μηχανικής μάθησης για τη βελτίωση του χρονοπρογραμματισμού και της χρήσης πόρων, με ιδιαίτερη έμφαση στη μέτρηση τυπικών μετρικών όπως ο μέσος χρόνος αναμονής, ο χρόνος διεκπεραίωσης και η απόδοση.

Η εφαρμογή προηγμένων μεθόδων τεχνητής νοημοσύνης στη διαχείριση εργασιών συνεχίζει να αποτελεί ερευνητικό πεδίο με πολλές πρακτικές εφαρμογές. Σύμφωνα με τους Moundridou & Virvou (2001), η ανάπτυξη ευφυών μοντέλων χρήστη μπορεί να βελτιώσει

σημαντικά την απόδοση εξατομικευμένων περιβαλλόντων εργασίας, επιτρέποντας την προσαρμογή των στρατηγικών προγραμματισμού με βάση τις ανάγκες κάθε χρήστη.

Επιπλέον, οι Tsiriga & Virvou (2003) μελετούν τη δυναμική προσαρμογή του μαθησιακού περιβάλλοντος στις ανάγκες του χρήστη, προσφέροντας πολύτιμες γνώσεις για την εφαρμογή εξατομίκευσης στη διαχείριση εργασιών.

Τεχνητή Νοημοσύνη στην Αλληλεπίδραση Ανθρώπου-Υπολογιστή και Κινητή Υπολογιστική

Η τεχνητή νοημοσύνη (ΤΝ) έχει επηρεάσει σημαντικά την αλληλεπίδραση ανθρώπου-υπολογιστή και την κινητή υπολογιστική, οδηγώντας σε εξελίξεις που αλλάζουν ριζικά τον τρόπο που οι χρήστες αλληλεπιδρούν με τις τεχνολογίες. Οι σύγχρονες έρευνες τονίζουν τον ρόλο της ΤΝ στη βελτίωση των διεπαφών χρήστη, την προσαρμογή των αλληλεπιδράσεων και την ανάπτυξη ευφυών συστημάτων.

Μελέτες όπως αυτές των Tsihrintzis, Virvou και Hatzilygeroudis (2023) αναδεικνύουν πώς η ΤΝ συμβάλλει στον επανασχεδιασμό των διεπαφών χρήστη, επιτρέποντας πιο ευέλικτες και προσαρμοστικές εμπειρίες. Παράλληλα, οι Virvou, Tsihrintzis, Bourbakis και Jain (2022) τονίζουν τη σημασία της ΤΝ στην ενίσχυση της μηχανικής λογισμικού, ειδικά στην ανάπτυξη έξυπνων εφαρμογών για κυβερνο-φυσικά συστήματα.

Η σχέση μεταξύ ΤΝ και αλληλεπίδρασης ανθρώπου-υπολογιστή έχει ερευνηθεί εκτενώς σε συλλογές άρθρων όπως αυτές των Tsihrintzis, Virvou και Jain (2022) και Weng, Shieh και Tsihrintzis (2023), που εξετάζουν την εφαρμογή προηγμένων τεχνικών πολυμέσων και κρυπτογραφίας για τη βελτίωση της εμπειρίας του χρήστη.

Ιστορική Εξέλιξη και Σύγχρονες Εφαρμογές

Η χρήση της ΤΝ στην αλληλεπίδραση ανθρώπου-υπολογιστή έχει βαθιές ρίζες. Παλαιότερες εργασίες, όπως των Virvou και Nakamura (2008), έθεσαν τις βάσεις για την ενσωμάτωση της τεχνητής νοημοσύνης σε εφαρμογές λογισμικού. Η έρευνα συνεχίστηκε με τη μελέτη των Tsihrintzis και Virvou (2010) που ανέλυσε τις προκλήσεις στην ανάπτυξη πολυμεσικών υπηρεσιών σε ευφυή περιβάλλοντα.

Η πρόοδος αυτή έχει οδηγήσει σε νέες ερευνητικές κατευθύνσεις, όπως αναδεικνύεται από τη συλλογή άρθρων των Tsihrintzis, Toro, Rios, Howlett και Jain (2023) στο πλαίσιο του **27th KES International Conference**. Αυτές οι έρευνες εξετάζουν τις τελευταίες τάσεις και προκλήσεις στην ΤΝ και την αλληλεπίδραση ανθρώπου-υπολογιστή.

Μηχανική Λογισμικού και Έξυπνες Εφαρμογές

Η ΤΝ έχει μετασχηματίσει τη μηχανική λογισμικού, επιτρέποντας τη δημιουργία προσαρμοστικών και έξυπνων εφαρμογών. Οι Virvou κ.ά. (2022α) εισάγουν νέες μεθόδους

ανάπτυξης λογισμικού, εστιάζοντας στην ενσωμάτωση τεχνικών ΤΝ για την ενίσχυση της λειτουργικότητας των εφαρμογών.

Οι Tsihrintzis κ.ά. (2022) εξετάζουν τις τελευταίες εξελίξεις στη μηχανική μάθηση και τη βαθιά μάθηση, ενώ οι Hatzilygeroudis κ.ά. (2023) προτείνουν την ενσωμάτωση διαφορετικών παραδειγμάτων μηχανικής μάθησης, όπως τα νευρωνικά δίκτυα και οι γενετικοί αλγόριθμοι, για τη βελτίωση των διαδραστικών εφαρμογών λογισμικού.

Συναισθηματική Εξατομίκευση και Προσαρμοστική Αλληλεπίδραση

Ένας αναδυόμενος τομέας είναι η συναισθηματική εξατομίκευση, όπου η ΤΝ χρησιμοποιείται για την προσαρμογή της εμπειρίας του χρήστη με βάση συναισθηματικές αντιδράσεις. Οι Tsihrintzis κ.ά. (2008) και Stathopoulou κ.ά. (2009) εξετάζουν τεχνικές αναγνώρισης συναισθημάτων μέσω πληκτρολογίου και προσώπου, ενώ οι Katsionis και Virvou (2004) εισάγουν ένα γνωστικό μοντέλο συναισθηματικής αλληλεπίδρασης στα εκπαιδευτικά παιχνίδια εικονικής πραγματικότητας.

Η Virvou (2022, 2023) επισημαίνει πώς αυτές οι εξελίξεις επηρεάζουν τις εμπειρίες χρηστών, οδηγώντας σε εφαρμογές που βασίζονται σε αντικειμενοστρεφή μοντέλα εξατομίκευσης, όπως μελετήθηκε από τους Alepis και Virvou (2014).

Συλλογισμός και Μοντελοποίηση Χρηστών

Η μοντελοποίηση χρηστών είναι κρίσιμη για την προσαρμογή των εφαρμογών στις ατομικές ανάγκες. Η Virvou (1999) ερευνά τη χρήση της ΤΝ για την κατανόηση και τη διόρθωση λαθών χρήστη, ενώ οι Tsiriga και Virvou (2003) υπογραμμίζουν τη σημασία της προσαρμογής στρατηγικών καθοδήγησης με βάση τη συμπεριφορά του χρήστη.

Στην εκπαίδευση, η Virvou (2018) εξετάζει τη χρήση της ΤΝ σε εξατομικευμένα εκπαιδευτικά περιβάλλοντα, ενώ οι Chrysafiadi και Virvou (2013, 2022) αναπτύσσουν προσαρμοστικά διαγωνίσματα και μεθόδους αξιολόγησης με χρήση ασαφούς λογικής.

Εξατομικευμένες Εφαρμογές και Ηθικές Προκλήσεις

Η εξατομίκευση επεκτείνεται σε πολλούς τομείς, από την υγεία έως τον τουρισμό. Οι Pavlakis, Alepis και Virvou (2012) παρουσίασαν μια έξυπνη εφαρμογή υποστήριξης ηλικιωμένων, ενώ οι Kontogianni κ.ά. (2024) εξετάζουν τον ρόλο της ΤΝ στον έξυπνο τουρισμό. Ωστόσο, ζητήματα προστασίας δεδομένων και ασφάλειας παραμένουν κρίσιμα, όπως δείχνουν οι Politou κ.ά. (2022) και Mougiakou & Virvou (2017).

Η ηθική χρήση της ΤΝ απαιτεί διαφάνεια και επεξηγησιμότητα, όπως προτείνουν οι Panagoulias, Virvou και Tsihrintzis (2024), ενώ η ενσωμάτωση ασαφούς λογικής για την εξατομίκευση κρίσιμων ειδοποιήσεων αναλύεται από τους Alonistioti κ.ά. (2023).

Μεγάλα Μοντέλα Γλώσσας και Εφαρμογές AI

Τα Μεγάλα Μοντέλα Γλώσσας (LLMs), όπως το ChatGPT, έχουν επηρεάσει σημαντικά την εκπαίδευση και την ανάπτυξη λογισμικού. Οι Virvou και Tsihrintzis (2023a) προτείνουν ένα

μοντέλο αξιοπιστίας για την ανάπτυξη λογισμικού με ΤΝ, ενώ οι Virvou και Tsihrintzis (2023b, 2023c) διερευνούν τη χρήση των LLMs στην εκπαίδευση και προτείνουν ένα ολιστικό πλαίσιο αξιολόγησης της αποτελεσματικότητάς τους.

Συνολικά, η τεχνητή νοημοσύνη διαμορφώνει το μέλλον της αλληλεπίδρασης ανθρώπου-υπολογιστή, με σημαντικές προκλήσεις αλλά και ευκαιρίες για καινοτομία και εξατομίκευση.

Οι πρόσφατες έρευνες στον τομέα της τεχνητής νοημοσύνης και των ευφυών συστημάτων διαχείρισης εργασιών επικεντρώνονται στην ενίσχυση της διαφάνειας και της επεξηγησιμότητας των αποφάσεων των αλγορίθμων. Σύμφωνα με τους Panagoulias et al. (2023), η εξηγησιμότητα της τεχνητής νοημοσύνης είναι κρίσιμη για την αποδοχή των αυτοματοποιημένων συστημάτων διαχείρισης εργασιών και την αποτελεσματική ενσωμάτωση της μηχανικής μάθησης στη διαδικασία λήψης αποφάσεων.

Τέλος, οι Papadimitriou et al. (2019) προτείνουν ένα σύστημα προσαρμοστικών σεναρίων που βασίζεται σε ασαφή λογική, το οποίο μπορεί να εφαρμοστεί σε περιβάλλοντα διαχείρισης εργασιών για τη βελτίωση της αλληλεπίδρασης με τον χρήστη και την ευελιξία των συστημάτων.

Οι σύγχρονες εφαρμογές διαχείρισης εργασιών στοχεύουν στη βελτίωση της διαδραστικότητας και της εξατομίκευσης μέσω της χρήσης τεχνικών τεχνητής νοημοσύνης. Οι Katsionis & Virvou (2008) αναλύουν τη χρήση διαδικτυακών υπηρεσιών σε εξατομικευμένες εφαρμογές ηλεκτρονικής μάθησης, προτείνοντας μεθόδους που μπορούν να προσαρμοστούν και στη διαχείριση εργασιών.

Επιπλέον, οι Stathopoulou et al. (2010) μελετούν τη δυνατότητα ενίσχυσης των συστημάτων αναγνώρισης συναισθημάτων με πληροφορίες από μοτίβα πληκτρολόγησης, προσφέροντας έναν νέο τρόπο προσαρμογής των στρατηγικών διαχείρισης εργασιών με βάση την ψυχολογική κατάσταση του χρήστη.

Η εφαρμογή μοντέλων τεχνητής νοημοσύνης στην ανάλυση δεδομένων και στη λήψη αποφάσεων αποτελεί βασική τάση στον τομέα της διαχείρισης εργασιών. Οι Panagoulias et al. (2021) προτείνουν τη χρήση βιοδεικτών διατροφής και μηχανικής μάθησης για τη δημιουργία εξατομικευμένων προγραμμάτων, μια προσέγγιση που μπορεί να προσαρμοστεί και στη διαχείριση εργασιών με βάση τις ανάγκες των χρηστών.

Οι Virvou & Kabassi (2002) αναλύουν ένα πολυπρακτορικό περιβάλλον μάθησης, το οποίο μπορεί να προσαρμοστεί σε συστήματα διαχείρισης εργασιών για τη βελτίωση της αλληλεπίδρασης και της προσαρμογής στις ανάγκες των χρηστών.

Τέλος, η τεχνητή νοημοσύνη και η εξατομίκευση συνεχίζουν να παίζουν κρίσιμο ρόλο στη βελτίωση της εμπειρίας χρήστη. Σύμφωνα με τη Virvou (2018), η ενσωμάτωση έξυπνων πρακτόρων και προσαρμοστικών χαρακτηριστικών στις εφαρμογές μπορεί να οδηγήσει σε πιο ανθρώπινη και φυσική αλληλεπίδραση με τα συστήματα διαχείρισης εργασιών.

ΚΕΦΑΛΑΙΟ 2. ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ

Στο κεφάλαιο αυτό, περιγράφεται λεπτομερώς η διαδικασία ανάλυσης και σχεδιασμού της εφαρμογής, η οποία αποτελεί το θεμέλιο για την ανάπτυξη της. Η προσέγγιση ακολουθεί τις αρχές του RUP, με έμφαση στις απαιτήσεις των χρηστών και τη βελτιστοποίηση της λειτουργικότητας του συστήματος.

Αρχικά, εξετάζονται οι λειτουργικές και μη λειτουργικές απαιτήσεις της εφαρμογής, ενώ στη συνέχεια γίνεται παρουσίαση των βασικών σχεδιαστικών αποφάσεων. Επιπλέον, περιγράφεται η υλοποίηση του γραφικού περιβάλλοντος (UI), συμπεριλαμβανομένων των παραμέτρων χρήσης του ημερολογίου και της διαχείρισης εργασιών. Ιδιαίτερη έμφαση δίνεται στην ενσωμάτωση των τριών αλγορίθμων χρονολόγησης: SJN, LLF, και RR, οι οποίοι θα υλοποιηθούν πλήρως και θα τεκμηριωθούν.

Η ανάλυση συνοδεύεται από UML διαγράμματα που περιγράφουν τη λειτουργική και δομική αρχιτεκτονική της εφαρμογής, ενώ γίνεται αναφορά σε πρότυπες μεθοδολογίες σχεδιασμού και τις απαραίτητες τεχνολογίες, όπως JavaFX και Gradle, που χρησιμοποιούνται για την υλοποίηση. Με αυτό τον τρόπο, το κεφάλαιο παρέχει τη βάση για την πλήρη κατανόηση του σχεδιασμού και της τεχνικής υλοποίησης της εφαρμογής.

2.1 Απαιτήσεις και Λειτουργικότητα της Εφαρμογής

Η ανάλυση των απαιτήσεων αποτελεί το πρώτο και πιο κρίσιμο βήμα για την ανάπτυξη της εφαρμογής. Η εφαρμογή έχει ως κύρια λειτουργία τη διαχείριση εργασιών μέσω ενός ευχάριστου και λειτουργικού γραφικού περιβάλλοντος. Οι χρήστες θα μπορούν να δημιουργούν, να επεξεργάζονται και να διαγράφουν εργασίες, ενώ θα έχουν τη δυνατότητα να τις προγραμματίζουν σε ημερήσια βάση μέσω ενός δυναμικού ημερολογίου. Επιπλέον, η επιλογή ενός από τους τρεις διαθέσιμους αλγόριθμους χρονολόγησης θα επιτρέπει την αυτόματη ταξινόμηση των εργασιών με βάση την προτεραιότητα, τη διάρκεια και την προθεσμία τους.

Για να επιτευχθεί η βέλτιστη εμπειρία χρήστη, η εφαρμογή πρέπει να είναι εύχρηστη, αποδοτική και αξιόπιστη. Χρησιμοποιείται το JavaFX για τη δημιουργία ενός σύγχρονου και εύχρηστου γραφικού περιβάλλοντος, ενώ το Gradle εξασφαλίζει τη σωστή διαχείριση εξαρτήσεων και τη σταθερότητα του project. Η εφαρμογή πρέπει να είναι επίσης επεκτάσιμη, ώστε να μπορούν να προστεθούν μελλοντικές λειτουργίες, και συμβατή με διαφορετικά λειτουργικά συστήματα. Ιδιαίτερη προσοχή δίνεται επίσης στη διαχείριση χρηστών, με υποστήριξη για πολλαπλά προφίλ και δυνατότητα σύνδεσης και αποσύνδεσης.

Η εφαρμογή στοχεύει στη βελτιστοποίηση της οργάνωσης εργασιών και στην ενίσχυση της παραγωγικότητας των χρηστών. Είναι ιδανική για φοιτητές, επαγγελματίες και ομάδες εργασίας που επιθυμούν έναν εύκολο και πρακτικό τρόπο διαχείρισης των καθηκόντων τους. Η

δυνατότητα επιλογής μεταξύ διαφορετικών αλγορίθμων χρονολόγησης προσφέρει στους χρήστες τη δυνατότητα να προσαρμόσουν τη χρήση της εφαρμογής στις προσωπικές τους ανάγκες. Επιπλέον, η προσεγμένη διεπαφή και οι γρήγορες αποκρίσεις καθιστούν την εφαρμογή κατάλληλη τόσο για αρχάριους όσο και για προχωρημένους χρήστες.

Η εφαρμογή παρέχει στους χρήστες ένα πλήρες σύνολο δυνατοτήτων για την αποτελεσματική διαχείριση των εργασιών τους. Μέσα από το ημερολόγιο, οι χρήστες μπορούν να προγραμματίζουν τις καθημερινές τους δραστηριότητες με βάση τον χρόνο και τις προτεραιότητες. Παράλληλα, παρέχεται η δυνατότητα δημιουργίας, επεξεργασίας και διαγραφής εργασιών μέσω ειδικά σχεδιασμένων παραθύρων διαλόγου. Οι χρήστες μπορούν επίσης να επιλέξουν τον αλγόριθμο ταξινόμησης που επιθυμούν, επιτρέποντας την προσαρμογή της εφαρμογής στις δικές τους ανάγκες και προτιμήσεις.

Η δυνατότητα δημιουργίας και διαχείρισης προφίλ χρήστη είναι μια από τις βασικές λειτουργίες της εφαρμογής. Οι χρήστες μπορούν να συνδεθούν με προσωπικό λογαριασμό, ο οποίος διασφαλίζει ότι οι ρυθμίσεις και τα δεδομένα τους είναι μοναδικά και ασφαλή. Όταν ένας χρήστης συνδεθεί, το σύστημα προσαρμόζεται στις δικές του απαιτήσεις, εμφανίζοντας μόνο τις εργασίες που του αφορούν. Επιπλέον, οι μη εξουσιοδοτημένοι χρήστες ξεκινούν ως επισκέπτες με περιορισμένη πρόσβαση στις λειτουργίες της εφαρμογής.

Ένας από τους σημαντικότερους στόχους της εφαρμογής είναι η αυτόματη ταξινόμηση των εργασιών μέσω των τριών αλγορίθμων χρονολόγησης. Ο κάθε αλγόριθμος εξυπηρετεί διαφορετικές ανάγκες προγραμματισμού, προσφέροντας ευελιξία στους χρήστες. Η ενσωμάτωση των αλγορίθμων αυτών γίνεται με τέτοιο τρόπο ώστε κάθε αλλαγή στις εργασίες ή στον επιλεγμένο αλγόριθμο να προκαλεί αυτόματη επανεπεξεργασία των δεδομένων και αναδιάταξη του ημερολογίου.

Η εφαρμογή αναπτύχθηκε χρησιμοποιώντας προηγμένα εργαλεία και τεχνολογίες, όπως το JavaFX για την κατασκευή του γραφικού περιβάλλοντος χρήστη και το Gradle για τη διαχείριση των εξαρτήσεων και την αυτοματοποίηση της διαδικασίας build. Η επιλογή αυτών των εργαλείων εξασφαλίζει τη σταθερότητα, την επεκτασιμότητα και τη συμβατότητα της εφαρμογής με πολλαπλά λειτουργικά συστήματα. Παράλληλα, χρησιμοποιούνται σύγχρονες τεχνικές προγραμματισμού, όπως η χρήση ξεχωριστών κλάσεων για τη διαχείριση των χρηστών και των δεδομένων, ώστε να διασφαλιστεί η καθαρότητα και η επαναχρησιμοποίηση του κώδικα.

Η ανάπτυξη της εφαρμογής βασίστηκε στο πλαίσιο της διαδικασίας RUP, το οποίο εξασφαλίζει ότι κάθε στάδιο υλοποίησης ακολουθεί μια σαφή και οργανωμένη μεθοδολογία. Από την ανάλυση των απαιτήσεων και το σχεδιασμό, έως την υλοποίηση και τη δοκιμή, η εφαρμογή αναπτύχθηκε με βάση τις αρχές της επανάληψης και της σταδιακής βελτίωσης. Οι UML απεικονίσεις που συνοδεύουν την ανάπτυξη παρέχουν μια καθαρή οπτική αναπαράσταση της αρχιτεκτονικής της εφαρμογής, διασφαλίζοντας τη σωστή επικοινωνία μεταξύ των τμημάτων και την ευκολία στην κατανόηση της συνολικής λειτουργίας.

2.2 Σχεδιαστική Προσέγγιση και Αρχιτεκτονική της Εφαρμογής

Η ανάπτυξη της εφαρμογής βασίστηκε στο μοτίβο σχεδιασμού Model-View-Controller (MVC), το οποίο διασφαλίζει τη σαφή διαχωρισμό των λειτουργιών της εφαρμογής. Στη συγκεκριμένη αρχιτεκτονική, το **Model** διαχειρίζεται τα δεδομένα και τη λογική της εφαρμογής, το **View** είναι υπεύθυνο για την παρουσίαση του περιβάλλοντος χρήστη, ενώ το **Controller** λειτουργεί ως γέφυρα μεταξύ των δύο. Αυτή η προσέγγιση εξασφαλίζει τη δυνατότητα μελλοντικής επεκτασιμότητας και συντηρησιμότητας, καθώς οι αλλαγές σε ένα επίπεδο δεν επηρεάζουν τα υπόλοιπα.

Η εφαρμογή οργανώνεται σε ξεχωριστά πακέτα που αντικατοπτρίζουν τις βασικές λειτουργικές περιοχές. Το πακέτο *utils* περιλαμβάνει τις κλάσεις διαχείρισης χρηστών και δεδομένων, όπως το *ProfileManager* και το *UserManager*, που διαχειρίζονται τη δημιουργία και την αυθεντικοποίηση προφίλ. Το πακέτο *components* περιέχει επαναχρησιμοποιούμενα γραφικά στοιχεία, όπως το *MenuBarComponent* και το *StatusBar*. Το κύριο πακέτο της εφαρμογής φιλοξενεί τη βασική κλάση *TaskManager* που συντονίζει τη λειτουργικότητα. Αυτή η διάρθρωση προάγει τη modular ανάπτυξη και επιτρέπει την εύκολη προσθήκη νέων χαρακτηριστικών.

Το γραφικό περιβάλλον της εφαρμογής σχεδιάστηκε για να είναι φιλικό προς τον χρήστη και λειτουργικό. Η κύρια διάταξη βασίζεται στο *BorderPane*, το οποίο οργανώνει τα στοιχεία σε περιοχές: το ημερολόγιο βρίσκεται στο κεντρικό τμήμα, η λίστα εργασιών στα δεξιά, ενώ η γραμμή κατάστασης και η μπάρα πλοήγησης βρίσκονται στο κάτω και στο πάνω μέρος αντίστοιχα. Ο σχεδιασμός εστιάζει στην απλότητα και τη χρηστικότητα, επιτρέποντας στον χρήστη να αλληλεπιδρά με τα δεδομένα του μέσω κουμπιών και παραθύρων διαλόγου, όπως φόρμες για την προσθήκη ή επεξεργασία εργασιών.

Η αλληλεπίδραση του χρήστη με την εφαρμογή βασίζεται στη χρήση ενός έξυπνου και ευέλικτου γραφικού περιβάλλοντος. Μέσω του ημερολογίου, ο χρήστης μπορεί να επιλέξει μια συγκεκριμένη ημέρα και να προβάλλει τις αντίστοιχες εργασίες, ενώ η λίστα εργασιών παρέχει εργαλεία για την προσθήκη, διαγραφή ή επεξεργασία τους. Όλες οι ενέργειες του χρήστη επηρεάζουν δυναμικά την εφαρμογή, με την ανανέωση του UI να πραγματοποιείται αυτόματα μετά από κάθε αλλαγή. Επιπλέον, το σύστημα παρέχει επιλογές μέσω των μενού για τη διαχείριση προφίλ χρηστών και την εφαρμογή αλγορίθμων ταξινόμησης.

Η διαχείριση δεδομένων βασίζεται στη χρήση JSON αρχείων για την αποθήκευση των προφίλ χρηστών και των πληροφοριών σχετικά με τις εργασίες. Το *ProfileManager* αναλαμβάνει τη δημιουργία, αποθήκευση και επαλήθευση των προφίλ χρηστών, ενώ το *UserManager* διαχειρίζεται τη σύνδεση και αποσύνδεση, διατηρώντας το τρέχον προφίλ στη μνήμη. Αυτή η διάκριση επιτρέπει την αποτελεσματική διαχείριση των δεδομένων και την εξασφάλιση ότι η

εφαρμογή μπορεί να προσαρμοστεί δυναμικά στις ανάγκες κάθε χρήστη, ενώ ταυτόχρονα διατηρεί ασφαλή τα προσωπικά δεδομένα.

Η εφαρμογή προσφέρει τρεις αλγορίθμους χρονολόγησης, οι οποίοι ενσωματώνονται μέσω μιας modular αρχιτεκτονικής. Ο κάθε αλγόριθμος (SJN, LLF, και RR) υλοποιείται σε ξεχωριστή κλάση που ακολουθεί το ίδιο interface, διευκολύνοντας έτσι την επιλογή και εφαρμογή τους από τον χρήστη. Όταν ο χρήστης επιλέγει έναν αλγόριθμο, το σύστημα επαναταξινομεί αυτόματα τις εργασίες και ενημερώνει το UI. Αυτή η δυναμική προσέγγιση επιτρέπει στους χρήστες να προσαρμόσουν τη λειτουργικότητα της εφαρμογής στις δικές τους ανάγκες, εξασφαλίζοντας παράλληλα ακρίβεια και ταχύτητα.

Η διαχείριση εργασιών υλοποιείται μέσω ενός ολοκληρωμένου συστήματος επεξεργασίας και χειρισμού δεδομένων. Οι χρήστες έχουν τη δυνατότητα να δημιουργήσουν νέες εργασίες, να επεξεργαστούν υπάρχουσες ή να διαγράψουν εργασίες που δεν χρειάζονται πλέον. Αυτές οι λειτουργίες παρέχονται μέσω διαδραστικών παραθύρων διαλόγου, στα οποία οι χρήστες μπορούν να εισάγουν πληροφορίες όπως τίτλο, περιγραφή, προθεσμία, προτεραιότητα και διάρκεια. Η ενημέρωση του συστήματος γίνεται σε πραγματικό χρόνο, με τις αλλαγές να αντικατοπτρίζονται άμεσα στο ημερολόγιο και τη λίστα εργασιών.

Τα UML διαγράμματα αποτελούν βασικό στοιχείο της ανάλυσης και σχεδιασμού, παρέχοντας μια σαφή οπτική αναπαράσταση της αρχιτεκτονικής και της ροής της εφαρμογής. Τα Use Case Diagrams καταγράφουν τις βασικές λειτουργίες και τις αλληλεπιδράσεις των χρηστών με το σύστημα, ενώ τα Class Diagrams περιγράφουν τη δομή των κλάσεων και τις σχέσεις μεταξύ τους. Επιπλέον, Sequence Diagrams απεικονίζουν τη ροή δεδομένων και εντολών, προσφέροντας μια ξεκάθαρη εικόνα του τρόπου με τον οποίο τα διαφορετικά μέρη της εφαρμογής συνεργάζονται. Αυτή η ανάλυση εξασφαλίζει την κατανόηση και επικοινωνία μεταξύ των μελών της ομάδας ανάπτυξης.

Η επιλεγμένη σχεδιαστική προσέγγιση βασίζεται σε αρχές επεκτασιμότητας, επαναχρησιμοποίησης κώδικα και αποδοτικότητας, καθιστώντας την εφαρμογή εύκολη στη συντήρηση και τη μελλοντική αναβάθμιση. Ο διαχωρισμός των ευθυνών σε διακριτές κλάσεις και πακέτα επιτρέπει την αποτελεσματική οργάνωση του κώδικα και μειώνει την πολυπλοκότητα. Επιπλέον, η ενσωμάτωση των αλγορίθμων σε modular αρχιτεκτονική διασφαλίζει τη λειτουργικότητα τους χωρίς να επηρεάζουν άλλες πτυχές της εφαρμογής. Συνολικά, το προσεκτικά σχεδιασμένο σύστημα συνδυάζει ευκολία στη χρήση και τεχνική αρτιότητα, καλύπτοντας τις ανάγκες των χρηστών και του project.

2.3 Σχεδιασμός και Διαχείριση Δεδομένων

Η διαχείριση δεδομένων είναι κρίσιμη για την ομαλή λειτουργία της εφαρμογής, καθώς απαιτείται αποθήκευση των πληροφοριών των χρηστών και των εργασιών με τρόπο αξιόπιστο

και αποδοτικό. Ο σχεδιασμός των δομών δεδομένων εστιάζει στη διατήρηση της ακεραιότητας και της ασφάλειας, ενώ παράλληλα εξασφαλίζει την ταχεία πρόσβαση και επεξεργασία. Η εφαρμογή χρησιμοποιεί αρχεία JSON ως βασικό μέσο αποθήκευσης, επιτρέποντας την εύκολη ανάγνωση και εγγραφή δεδομένων χωρίς την ανάγκη σύνθετων συστημάτων βάσης δεδομένων.

Τα δεδομένα της εφαρμογής οργανώνονται σε δύο βασικές κατηγορίες: προφίλ χρηστών και εργασίες. Για τα προφίλ χρηστών, το JSON αρχείο περιέχει τα πεδία "username" και "password", διασφαλίζοντας μοναδικότητα μέσω της χρήσης του ονόματος χρήστη ως βασικού κλειδιού. Οι εργασίες αποθηκεύονται με πληροφορίες όπως τίτλος, περιγραφή, προθεσμία, προτεραιότητα και διάρκεια. Αυτή η δομή επιτρέπει την εύκολη συσχέτιση των εργασιών με τον χρήστη που τις έχει δημιουργήσει, διατηρώντας τα δεδομένα οργανωμένα και ευανάγνωστα.

Κατά την ανάγνωση και εγγραφή στα αρχεία JSON, γίνεται χρήση μηχανισμών διαχείρισης σφαλμάτων για να διασφαλιστεί η ακεραιότητα των αποθηκευμένων δεδομένων, ακόμη και σε περίπτωση απρόβλεπτων συνθηκών.

Ο σχεδιασμός της αποθήκευσης δεδομένων έχει λάβει υπόψη την κλιμακωσιμότητα της εφαρμογής. Τα αρχεία JSON είναι κατάλληλα για την αποθήκευση δεδομένων σε μικρή κλίμακα, αλλά παρέχεται και η δυνατότητα μελλοντικής μετάβασης σε μία πιο ολοκληρωμένη βάση δεδομένων, όπως MySQL ή PostgreSQL, εάν αυξηθούν οι απαιτήσεις σε όγκο ή ταχύτητα. Η αρχιτεκτονική της εφαρμογής είναι σχεδιασμένη με τέτοιο τρόπο ώστε η αλλαγή του συστήματος αποθήκευσης να απαιτεί ελάχιστες προσαρμογές.

Η διαδικασία εισαγωγής και ανάκτησης δεδομένων από τα JSON αρχεία είναι βασισμένη στη χρήση της βιβλιοθήκης Gson, η οποία παρέχει δυνατότητες για γρήγορη μετατροπή αντικειμένων Java σε JSON και αντίστροφα. Για την αποθήκευση νέων δεδομένων, κάθε αντικείμενο μετατρέπεται σε JSON string και αποθηκεύεται στο αντίστοιχο αρχείο. Κατά την εκκίνηση της εφαρμογής, τα αρχεία διαβάζονται και τα δεδομένα φορτώνονται στη μνήμη, ώστε να είναι άμεσα διαθέσιμα για χρήση. Αυτή η προσέγγιση ελαχιστοποιεί την καθυστέρηση κατά τη διάρκεια της λειτουργίας.

Η σύνδεση των δεδομένων χρήστη με τις εργασίες επιτυγχάνεται μέσω της χρήσης μοναδικών αναγνωριστικών για κάθε χρήστη. Κάθε εργασία περιλαμβάνει ένα πεδίο που αναφέρεται στο αντίστοιχο προφίλ χρήστη, εξασφαλίζοντας τη σωστή αντιστοίχιση. Αυτή η διάρθρωση επιτρέπει την ταυτόχρονη διαχείριση πολλών χρηστών, χωρίς να επηρεάζεται η απόδοση ή η πολυπλοκότητα του συστήματος. Η ενσωμάτωση αυτής της σχέσης στο JSON αρχείο παρέχει ένα σαφές και οργανωμένο μοντέλο δεδομένων που διευκολύνει τη διαχείριση και την επεξεργασία.

Ο σχεδιασμός του συστήματος αποθήκευσης δεδομένων έχει λάβει υπόψη τις μελλοντικές απαιτήσεις της εφαρμογής. Τα JSON αρχεία παρέχουν ευελιξία για την αποθήκευση σύνθετων δεδομένων, ενώ μπορούν να μετατραπούν εύκολα σε συμβατά φορμάτ με συστήματα βάσεων δεδομένων. Επιπλέον, η λογική αποθήκευσης έχει υλοποιηθεί με τρόπο που υποστηρίζει την επέκταση, επιτρέποντας την προσθήκη νέων τύπων δεδομένων, όπως ετικέτες ή κατηγορίες

για τις εργασίες. Με τη χρήση κατάλληλων διεπαφών και εργαλείων διαχείρισης, η εφαρμογή είναι έτοιμη να ενσωματώσει επιπλέον δυνατότητες, διατηρώντας ταυτόχρονα την ευχρηστία και την αποδοτικότητα της.

2.4 UML Διαγράμματα

Η διαδικασία Rational Unified Process (RUP) αποτελεί μία από τις πιο διαδεδομένες μεθοδολογίες ανάπτυξης λογισμικού, προσφέροντας μια συστηματική προσέγγιση για την ανάλυση, το σχεδιασμό, την υλοποίηση και τη δοκιμή συστημάτων. Στο πλαίσιο αυτής της πτυχιακής εργασίας, η διαδικασία RUP χρησιμοποιήθηκε για την ανάπτυξη μιας εφαρμογής διαχείρισης εργασιών, ενσωματώνοντας διαγράμματα UML σε κάθε φάση της ανάπτυξης. Ο συνδυασμός RUP και UML εξασφάλισε τη σαφή περιγραφή των λειτουργιών, των δομών και των ροών της εφαρμογής, επιτρέποντας την οργανωμένη και αποδοτική πρόοδο του έργου.

Τα UML διαγράμματα αποτελούν αναπόσπαστο εργαλείο στη διαδικασία ανάπτυξης λογισμικού, καθώς προσφέρουν έναν οπτικό τρόπο αναπαράστασης του συστήματος. Από τα Διαγράμματα Περιπτώσεων Χρήσης, που αποτυπώνουν τις κύριες λειτουργίες και τις αλληλεπιδράσεις χρηστών, έως τα Διαγράμματα Καταστάσεων και Διανομής, που περιγράφουν τις δυναμικές ροές και την υποδομή του συστήματος, η χρήση τους διευκόλυνε την αποτύπωση των απαιτήσεων και την υλοποίηση της εφαρμογής. Επιπλέον, κάθε κατηγορία διαγραμμάτων συνέβαλε στην κατανόηση διαφορετικών πτυχών του συστήματος, εξυπηρετώντας συγκεκριμένους στόχους σε κάθε φάση του RUP.

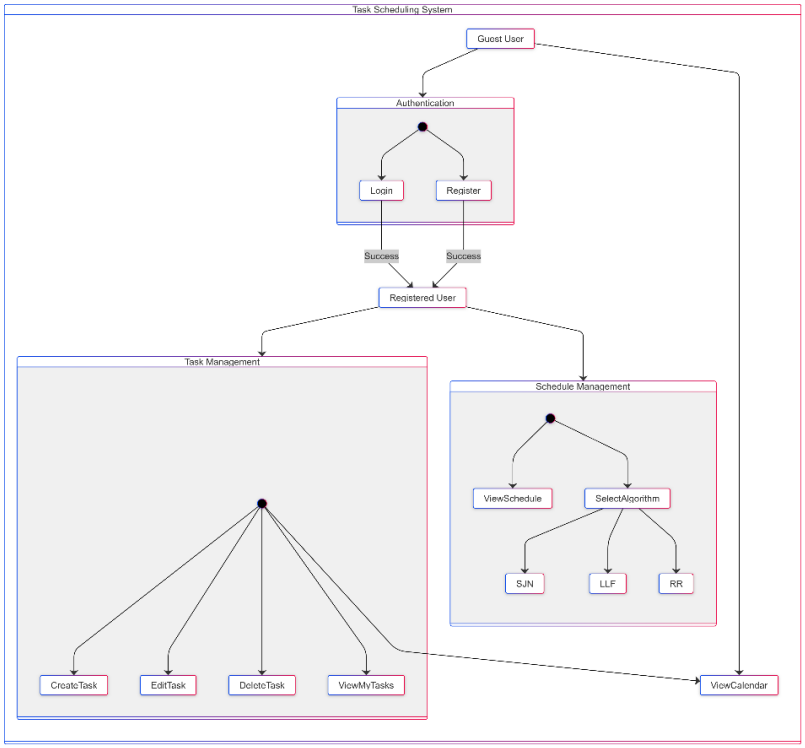
Η ενότητα αυτή είναι οργανωμένη σύμφωνα με τη διαδικασία RUP, καλύπτοντας τις φάσεις της Έναρξης, της Εκπόνησης Μελέτης και της Κατασκευής. Σε κάθε φάση παρουσιάζονται τα διαγράμματα που δημιουργήθηκαν, μαζί με την ανάλυση του τρόπου με τον οποίο συνδέονται με την ανάπτυξη της εφαρμογής. Έμφαση δίνεται στη χρήση των UML διαγραμμάτων ως εργαλείων για την οπτική και συστηματική καταγραφή της λειτουργικότητας και της δομής της εφαρμογής, εξασφαλίζοντας την πληρότητα και τη συνέπεια του έργου.

Τέλος, η ενότητα αναδεικνύει τη σημασία του συνδυασμού της διαδικασίας RUP με τα UML διαγράμματα για την επιτυχή ολοκλήρωση του έργου. Μέσω αυτής της συνδυαστικής προσέγγισης, η εφαρμογή αναπτύχθηκε με τρόπο που διευκολύνει τη συντήρηση και την περαιτέρω επέκταση, ενώ ταυτόχρονα ανταποκρίνεται στις ανάγκες των χρηστών. Με αυτό τον τρόπο, αποδεικνύεται η αξία της οπτικοποίησης και της δομημένης ανάλυσης κατά τη διαδικασία ανάπτυξης λογισμικού.

Φάση Έναρξης (Inception)

Η φάση της έναρξης αποτελεί το πρώτο στάδιο στη διαδικασία ανάπτυξης του συστήματος και έχει ως στόχο τον καθορισμό των βασικών λειτουργιών και απαιτήσεων της

εφαρμογής. Στη συγκεκριμένη φάση, η ανάλυση επικεντρώνεται στη διαμόρφωση της κύριας αρχιτεκτονικής και στη διερεύνηση των βασικών σεναρίων χρήσης, ώστε να εξασφαλιστεί η πληρότητα και η σαφήνεια των απαιτήσεων. Τα δύο διαγράμματα που δημιουργήθηκαν κατά τη φάση αυτή – το διάγραμμα περιπτώσεων χρήσης (Εικόνα 1) και το διάγραμμα τάξεων (Εικόνα 2) – αποτυπώνουν τα βασικά στοιχεία του συστήματος και θέτουν τη βάση για την ανάπτυξη του έργου.



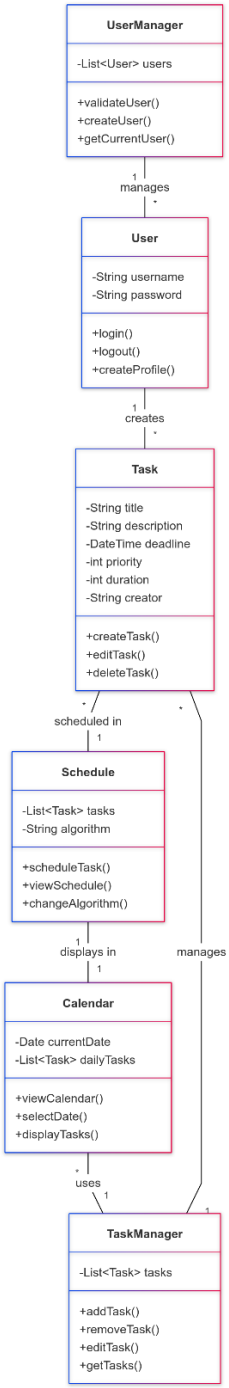
Εικόνα 1. Διάγραμμα Περιπτώσεων Χρήσης στη φάση της έναρξης.

Το διάγραμμα περιπτώσεων χρήσης απεικονίζει τις βασικές αλληλεπιδράσεις των χρηστών με την εφαρμογή. Στην Εικόνα 1, περιγράφονται οι κύριες ενέργειες του συστήματος, όπως η "Εγγραφή", η "Σύνδεση", και η "Διαχείριση Εργασιών". Ο Guest User μπορεί να πλοηγηθεί στο σύστημα και να εκτελέσει βασικές ενέργειες, όπως η προβολή εργασιών, ενώ ο Registered User έχει πρόσβαση σε προηγμένες λειτουργίες, όπως η δημιουργία, επεξεργασία και διαγραφή εργασιών. Η διαχείριση των εργασιών υποστηρίζεται από δυνατότητες προγραμματισμού, όπως η επιλογή αλγορίθμου χρονολόγησης, δίνοντας στους χρήστες τη δυνατότητα να προσαρμόσουν την εμπειρία τους.

Το διάγραμμα τάξεων, που παρουσιάζεται στην Εικόνα 2, περιγράφει τη δομή του συστήματος σε επίπεδο κλάσεων και αντικειμένων. Η κλάση *UserManager* διαχειρίζεται τους χρήστες και τις βασικές λειτουργίες αυθεντικοποίησης, όπως η επικύρωση και η δημιουργία προφίλ. Η κλάση *Task* περιλαμβάνει πληροφορίες σχετικά με τις εργασίες, όπως ο τίτλος, η περιγραφή, η προθεσμία, και η προτεραιότητα, ενώ η κλάση *TaskManager* οργανώνει και διαχειρίζεται τις εργασίες. Οι αλληλεπιδράσεις μεταξύ των τάξεων αποτυπώνουν τη βασική ροή

δεδομένων και λειτουργιών του συστήματος, όπως η αντιστοίχιση χρηστών με εργασίες μέσω του *UserManager*.

Η φάση της έναρξης θέτει τη βάση για τη σωστή ανάπτυξη του συστήματος, παρέχοντας σαφείς κατευθύνσεις μέσω των UML διαγραμμάτων. Το διάγραμμα περιπτώσεων χρήσης επισημαίνει τις κύριες λειτουργίες και τις σχέσεις των χρηστών με το σύστημα, ενώ το διάγραμμα τάξεων αποτυπώνει τη λογική αρχιτεκτονική και τις βασικές κλάσεις. Αυτά τα δύο διαγράμματα λειτουργούν συμπληρωματικά, προσφέροντας μια σφαιρική εικόνα για τη λειτουργικότητα και τη δομή του συστήματος, προετοιμάζοντας το έδαφος για τις επόμενες φάσεις της ανάπτυξης.



Εικόνα 2. Διάγραμμα Κλάσεων στη φάση της έναρξης.

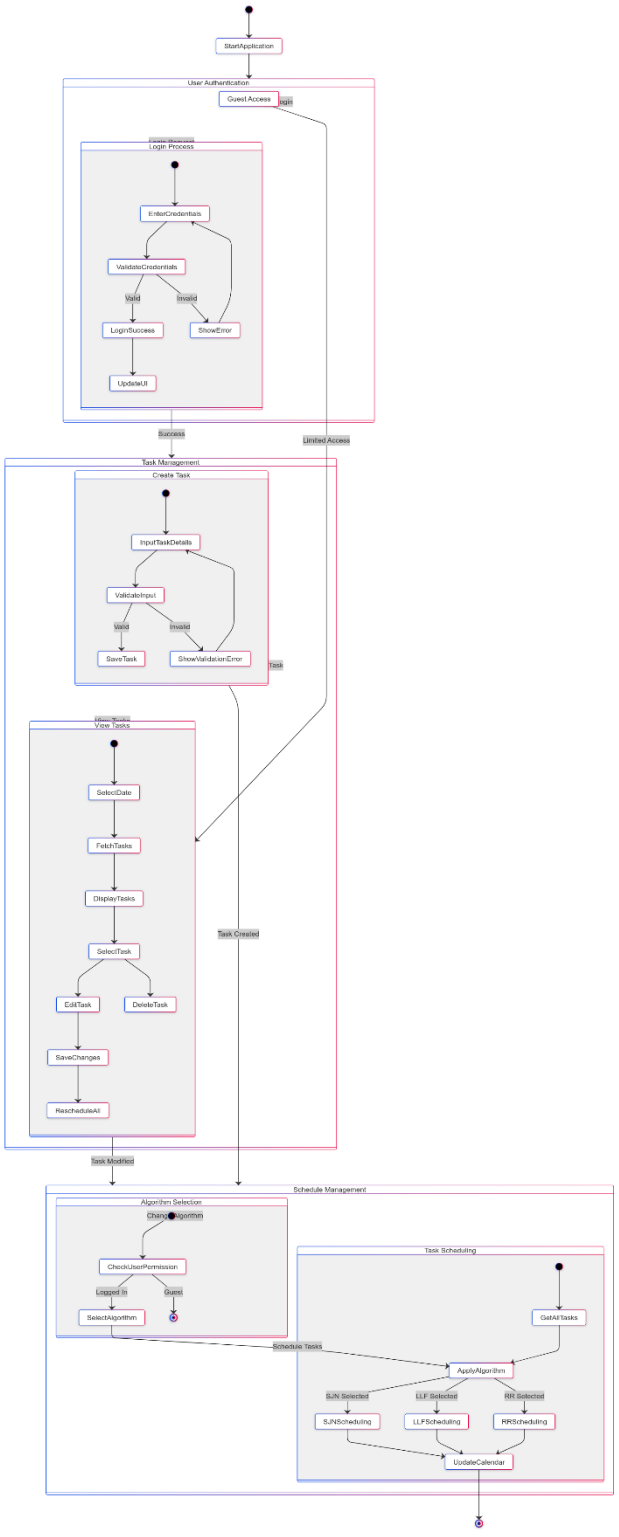
Φάση Εκπόνησης Μελέτης (Elaboration)

Η φάση της Εκπόνησης Μελέτης αποτελεί το επόμενο κρίσιμο βήμα μετά την Έναρξη, όπου το σύστημα αρχίζει να αποκτά πιο ξεκάθαρη και λεπτομερή δομή. Σε αυτό το στάδιο, επικεντρωνόμαστε στην ανάλυση των απαιτήσεων, στον εμπλουτισμό του σχεδιασμού, και στη λεπτομερή αποτύπωση των αλληλεπιδράσεων και της δυναμικής του συστήματος. Τα UML

διαγράμματα που δημιουργήθηκαν σε αυτή τη φάση, όπως τα διαγράμματα τάξεων, συνεργασίας, και καταστάσεων, συμβάλλουν στη βαθύτερη κατανόηση και στην αποδοτικότερη υλοποίηση της εφαρμογής.

Στη συνέχεια, θα αναλυθούν τα βασικά διαγράμματα που σχεδιάστηκαν, τα οποία ενισχύουν την πληρότητα και τη συνοχή του συστήματος, επιτρέποντας την ομαλή μετάβαση στη φάση της Κατασκευής.

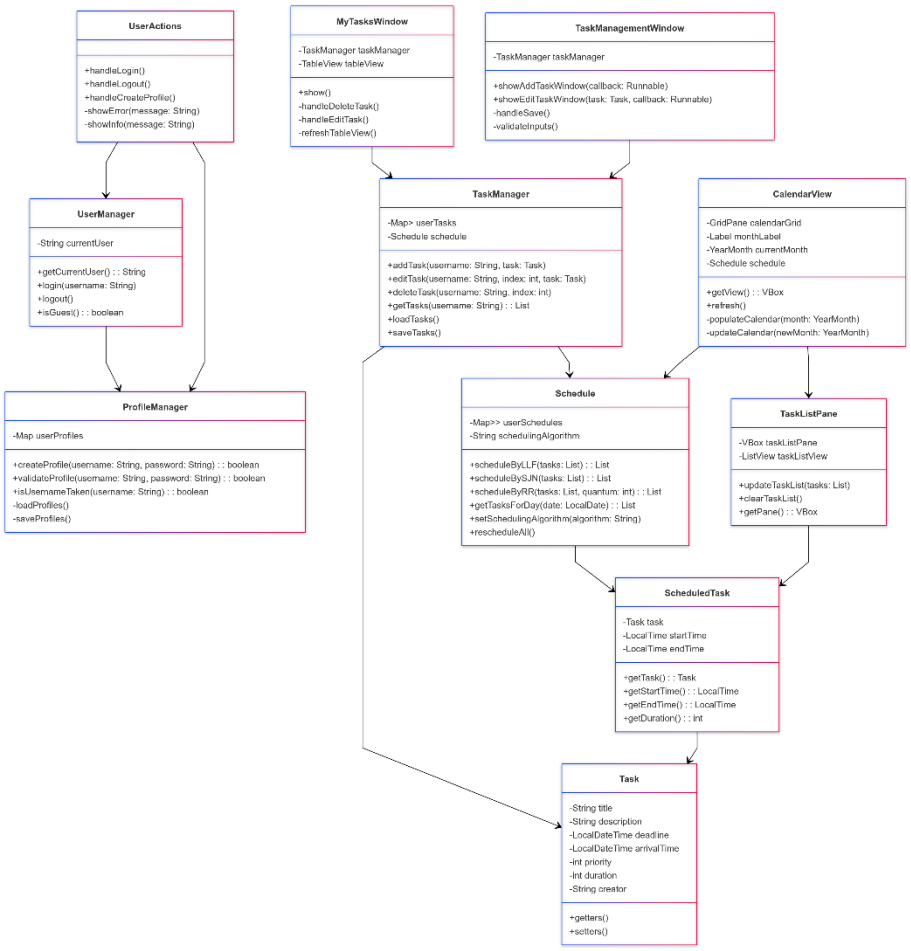
Το Διάγραμμα Δραστηριοτήτων χαρτογραφεί τις κρίσιμες λειτουργίες του συστήματος, από την ταυτοποίηση χρήστη έως τον χρονοπρογραμματισμό εργασιών. Η Εικόνα 3 αποτυπώνει τις διακλαδώσεις και τους ελέγχους εγκυρότητας που διασφαλίζουν την ορθότητα των δεδομένων εισόδου. Μέσα από τη διαδραστική ροή εργασιών, αναδεικνύεται η δυναμική σχέση χρήστη-συστήματος, με έμφαση στις επιλογές χρονοπρογραμματισμού και τη διαμόρφωση της τελικής εμπειρίας.



Εικόνα 3. Διάγραμμα Δραστηριοτήτων στη φάση της εκπόνησης μελέτης.

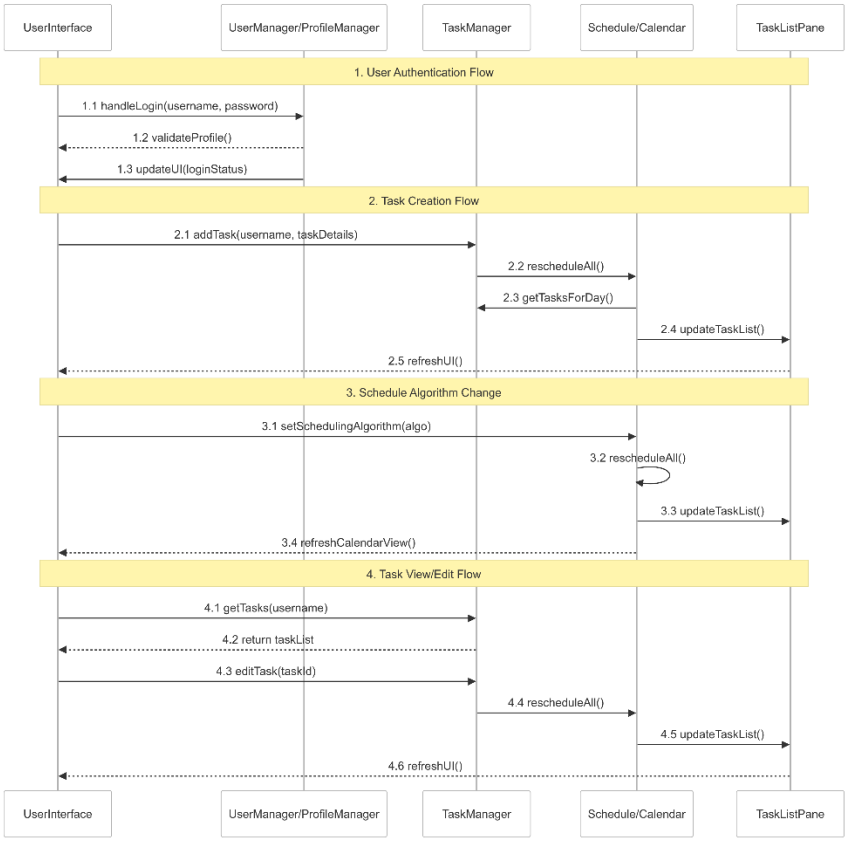
Για την υλοποίηση αυτών των λειτουργιών, η Εικόνα 4 παρουσιάζει την εξέλιξη του Διαγράμματος Τάξεων, εμπλουτίζοντας την αρχιτεκτονική του συστήματος με νέα δομικά στοιχεία. Κεντρικό ρόλο διαδραματίζουν δύο νέες κλάσεις: η Schedule, που ενορχηστρώνει τους αλγορίθμους χρονοπρογραμματισμού, και η TaskListPane, που επιμελείται τη γραφική διεπαφή

των εργασιών. Η αλληλεπίδραση μεταξύ του TaskManager και της Schedule αναδεικνύει τη συνέργεια των λειτουργικών μονάδων, διασφαλίζοντας την αποτελεσματική διαχείρισή τους.



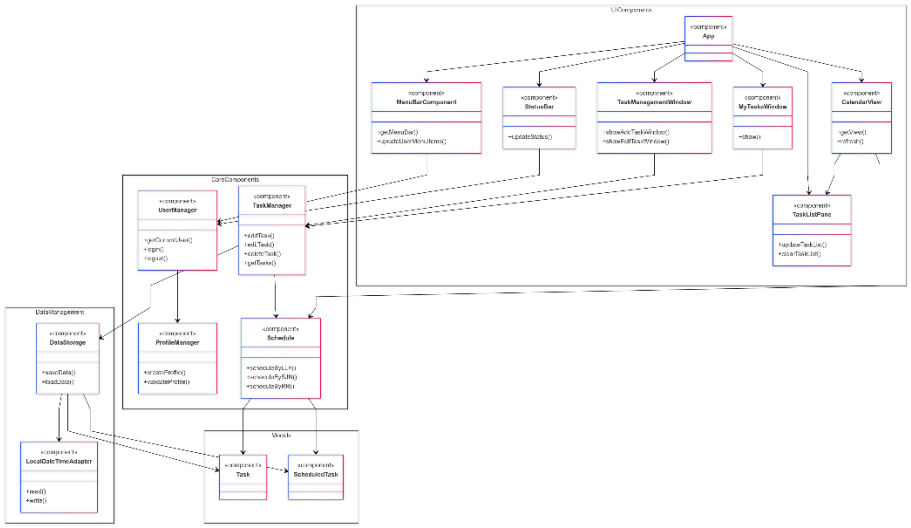
Εικόνα 4. Διάγραμμα Τάξεων στη φάση της εκπόνησης μελέτης.

Η πρακτική εφαρμογή αυτής της αρχιτεκτονικής αποτυπώνεται στην Εικόνα 5, όπου το Διάγραμμα Συνεργασίας προσφέρει μια δυναμική απεικόνιση των αλληλεπιδράσεων μεταξύ των συστατικών στοιχείων. Από την ταυτοποίηση χρήστη μέχρι την ενημέρωση του ημερολογίου μέσω του CalendarView, το διάγραμμα αποτυπώνει την αλυσίδα επικοινωνίας μεταξύ των κύριων μονάδων. Η αρμονική συνεργασία του τρίπτυχου UserManager, TaskManager και Schedule εξασφαλίζει την ομαλή ροή δεδομένων και την απρόσκοπτη εκτέλεση των λειτουργιών του συστήματος.



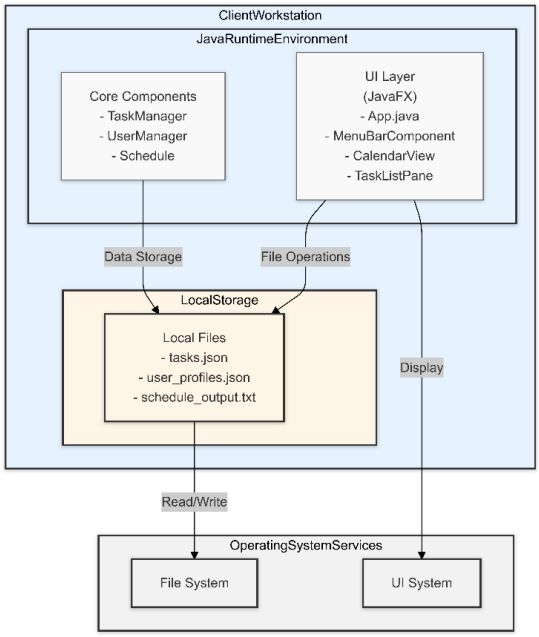
Εικόνα 5. Διάγραμμα Συνεργασίας στη φάση της εκπόνησης μελέτης.

Συμπληρώνοντας την αρχιτεκτονική απεικόνιση, το Διάγραμμα Εξαρτημάτων στην Εικόνα 6 οργανώνει το σύστημα σε διακριτές λειτουργικές μονάδες. Στον πυρήνα του σχεδιασμού βρίσκονται τα Core Components με τους διαχειριστές UserManager και TaskManager, ενώ η διεπαφή χρήστη υλοποιείται μέσω των UI Components όπως το TaskManagementWindow και το CalendarView. Παράλληλα, το επίπεδο διαχείρισης δεδομένων, με τις μονάδες DataStorage και ProfileManager, εξασφαλίζει την αξιόπιστη διαχείριση των πληροφοριών χρήστη. Αυτή η διαστρωματωμένη προσέγγιση όχι μόνο διευκολύνει τη μελλοντική επέκταση του συστήματος, αλλά και εγγυάται τον καθαρό διαχωρισμό μεταξύ επιχειρησιακής λογικής και παρουσίασης.



Εικόνα 6. Διάγραμμα Εξαρτημάτων στη φάση της εκπόνησης μελέτης.

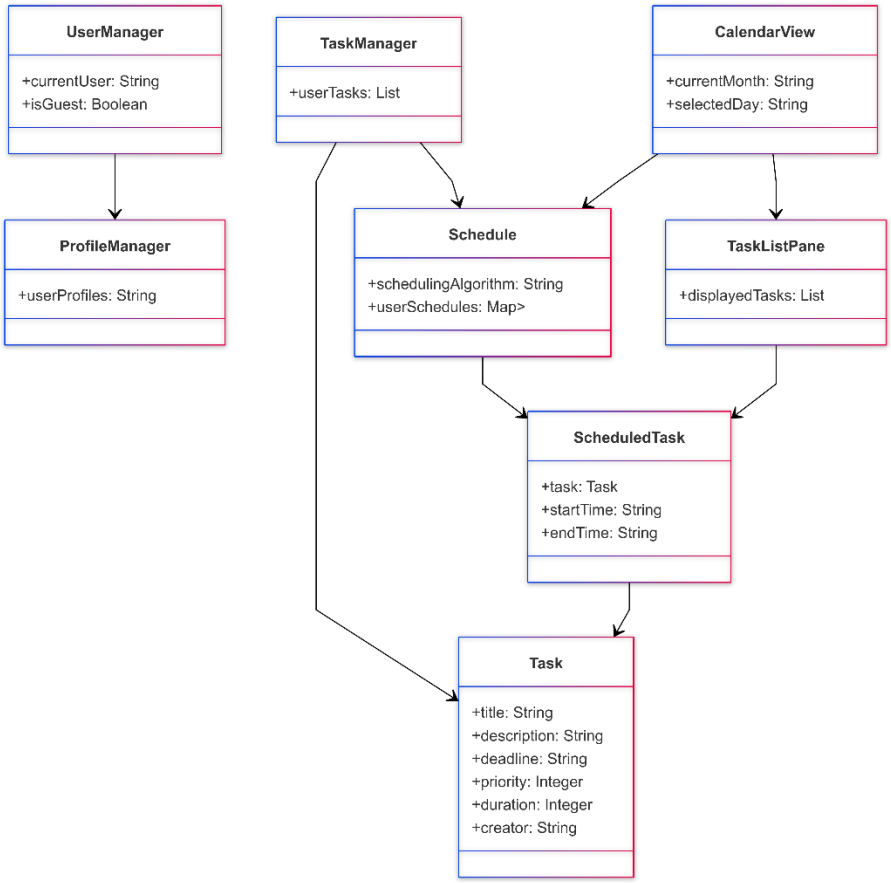
Ενώ το Διάγραμμα Εξαρτημάτων εστιάζει στη λογική δομή, το Διάγραμμα Διανομής στην Εικόνα 7 μεταφέρει το σύστημα στο φυσικό επίπεδο υλοποίησης. Η αρχιτεκτονική προσέγγιση επιλέγει τη λειτουργία σε Client Workstation, με το Java Runtime Environment να αναλαμβάνει την εκτέλεση της επιχειρησιακής λογικής. Η τοπική αποθήκευση των δεδομένων σε αρχεία tasks.json και user_profiles.json, σε συνδυασμό με τη χρήση του LocalStorage για την εσωτερική επικοινωνία, δημιουργεί ένα αυτόνομο περιβάλλον λειτουργίας. Αυτός ο σχεδιασμός όχι μόνο απελευθερώνει το σύστημα από εξωτερικές εξαρτήσεις, αλλά και ενισχύει την ασφάλεια κατά την προσωπική χρήση.



Εικόνα 7. Διάγραμμα Διανομής στη φάση της εκπόνησης μελέτης.

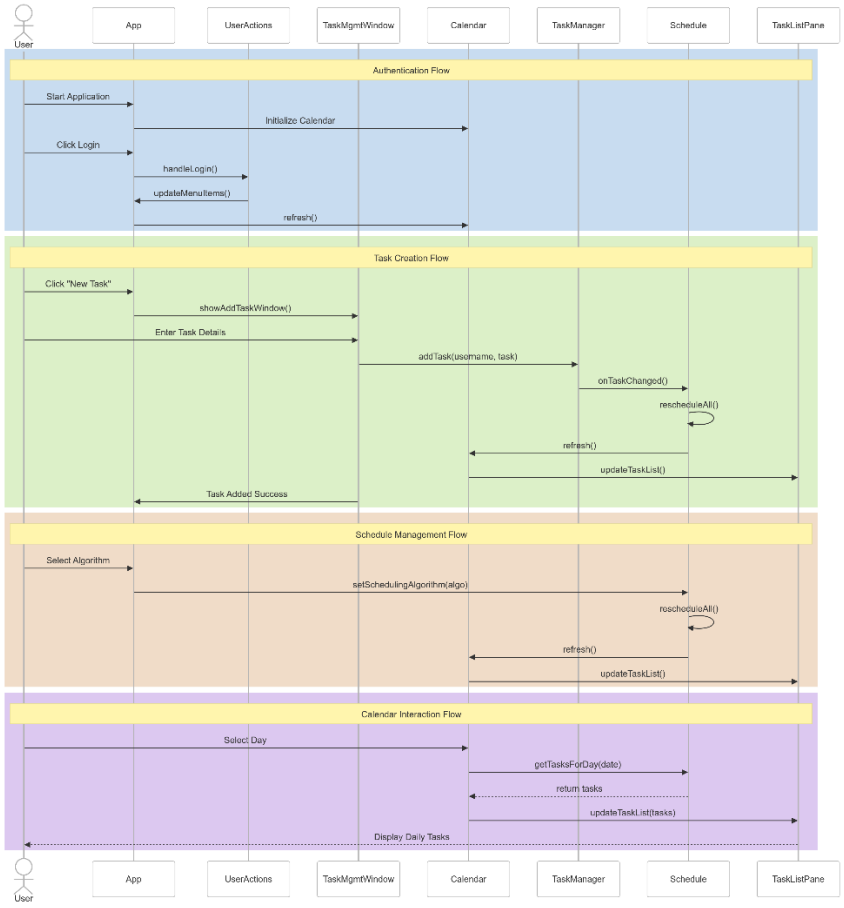
Μεταβαίνοντας από τη φυσική υποδομή στη δυναμική συμπεριφορά του συστήματος, το Διάγραμμα Αντικειμένων στην Εικόνα 8 παρέχει στιγμιότυπα της εφαρμογής εν λειτουργία. Η

εξέλιξη από την απλή κλάση Task, με τα βασικά χαρακτηριστικά τίτλου, προθεσμίας και προτεραιότητας, στην εμπλουτισμένη ScheduledTask, με τις επιπρόσθετες χρονικές παραμέτρους, αναδεικνύει την ιεραρχική οργάνωση των δεδομένων. Μέσα από αυτή την απεικόνιση αποκαλύπτεται η πραγματική αλληλεπίδραση των κλάσεων κατά την εκτέλεση των λειτουργιών του συστήματος.



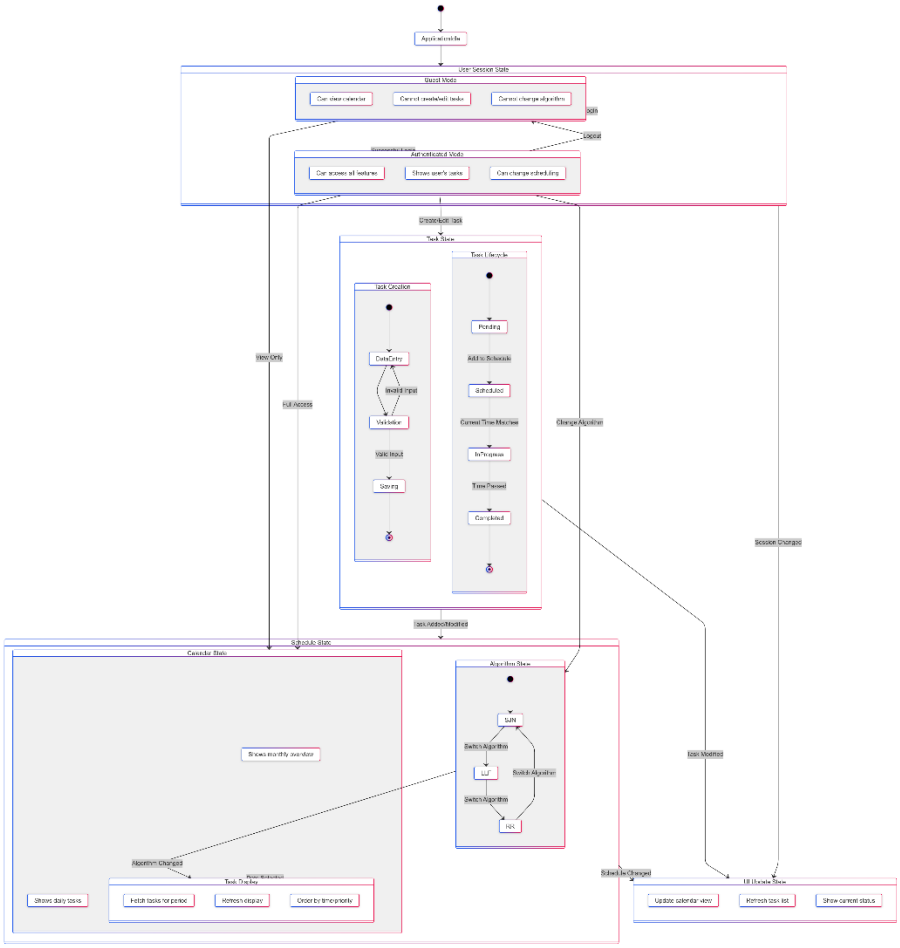
Εικόνα 8. Διάγραμμα Αντικειμένων στη φάση της εκπόνησης μελέτης.

Συμπληρωματικά με τα στιγμιότυπα των αντικειμένων, το Διάγραμμα Σειράς στην Εικόνα 9 ξετυλίγει τη χρονική αλληλουχία των αλληλεπιδράσεων στο σύστημα. Από την αρχική ταυτοποίηση του χρήστη έως την τελική ενημέρωση της διεπαφής, το διάγραμμα παρακολουθεί τη διαδρομή κάθε λειτουργίας μέσα στο σύστημα. Η αναλυτική απεικόνιση των μηνυμάτων μεταξύ των συστατικών στοιχείων - από την είσοδο δεδομένων μέχρι την επεξεργασία από τον TaskManager - αποκαλύπτει τον πλήρη κύκλο ζωής κάθε ενέργειας χρήστη.



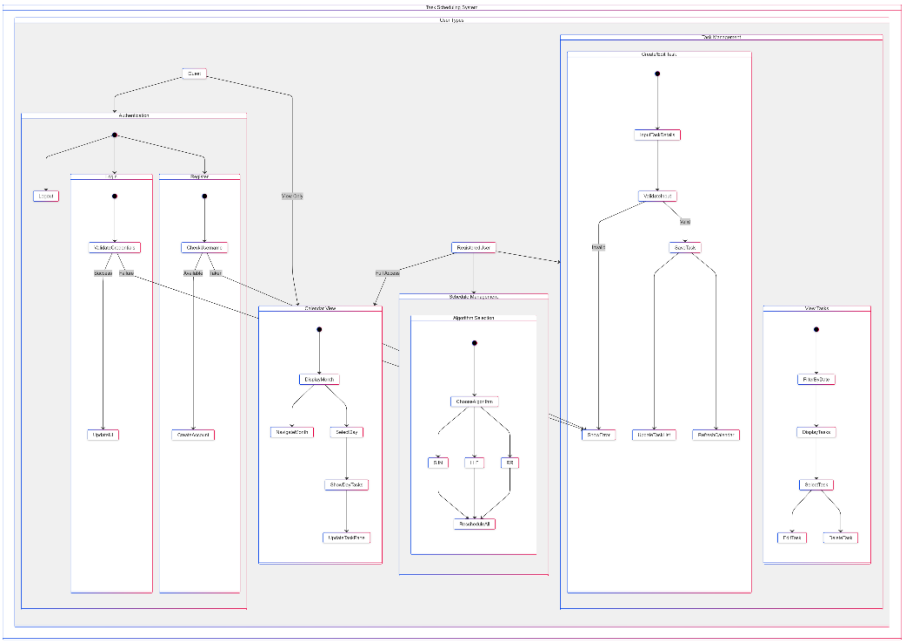
Εικόνα 9. Διάγραμμα Σειράς στη φάση της εκπόνησης μελέτης.

Ενώ το Διάγραμμα Σειράς εστιάζει στην αλληλουχία των ενεργειών, το Διάγραμμα Καταστάσεων στην Εικόνα 10 αναδεικνύει τους μετασχηματισμούς των αντικειμένων κατά τη διάρκεια του κύκλου ζωής τους. Κάθε εργασία διατρέχει μια προδιαγεγραμμένη πορεία εξέλιξης - από την αρχική κατάσταση "Pending", μέσω του προγραμματισμού ("Scheduled"), έως την τελική ολοκλήρωση ("Completed"). Οι μεταβάσεις αυτές πυροδοτούνται από συγκεκριμένα γεγονότα, είτε πρόκειται για αλλαγές στον αλγόριθμο είτε για παρεμβάσεις του χρήστη, σκιαγραφώντας έτσι τη δυναμική συμπεριφορά του συστήματος στο χρόνο.



Εικόνα 10. Διάγραμμα Καταστάσεων στη φάση της εκπόνησης μελέτης.

Το δεύτερο Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagram) στην Εικόνα 11 επεκτείνει το πρώτο, προσθέτοντας περισσότερες λεπτομέρειες για τις λειτουργίες και τη διαχείριση του συστήματος. Εκτός από τις βασικές ενέργειες, όπως η σύνδεση και η δημιουργία εργασιών, περιλαμβάνονται πιο σύνθετες λειτουργίες, όπως η αλλαγή αλγορίθμου και η προγραμματισμένη ανανέωση των δεδομένων στο ημερολόγιο. Οι συσχετίσεις μεταξύ των ρόλων (π.χ., Guest User, Registered User) και των περιπτώσεων χρήσης δείχνουν τη διαβάθμιση των δικαιωμάτων πρόσβασης και τη λειτουργικότητα που παρέχεται σε κάθε κατηγορία χρήστη. Αυτό το διάγραμμα ολοκληρώνει τη φάση της εκπόνησης μελέτης, δίνοντας μια συνολική εικόνα για τις δυνατότητες του συστήματος.

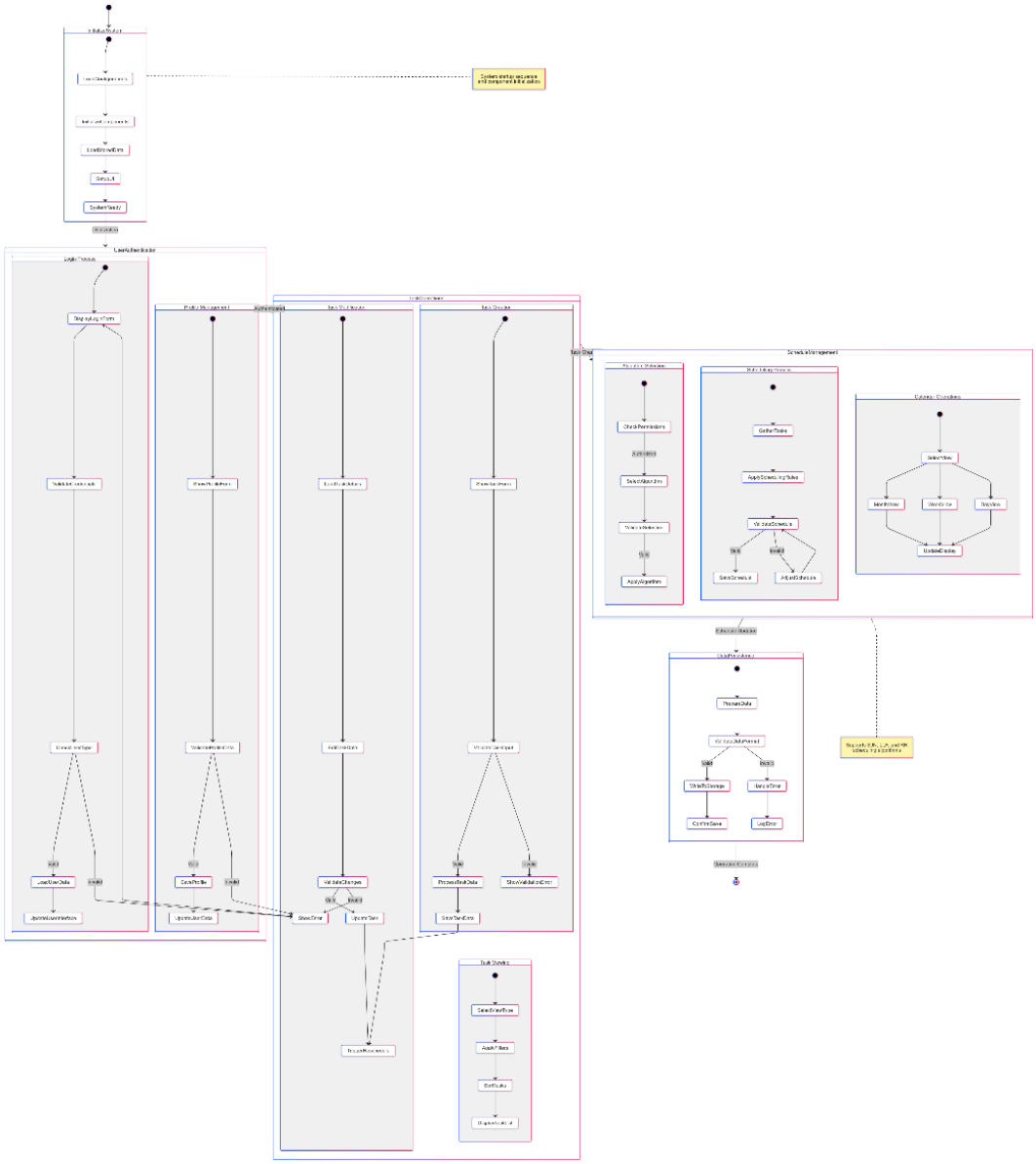


Εικόνα 11. Διάγραμμα Περιπτώσεων Χρήσης στη φάση της εκπόνησης μελέτης.

Φάση Κατασκευής (Construction)

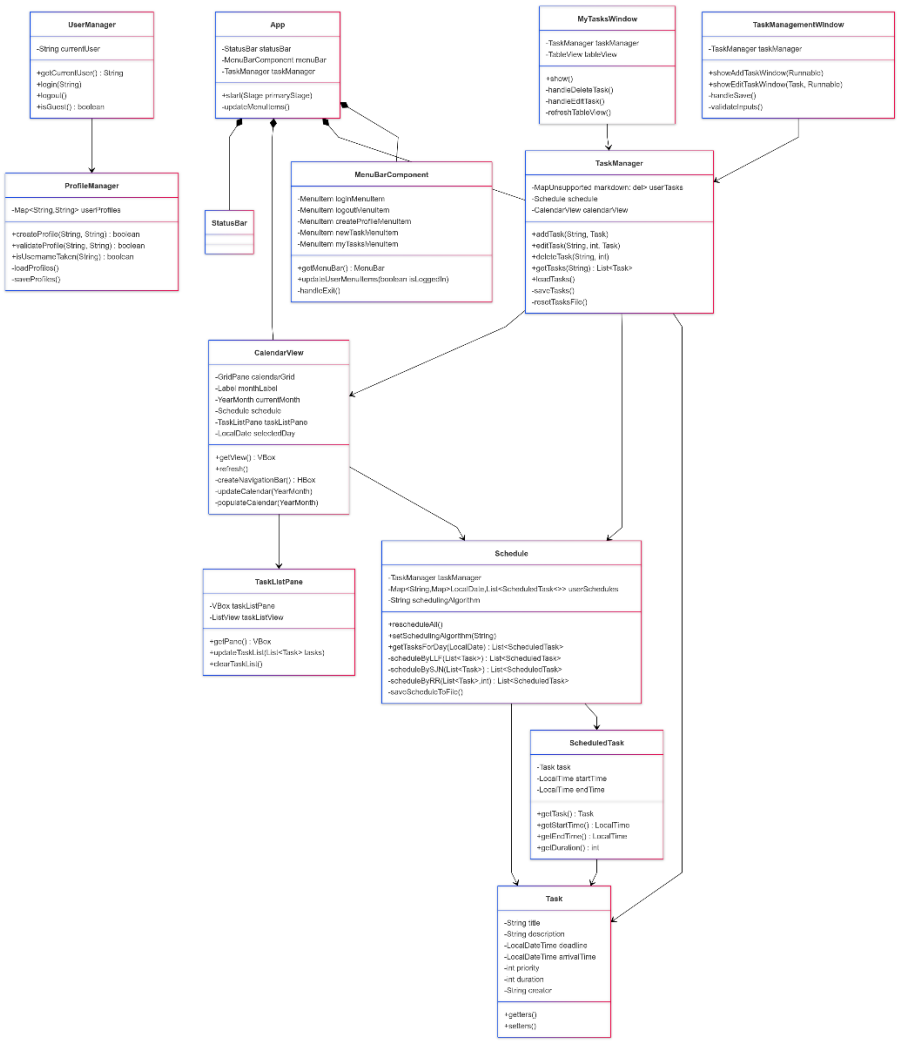
Η φάση της Κατασκευής αποτελεί το στάδιο όπου το σύστημα περνά από τη θεωρία στην πράξη, με την πλήρη υλοποίηση των λειτουργιών του. Στόχος είναι η οριστικοποίηση της αρχιτεκτονικής, η ανάπτυξη όλων των προγραμματιστικών μονάδων και η δοκιμή του συστήματος για τη διασφάλιση της λειτουργικότητας. Τα διαγράμματα που δημιουργήθηκαν σε αυτό το στάδιο παρέχουν μια ολοκληρωμένη εικόνα της αρχιτεκτονικής, της ροής εργασιών, και της διασύνδεσης των υποσυστημάτων.

Το πρώτο βήμα αυτής της φάσης αποτυπώνεται στο Διάγραμμα Δραστηριοτήτων της Εικόνα 12, το οποίο εμβαθύνει στις ροές εργασίας του συστήματος. Από την ταυτοποίηση χρηστών και τη δημιουργία εργασιών μέχρι την επιλογή αλγορίθμων και τη διαχείριση δεδομένων, το διάγραμμα χαρτογραφεί κάθε πτυχή της λειτουργικότητας. Ιδιαίτερη έμφαση δίνεται στην αντιμετώπιση εξαιρέσεων και στη διαχείριση σύνθετων αλληλεπιδράσεων, όπως η ενημέρωση του ημερολογίου, διασφαλίζοντας έτσι την ομαλή εκτέλεση όλων των διαδικασιών.



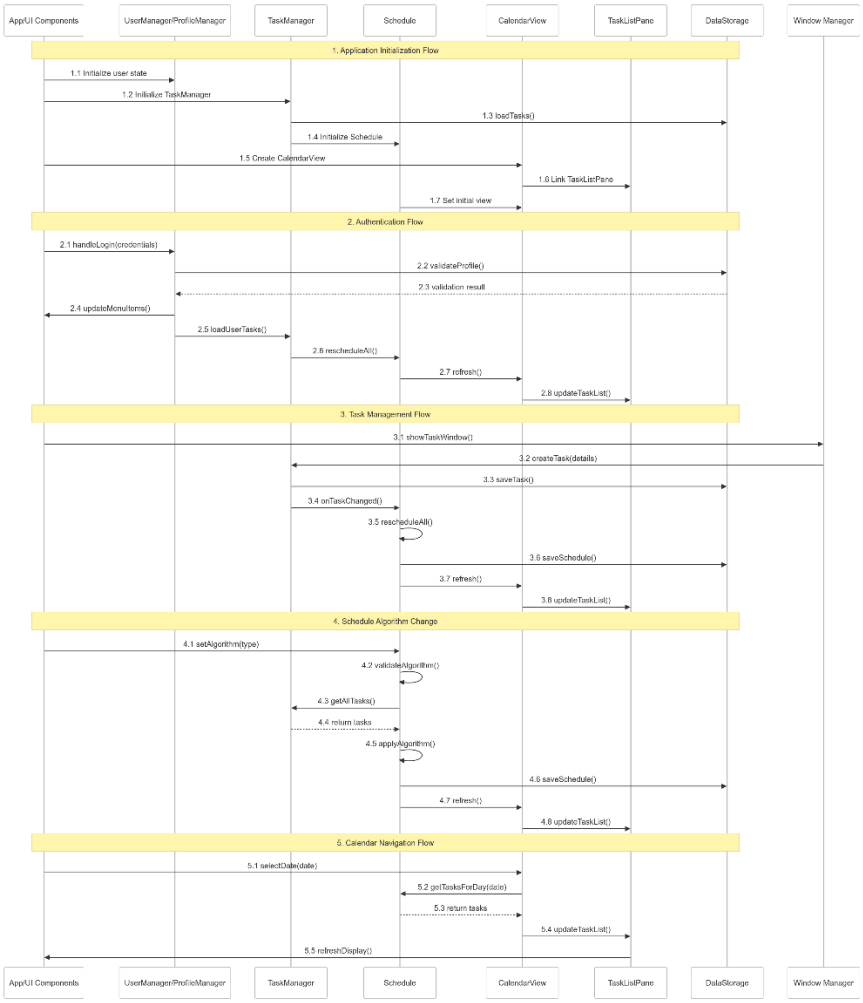
Εικόνα 12. Διάγραμμα Δραστηριοτήτων στη φάση της κατασκευής.

Παράλληλα με τη χαρτογράφηση των δραστηριοτήτων, το τρίτο Διάγραμμα Τάξεων στην Εικόνα 13 παρουσιάζει την ώριμη πλέον αρχιτεκτονική του συστήματος. Η εξέλιξη του σχεδιασμού αντικατοπτρίζεται στην προσθήκη εξειδικευμένων κλάσεων - η MenuBarComponent αναλαμβάνει τον έλεγχο της διεπαφής χρήστη, ενώ η αναβαθμισμένη TaskManager συγκεντρώνει το σύνολο των λειτουργιών διαχείρισης εργασιών. Ιδιαίτερη σημασία αποκτά η διασύνδεση μεταξύ Schedule και TaskListPane, βελτιστοποιώντας το συγχρονισμό προγραμματισμού και απεικόνισης των εργασιών. Το αποτέλεσμα είναι ένα πολύπλοκο αλλά καλά ενορχηστρωμένο σύστημα συνεργαζόμενων μονάδων.



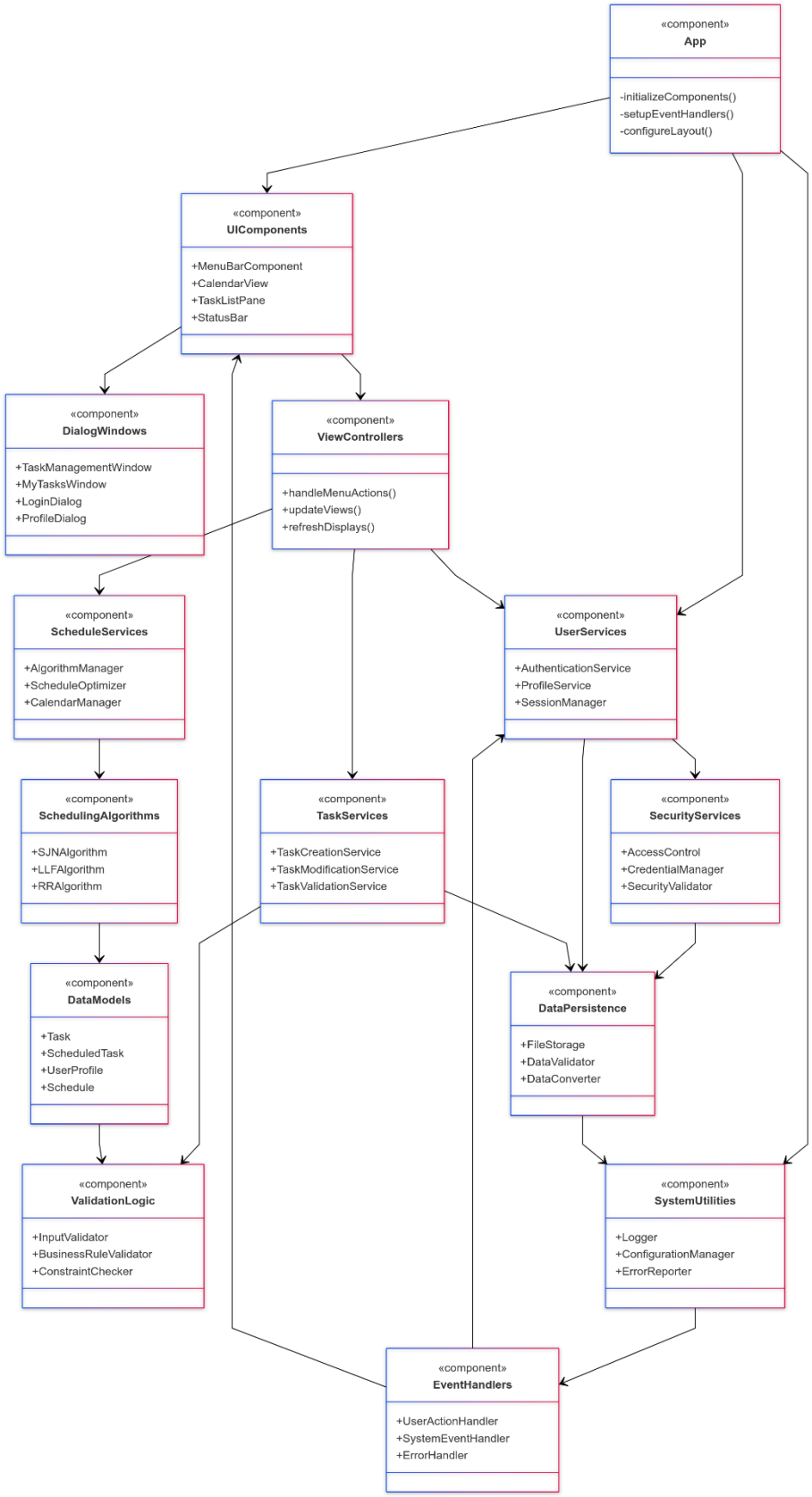
Εικόνα 13. Διάγραμμα Τάξεων στη φάση της κατασκευής.

Ενώ το Διάγραμμα Τάξεων καθορίζει τη στατική δομή, το δεύτερο Διάγραμμα Συνεργασίας στην Εικόνα 14 ζωντανεύει τη δυναμική αλληλεπίδραση των συστατικών στοιχείων. Η πολύπλευρη επικοινωνία μεταξύ UserManager, TaskManager και CalendarView αποκαλύπτει τον τρόπο με τον οποίο το σύστημα διαχειρίζεται πραγματικά σενάρια χρήσης - από την επεξεργασία έως την οπτικοποίηση των εργασιών. Η λεπτομερής απεικόνιση της ροής μηνυμάτων μεταξύ των μονάδων υπογραμμίζει πώς η συντονισμένη επικοινωνία εξασφαλίζει την αρμονική λειτουργία του συστήματος.



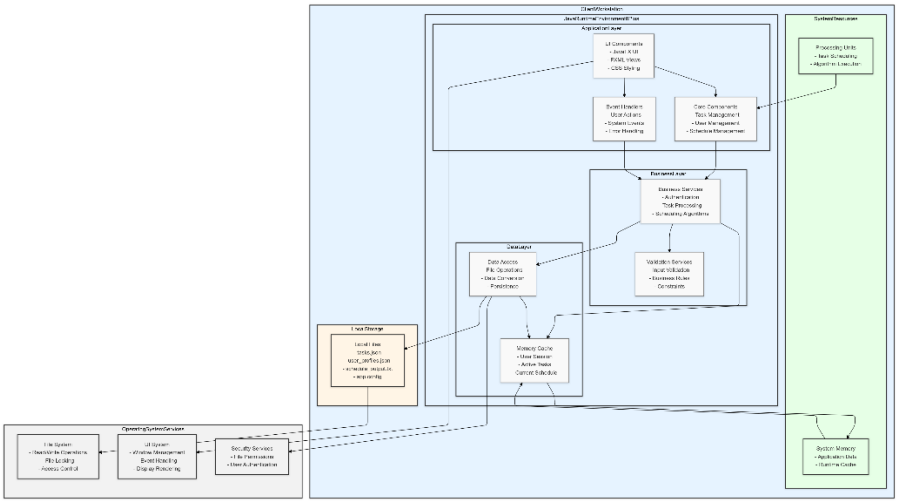
Εικόνα 14. Διάγραμμα Συνεργασίας στη φάση της κατασκευής.

Μετά την ανάλυση των δυναμικών αλληλεπιδράσεων, το δεύτερο Διάγραμμα Εξαρτημάτων στην Εικόνα 15 επιστρέφει στην αρχιτεκτονική οργάνωση του συστήματος, αυτή τη φορά σε επίπεδο λειτουργικών μονάδων. Η επέκταση της διεπαφής χρήστη εμπλουτίζεται με εξειδικευμένα στοιχεία - το StatusBar για την παρακολούθηση της κατάστασης χρήστη και το MenuBarComponent για την απρόσκοπτη πλοήγηση. Παράλληλα, η αρχιτεκτονική ωριμάζει με την προσθήκη εξειδικευμένων υπηρεσιών όπως οι SchedulingAlgorithms και οι TaskServices, οι οποίες επικοινωνούν με τη διεπαφή χρήστη μέσω καθορισμένων διασυνδέσεων. Αυτός ο δομημένος διαχωρισμός αρμοδιοτήτων όχι μόνο διευκολύνει τη συντήρηση, αλλά και θέτει τα θεμέλια για μελλοντικές επεκτάσεις του συστήματος.



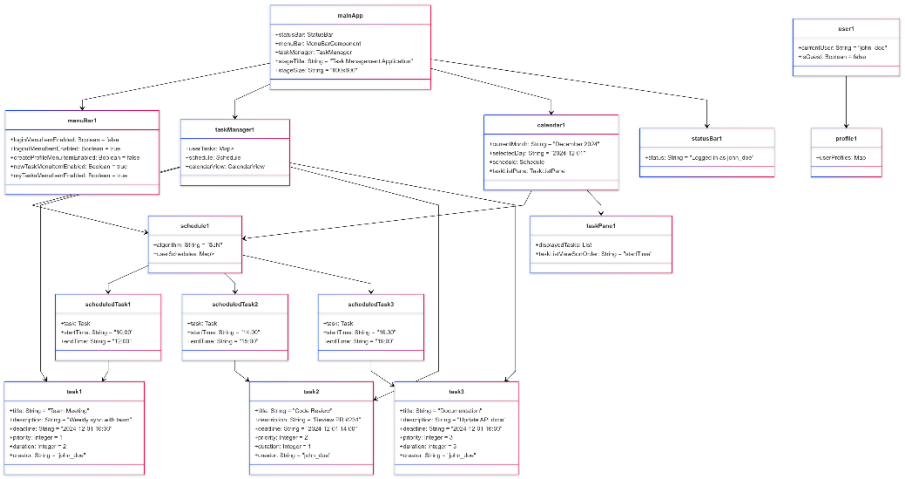
Εικόνα 15. Διάγραμμα Εξαρτημάτων στη φάση της κατασκευής.

Ενώ το Διάγραμμα Εξαρτημάτων επικεντρώνεται στην οργάνωση των λειτουργικών μονάδων, το δεύτερο Διάγραμμα Διανομής στην Εικόνα 16 μεταφράζει αυτή την αρχιτεκτονική σε φυσική υποδομή. Διατηρώντας το Java Runtime Environment ως βάση εκτέλεσης, το σύστημα εξελίσσεται με την εισαγωγή της εξειδικευμένης μονάδας DataLayer, η οποία ενοποιεί τις λειτουργίες διαχείρισης δεδομένων. Η προσθήκη του Memory Cache αποτελεί στρατηγική βελτίωση της απόδοσης, δημιουργώντας ένα ενδιάμεσο επίπεδο προσωρινής αποθήκευσης για ταχύτερη πρόσβαση στα συχνά χρησιμοποιούμενα δεδομένα. Έτσι, το διάγραμμα γεφυρώνει το χάσμα μεταξύ λογικής σχεδίασης και φυσικής υλοποίησης.



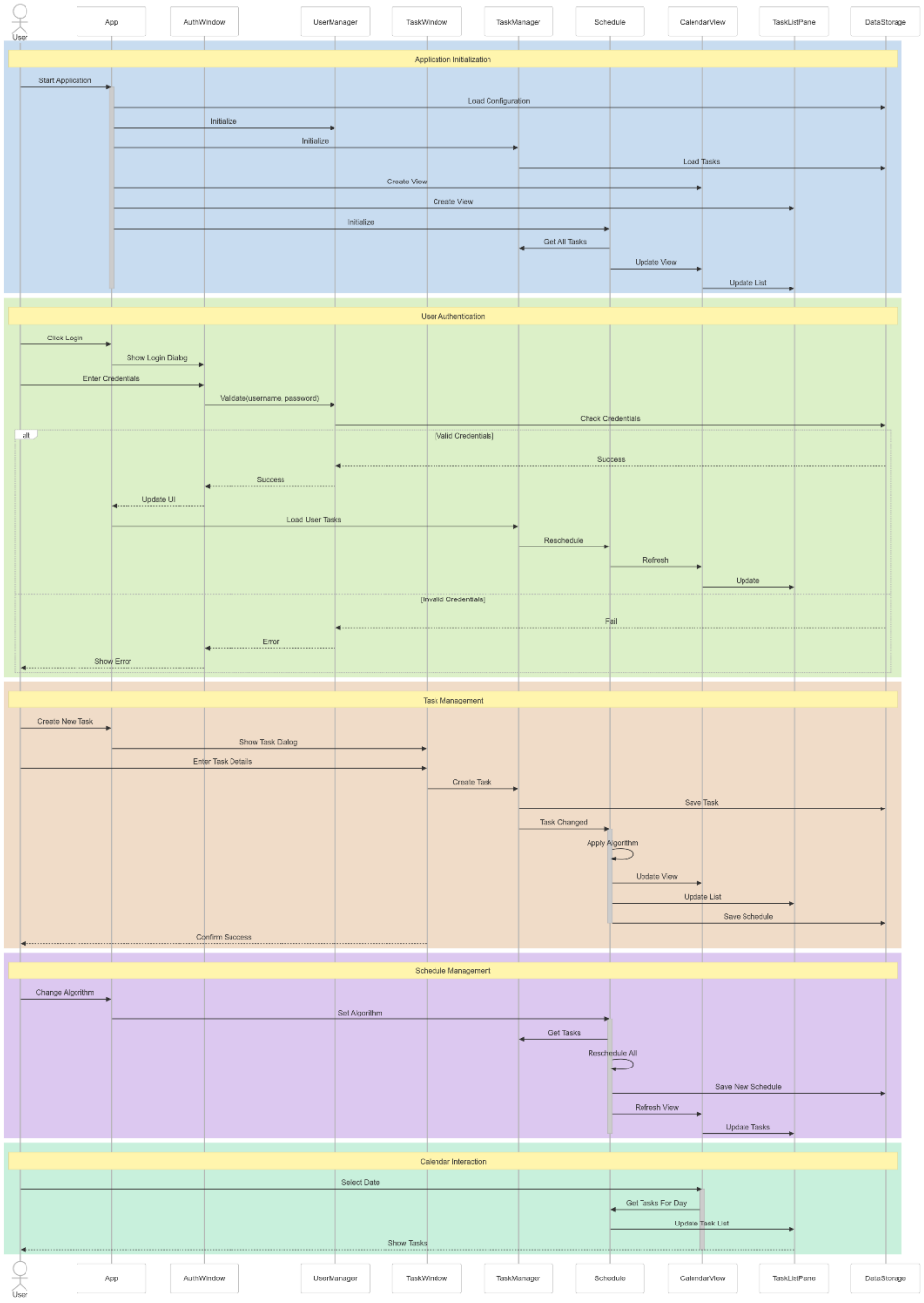
Εικόνα 16. Διάγραμμα Διανομής στη φάση της κατασκευής.

Μεταβαίνοντας από τη φυσική υποδομή στην πρακτική λειτουργία, το δεύτερο Διάγραμμα Αντικειμένων στην Εικόνα 17 προσφέρει μια ζωντανή απεικόνιση του συστήματος εν δράσει. Η δυναμική φύση της εφαρμογής αποτυπώνεται μέσα από συγκεκριμένα παραδείγματα - οι εργασίες task1, task2 και task3 αποκαλύπτουν την ποικιλομορφία των δεδομένων, με κάθε αντικείμενο να φέρει τα δικά του χαρακτηριστικά τίτλου, προτεραιότητας και διάρκειας. Παράλληλα, τα στιγμιότυπα της κλάσης Schedule επιδεικνύουν την εφαρμογή προηγμένων αλγορίθμων χρονοπρογραμματισμού, όπως το SJN, παρέχοντας μια απτή εικόνα της διαχείρισης χρονοδιαγραμμάτων στην πράξη.



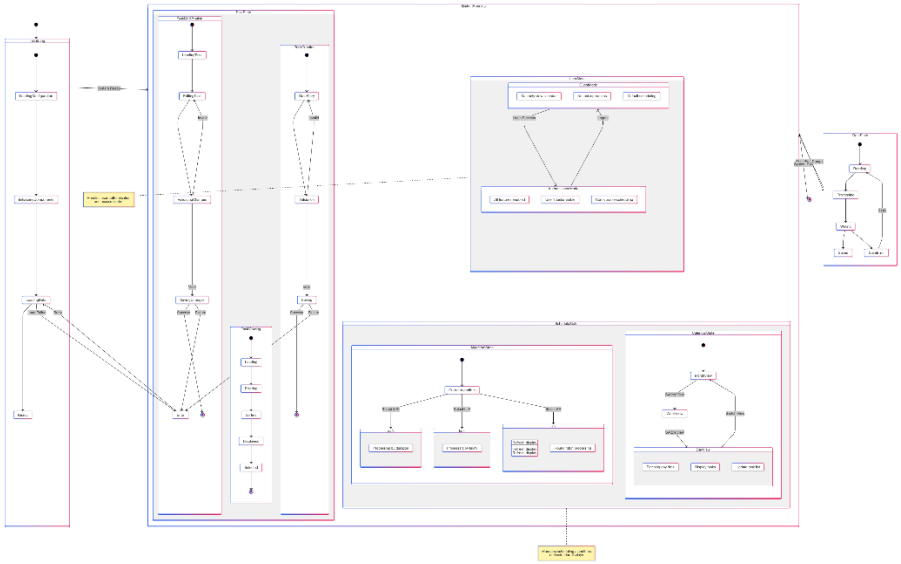
Εικόνα 17. Διάγραμμα Αντικειμένων στη φάση της κατασκευής.

Συμπληρώνοντας την πρακτική απεικόνιση των αντικειμένων, το δεύτερο Διάγραμμα Σειράς στην Εικόνα 18 εμβαθύνει στη χρονική αλληλουχία των πιο σύνθετων λειτουργιών του συστήματος. Ξεκινώντας από την ταυτοποίηση χρήστη και προχωρώντας στη δημιουργία εργασιών και την επιλογή αλγορίθμων χρονολόγησης, το διάγραμμα ξετυλίγει τη διαδοχική αλυσίδα των επιμέρους βημάτων. Κάθε στάδιο της διαδικασίας αποκτά συγκεκριμένο ρόλο - ο UserManager επικυρώνει τα δεδομένα, ο TaskManager διαχειρίζεται τις εργασίες, και η CalendarView συγχρονίζει το ημερολόγιο, δημιουργώντας έτσι ένα αρμονικό σύνολο αλληλεπιδράσεων σε πραγματικό χρόνο.



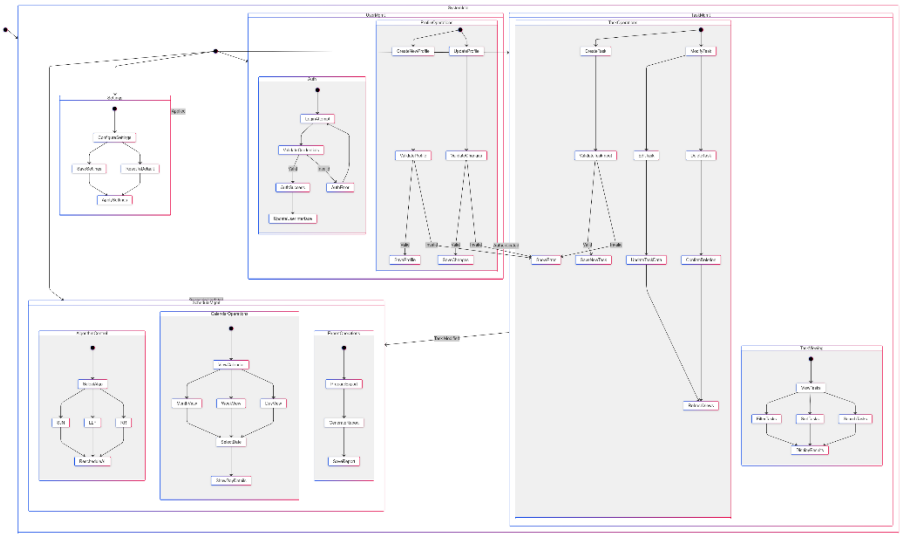
Εικόνα 18. Διάγραμμα Σειράς στη φάση της κατασκευής.

Το δεύτερο Διάγραμμα Καταστάσεων στην Εικόνα 19 απεικονίζει τις μεταβάσεις μεταξύ διαφορετικών καταστάσεων των βασικών στοιχείων του συστήματος. Για παράδειγμα, μια εργασία μπορεί να ξεκινήσει ως "Pending", να μεταβεί σε "Scheduled" και τελικά σε "Completed". Η κατάσταση του αλγορίθμου προγραμματισμού αλλάζει ανάλογα με την επιλογή του χρήστη, ενώ το ημερολόγιο ενημερώνεται με βάση τα νέα δεδομένα. Αυτό το διάγραμμα είναι κρίσιμο για την κατανόηση της δυναμικής συμπεριφοράς του συστήματος και του τρόπου με τον οποίο αντιδρά σε αλλαγές ή ενέργειες χρηστών.



Εικόνα 19. Διάγραμμα Καταστάσεων στη φάση της κατασκευής.

Το τρίτο Διάγραμμα Περιπτώσεων Χρήσης στην Εικόνα 20 παρουσιάζει την πλήρη λειτουργικότητα του συστήματος, συμπεριλαμβανομένων όλων των βασικών και προηγμένων λειτουργιών. Οι περιπτώσεις χρήσης, όπως η προβολή εργασιών, η αλλαγή αλγορίθμου, και η διαχείριση χρονοδιαγραμμάτων, συνδέονται με διαφορετικούς ρόλους χρηστών. Οι ρόλοι Guest User και Registered User παρουσιάζονται με ξεκάθαρες διαβαθμίσεις πρόσβασης. Το διάγραμμα αυτό ολοκληρώνει την απεικόνιση της λειτουργικότητας του συστήματος, επιδεικνύοντας τη συνολική του δυνατότητα να καλύψει τις ανάγκες διαφορετικών χρηστών.



Εικόνα 20. Διάγραμμα Περιπτώσεων Χρήσης στη φάση της κατασκευής.

Η φάση της Κατασκευής ολοκληρώνει την ανάπτυξη του συστήματος, μετατρέποντας τον σχεδιασμό σε λειτουργική εφαρμογή. Τα διαγράμματα που δημιουργήθηκαν σε αυτή τη φάση, όπως τα διαγράμματα δραστηριοτήτων, καταστάσεων, και περιπτώσεων χρήσης, συνδυάζονται για να προσφέρουν μια ολοκληρωμένη εικόνα της λειτουργίας του συστήματος. Κάθε διάγραμμα, με την εξειδικευμένη του προσέγγιση, συμβάλλει στην κατανόηση και την τελειοποίηση των

διαδικασιών, εξασφαλίζοντας ότι το τελικό σύστημα είναι πλήρως λειτουργικό, αποδοτικό και εύκολα επεκτάσιμο. Με αυτόν τον τρόπο, η φάση της Κατασκευής θέτει τα θεμέλια για την επιτυχή παράδοση της εφαρμογής, ανταποκρινόμενη πλήρως στις απαιτήσεις των χρηστών.

Το κεφάλαιο αυτό ανέδειξε τη σημασία της διαδικασίας RUP στη συστηματική ανάπτυξη του συστήματος. Με τη διαίρεση της ανάπτυξης σε φάσεις (Έναρξη, Εκπόνηση Μελέτης, Κατασκευή), καταφέραμε να οργανώσουμε αποτελεσματικά τη ροή εργασιών και να διασφαλίσουμε τη συνέπεια ανάμεσα στον σχεδιασμό και την υλοποίηση. Η δομή που προσφέρει το RUP επέτρεψε την προσαρμογή του σχεδιασμού στις ανάγκες του έργου, εστιάζοντας στις προτεραιότητες κάθε φάσης και προλαμβάνοντας πιθανά προβλήματα.

Τα UML διαγράμματα αποτέλεσαν τον ακρογωνιαίο λίθο της ανάλυσης και του σχεδιασμού, διευκολύνοντας την αποτύπωση και κατανόηση του συστήματος. Από τα διαγράμματα περιπτώσεων χρήσης, που ανέδειξαν τις βασικές λειτουργίες, έως τα διαγράμματα καταστάσεων και ακολουθίας, που κατέγραψαν την πολυπλοκότητα της ροής δεδομένων και λειτουργιών, η χρήση τους εξασφάλισε την ακρίβεια και την οργάνωση σε κάθε στάδιο. Μέσα από αυτά, καταφέραμε να ενσωματώσουμε την απαραίτητη πολυπλοκότητα, διατηρώντας όμως την ευελιξία και την κατανοητότητα του συστήματος.

Η ολοκλήρωση του σχεδιασμού και της ανάπτυξης κατέληξε σε ένα λειτουργικό και επεκτάσιμο σύστημα, έτοιμο να ανταποκριθεί στις ανάγκες των χρηστών. Η προσέγγιση που ακολουθήθηκε προσφέρει μια σταθερή βάση για μελλοντικές βελτιώσεις, είτε αυτές αφορούν την προσθήκη νέων λειτουργιών είτε την ενσωμάτωση τεχνολογιών. Μέσα από αυτή τη διαδικασία, καταδείχθηκε η αξία της μεθοδικής προσέγγισης στην ανάπτυξη λογισμικού, καθιστώντας το σύστημα όχι μόνο εργαλείο διαχείρισης αλλά και μοντέλο ανάπτυξης για μελλοντικά έργα.

ΚΕΦΑΛΑΙΟ 3. ΥΛΟΠΟΙΗΣΗ

Η διεπαφή χρήστη αποτελεί ένα κρίσιμο συστατικό της εφαρμογής διαχείρισης εργασιών, καθώς είναι υπεύθυνη για την αποτελεσματική αλληλεπίδραση μεταξύ των χρηστών και του συστήματος χρονοπρογραμματισμού. Η υλοποίησή της βασίστηκε στο πλαίσιο JavaFX, το οποίο παρέχει ένα ισχυρό σύνολο εργαλείων για τη δημιουργία σύγχρονων desktop εφαρμογών. Η διεπαφή σχεδιάστηκε με γνώμονα τέσσερις βασικές αρχές: την ευχρηστία, την αποδοτικότητα, την προσβασιμότητα και την αισθητική. Το σύστημα αποτελείται από διάφορα επιμέρους components, καθένα από τα οποία εξυπηρετεί συγκεκριμένες λειτουργικές απαιτήσεις, ενώ παράλληλα διατηρεί μια συνεκτική και διαισθητική εμπειρία χρήσης. Στις επόμενες ενότητες αναλύεται λεπτομερώς η υλοποίηση κάθε component, καθώς και ο τρόπος με τον οποίο αυτά συνεργάζονται για να παρέχουν μια ολοκληρωμένη λύση διαχείρισης εργασιών.

3.1 Περιβάλλον Ανάπτυξης και Εργαλεία

Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε σε περιβάλλον Java χρησιμοποιώντας τη βιβλιοθήκη JavaFX για τη δημιουργία της διεπαφής χρήστη. Το IntelliJ IDEA Ultimate Edition επιλέχθηκε ως το βασικό εργαλείο ανάπτυξης, προσφέροντας ισχυρή υποστήριξη για την πλατφόρμα JavaFX και το σύστημα Gradle. Μέσα από το IDE, ήταν εφικτή η εύκολη διαχείριση του project, η ενσωμάτωση εξαρτήσεων, και η διευκόλυνση της διαδικασίας debugging.

Το project ακολουθεί μια σαφή και οργανωμένη δομή, όπου κάθε πακέτο εξυπηρετεί συγκεκριμένες λειτουργίες. Στο πακέτο components υλοποιούνται τα στοιχεία της διεπαφής χρήστη, όπως το *CalendarView*, το *TaskListPane* και το *MenuBarComponent*. Το πακέτο models περιέχει τις κλάσεις για τη διαχείριση δεδομένων, όπως οι κλάσεις *Task* και *ScheduledTask*. Παράλληλα, το πακέτο utils περιλαμβάνει βοηθητικές κλάσεις, όπως το Schedule για τη διαχείριση των αλγορίθμων χρονολόγησης και το για τη διαχείριση προφίλ χρηστών. Τέλος, το πακέτο views φιλοξενεί τα παράθυρα διαχείρισης εργασιών, προσφέροντας εύκολη πρόσβαση σε βασικές λειτουργίες της εφαρμογής.

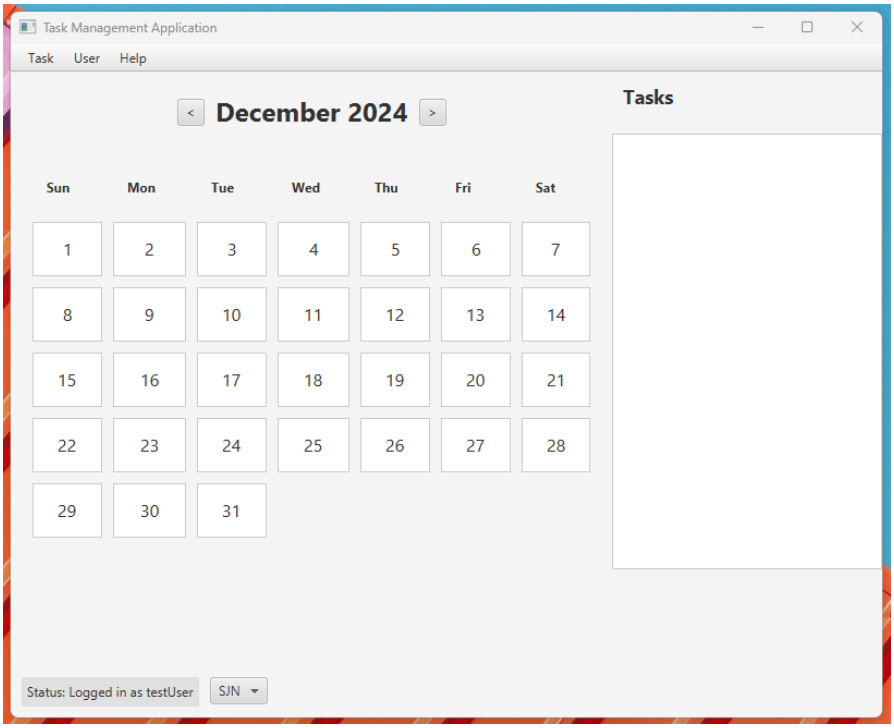
Για τη διαχείριση εξαρτήσεων και το build process χρησιμοποιήθηκε το Gradle. Το αρχείο build.gradle περιλαμβάνει δηλώσεις εξαρτήσεων για βιβλιοθήκες όπως το Gson, που χρησιμοποιείται για την αποθήκευση και ανάκτηση δεδομένων σε JSON μορφή, και το ControlsFX, το οποίο παρέχει επιπλέον εργαλεία για τη βελτίωση της διεπαφής χρήστη. Η χρήση του Gradle απλοποίησε τη διαδικασία ανάπτυξης, εξασφαλίζοντας ταυτόχρονα την ενημέρωση των βιβλιοθηκών και την εύκολη διαχείριση των builds.

Τέλος, το module-info.java προσδιορίζει τη δομή του module com.example.taskmanagement και τις εξαρτήσεις του. Δηλώνει ότι το project απαιτεί τα modules javafx.controls και javafx.fxml, ενώ εκθέτει τα πακέτα com.example.taskmanagement και

components σε άλλα modules. Αυτή η προσέγγιση εξασφαλίζει τη συμβατότητα και τη modular ανάπτυξη της εφαρμογής, παρέχοντας παράλληλα ένα ευέλικτο περιβάλλον εργασίας για περαιτέρω επεκτάσεις.

3.2 Υλοποίηση Διεπαφής Χρήστη

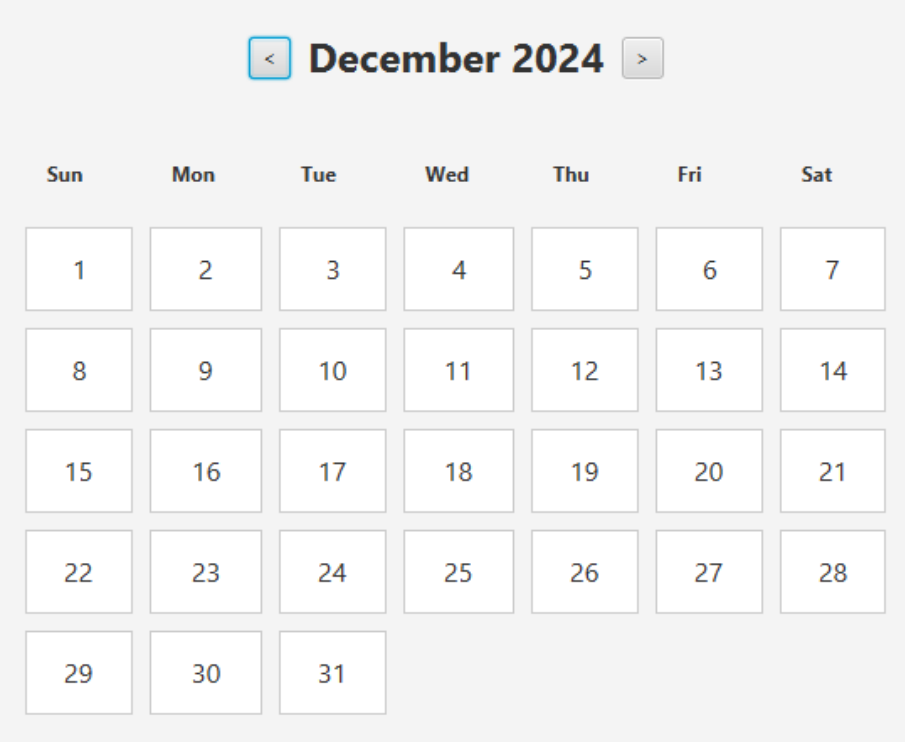
Η υλοποίηση του user interface της εφαρμογής βασίστηκε στο πλαίσιο JavaFX, το οποίο επιλέχθηκε για την ευελιξία του και τη δυνατότητα δημιουργίας σύγχρονων desktop εφαρμογών. Το κύριο παράθυρο της εφαρμογής (App.java) υιοθετεί μια διάταξη τύπου BorderPane, η οποία χωρίζει το παράθυρο σε πέντε διακριτές περιοχές. Στην κορυφή τοποθετείται η μπάρα μενού, στο κέντρο βρίσκεται το ημερολόγιο για την προβολή των εργασιών, στα δεξιά εμφανίζεται η λίστα των εργασιών της επιλεγμένης ημέρας, ενώ στο κάτω μέρος υπάρχει η γραμμή κατάστασης μαζί με το μενού επιλογής αλγορίθμου χρονοπρογραμματισμού. Το παράθυρο έχει προκαθορισμένες διαστάσεις 800x600 pixels, παρέχοντας επαρκή χώρο για όλα τα στοιχεία διεπαφής, ενώ παράλληλα διατηρεί την εφαρμογή συμπαγή και εύχρηστη.



Εικόνα 21. Το κύριο παράθυρο της εφαρμογής.

Το σύστημα ημερολογίου υλοποιείται μέσω της κλάσης *CalendarView*, η οποία αποτελεί ένα από τα κεντρικότερα στοιχεία της διεπαφής χρήστη. Η υλοποίηση βασίζεται σε ένα GridPane που διαμορφώνει ένα πλέγμα 7x6 για την αναπαράσταση των ημερών του μήνα. Στην κορυφή του ημερολογίου υπάρχει μια γραμμή πλοήγησης που περιλαμβάνει κουμπιά για μετακίνηση στον προηγούμενο και επόμενο μήνα, καθώς και την τρέχουσα ημερομηνία σε μορφή "Μήνας Έτος". Κάθε κελί του ημερολογίου αντιπροσωπεύει μια ημέρα και έχει ελάχιστες διαστάσεις 50x50 pixels,

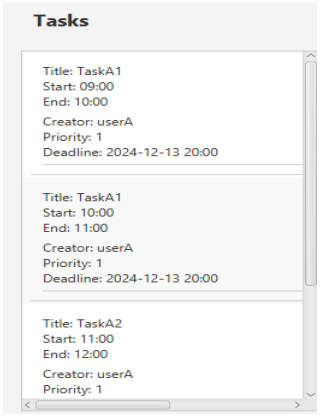
ενώ μπορεί να επεκταθεί ανάλογα με το μέγεθος του παραθύρου. Η επιλογή μιας ημέρας ενεργοποιεί την εμφάνιση των αντίστοιχων προγραμματισμένων εργασιών στο παράθυρο λίστας εργασιών, ενώ η ανανέωση του ημερολογίου (*refresh*) πραγματοποιείται αυτόματα κάθε φορά που υπάρχει αλλαγή στον προγραμματισμό των εργασιών ή στον επιλεγμένο αλγόριθμο χρονοπρογραμματισμού.



Εικόνα 22. Το interface του CalendarView, όπως απεικονίζεται, παρέχει μια αναλυτική προβολή του μήνα Δεκέμβριου του 2024.

Η λίστα εργασιών υλοποιείται μέσω της κλάσης *TaskListPane*, η οποία εμφανίζεται στο δεξί τμήμα της εφαρμογής και παρέχει λεπτομερή προβολή των προγραμματισμένων εργασιών για την επιλεγμένη ημέρα. Η κλάση χρησιμοποιεί ένα VBox ως βασικό container και ενσωματώνει ένα εξειδικευμένο ListView<ScheduledTask> για την παρουσίαση των εργασιών. Κάθε εργασία στη λίστα αναπαρίσταται από ένα προσαρμοσμένο ListCell, το οποίο παρέχεται μέσω της εσωτερικής κλάσης *TaskCell*. Το κελί κάθε εργασίας εμφανίζει τον τίτλο, την ώρα έναρξης και λήξης, τον δημιουργό, την προτεραιότητα και την προθεσμία της εργασίας, με τα στοιχεία να διαχωρίζονται οπτικά μέσω μιας λεπτής γραμμής. Η λίστα ταξινομείται αυτόματα με βάση την ώρα έναρξης των εργασιών, ενώ υπάρχει δυνατότητα αυτόματης ανανέωσης (*updateTaskList*) όταν αλλάζει η επιλεγμένη ημέρα ή όταν τροποποιούνται οι εργασίες.

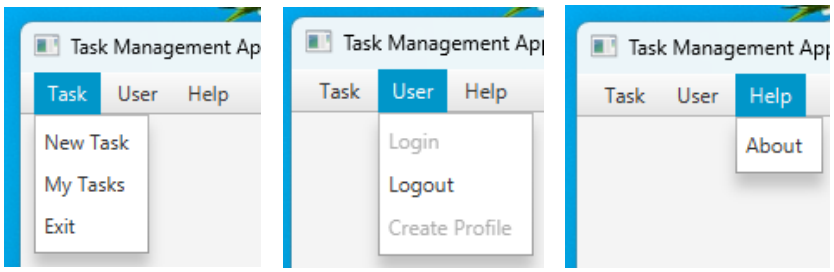
```java
public TaskListPane() {
    taskListPane = new VBox();
    taskListPane.setSpacing(10);

    Label titleLabel = new Label( s: "Tasks");
    titleLabel.setStyle("-fx-font-size: 18px; " +
            "-fx-font-weight: bold; -fx-padding: 10; " +
            "-fx-alignment: center");
    taskListView = new ListView<>();
    taskListView.setCellFactory(new TaskCellFactory());
    taskListPane.getChildren().addAll(titleLabel, taskListView);
}
```




```java
public void updateTaskList(List<ScheduledTask> tasksForDay) {  2 usages
    taskListView.getItems().clear();
    tasksForDay.sort(( ScheduledTask t1,  ScheduledTask t2)
            -> t1.getStartTime().compareTo(t2.getStartTime())); // Sort by start time
    taskListView.getItems().addAll(tasksForDay);
}
```

Εικόνα 23. Ο κατασκευαστής του component που αρχικοποιεί την κλάση με χρήση ενός VBox και προσαρμόζει τον τίτλο 'Tasks' με στιλιστικά χαρακτηριστικά, ενώ η λίστα εργασιών (taskListView) προετοιμάζεται με την custom κλάση TaskCellFactory (Πάνω-Αριστερά). Η μέθοδος updateTaskList αναλαμβάνει τη δυναμική ενημέρωση της λίστας, οργανώνοντας τις εργασίες με βάση την ώρα έναρξης και προσθέτοντάς τες στη λίστα εργασιών (Κάτω). Το interface της κλάσης με το οποίο αλληλοεπιδρά ο χρήστης κατά τη λειτουργία της εφαρμογής (Πάνω-Δεξιά).

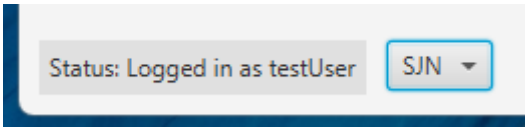
Η εφαρμογή ενσωματώνει ένα ολοκληρωμένο σύστημα πλοήγησης μέσω της κλάσης *MenuBarComponent*, η οποία δημιουργεί μια δομημένη μπάρα μενού στο πάνω μέρος του παραθύρου. Το μενού αποτελείται από τρεις βασικές κατηγορίες: "Task", "User" και "Help". Το μενού "Task" περιλαμβάνει επιλογές για τη δημιουργία νέας εργασίας, την προβολή των εργασιών του χρήστη και την έξοδο από την εφαρμογή, ενώ το μενού "User" διαχειρίζεται τις λειτουργίες σύνδεσης, αποσύνδεσης και δημιουργίας προφίλ. Η κατάσταση κάθε στοιχείου του μενού προσαρμόζεται δυναμικά μέσω της μεθόδου *updateUserMenuItems()* ανάλογα με το αν ο χρήστης είναι συνδεδεμένος ή όχι.



Εικόνα 24. Το μενού της εφαρμογής περιλαμβάνει επιλογές για διαχείριση εργασιών ('Task'), διαχείριση χρήστη ('User') και πληροφορίες εφαρμογής ('Help'), με δυνατότητες όπως δημιουργία προφίλ, σύνδεση, και προβολή πληροφοριών.

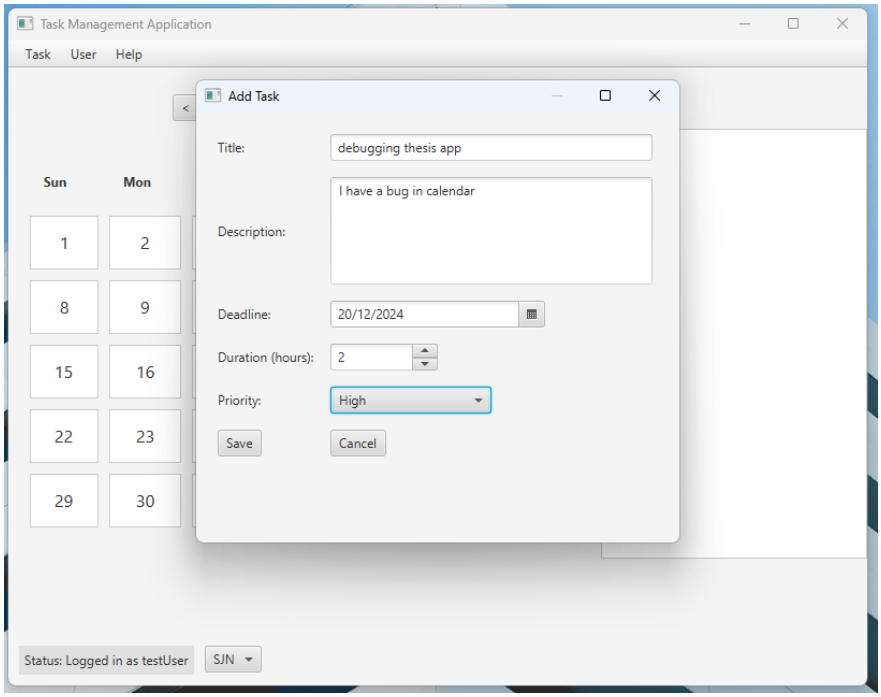
Στο κάτω μέρος της εφαρμογής, η κλάση *StatusBar* υλοποιεί μια γραμμή κατάστασης που εμφανίζει πληροφορίες για τον τρέχοντα συνδεδεμένο χρήστη, με ένα διακριτικό γκρι φόντο και κατάλληλη εσοχή για πιο εύκολη ανάγνωση. Δίπλα στη γραμμή κατάστασης τοποθετείται ένα

ChoiceBox για την επιλογή του αλγορίθμου χρονοπρογραμματισμού, το οποίο ενεργοποιείται μόνο όταν ο χρήστης είναι συνδεδεμένος.



Εικόνα 25. Η γραμμή κατάστασης της εφαρμογής που δείχνει αν ο χρήστης έχει ταυτοποιηθεί, καθώς και τον τρέχων αλγόριθμο χρονολόγησης.

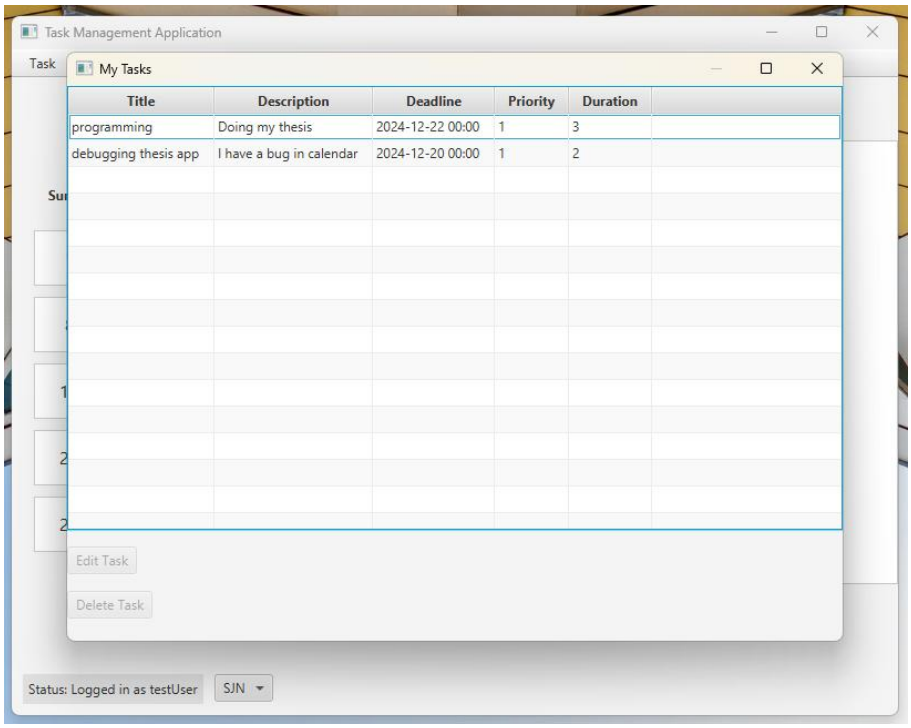
Το παράθυρο διαχείρισης εργασιών υλοποιείται μέσω της κλάσης *TaskManagementWindow* και προσφέρει μια ολοκληρωμένη φόρμα για τη δημιουργία και επεξεργασία εργασιών. Η διεπαφή χρησιμοποιεί ένα GridPane για την οργάνωση των στοιχείων εισόδου, με κατάλληλα διαστήματα (padding 20 pixels και gaps 15 pixels) για βέλτιστη αναγνωσιμότητα. Τα πεδία εισαγωγής περιλαμβάνουν ένα TextField για τον τίτλο με πλάτος 300 pixels, ένα TextArea για την περιγραφή με το ίδιο πλάτος και ύψος 100 pixels για την υποστήριξη πολλαπλών γραμμών, ένα DatePicker για την επιλογή προθεσμίας, ένα Spinner για τον καθορισμό της διάρκειας σε ώρες (με εύρος 1-24) και ένα ComboBox για την επιλογή προτεραιότητας με τρεις επιλογές (High, Medium, Low). Το παράθυρο λειτουργεί σε modal mode, εμποδίζοντας την αλληλεπίδραση με το κύριο παράθυρο μέχρι την ολοκλήρωση της επεξεργασίας, ενώ ενσωματώνει πλήρη επικύρωση των δεδομένων εισόδου πριν την αποθήκευση.



Εικόνα 26. Το παράθυρο της δημιουργίας νέων tasks ή επεξεργασίας των υπάρχοντων.

Το παράθυρο προβολής εργασιών χρήστη υλοποιείται μέσω της κλάσης *MyTasksWindow* και παρέχει μια συγκεντρωτική προβολή όλων των εργασιών του συνδεδεμένου χρήστη σε μορφή πίνακα. Ο πίνακας υλοποιείται με χρήση του TableView της JavaFX και περιλαμβάνει πέντε βασικές στήλες: τίτλο, περιγραφή, προθεσμία, προτεραιότητα και διάρκεια. Η

στήλη της προθεσμίας χρησιμοποιεί έναν προσαρμοσμένο CellValueFactory για τη μορφοποίηση της ημερομηνίας σε αναγνώσιμη μορφή (yyyy-MM-dd HH:mm). Κάτω από τον πίνακα υπάρχουν δύο κουμπιά, "Edit Task" και "Delete Task", τα οποία ενεργοποιούνται μόνο όταν έχει επιλεγεί κάποια εργασία. Το παράθυρο έχει διαστάσεις 700x500 pixels και υποστηρίζει αυτόματη αποεπιλογή εργασιών όταν ο χρήστης κάνει κλικ εκτός του πίνακα, χάρη σε έναν ειδικό MouseEvent handler στο VBox layout.



Εικόνα 27. Το παράθυρο που δίνει τη δυνατότητα στο χρήστη να βλέπει μεμονομένα τις εργασίες του και να τις διαχειρίζεται.

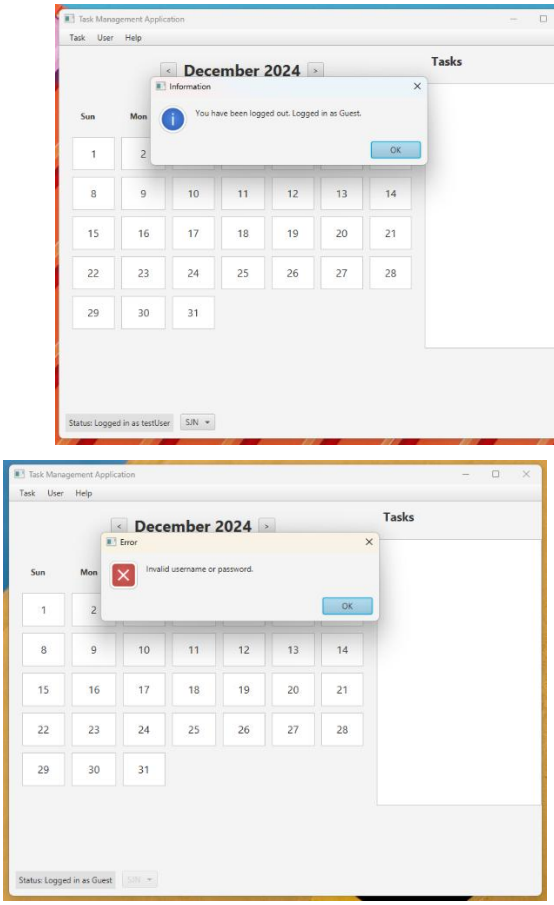
Η επικοινωνία μεταξύ των διαφόρων components του UI υλοποιείται μέσω ενός συστήματος αναφορών και callbacks που εξασφαλίζει τη συνεπή ενημέρωση όλων των στοιχείων της διεπαφής. Στο επίκεντρο αυτού του συστήματος βρίσκεται η κλάση *TaskManager*, η οποία λειτουργεί ως κεντρικός διαχειριστής των εργασιών και διατηρεί αναφορές τόσο στο Schedule όσο και στο *CalendarView*. Όταν πραγματοποιείται μια αλλαγή στις εργασίες (προσθήκη, επεξεργασία ή διαγραφή), ο *TaskManager* ενημερώνει το Schedule μέσω της μεθόδου *onTaskChanged()*, το οποίο με τη σειρά του προκαλεί την ανανέωση του *CalendarView* και του *TaskListPane*. Αντίστοιχα, όταν αλλάζει ο επιλεγμένος αλγόριθμος χρονοπρογραμματισμού, το Schedule ενημερώνει το *CalendarView* μέσω της μεθόδου *refresh()*, διασφαλίζοντας ότι η οπτική αναπαράσταση των εργασιών παραμένει συγχρονισμένη με την τρέχουσα κατάσταση του συστήματος.

```java
// Add a task for a specific user
public void addTask(String username, Task task) {  1 usage
    userTasks.computeIfAbsent(username, String k -> new ArrayList<>()).add(task);
    saveTasks();

    // Notify the schedule
    if (schedule != null) {
        schedule.onTaskChanged();
    }
    // Notify the calendar
    if (calendarView != null) {
        calendarView.refresh();
    }
}
```

Εικόνα 28. Η μέθοδος της δημιουργίας νέου task της κλάσης TaskManager που καλείται όταν ο χρήστης έχει συμπληρώσει τα πεδία της φόρμας και πατάει το 'Save'.

Η διαχείριση των γεγονότων χρήστη υλοποιείται ακολουθώντας το πρότυπο event-driven programming της JavaFX. Στην κλάση App, οι λειτουργίες του μενού συνδέονται με τους αντίστοιχους handlers μέσω της μεθόδου setOnAction(), όπως για παράδειγμα στη διαχείριση σύνδεσης/αποσύνδεσης χρήστη που υλοποιείται μέσω της κλάσης *UserActions*. Στο ημερολόγιο, κάθε κελί-ημέρα διαθέτει έναν MouseEvent handler που ενεργοποιείται με το κλικ και ενημερώνει το *TaskListPane* για τις εργασίες της επιλεγμένης ημέρας. Το ChoiceBox του αλγορίθμου χρονοπρογραμματισμού χρησιμοποιεί έναν ChangeListener που παρακολουθεί τις αλλαγές στην επιλογή του χρήστη και ενημερώνει αυτόματα το Schedule. Παράλληλα, όλα τα παράθυρα διαλόγου (login, task management) περιλαμβάνουν ενσωματωμένους μηχανισμούς επικύρωσης που ελέγχουν την εγκυρότητα των δεδομένων πριν την εκτέλεση οποιασδήποτε ενέργειας.

Η εφαρμογή ενσωματώνει ένα ολοκληρωμένο σύστημα διαχείρισης σφαλμάτων και ανατροφοδότησης προς τον χρήστη μέσω της χρήσης των διαλόγων Alert της JavaFX. Συγκεκριμένα, τα σφάλματα εμφανίζονται με διαλόγους τύπου Alert.AlertType.ERROR, ενώ τα επιτυχή μηνύματα με διαλόγους τύπου Alert.AlertType.INFORMATION.



Εικόνα 29. Δύο είδη παραθύρων ενημέρωσης του χρήστη. Αριστερά, η εφαρμογή ενημερώνει τον χρήστη ότι επιτυχώς αποσυνδέθηκε, ενώ, δεξιά, ότι έχει βάλει λάθος τα στοιχεία σύνδεσης.

Το σύστημα επικύρωσης στο *TaskManagementWindow* ελέγχει την πληρότητα και εγκυρότητα των δεδομένων πριν την αποθήκευση μιας εργασίας, εμφανίζοντας κατάλληλα μηνύματα λάθους όταν λείπουν υποχρεωτικά πεδία. Αντίστοιχα, στο σύστημα διαχείρισης χρηστών, η κλάση *UserActions* παρέχει εξειδικευμένα μηνύματα για περιπτώσεις όπως η αποτυχημένη σύνδεση, η προσπάθεια δημιουργίας λογαριασμού με όνομα που ήδη υπάρχει, ή η προσπάθεια διαχείρισης εργασιών χωρίς σύνδεση χρήστη. Τα μηνύματα αυτά εμφανίζονται σε modal παράθυρα που απαιτούν την προσοχή του χρήστη πριν συνεχίσει η αλληλεπίδραση με την εφαρμογή.

Η οπτική σχεδίαση της εφαρμογής βασίζεται σε μια συνεπή προσέγγιση στυλ που εφαρμόζεται σε όλα τα επιμέρους στοιχεία της διεπαφής. Στο ημερολόγιο, κάθε κελί-ημέρα διαμορφώνεται με λευκό φόντο και περίγραμμα σε απαλό γκρι χρώμα (#ccc), ενώ χρησιμοποιείται γραμματοσειρά μεγέθους 16 pixels για βέλτιστη αναγνωσιμότητα. Στο *TaskListPane*, κάθε εγγραφή εργασίας διαχωρίζεται με λεπτές γραμμές και διαθέτει κατάλληλο padding 10 pixels για καλύτερη οπτική ιεράρχηση της πληροφορίας. Η γραμμή κατάστασης χρησιμοποιεί ένα διακριτικό γκρι φόντο (#e0e0e0) που τη διαχωρίζει οπτικά από το υπόλοιπο περιεχόμενο, ενώ ο τίτλος του μήνα στο ημερολόγιο τονίζεται με έντονη γραμματοσειρά μεγέθους 24 pixels. Όλα τα παράθυρα

διαλόγου διατηρούν συνεπή αποστάσεις μεταξύ των στοιχείων τους, με προκαθορισμένα διαστήματα (gaps) 10-15 pixels για εξασφάλιση καθαρής και επαγγελματικής εμφάνισης.

Η εφαρμογή ενσωματώνει πολλαπλά χαρακτηριστικά που βελτιώνουν την προσβασιμότητα και τη συνολική εμπειρία χρήσης. Όλα τα διαδραστικά στοιχεία, όπως τα κουμπιά διαχείρισης εργασιών και τα στοιχεία του μενού, απενεργοποιούνται αυτόματα (setDisable) όταν δεν είναι διαθέσιμα, παρέχοντας σαφή οπτική ένδειξη για τις επιτρεπόμενες ενέργειες. Στο *MyTasksWindow*, τα κουμπιά επεξεργασίας και διαγραφής συνδέονται με την επιλογή εργασίας μέσω του bind στην ιδιότητα selectedItemProperty του TableView, εξασφαλίζοντας ότι ο χρήστης δεν μπορεί να εκτελέσει ενέργειες χωρίς να έχει επιλέξει πρώτα μια εργασία. Το *TaskManagementWindow* παρέχει προκαθορισμένες τιμές και περιορισμούς εύρους στα πεδία εισαγωγής, όπως το Spinner διάρκειας που περιορίζεται σε τιμές 1-24 ώρες, αποτρέποντας την εισαγωγή μη έγκυρων δεδομένων. Επιπλέον, όλα τα παράθυρα διαλόγου λειτουργούν σε modal mode, εστιάζοντας την προσοχή του χρήστη στην τρέχουσα εργασία και αποτρέποντας πιθανά σφάλματα από παράλληλες ενέργειες.

Η απόδοση και η αποκρισιμότητα της διεπαφής διασφαλίζονται μέσω μιας σειράς βελτιστοποιήσεων στην υλοποίηση των components. Στο *CalendarView*, η ανανέωση του περιεχομένου πραγματοποιείται επιλεκτικά μόνο στα επηρεαζόμενα στοιχεία μέσω της μεθόδου *refresh()*, αποφεύγοντας την πλήρη επανασχεδίαση του ημερολογίου. Το *TaskListPane* χρησιμοποιεί την τεχνική virtualization της JavaFX ListView, φορτώνοντας στη μνήμη μόνο τα ορατά στοιχεία της λίστας, γεγονός που επιτρέπει την αποδοτική διαχείριση μεγάλου αριθμού εργασιών. Η εφαρμογή υιοθετεί ασύγχρονη φόρτωση των δεδομένων κατά την εκκίνηση, με τον *TaskManager* να φορτώνει τις εργασίες παράλληλα με την αρχικοποίηση του UI. Επιπλέον, η χρήση του μοτίβου observer για την ενημέρωση των components εξασφαλίζει ότι οι ενημερώσεις της διεπαφής πραγματοποιούνται μόνο όταν είναι απαραίτητο, ελαχιστοποιώντας την επιβάρυνση του συστήματος και διατηρώντας την εφαρμογή αποκρίσιμη ακόμη και κατά τη διαχείριση πολύπλοκων χρονοπρογραμματισμών.

3.3 Υλοποίηση Αλγορίθμων Χρονολόγησης

Σε αυτή την ενότητα παρουσιάζεται η λεπτομερής υλοποίηση των τριών αλγορίθμων χρονολόγησης που ενσωματώθηκαν στην εφαρμογή: Shortest Job Next, Least Laxity First και Round Robin. Οι αλγόριθμοι υλοποιήθηκαν στην κλάση *Schedule* της εφαρμογής και προσαρμόστηκαν ώστε να ανταποκρίνονται στις ιδιαίτερες απαιτήσεις του συστήματος διαχείρισης εργασιών. Κάθε αλγόριθμος έχει βελτιστοποιηθεί για να διαχειρίζεται αποτελεσματικά τους περιορισμούς του συστήματος, όπως το όριο των 8 ωρών ημερήσιας εργασίας ανά χρήστη και το καθορισμένο ωράριο λειτουργίας του συστήματος. Η εναλλαγή μεταξύ των αλγορίθμων γίνεται δυναμικά μέσω του γραφικού περιβάλλοντος, επιτρέποντας στους χρήστες να επιλέξουν

τον καταλληλότερο αλγόριθμο για τις ανάγκες τους, ενώ το σύστημα αναλαμβάνει αυτόματα τον επαναπρογραμματισμό των εργασιών σύμφωνα με τη λογική του επιλεγμένου αλγορίθμου.

3.3.1 Shortest Job Next

Η υλοποίηση του αλγορίθμου Shortest Job Next στην εφαρμογή πραγματοποιήθηκε μέσω της μεθόδου *scheduleBySJN* στην κλάση *Schedule*. Ο αλγόριθμος τροποποιήθηκε ώστε να λαμβάνει υπόψη όχι μόνο τη διάρκεια των εργασιών, αλλά και τον χρόνο άφιξής τους στο σύστημα, διασφαλίζοντας έτσι μια πιο ρεαλιστική προσέγγιση στη διαχείριση των εργασιών. Η μέθοδος δέχεται ως είσοδο μια λίστα αντικειμένων *Task* και επιστρέφει μια λίστα από *ScheduledTask*, που περιέχει τις χρονοπρογραμματισμένες εργασίες με τους ακριβείς χρόνους έναρξης και λήξης τους.

Η διαδικασία χρονοπρογραμματισμού ξεκινά με την ταξινόμηση των εργασιών με βάση τον χρόνο άφιξής τους, χρησιμοποιώντας τη μέθοδο sort() της Java Collections Framework με έναν custom Comparator. Στη συνέχεια, δημιουργείται μια ουρά προτεραιότητας (PriorityQueue) που διατηρεί τις διαθέσιμες εργασίες ταξινομημένες με βάση τη διάρκειά τους. Η χρήση της ουράς προτεραιότητας εξασφαλίζει ότι πάντα επιλέγεται η συντομότερη εργασία από τις διαθέσιμες, ακολουθώντας την βασική αρχή του αλγορίθμου SJN.

Ο αλγόριθμος λειτουργεί επαναληπτικά, διατηρώντας έναν τρέχοντα χρόνο (currentTime) που ξεκινά από την καθορισμένη ώρα έναρξης του συστήματος (SYSTEM_START_TIME). Σε κάθε επανάληψη, ελέγχει ποιες εργασίες έχουν φτάσει στο σύστημα μέχρι τον τρέχοντα χρόνο και τις προσθέτει στην ουρά προτεραιότητας. Εάν υπάρχουν διαθέσιμες εργασίες στην ουρά, επιλέγει την συντομότερη (μέσω της μεθόδου *poll()*), τη χρονοπρογραμματεί στον τρέχοντα χρόνο και ενημερώνει τον τρέχοντα χρόνο προσθέτοντας τη διάρκεια της εργασίας που μόλις προγραμματίστηκε.

Μία σημαντική βελτίωση που προστέθηκε στην υλοποίηση είναι η διαχείριση των περιορισμών του συστήματος σχετικά με το ωράριο εργασίας. Συγκεκριμένα, η μέθοδος *distributeScheduledTasks* διασφαλίζει ότι κανένας χρήστης δεν προγραμματίζεται για περισσότερες από 8 ώρες εργασίας ημερησίως, όπως ορίζεται από τη σταθερά DAILY_USER_WORK_HOURS. Όταν μια εργασία δεν μπορεί να ολοκληρωθεί εντός των διαθέσιμων ωρών μιας ημέρας, διαχωρίζεται αυτόματα και το υπόλοιπο μέρος της μεταφέρεται στην επόμενη διαθέσιμη ημέρα.

Η εφαρμογή επίσης ενσωματώνει έναν μηχανισμό αποφυγής επικαλύψεων μέσω της μεθόδου *findNextAvailableTimeSlot*. Αυτή η μέθοδος εξετάζει το υπάρχον πρόγραμμα του χρήστη για μια συγκεκριμένη ημέρα και εντοπίζει το επόμενο διαθέσιμο χρονικό διάστημα που μπορεί να φιλοξενήσει μια νέα εργασία χωρίς να δημιουργεί συγκρούσεις με τις ήδη προγραμματισμένες εργασίες. Εάν δεν βρεθεί κατάλληλο διάστημα εντός της τρέχουσας ημέρας, η αναζήτηση συνεχίζεται στις επόμενες ημέρες μέχρι να βρεθεί κατάλληλος χρόνος.

Τέλος, ο αλγόριθμος SJN συνδέεται με το υπόλοιπο σύστημα μέσω του μηχανισμού ανανέωσης του προγράμματος. Κάθε φορά που προστίθεται, τροποποιείται ή διαγράφεται μια εργασία, καλείται η μέθοδος *rescheduleAll()* η οποία επαναϋπολογίζει ολόκληρο το πρόγραμμα. Αυτό εξασφαλίζει ότι διατηρείται πάντα η βέλτιστη σειρά εκτέλεσης των εργασιών σύμφωνα με τα κριτήρια του αλγορίθμου SJN. Η ενημέρωση του προγράμματος αντανακλάται άμεσα στη διεπαφή χρήστη μέσω της μεθόδου *refresh()* του *CalendarView*, επιτρέποντας στους χρήστες να βλέπουν τις αλλαγές σε πραγματικό χρόνο.

3.3.2 Least Laxity First

Η υλοποίηση του αλγορίθμου Least Laxity First πραγματοποιήθηκε στην εφαρμογή μέσω της μεθόδου *scheduleByLLF* στην κλάση *Schedule*. Ο LLF είναι ένας δυναμικός αλγόριθμος χρονοπρογραμματισμού που λαμβάνει αποφάσεις με βάση την έννοια της χαλαρότητας (laxity). Η χαλαρότητα μιας εργασίας υπολογίζεται ως η διαφορά μεταξύ του χρόνου που απομένει μέχρι την προθεσμία της και του χρόνου που απαιτείται για την εκτέλεσή της. Η υλοποίησή μας δέχεται ως είσοδο μια λίστα αντικειμένων *Task* και επιστρέφει μια λίστα χρονοπρογραμματισμένων εργασιών *ScheduledTask*.

Ο κεντρικός μηχανισμός του αλγορίθμου βασίζεται στη μέθοδο *calculateLaxity*, η οποία υπολογίζει τη χαλαρότητα κάθε εργασίας σε λεπτά για μεγαλύτερη ακρίβεια. Η μέθοδος λαμβάνει υπόψη τρεις βασικές παραμέτρους: τον χρόνο άφιξης της εργασίας (arrivalTime), την προθεσμία ολοκλήρωσης (deadline) και τον απαιτούμενο χρόνο εκτέλεσης (remainingExecutionTime). Για εργασίες που δεν έχουν φτάσει ακόμη στο σύστημα (currentTime < arrivalTime), η χαλαρότητα ορίζεται ως η μέγιστη δυνατή τιμή (Integer.MAX_VALUE), εξασφαλίζοντας έτσι ότι δεν θα επιλεγούν για εκτέλεση πριν από τον χρόνο άφιξής τους.

Η διαδικασία χρονοπρογραμματισμού υλοποιείται με έναν επαναληπτικό αλγόριθμο που διατηρεί μια λίστα με τις εναπομείνασες εργασίες (remainingTasks) και έναν τρέχοντα χρόνο (currentTime). Σε κάθε επανάληψη, ο αλγόριθμος υπολογίζει τη χαλαρότητα για όλες τις εναπομείνασες εργασίες χρησιμοποιώντας ένα Map (laxityMap) και επιλέγει την εργασία με τη μικρότερη χαλαρότητα μέσω της μεθόδου Collections.min(). Αυτή η προσέγγιση εξασφαλίζει ότι προτεραιότητα δίνεται στις εργασίες που έχουν τον μικρότερο "ελεύθερο" χρόνο μέχρι την προθεσμία τους.

Μια σημαντική βελτίωση στην υλοποίηση του LLF αποτελεί η διαχείριση των ανέφικτων χρονοπρογραμματισμών (infeasible scheduling). Συγκεκριμένα, όταν ο διαθέσιμος χρόνος μέχρι την προθεσμία μιας εργασίας είναι μικρότερος από τον απαιτούμενο χρόνο εκτέλεσης, η μέθοδος *calculateLaxity* επιστρέφει τη μέγιστη δυνατή τιμή (Integer.MAX_VALUE), υποδεικνύοντας ότι η εργασία αυτή δεν μπορεί να ολοκληρωθεί εγκαίρως. Αυτός ο μηχανισμός αποτρέπει τον χρονοπρογραμματισμό εργασιών που είναι αδύνατον να ολοκληρωθούν εντός των προθεσμιών τους, βελτιώνοντας έτσι τη συνολική αποτελεσματικότητα του συστήματος.

Η ενσωμάτωση του περιορισμού των 8 ωρών ημερήσιας εργασίας υλοποιείται μέσω της μεθόδου *distributeScheduledTasks*. Αφού ο LLF δημιουργήσει την αρχική ακολουθία εκτέλεσης των εργασιών, η μέθοδος αυτή αναλαμβάνει να κατανείμει τις εργασίες στις διαθέσιμες ημέρες, τηρώντας το όριο των 8 ωρών. Όταν μια εργασία δεν μπορεί να ολοκληρωθεί εντός του ημερήσιου ορίου, διασπάται αυτόματα και το υπόλοιπο μέρος της μεταφέρεται στην επόμενη διαθέσιμη ημέρα, διατηρώντας παράλληλα τη σχετική σειρά προτεραιότητας που καθορίστηκε από τον LLF.

Τέλος, η διασύνδεση του αλγορίθμου με το υπόλοιπο σύστημα επιτυγχάνεται μέσω του μηχανισμού ενημέρωσης του προγράμματος. Όταν ο χρήστης επιλέγει τον LLF ως αλγόριθμο χρονοπρογραμματισμού από το γραφικό περιβάλλον ή όταν γίνονται αλλαγές στις εργασίες, καλείται η μέθοδος *rescheduleAll()*. Η μέθοδος αυτή ενεργοποιεί τον LLF αλγόριθμο και ανανεώνει το πρόγραμμα, το οποίο στη συνέχεια αντανακλάται στο ημερολόγιο της εφαρμογής μέσω της μεθόδου *refresh()* του *CalendarView*, επιτρέποντας στους χρήστες να βλέπουν άμεσα τις αλλαγές στο πρόγραμμά τους.

3.3.3 Round Robin

Η υλοποίηση του αλγορίθμου Round Robin στην εφαρμογή πραγματοποιήθηκε μέσω της μεθόδου *scheduleByRR* στην κλάση *Schedule*. Ο αλγόριθμος υλοποιεί μια δίκαιη κατανομή του χρόνου εκτέλεσης μεταξύ των εργασιών, χρησιμοποιώντας ένα σταθερό κβάντο χρόνου (time quantum) για κάθε εργασία. Στην περίπτωση της εφαρμογής μας, το κβάντο χρόνου ορίστηκε στη μία ώρα, επιτρέποντας έτσι μια ισορροπημένη εναλλαγή μεταξύ των εργασιών χωρίς να δημιουργείται υπερβολική επιβάρυνση από τις εναλλαγές.

Η βασική δομή του αλγορίθμου στηρίζεται σε δύο ουρές: την activeQueue για τις ενεργές εργασίες που είναι έτοιμες προς εκτέλεση και την pendingQueue για τις εργασίες που δεν έχουν φτάσει ακόμη στον χρόνο άφιξής τους. Η pendingQueue υλοποιείται ως PriorityQueue που ταξινομεί τις εργασίες με βάση τον χρόνο άφιξής τους, εξασφαλίζοντας έτσι ότι οι εργασίες θα εισαχθούν στην activeQueue με τη σωστή χρονική σειρά. Για κάθε εργασία διατηρείται επίσης ένας μετρητής του εναπομείναντα χρόνου εκτέλεσης στο Map remainingDurations.

Μια σημαντική βελτίωση που προστέθηκε στον κλασικό αλγόριθμο Round Robin είναι η διαχείριση των χρονικών περιορισμών του συστήματος. Συγκεκριμένα, η υλοποίηση λαμβάνει υπόψη το καθορισμένο ωράριο λειτουργίας του συστήματος (SYSTEM_START_TIME και SYSTEM_END_TIME). Όταν μια εργασία δεν μπορεί να ολοκληρωθεί εντός του τρέχοντος ωραρίου λειτουργίας, ο αλγόριθμος αυτόματα μεταφέρει την εκτέλεσή της στην επόμενη εργάσιμη ημέρα, διατηρώντας παράλληλα τη δίκαιη κατανομή του χρόνου μεταξύ όλων των εργασιών.

Η διαχείριση των εργασιών που υπερβαίνουν το ημερήσιο όριο των 8 ωρών γίνεται μέσω της μεθόδου *distributeForRoundRobin*. Η μέθοδος αυτή εξασφαλίζει ότι κάθε χρήστης δεν θα προγραμματιστεί για περισσότερες από 8 ώρες εργασίας ανά ημέρα, όπως ορίζεται από τη σταθερά DAILY_USER_WORK_HOURS. Όταν προκύπτει υπέρβαση του ορίου, η εργασία

μεταφέρεται αυτόματα στην επόμενη διαθέσιμη ημέρα, διατηρώντας την ακολουθία εκτέλεσης του Round Robin και τη συνέπεια του προγράμματος.

Η υλοποίηση περιλαμβάνει επίσης έναν εξελιγμένο μηχανισμό διαχείρισης των χρονοθυρίδων μέσω της μεθόδου *findNextAvailableTimeSlot*. Ο μηχανισμός αυτός εξετάζει το υπάρχον πρόγραμμα και εντοπίζει τα επόμενα διαθέσιμα χρονικά διαστήματα που μπορούν να φιλοξενήσουν κάθε κβάντο χρόνου μιας εργασίας. Σε περίπτωση που μια εργασία δεν μπορεί να ολοκληρωθεί στο τρέχον κβάντο χρόνου, επιστρέφει στην ουρά για μελλοντική εκτέλεση, ενώ παράλληλα ενημερώνεται ο εναπομείνας χρόνος εκτέλεσής της στο Map remainingDurations.

Η ενσωμάτωση του αλγορίθμου στο συνολικό σύστημα γίνεται μέσω του μηχανισμού επαναχρονοπρογραμματισμού. Κάθε φορά που γίνεται αλλαγή στις εργασίες (προσθήκη, τροποποίηση ή διαγραφή) ή όταν ο χρήστης επιλέγει τον αλγόριθμο Round Robin από το γραφικό περιβάλλον, καλείται η μέθοδος *rescheduleAll()*. Αυτή με τη σειρά της ενεργοποιεί τον Round Robin αλγόριθμο και ανανεώνει το πρόγραμμα, το οποίο στη συνέχεια αντανακλάται στο ημερολόγιο της εφαρμογής μέσω της μεθόδου *refresh()* του *CalendarView*.

3.4 Διαχείριση Δεδομένων

Η διαχείριση των δεδομένων στην εφαρμογή υλοποιείται κυρίως μέσω της κλάσης TaskManager, η οποία αποτελεί τον κεντρικό μηχανισμό για την αποθήκευση και διαχείριση των εργασιών των χρηστών. Η κλάση αυτή χρησιμοποιεί μια δομή HashMap (*userTasks*) για την αποθήκευση των εργασιών, όπου το κλειδί είναι το όνομα χρήστη και η τιμή είναι μια λίστα με τις εργασίες του συγκεκριμένου χρήστη. Αυτή η προσέγγιση επιτρέπει την αποτελεσματική οργάνωση και πρόσβαση στα δεδομένα ανά χρήστη, διασφαλίζοντας παράλληλα τον διαχωρισμό των εργασιών μεταξύ διαφορετικών χρηστών του συστήματος.

Για την μόνιμη αποθήκευση των δεδομένων, η εφαρμογή χρησιμοποιεί τη βιβλιοθήκη Gson της Google, η οποία παρέχει λειτουργίες μετατροπής των αντικειμένων Java σε μορφή JSON και αντίστροφα. Συγκεκριμένα, υλοποιείται ένας προσαρμοσμένος TypeAdapter (*LocalDateTimeAdapter*) για τη σωστή σειριοποίηση και αποσειριοποίηση των αντικειμένων *LocalDateTime*, εξασφαλίζοντας την ακριβή αποθήκευση των χρονικών δεδομένων. Τα δεδομένα αποθηκεύονται σε ένα αρχείο *"tasks.json"*, το οποίο ενημερώνεται αυτόματα μετά από κάθε προσθήκη, τροποποίηση ή διαγραφή εργασίας.

Η αρχιτεκτονική του συστήματος υποστηρίζει τη διαχείριση των δεδομένων μέσω ενός παρατηρητή (observer pattern), όπου οι αλλαγές στα δεδομένα των εργασιών ενημερώνουν αυτόματα τα σχετικά components της εφαρμογής. Συγκεκριμένα, όταν πραγματοποιείται μια αλλαγή στα δεδομένα μέσω του *TaskManager*, ενημερώνονται αυτόματα τόσο το Schedule component για τον επαναπρογραμματισμό των εργασιών, όσο και το *CalendarView* για την

ανανέωση της οπτικής αναπαράστασης του ημερολογίου. Αυτή η προσέγγιση εξασφαλίζει τη συνέπεια των δεδομένων σε όλα τα επίπεδα της εφαρμογής.

Η εφαρμογή υλοποιεί έναν εξελιγμένο μηχανισμό διαχείρισης σφαλμάτων για την προστασία της ακεραιότητας των δεδομένων. Σε περίπτωση που εντοπιστεί πρόβλημα κατά την ανάγνωση του αρχείου JSON (JsonSyntaxException), το σύστημα έχει τη δυνατότητα να επαναφέρει τα δεδομένα σε μια σταθερή κατάσταση μέσω της μεθόδου *resetTasksFile()*. Επιπλέον, κατά την φόρτωση των δεδομένων, πραγματοποιείται έλεγχος για την ύπαρξη και εγκυρότητα του αρχείου tasks.json, ενώ σε περίπτωση κενού ή κατεστραμμένου αρχείου, το σύστημα δημιουργεί αυτόματα ένα νέο, καθαρό αρχείο για την αποθήκευση των μελλοντικών δεδομένων.

Η διαχείριση των χρονοπρογραμματισμένων εργασιών (ScheduledTasks) πραγματοποιείται μέσω ενός εξειδικευμένου συστήματος που συνδυάζει τις κλάσεις *Schedule* και *ScheduledTaskWithUser*. Το σύστημα διατηρεί δύο επίπεδα προγραμματισμού: ένα για τις εργασίες ανά χρήστη (userSchedules) και ένα για το συνολικό σύστημα (systemSchedule). Αυτή η διπλή καταγραφή επιτρέπει τόσο την αποτελεσματική διαχείριση των περιορισμών χρόνου ανά χρήστη (8 ώρες ημερησίως) όσο και τη συνολική εποπτεία του προγράμματος. Τα δεδομένα του προγραμματισμού αποθηκεύονται σε ένα ξεχωριστό αρχείο *"schedule_output.txt"* με αναλυτικές πληροφορίες για κάθε προγραμματισμένη εργασία.

Για τη διαχείριση των προφίλ χρηστών, η εφαρμογή χρησιμοποιεί την κλάση *ProfileManager*, η οποία διατηρεί τα δεδομένα των χρηστών σε ένα ξεχωριστό αρχείο *"user_profiles.json"*. Η κλάση αυτή υλοποιεί μεθόδους για τη δημιουργία νέων προφίλ, την επικύρωση των διαπιστευτηρίων κατά τη σύνδεση και τον έλεγχο διαθεσιμότητας ονομάτων χρήστη. Τα δεδομένα των προφίλ αποθηκεύονται σε μορφή HashMap, όπου το κλειδί είναι το όνομα χρήστη και η τιμή είναι ο κωδικός πρόσβασης. Η διαχείριση της τρέχουσας συνεδρίας χρήστη γίνεται μέσω της κλάσης UserManager, η οποία διατηρεί πληροφορίες για τον συνδεδεμένο χρήστη και επιτρέπει την κατάλληλη προσαρμογή της διεπαφής και των λειτουργιών της εφαρμογής.

ΚΕΦΑΛΑΙΟ 4. ΕΛΕΓΧΟΣ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

Ο έλεγχος λογισμικού αποτελεί ένα κρίσιμο στάδιο στον κύκλο ζωής ανάπτυξης εφαρμογών, καθώς διασφαλίζει την ορθή λειτουργία και την αξιοπιστία του τελικού προϊόντος. Στο παρόν κεφάλαιο παρουσιάζεται η διαδικασία ελέγχου της εφαρμογής διαχείρισης εργασιών, με ιδιαίτερη έμφαση στην αξιολόγηση των τριών υλοποιημένων αλγορίθμων χρονοπρογραμματισμού (SJN, LLF και RR). Η διαδικασία ελέγχου περιλαμβάνει τόσο τον έλεγχο των επιμέρους λειτουργικών μονάδων όσο και τον έλεγχο του συστήματος ως σύνολο, με στόχο την επαλήθευση της ορθής λειτουργίας του συστήματος και την αξιολόγηση της αποτελεσματικότητας των διαφορετικών αλγορίθμων χρονοπρογραμματισμού. Επιπλέον, εξετάζεται η συμμόρφωση με τις προδιαγραφές του συστήματος, όπως ο περιορισμός του 8ώρου εργασίας και η σωστή διαχείριση των προτεραιοτήτων των εργασιών. Τα αποτελέσματα του ελέγχου παρουσιάζονται αναλυτικά και αξιολογούνται με βάση συγκεκριμένες μετρικές απόδοσης και ποιότητας.

4.1 Μεθοδολογία Ελέγχου

Η διαδικασία ελέγχου της εφαρμογής επικεντρώθηκε στην αξιολόγηση της ορθής λειτουργίας των τριών αλγορίθμων χρονοπρογραμματισμού (SJN, LLF και RR) και της συνολικής λειτουργικότητας του συστήματος. Ο έλεγχος πραγματοποιήθηκε σε τρία επίπεδα:

1. Έλεγχος Μεμονωμένων Λειτουργιών (Component Testing):
   - Έλεγχος κάθε component ξεχωριστά μετά την υλοποίησή του
   - Επιβεβαίωση της ορθής λειτουργίας του UI και των αλληλεπιδράσεών του
   - Έλεγχος των λειτουργιών διαχείρισης χρηστών (εγγραφή, σύνδεση, αποσύνδεση)
2. Έλεγχος Αλγορίθμων Χρονοπρογραμματισμού:
   - Δοκιμή κάθε αλγορίθμου με διαφορετικά σενάρια εργασιών
   - Επαλήθευση της ορθής εφαρμογής των κανόνων κάθε αλγορίθμου
   - Έλεγχος των περιορισμών (π.χ., 8ωρο ανά χρήστη) και της λειτουργίας του συστήματος σε 24ωρη βάση
3. Έλεγχος Ολοκληρωμένου Συστήματος:
   - Επαλήθευση της συνολικής λειτουργίας της εφαρμογής
   - Έλεγχος της αλληλεπίδρασης μεταξύ των διαφόρων components
   - Δοκιμή σεναρίων πολλαπλών χρηστών και ταυτόχρονων εργασιών

Η μεθοδολογία ελέγχου βασίστηκε στην άμεση παρατήρηση της λειτουργίας της εφαρμογής και στην ανάλυση των αποτελεσμάτων που καταγράφονται στο αρχείο schedule_output.txt. Συγκεκριμένα:

- Χρησιμοποιήθηκαν εκτυπώσεις στην κονσόλα για τον εντοπισμό σφαλμάτων
- Αναλύθηκε το αρχείο καταγραφής του χρονοπρογραμματισμού για την επαλήθευση της ορθής κατανομής των εργασιών
- Εφαρμόστηκε επαναληπτική διαδικασία ελέγχου και διόρθωσης για τη βελτιστοποίηση της λειτουργίας των αλγορίθμων

Τα κριτήρια επιτυχίας για τον έλεγχο περιλάμβαναν:

- Την ορθή λειτουργία του UI χωρίς σφάλματα
- Την τήρηση των προδιαγραφών των αλγορίθμων χρονοπρογραμματισμού
- Την αποτελεσματική διαχείριση των περιορισμών του συστήματος (χρονικοί περιορισμοί, όρια χρηστών)
- Την ακριβή καταγραφή και παρουσίαση του προγράμματος εργασιών

Η διαδικασία ελέγχου ήταν συνεχής κατά τη διάρκεια της ανάπτυξης, επιτρέποντας την έγκαιρη αναγνώριση και διόρθωση σφαλμάτων και την βελτιστοποίηση της λειτουργίας της εφαρμογής.

4.2 Σενάρια Ελέγχου

Στην παρούσα ενότητα παρουσιάζονται τα σενάρια ελέγχου που εκτελέστηκαν για την επαλήθευση της ορθής λειτουργίας των τριών αλγορίθμων χρονοπρογραμματισμού. Για κάθε αλγόριθμο πραγματοποιήθηκαν διαφορετικά σενάρια δοκιμών με στόχο την επιβεβαίωση της συμμόρφωσής τους με τις θεωρητικές προδιαγραφές τους και την αποτελεσματική διαχείριση των εργασιών του συστήματος. Τα αποτελέσματα των δοκιμών καταγράφηκαν και αναλύθηκαν για την αξιολόγηση της απόδοσης κάθε αλγορίθμου.

4.2.1 Έλεγχος Shortest Job Next

Ο έλεγχος του αλγορίθμου SJN πραγματοποιήθηκε μέσω εννέα διαφορετικών σεναρίων που σχεδιάστηκαν για να αξιολογήσουν την απόδοση και την ορθή λειτουργία του αλγορίθμου σε διάφορες συνθήκες. Τα σενάρια χωρίστηκαν σε τρεις βασικές κατηγορίες: βασικά σενάρια, σενάρια πολλαπλών χρηστών και ακραίες περιπτώσεις.

Στα βασικά σενάρια, ελέγχθηκε η συμπεριφορά του αλγορίθμου σε περιπτώσεις εργασιών ίσης και διαφορετικής διάρκειας. Για παράδειγμα, σε ένα σενάριο δοκιμής δύο εργασίες

με διάρκειες 4 και 1 ώρες αντίστοιχα, ίδια προτεραιότητα και ταυτόχρονη άφιξη (08:00) υποβλήθηκαν στο σύστημα. Το πρόγραμμα που δημιουργήθηκε προγραμμάτισε τη σύντομη εργασία ("Short Task") στο διάστημα 09:00-10:00 και την μεγαλύτερη σε διάρκεια εργασία ("Long Task") στο διάστημα 10:00-14:00, επιβεβαιώνοντας την ορθή εφαρμογή της βασικής αρχής του αλγορίθμου SJN.

Εικόνα 30. Σενάριο με εργασίες με διαφορετική διάρκεια.

Τα σενάρια πολλαπλών χρηστών επιβεβαίωσαν ότι το σύστημα διατηρεί ανεξάρτητο προγραμματισμό για κάθε χρήστη, τηρώντας το όριο των 8 ωρών ημερησίως ανά χρήστη, ενώ επιτρέπει την ταυτόχρονη εκτέλεση εργασιών διαφορετικών χρηστών. Σε περιπτώσεις υπέρβασης του ημερήσιου ορίου, οι επιπλέον εργασίες προγραμματίζονται αυτόματα για την επόμενη διαθέσιμη ημέρα.

Στις ακραίες περιπτώσεις, ελέγχθηκε η συμπεριφορά του συστήματος σε διάφορα σενάρια, με πιο χαρακτηριστικό αυτό της σύγκρουσης μεταξύ προτεραιότητας και διάρκειας. Συγκεκριμένα, δοκιμάστηκε ένα σενάριο όπου μια εργασία μεγάλης διάρκειας (4 ώρες) με χαμηλή προτεραιότητα (3) και μια εργασία μικρής διάρκειας (1 ώρα) με υψηλή προτεραιότητα (1) υποβλήθηκαν ταυτόχρονα. Το παραγόμενο πρόγραμμα έδωσε προτεραιότητα στη σύντομη εργασία (09:00-10:00) παρά τη χαμηλότερη προτεραιότητά της, ακολουθούμενη από τη

μεγαλύτερη εργασία (10:00-14:00), επιβεβαιώνοντας ότι ο αλγόριθμος SJN βασίζει τις αποφάσεις του αποκλειστικά στη διάρκεια των εργασιών.

Εικόνα 31. Σενάριο που ελέγχει πως το σύστημα συγκρίνει το priority με το duration

Τα αποτελέσματα όλων των δοκιμών καταγράφηκαν στο αρχείο schedule_output.txt, επιτρέποντας τη λεπτομερή ανάλυση της συμπεριφοράς του αλγορίθμου και την επιβεβαίωση της ορθής υλοποίησής του σύμφωνα με τις θεωρητικές προδιαγραφές του SJN.

4.2.2 Έλεγχος Least Laxity First

Ο έλεγχος του αλγορίθμου Least Laxity First LLF πραγματοποιήθηκε μέσω διαφόρων σεναρίων που σχεδιάστηκαν για να αξιολογήσουν την ικανότητα του αλγορίθμου να προγραμματίζει εργασίες με βάση το περιθώριο χαλαρότητας (laxity). Τα σενάρια περιλάμβαναν περιπτώσεις με διαφορετικές προθεσμίες, διάρκειες εργασιών και χρόνους άφιξης, εστιάζοντας στην αποτελεσματική διαχείριση των χρονικών περιορισμών.

Στα βασικά σενάρια, δοκιμάστηκε η συμπεριφορά του αλγορίθμου με εργασίες που είχαν διαφορετική χαλαρότητα. Για παράδειγμα, σε ένα σενάριο δοκιμής, τρεις εργασίες με ίδια ώρα άφιξης (09:00) αλλά διαφορετικές προθεσμίες (11:00, 14:00 και 17:00) και διάρκειες (1, 2 και 3 ώρες αντίστοιχα) υποβλήθηκαν στο σύστημα. Το πρόγραμμα που παράχθηκε προγραμμάτισε

πρώτα την εργασία με τη μικρότερη χαλαρότητα (προθεσμία μείον τρέχων χρόνος μείον διάρκεια), επιβεβαιώνοντας την ορθή εφαρμογή της βασικής αρχής του αλγορίθμου LLF.

Σε πιο σύνθετα σενάρια, εξετάστηκε η συμπεριφορά του αλγορίθμου σε περιπτώσεις όπου οι εργασίες είχαν διαφορετικούς χρόνους άφιξης και επικαλυπτόμενες προθεσμίες. Ιδιαίτερη έμφαση δόθηκε στην τήρηση του περιορισμού των 8 ωρών εργασίας ανά ημέρα, με τον αλγόριθμο να μεταφέρει αυτόματα τις εργασίες στην επόμενη διαθέσιμη ημέρα όταν υπερβαίνεται το ημερήσιο όριο. Επιπλέον, ο αλγόριθμος επέδειξε σωστή διαχείριση των περιπτώσεων όπου εργασίες έπρεπε να προγραμματιστούν μόνο μετά την άφιξή τους στο σύστημα, διατηρώντας παράλληλα τη βέλτιστη σειρά με βάση τη χαλαρότητα.

Στις ακραίες περιπτώσεις, ελέγχθηκε η αντιμετώπιση εργασιών με μηδενική ή αρνητική χαλαρότητα, καθώς και περιπτώσεις όπου πολλαπλές εργασίες είχαν την ίδια προθεσμία αλλά διαφορετικές διάρκειες. Σε αυτές τις περιπτώσεις, ο αλγόριθμος έδωσε προτεραιότητα στις εργασίες με τη μικρότερη χαλαρότητα, ενώ σε περιπτώσεις ίσης χαλαρότητας, προτιμήθηκαν οι εργασίες μικρότερης διάρκειας για τη βελτιστοποίηση του συνολικού προγράμματος.

Όλα τα αποτελέσματα των δοκιμών καταγράφηκαν στο αρχείο schedule_output.txt, επιτρέποντας τη λεπτομερή ανάλυση και επιβεβαίωση της ορθής λειτουργίας του αλγορίθμου LLF σύμφωνα με τις θεωρητικές του προδιαγραφές.

4.2.3 Έλεγχος Round Robin

Ο έλεγχος του αλγορίθμου Round Robin πραγματοποιήθηκε μέσω διαφορετικών σεναρίων που σχεδιάστηκαν για να αξιολογήσουν την απόδοση και την ορθή λειτουργία του αλγορίθμου με έμφαση στη δίκαιη κατανομή του χρόνου εκτέλεσης και το σεβασμό των χρόνων άφιξης των εργασιών. Τα σενάρια χωρίστηκαν σε τρεις βασικές κατηγορίες: βασικά σενάρια εναλλαγής εργασιών, σενάρια διαφορετικών χρόνων άφιξης και σενάρια με περιορισμούς συστήματος. Στα βασικά σενάρια, ελέγχθηκε η θεμελιώδης λειτουργία του αλγορίθμου RR σχετικά με την εναλλαγή των εργασιών. Για παράδειγμα, σε ένα σενάριο δύο εργασίες με διάρκειες 5 και 6 ώρες αντίστοιχα και ταυτόχρονη άφιξη (09:00) υποβλήθηκαν στο σύστημα. Το πρόγραμμα που δημιουργήθηκε εναλλάσσει τις εργασίες κάθε μία ώρα (quantum), ξεκινώντας με την πρώτη εργασία στο διάστημα 09:00-10:00, συνεχίζοντας με τη δεύτερη στο 10:00-11:00, και ούτω καθεξής, επιβεβαιώνοντας την ορθή εφαρμογή της αρχής της δίκαιης χρονοδρομολόγησης.

Τα σενάρια διαφορετικών χρόνων άφιξης επέδειξαν την ικανότητα του αλγορίθμου να διαχειρίζεται εργασίες που εισέρχονται στο σύστημα σε διαφορετικές χρονικές στιγμές. Συγκεκριμένα, σε ένα χαρακτηριστικό σενάριο, μια εργασία με χρόνο άφιξης 09:00 και μια δεύτερη με χρόνο άφιξης 11:00 υποβλήθηκαν στο σύστημα. Ο αλγόριθμος εκτέλεσε αποκλειστικά την πρώτη εργασία μέχρι την άφιξη της δεύτερης στις 11:00, οπότε και ξεκίνησε η εναλλαγή μεταξύ των δύο εργασιών. Αυτό επιβεβαίωσε ότι ο αλγόριθμος σέβεται τους χρόνους άφιξης και προσαρμόζει δυναμικά το πρόγραμμα εκτέλεσης. Στα σενάρια περιορισμών συστήματος,

ελέγχθηκε η συμπεριφορά του αλγορίθμου σε σχέση με το ημερήσιο όριο των 8 ωρών ανά χρήστη και τα όρια λειτουργίας του συστήματος (09:00-23:59). Στην περίπτωση υπέρβασης του ημερήσιου ορίου, οι εναπομείνασες εργασίες μεταφέρονται αυτόματα στην επόμενη διαθέσιμη ημέρα, διατηρώντας τη σειρά εναλλαγής τους. Επιπλέον, επιβεβαιώθηκε ότι ο αλγόριθμος RR διατηρεί το quantum του ενός ώρας ακόμα και όταν οι εργασίες μεταφέρονται σε επόμενη ημέρα.

Η συνολική αξιολόγηση των αποτελεσμάτων επιβεβαίωσε την ορθή υλοποίηση του αλγορίθμου Round Robin σύμφωνα με τις θεωρητικές του προδιαγραφές. Ο αλγόριθμος επέδειξε συνέπεια στη διατήρηση της δίκαιης χρονοδρομολόγησης, στο σεβασμό των χρόνων άφιξης και στην τήρηση των περιορισμών του συστήματος. Ιδιαίτερα σημαντική ήταν η επιβεβαίωση της ομαλής μετάβασης μεταξύ διαφορετικών καταστάσεων του συστήματος, όπως η είσοδος νέων εργασιών και η αλλαγή ημέρας, χωρίς να διαταράσσεται η βασική λειτουργία του αλγορίθμου.

4.3 Προβλήματα και προκλήσεις

Κατά την υλοποίηση και τον έλεγχο της εφαρμογής, αντιμετωπίστηκαν διάφορες προκλήσεις που απαιτούσαν προσεκτικό σχεδιασμό και αντιμετώπιση. Μία από τις κύριες δυσκολίες ήταν η ορθή υλοποίηση των αλγορίθμων χρονολόγησης, καθώς έπρεπε να διασφαλιστεί η σωστή λειτουργία τους σε διάφορα σενάρια χρήσης. Ιδιαίτερη προσοχή δόθηκε στην υλοποίηση του Round Robin scheduler, ο οποίος έπρεπε να διαχειρίζεται σωστά τις εργασίες σε κβάντα χρόνου, διατηρώντας παράλληλα τον περιορισμό των 8 ωρών ημερήσιας εργασίας ανά χρήστη.

Ένα επιπλέον ζήτημα που έπρεπε να αντιμετωπιστεί ήταν η αποτελεσματική διαχείριση των εργασιών όταν ξεπερνούσαν το ημερήσιο όριο εργασίας. Σε αυτές τις περιπτώσεις, έπρεπε να διασφαλιστεί ότι οι εργασίες μεταφέρονται σωστά στην επόμενη διαθέσιμη ημέρα, διατηρώντας τη σειρά προτεραιότητας και τις απαιτήσεις του εκάστοτε αλγορίθμου χρονολόγησης. Η υλοποίηση αυτής της λειτουργικότητας απαιτούσε προσεκτικό χειρισμό των χρονικών διαστημάτων και των περιορισμών του συστήματος.

Η διαχείριση της κατάστασης του συστήματος και των δεδομένων αποτέλεσε επίσης μια σημαντική πρόκληση. Έπρεπε να διασφαλιστεί η συνέπεια των δεδομένων κατά την αποθήκευση και ανάκτησή τους από το αρχείο JSON, ειδικά όταν γίνονταν αλλαγές στον προγραμματισμό των εργασιών. Επιπλέον, η διαχείριση των αντικειμένων Task και ScheduledTask απαιτούσε προσεκτικό χειρισμό για την αποφυγή διπλοεγγραφών και την ορθή ενημέρωση των πληροφοριών χρονοπρογραμματισμού.

Μια επιπρόσθετη πρόκληση ήταν η ανάπτυξη της διεπαφής χρήστη με τρόπο που να είναι φιλική και εύχρηστη. Η υλοποίηση του ημερολογίου και της λίστας εργασιών απαιτούσε προσεκτικό σχεδιασμό ώστε να παρουσιάζονται οι πληροφορίες με σαφήνεια και να επιτρέπεται η εύκολη διαχείριση των εργασιών. Ιδιαίτερη έμφαση δόθηκε στην ορθή απεικόνιση των

χρονοπρογραμματισμένων εργασιών και στην άμεση ενημέρωση της διεπαφής μετά από κάθε αλλαγή στον προγραμματισμό.

Επιπλέον, η διαχείριση των χρηστών και των δικαιωμάτων τους αποτέλεσε ένα σημαντικό κομμάτι της υλοποίησης. Έπρεπε να διασφαλιστεί ότι κάθε χρήστης έχει πρόσβαση μόνο στις δικές του εργασίες και ότι οι περιορισμοί του συστήματος (όπως το όριο των 8 ωρών ημερήσιας εργασίας) εφαρμόζονται σωστά για κάθε χρήστη ξεχωριστά. Η υλοποίηση του συστήματος ελέγχου πρόσβασης και η διαχείριση των συνεδριών των χρηστών απαιτούσε προσεκτικό σχεδιασμό για την αποφυγή προβλημάτων ασφάλειας και συνέπειας των δεδομένων.

Τέλος, η διασφάλιση της σωστής λειτουργίας του συστήματος σε διάφορα σενάρια χρήσης απαιτούσε εκτενείς ελέγχους και δοκιμές. Ιδιαίτερη προσοχή δόθηκε στον έλεγχο των διαφορετικών αλγορίθμων χρονολόγησης και στην επαλήθευση της ορθής λειτουργίας τους σε περιπτώσεις υψηλού φόρτου εργασίας ή ειδικών συνθηκών. Η αντιμετώπιση των προβλημάτων που εντοπίστηκαν κατά τη διάρκεια των δοκιμών οδήγησε σε βελτιώσεις και προσαρμογές του κώδικα για την επίτευξη καλύτερης απόδοσης και αξιοπιστίας του συστήματος.

ΚΕΦΑΛΑΙΟ 5. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Το πέμπτο κεφάλαιο συνοψίζει τα βασικά συμπεράσματα που προέκυψαν από την ανάπτυξη και αξιολόγηση του συστήματος διαχείρισης εργασιών, ενώ παράλληλα προτείνει μελλοντικές επεκτάσεις και βελτιώσεις. Η ανάπτυξη της εφαρμογής, που ενσωματώνει τρεις διαφορετικούς αλγόριθμους χρονοπρογραμματισμού παρείχε σημαντικά συμπεράσματα σχετικά με την αποτελεσματικότητα και την καταλληλότητα του κάθε αλγορίθμου σε διαφορετικά σενάρια χρήσης. Η εμπειρία από την υλοποίηση και τον έλεγχο του συστήματος ανέδειξε τόσο τα πλεονεκτήματα όσο και τους περιορισμούς της τρέχουσας προσέγγισης, δημιουργώντας παράλληλα ευκαιρίες για περαιτέρω βελτιώσεις και επεκτάσεις. Στις ενότητες που ακολουθούν, αναλύονται λεπτομερώς τα συμπεράσματα που προέκυψαν και παρουσιάζονται προτάσεις για μελλοντικές επεκτάσεις που θα μπορούσαν να ενισχύσουν περαιτέρω τη λειτουργικότητα και την αποτελεσματικότητα του συστήματος.

5.1 Συμπεράσματα

Η ανάπτυξη και υλοποίηση της εφαρμογής διαχείρισης εργασιών με τρεις διαφορετικούς αλγορίθμους χρονοπρογραμματισμού οδήγησε σε σημαντικά συμπεράσματα τόσο σε τεχνικό επίπεδο όσο και σε επίπεδο χρηστικότητας. Η εφαρμογή πέτυχε τον βασικό της στόχο, παρέχοντας ένα ολοκληρωμένο περιβάλλον για τη διαχείριση και τον προγραμματισμό εργασιών, ενώ παράλληλα προσέφερε στους χρήστες την ευελιξία επιλογής του καταλληλότερου αλγορίθμου για τις ανάγκες τους. Η χρήση του JavaFX για την ανάπτυξη της διεπαφής χρήστη αποδείχθηκε επιτυχής επιλογή, καθώς επέτρεψε τη δημιουργία ενός σύγχρονου και διαισθητικού περιβάλλοντος που διευκολύνει την αλληλεπίδραση με το σύστημα.

Η υλοποίηση των τριών αλγορίθμων χρονοπρογραμματισμού (SJN, LLF και Round Robin) ανέδειξε τα ιδιαίτερα χαρακτηριστικά και τις δυνατότητες του καθενός σε διαφορετικά σενάρια χρήσης. Ο αλγόριθμος SJN αποδείχθηκε ιδιαίτερα αποτελεσματικός στην ελαχιστοποίηση του συνολικού χρόνου αναμονής των εργασιών, ειδικά σε περιπτώσεις όπου υπήρχαν πολλές σύντομες εργασίες. Ο LLF παρείχε καλύτερη διαχείριση των προθεσμιών, εξασφαλίζοντας ότι οι εργασίες με περιορισμένο χρονικό περιθώριο λαμβάνουν την απαραίτητη προτεραιότητα. Ο Round Robin, με το σταθερό κβάντο χρόνου του μίας ώρας, πέτυχε δίκαιη κατανομή του χρόνου εκτέλεσης μεταξύ των εργασιών, αποτρέποντας την μονοπώληση των πόρων του συστήματος από μεμονωμένες εργασίες.

Η εφαρμογή του περιορισμού των 8 ωρών ημερήσιας εργασίας ανά χρήστη αποτέλεσε μια σημαντική πρόκληση που αντιμετωπίστηκε επιτυχώς μέσω του μηχανισμού αυτόματης μεταφοράς εργασιών στην επόμενη διαθέσιμη ημέρα. Αυτή η προσέγγιση όχι μόνο διασφάλισε

τη συμμόρφωση με τους εργασιακούς περιορισμούς, αλλά παράλληλα επέτρεψε την αποτελεσματική διαχείριση του φόρτου εργασίας σε βάθος χρόνου. Η υλοποίηση αυτού του μηχανισμού απαίτησε προσεκτικό σχεδιασμό και εκτενείς δοκιμές για να εξασφαλιστεί η ομαλή λειτουργία του συστήματος σε διάφορα σενάρια χρήσης.

Η αρχιτεκτονική του συστήματος, βασισμένη στο πρότυπο Model-View-Controller, αποδείχθηκε ιδιαίτερα αποτελεσματική στη διαχείριση της πολυπλοκότητας της εφαρμογής. Ο διαχωρισμός των ευθυνών μεταξύ των διαφορετικών επιπέδων επέτρεψε την ευκολότερη συντήρηση και επέκταση του κώδικα, ενώ παράλληλα διευκόλυνε τον εντοπισμό και τη διόρθωση σφαλμάτων κατά τη φάση της ανάπτυξης. Η χρήση του Gradle ως εργαλείου διαχείρισης εξαρτήσεων και αυτοματοποίησης της διαδικασίας κατασκευής συνέβαλε σημαντικά στην αποτελεσματική οργάνωση του project και στην απλοποίηση της διαδικασίας ανάπτυξης.

Η διαχείριση των δεδομένων μέσω αρχείων JSON αποδείχθηκε επαρκής για τις ανάγκες της εφαρμογής, προσφέροντας ευελιξία στην αποθήκευση και ανάκτηση των πληροφοριών χρηστών και εργασιών. Η χρήση της βιβλιοθήκης Gson διευκόλυνε σημαντικά τη διαδικασία σειριοποίησης και αποσειριοποίησης των δεδομένων, ενώ ο μηχανισμός διαχείρισης σφαλμάτων που υλοποιήθηκε εξασφάλισε την ακεραιότητα των δεδομένων ακόμη και σε περιπτώσεις απρόβλεπτων συνθηκών. Ωστόσο, για μεγαλύτερη κλίμακα χρήσης, η μετάβαση σε ένα πιο ολοκληρωμένο σύστημα διαχείρισης βάσεων δεδομένων θα μπορούσε να προσφέρει καλύτερη απόδοση και περισσότερες δυνατότητες.

Η διαδικασία ελέγχου της εφαρμογής ανέδειξε τη σημασία του συστηματικού testing σε όλα τα επίπεδα του συστήματος. Οι εκτενείς δοκιμές των αλγορίθμων χρονοπρογραμματισμού σε διάφορα σενάρια χρήσης επέτρεψαν τον εντοπισμό και τη διόρθωση σημαντικών ζητημάτων, βελτιώνοντας την αξιοπιστία του συστήματος. Ιδιαίτερα σημαντική αποδείχθηκε η δοκιμή ακραίων περιπτώσεων (edge cases) για κάθε αλγόριθμο, καθώς αποκάλυψε πιθανές αδυναμίες και οδήγησε σε βελτιώσεις στη λογική του χρονοπρογραμματισμού.

Η εμπειρία χρήστη αποτέλεσε βασική προτεραιότητα κατά την ανάπτυξη της εφαρμογής, με ιδιαίτερη έμφαση στην απλότητα και τη λειτουργικότητα της διεπαφής. Το σύστημα ημερολογίου, σε συνδυασμό με την εύχρηστη λίστα εργασιών, προσέφερε στους χρήστες μια ξεκάθαρη εικόνα του προγράμματός τους και των επερχόμενων υποχρεώσεών τους. Η δυνατότητα εναλλαγής μεταξύ των τριών αλγορίθμων χρονοπρογραμματισμού μέσω ενός απλού μενού επιλογής επέτρεψε στους χρήστες να πειραματιστούν με διαφορετικές στρατηγικές οργάνωσης των εργασιών τους, προσαρμόζοντας το σύστημα στις προσωπικές τους ανάγκες και προτιμήσεις.

Συνολικά, η ανάπτυξη της εφαρμογής απέδειξε ότι είναι εφικτή η δημιουργία ενός αποτελεσματικού συστήματος διαχείρισης εργασιών που συνδυάζει προηγμένους αλγόριθμους χρονοπρογραμματισμού με μια φιλική προς το χρήστη διεπαφή. Η επιτυχής ενσωμάτωση των τριών διαφορετικών αλγορίθμων και η αποτελεσματική διαχείριση των περιορισμών του συστήματος δημιούργησαν μια στέρεη βάση για μελλοντικές επεκτάσεις και βελτιώσεις.

Παράλληλα, η εμπειρία που αποκτήθηκε κατά την ανάπτυξη και τον έλεγχο του συστήματος προσέφερε πολύτιμες γνώσεις για τη βελτιστοποίηση παρόμοιων εφαρμογών στο μέλλον.

Η προσωπική μου συμβολή στην ανάπτυξη της εφαρμογής διαχείρισης εργασιών επικεντρώθηκε σε αρκετές πτυχές του έργου, συμβάλλοντας τόσο στην τεχνική υλοποίηση όσο και στη διαμόρφωση της συνολικής εμπειρίας χρήστη. Συγκεκριμένα:

- **Ανάπτυξη και ενσωμάτωση αλγορίθμων χρονοπρογραμματισμού:**
  Συμμετείχα στην ανάλυση και υλοποίηση των τριών βασικών αλγορίθμων χρονοπρογραμματισμού (Shortest Job Next, Least Laxity First, Round Robin), διασφαλίζοντας ότι η εφαρμογή μπορεί να εξυπηρετήσει διαφορετικές στρατηγικές κατανομής εργασιών.
- **Υλοποίηση του μηχανισμού περιορισμού των 8 ωρών εργασίας:**
  Ανέπτυξα και βελτιστοποίησα τον μηχανισμό αυτόματης μεταφοράς εργασιών στην επόμενη διαθέσιμη ημέρα, διασφαλίζοντας τη συμμόρφωση με τους εργασιακούς περιορισμούς και τη βέλτιστη κατανομή του φόρτου εργασίας.
- **Σχεδιασμός και ανάπτυξη της διεπαφής χρήστη (UI) με JavaFX:**
  Συμμετείχα στη δημιουργία ενός καθαρού και ευχάριστου περιβάλλοντος χρήσης, που επιτρέπει την εύκολη πλοήγηση και διαχείριση εργασιών. Ιδιαίτερη έμφαση δόθηκε στην οπτικοποίηση του προγράμματος μέσω του ενσωματωμένου ημερολογίου.
- **Βελτιστοποίηση της αποθήκευσης και ανάκτησης δεδομένων με JSON:**
  Ανέλαβα την υλοποίηση της διαχείρισης δεδομένων μέσω αρχείων JSON, χρησιμοποιώντας τη βιβλιοθήκη Gson για αποθήκευση και ανάκτηση εργασιών, επιτρέποντας την εύκολη επέκταση της εφαρμογής.
- **Δοκιμές και βελτιώσεις (Testing & Debugging):**
  Εκτέλεσα εκτενείς δοκιμές για τον έλεγχο της απόδοσης των αλγορίθμων, τη σταθερότητα της εφαρμογής και τη σωστή λειτουργία του μηχανισμού χρονοπρογραμματισμού, διασφαλίζοντας την αξιοπιστία του συστήματος.

Η ανάπτυξη και υλοποίηση της εφαρμογής διαχείρισης εργασιών με τρεις διαφορετικούς αλγορίθμους χρονοπρογραμματισμού οδήγησε σε σημαντικά συμπεράσματα τόσο σε τεχνικό επίπεδο όσο και σε επίπεδο χρηστικότητας. Η εφαρμογή πέτυχε τον βασικό της στόχο,

παρέχοντας ένα ολοκληρωμένο περιβάλλον για τη διαχείριση και τον προγραμματισμό εργασιών, ενώ παράλληλα προσέφερε στους χρήστες την ευελιξία επιλογής του καταλληλότερου αλγορίθμου για τις ανάγκες τους.

Η χρήση του **JavaFX** για την ανάπτυξη της διεπαφής χρήστη αποδείχθηκε επιτυχής επιλογή, καθώς επέτρεψε τη δημιουργία ενός σύγχρονου και διαισθητικού περιβάλλοντος που διευκολύνει την αλληλεπίδραση με το σύστημα. Ο σχεδιασμός του UI είχε ως βασική αρχή την ευχρηστία, εξασφαλίζοντας ότι οι λειτουργίες είναι προσβάσιμες ακόμη και από χρήστες χωρίς προηγούμενη εμπειρία σε παρόμοιες εφαρμογές.

Η υλοποίηση των τριών αλγορίθμων χρονοπρογραμματισμού **(SJN, LLF και Round Robin)** ανέδειξε τα ιδιαίτερα χαρακτηριστικά και τις δυνατότητες του καθενός σε διαφορετικά σενάρια χρήσης:

- **Ο SJN** αποδείχθηκε ιδιαίτερα αποτελεσματικός στην ελαχιστοποίηση του συνολικού χρόνου αναμονής των εργασιών, ειδικά σε περιπτώσεις όπου υπήρχαν πολλές σύντομες εργασίες.
- **Ο LLF** παρείχε καλύτερη διαχείριση των προθεσμιών, εξασφαλίζοντας ότι οι εργασίες με περιορισμένο χρονικό περιθώριο λαμβάνουν την απαραίτητη προτεραιότητα.
- **Ο Round Robin**, με το σταθερό κβάντο χρόνου του μίας ώρας, πέτυχε δίκαιη κατανομή του χρόνου εκτέλεσης μεταξύ των εργασιών, αποτρέποντας την μονοπώληση των πόρων του συστήματος από μεμονωμένες εργασίες.

Η εφαρμογή του περιορισμού των **8 ωρών ημερήσιας εργασίας** ανά χρήστη αποτέλεσε μια σημαντική πρόκληση που αντιμετωπίστηκε επιτυχώς μέσω του μηχανισμού αυτόματης μεταφοράς εργασιών στην επόμενη διαθέσιμη ημέρα. Αυτή η προσέγγιση όχι μόνο διασφάλισε τη συμμόρφωση με τους εργασιακούς περιορισμούς, αλλά παράλληλα επέτρεψε την αποτελεσματική διαχείριση του φόρτου εργασίας σε βάθος χρόνου.

Τεχνικές και Λειτουργικές Επιτυχίες

Η αρχιτεκτονική του συστήματος, βασισμένη στο πρότυπο **Model-View-Controller (MVC)**, αποδείχθηκε ιδιαίτερα αποτελεσματική στη διαχείριση της πολυπλοκότητας της εφαρμογής. Ο διαχωρισμός των ευθυνών μεταξύ των διαφορετικών επιπέδων επέτρεψε την ευκολότερη συντήρηση και επέκταση του κώδικα, ενώ παράλληλα διευκόλυνε τον εντοπισμό και τη διόρθωση σφαλμάτων κατά τη φάση της ανάπτυξης.

Η χρήση του **Gradle** ως εργαλείου διαχείρισης εξαρτήσεων και αυτοματοποίησης της διαδικασίας κατασκευής συνέβαλε σημαντικά στην οργάνωση του project και στην απλοποίηση

της διαδικασίας ανάπτυξης. Η **διαχείριση των δεδομένων μέσω αρχείων JSON** αποδείχθηκε επαρκής, προσφέροντας ευελιξία στην αποθήκευση και ανάκτηση των πληροφοριών χρηστών και εργασιών.

Η διαδικασία ελέγχου της εφαρμογής ανέδειξε τη σημασία του **συστηματικού testing** σε όλα τα επίπεδα του συστήματος. Οι **εκτενείς δοκιμές των αλγορίθμων χρονοπρογραμματισμού** σε διάφορα σενάρια χρήσης επέτρεψαν τον εντοπισμό και τη διόρθωση σημαντικών ζητημάτων, βελτιώνοντας την αξιοπιστία του συστήματος.

Ιδιαίτερα σημαντική αποδείχθηκε η δοκιμή **ακραίων περιπτώσεων (edge cases)** για κάθε αλγόριθμο, καθώς αποκάλυψε πιθανές αδυναμίες και οδήγησε σε βελτιώσεις στη λογική του χρονοπρογραμματισμού. Για παράδειγμα, εντοπίστηκαν και διορθώθηκαν προβλήματα στον LLF όταν πολλές εργασίες είχαν την ίδια χαμηλή χαλαρότητα (laxity), καθώς και στη δυναμική διαχείριση προτεραιοτήτων στον Round Robin.

Προτάσεις Τρόπων Χρήσης της Εφαρμογής

Η εφαρμογή προσφέρει πολλαπλές δυνατότητες προσαρμογής στις ανάγκες του χρήστη, επιτρέποντας διαφορετικούς τρόπους χρήσης ανάλογα με το προφίλ και τις απαιτήσεις κάθε χρήστη. Μερικές προτεινόμενες περιπτώσεις χρήσης περιλαμβάνουν:

- **Διαχείριση εργασιών για επαγγελματίες και ελεύθερους επαγγελματίες:**
  - Χρήση του **Round Robin** για δίκαιη κατανομή χρόνου εργασίας μεταξύ πολλαπλών project.
  - Χρήση του **SJN** για προτεραιοποίηση μικρών εργασιών και ταχύτερη ολοκλήρωσή τους.
  - Χρήση του **LLF** για αυστηρή διαχείριση προθεσμιών σε έργα με συγκεκριμένα deadlines.
- **Εκπαιδευτική χρήση και προγραμματισμός μελέτης:**
  - Οι μαθητές και οι φοιτητές μπορούν να χρησιμοποιήσουν τον αλγόριθμο **SJN** για να διαχειριστούν γρήγορες αναθεωρήσεις πριν από εξετάσεις.
  - Ο **LLF** μπορεί να χρησιμοποιηθεί για την προτεραιοποίηση μαθημάτων που πλησιάζουν στις προθεσμίες εξετάσεων ή εργασιών.
- **Προγραμματισμός βαρδιών και εργασιακών υποχρεώσεων:**

- Εφαρμογή σε ομάδες εργασίας για τη διαχείριση βαρδιών μέσω του **Round Robin**, διασφαλίζοντας ίση κατανομή μεταξύ των εργαζομένων.
- Συνδυασμός **SJN και LLF** για τη διαχείριση καθημερινών εργασιών σε δυναμικά περιβάλλοντα, όπως ο τομέας του IT ή η εξυπηρέτηση πελατών.

- **Διαχείριση προσωπικών εργασιών και καθημερινής ρουτίνας:**
  - Ο χρήστης μπορεί να οργανώσει προσωπικά tasks (π.χ. προπονήσεις, αναθέσεις, ψώνια) και να τα κατανείμει ανάλογα με τις προτιμήσεις του.
  - Η εφαρμογή μπορεί να λειτουργήσει ως προσωπικός βοηθός υπενθύμισης, βοηθώντας στην καλύτερη κατανομή του χρόνου μέσα στην ημέρα.

Η **εμπειρία χρήστη (UX)** αποτέλεσε βασική προτεραιότητα κατά την ανάπτυξη της εφαρμογής. Το **σύστημα ημερολογίου**, σε συνδυασμό με την εύχρηστη λίστα εργασιών, προσέφερε στους χρήστες μια ξεκάθαρη εικόνα του προγράμματός τους και των επερχόμενων υποχρεώσεών τους.

Η δυνατότητα εναλλαγής μεταξύ των τριών αλγορίθμων χρονοπρογραμματισμού μέσω ενός απλού μενού επιλογής επέτρεψε στους χρήστες να **πειραματιστούν με διαφορετικές στρατηγικές οργάνωσης των εργασιών τους**, προσαρμόζοντας το σύστημα στις προσωπικές τους ανάγκες και προτιμήσεις.

Η εφαρμογή αποδείχθηκε ιδιαίτερα χρήσιμη σε διαφορετικά περιβάλλοντα χρήσης, όπως:

- **Επαγγελματική διαχείριση εργασιών** για τη διαχείριση project-based workflows.
- **Προγραμματισμός μαθημάτων και μελέτης**, με δυνατότητα προτεραιοποίησης αναλόγως των deadlines.
- **Προγραμματισμός βαρδιών** σε ομάδες εργασίας, αξιοποιώντας το Round Robin για δίκαιη κατανομή του φόρτου εργασίας.
- **Διαχείριση προσωπικών εργασιών και υποχρεώσεων**, λειτουργώντας ως ένας έξυπνος προσωπικός βοηθός για καλύτερη διαχείριση χρόνου.

Η επιτυχής ενσωμάτωση των τριών διαφορετικών αλγορίθμων και η αποτελεσματική διαχείριση των περιορισμών του συστήματος δημιούργησαν μια στέρεη βάση για τη χρήση της εφαρμογής σε **πραγματικά σενάρια διαχείρισης εργασιών**.

5.2 Μελλοντικές Επεκτάσεις

Η εφαρμογή έχει τη δυνατότητα να εξελιχθεί σε ένα ολοκληρωμένο εργαλείο διαχείρισης εργασιών, μέσα από την ενσωμάτωση νέων αλγορίθμων, την υποστήριξη πολλαπλών πλατφορμών και τη διασύνδεση με εξωτερικές υπηρεσίες. Μία από τις κύριες προοπτικές εξέλιξης της εφαρμογής είναι η ενσωμάτωση επιπλέον αλγορίθμων χρονοπρογραμματισμού, όπως ο Genetic Algorithm (GA) ή ο Particle Swarm Optimization (PSO). Αυτοί οι αλγόριθμοι μπορούν να ενισχύσουν την προσαρμοστικότητα και την απόδοση του συστήματος, επιτρέποντας την καλύτερη αντιμετώπιση περίπλοκων σεναρίων, όπως η διαχείριση πολλαπλών χρηστών ή δυναμικά μεταβαλλόμενων προθεσμιών. Η ενσωμάτωση αυτών των μεθόδων θα μπορούσε να γίνει μέσα από ένα modular σύστημα, διατηρώντας την επεκτασιμότητα της εφαρμογής.

Ένα σημαντικό βήμα προς αυτή την κατεύθυνση είναι η ενσωμάτωση επιπλέον αλγορίθμων χρονοπρογραμματισμού, οι οποίοι θα βελτιώσουν την αποδοτικότητα του συστήματος και θα καλύψουν πιο περίπλοκες απαιτήσεις χρηστών. Η μελλοντική εξέλιξη της εφαρμογής περιλαμβάνει τη δυνατότητα υποστήριξης πολλαπλών πλατφορμών, όπως κινητά τηλέφωνα και tablets. Η ανάπτυξη εφαρμογών για λειτουργικά συστήματα Android και iOS θα επιτρέψει την πρόσβαση των χρηστών στις εργασίες τους από οπουδήποτε. Επιπλέον, η προσθήκη cloud-based συγχρονισμού μπορεί να εξασφαλίσει τη συνεχή αποθήκευση και ανάκτηση δεδομένων σε πραγματικό χρόνο, βελτιώνοντας την εμπειρία χρήστη.

Παράλληλα, η δυνατότητα υποστήριξης πολλαπλών πλατφορμών και η ανάπτυξη mobile εφαρμογών μπορεί να επεκτείνει τη χρηστικότητα της εφαρμογής, επιτρέποντας στους χρήστες να έχουν πρόσβαση στις εργασίες τους από οπουδήποτε. Η εξέλιξη της εφαρμογής μπορεί να επεκταθεί με τη δημιουργία λειτουργιών για επιχειρησιακή χρήση, όπως η υποστήριξη συνεργατικών ομάδων εργασίας. Οι δυνατότητες αυτές μπορεί να περιλαμβάνουν κοινόχρηστα ημερολόγια, ειδοποιήσεις για προθεσμίες και σύστημα ανάθεσης εργασιών σε διαφορετικά μέλη ομάδας. Η ενσωμάτωση ρόλων και δικαιωμάτων πρόσβασης θα ενισχύσει την ασφάλεια και τη διαφάνεια στις εργασίες.

Επιπλέον, η ανάλυση δεδομένων και η χρήση τεχνητής νοημοσύνης μπορούν να υποστηρίξουν τις προηγούμενες λειτουργίες, προσφέροντας στους χρήστες προτάσεις για καλύτερη διαχείριση του χρόνου και εξατομικευμένες λύσεις. Η ενσωμάτωση εργαλείων ανάλυσης δεδομένων θα μπορούσε να προσφέρει στους χρήστες πολύτιμες πληροφορίες σχετικά με τη διαχείριση του χρόνου τους. Με τη χρήση τεχνητής νοημοσύνης (AI), η εφαρμογή μπορεί να

προτείνει βελτιστοποιήσεις στο χρονοδιάγραμμα των εργασιών, βασιζόμενη σε μοτίβα χρήσης. Επίσης, η χρήση αλγορίθμων μηχανικής μάθησης θα μπορούσε να προσαρμόσει τις προτεραιότητες των εργασιών ανάλογα με τις συνήθειες του χρήστη.

Τέλος, η διασύνδεση με εξωτερικές εφαρμογές, όπως εργαλεία διαχείρισης έργων ή ημερολόγια, θα ενοποιήσει όλες τις λειτουργίες σε ένα συνεκτικό σύστημα, καθιστώντας την εφαρμογή ένα πολύτιμο εργαλείο παραγωγικότητας για τους χρήστες. Η ανάπτυξη API για τη σύνδεση της εφαρμογής με άλλες υπηρεσίες, όπως εφαρμογές διαχείρισης έργων (π.χ., Trello, Asana) ή ημερολόγια (Google Calendar, Outlook), θα επιτρέψει τη διαχείριση εργασιών από μια ενιαία πλατφόρμα. Επιπλέον, η ενσωμάτωση με εργαλεία παραγωγικότητας, όπως το Microsoft Teams ή το Slack, μπορεί να διευκολύνει τη συνεργασία και την επικοινωνία μεταξύ των χρηστών.

Η εφαρμογή έχει τη δυνατότητα να εξελιχθεί σε ένα ολοκληρωμένο εργαλείο διαχείρισης εργασιών, μέσα από την ενσωμάτωση νέων λειτουργιών, την προσαρμογή στις ανάγκες των χρηστών και τη βελτιστοποίηση των υπαρχόντων αλγορίθμων. Οι επεκτάσεις αυτές μπορούν να βελτιώσουν τόσο την αποδοτικότητα του συστήματος όσο και τη συνολική εμπειρία χρήστη.

Εξατομίκευση και Προσαρμογή στις Ανάγκες του Χρήστη

Μία από τις κύριες προοπτικές εξέλιξης της εφαρμογής είναι η **εξατομίκευση των λειτουργιών της** ώστε να προσαρμόζεται στις ανάγκες του εκάστοτε χρήστη. Μέσα από τη συλλογή δεδομένων χρήσης και την ανάλυση προτιμήσεων, η εφαρμογή μπορεί να προτείνει βέλτιστες στρατηγικές χρονοπρογραμματισμού. Για παράδειγμα:

- Οι χρήστες που ολοκληρώνουν συχνά μικρές εργασίες μπορεί να λαμβάνουν **σύσταση χρήσης του αλγορίθμου Shortest Job Next (SJN)**.
- Όσοι ακολουθούν αυστηρά προγράμματα με σταθερές προθεσμίες μπορούν να καθοδηγούνται προς **τον Least Laxity First (LLF)**.
- Για χρήστες που εργάζονται σε πολλές εργασίες ταυτόχρονα, η εφαρμογή μπορεί να προτείνει **τον Round Robin για ισορροπημένη κατανομή του χρόνου**.

Η εξατομίκευση μπορεί επίσης να επεκταθεί στη διεπαφή χρήστη, επιτρέποντας την προσαρμογή του UI με βάση τις προτιμήσεις του χρήστη, όπως σκοτεινή λειτουργία (dark mode), προσαρμοσμένες ειδοποιήσεις ή δυνατότητα εμφάνισης των εργασιών με διαφορετικούς τρόπους (π.χ. σε λίστα, ημερολόγιο ή kanban board).

Σύστημα Προτάσεων (Recommendation System)

Η προσθήκη ενός **συστήματος προτάσεων** θα μπορούσε να βελτιώσει την αποδοτικότητα της εφαρμογής, επιτρέποντας στους χρήστες να λαμβάνουν προτάσεις για την καλύτερη διαχείριση των εργασιών τους. Ένα recommendation system θα μπορούσε να λειτουργήσει ως εξής:

- **Προτάσεις για προτεραιοποίηση εργασιών**, βασισμένες σε παλαιότερη συμπεριφορά χρήσης και επαναλαμβανόμενα μοτίβα.
- **Αυτόματη πρόβλεψη των πιο πιθανών ωρών εργασίας** για συγκεκριμένες εργασίες, λαμβάνοντας υπόψη το ημερήσιο πρόγραμμα και τις προτιμήσεις του χρήστη.
- **Ειδοποιήσεις για υπερφόρτωση εργασιών**, προειδοποιώντας τον χρήστη εάν έχει προγραμματίσει περισσότερες εργασίες από όσες μπορεί να ολοκληρώσει μέσα σε μια ημέρα.

Η εφαρμογή θα μπορούσε επίσης να ενσωματώσει **αλγόριθμους μηχανικής μάθησης** που θα αναλύουν το ιστορικό εργασιών και θα προσφέρουν δυναμικές προσαρμογές στο χρονοδιάγραμμα.

Βελτιώσεις στους Υπάρχοντες Αλγορίθμους Χρονοπρογραμματισμού

Παρόλο που οι τρεις αλγόριθμοι που χρησιμοποιούνται είναι αποδοτικοί, υπάρχουν περιθώρια βελτίωσης που θα μπορούσαν να ενισχύσουν την απόδοσή τους σε πραγματικά σενάρια χρήσης:

1. **Βελτίωση του Shortest Job Next (SJN)**
   - Μπορεί να προστεθεί ένας **μηχανισμός πρόβλεψης** που να εκτιμά τον χρόνο εκτέλεσης των εργασιών με βάση το ιστορικό του χρήστη, ώστε να αποφεύγονται ανακριβείς εκτιμήσεις.
   - Ένα **σύστημα προτεραιότητας** θα μπορούσε να αποτρέψει τη συνεχή καθυστέρηση μεγαλύτερων εργασιών, μειώνοντας το φαινόμενο της "ασιτίας" (starvation).
2. **Βελτίωση του Least Laxity First (LLF)**
   - Μια πιο **δυναμική προσαρμογή της χαλαρότητας** (laxity) θα μπορούσε να λαμβάνει υπόψη πρόσθετους παράγοντες, όπως η

σημαντικότητα της εργασίας ή η πιθανότητα να καθυστερήσει λόγω εξωτερικών παραγόντων.

- Μπορεί να εφαρμοστεί ένας **μηχανισμός εξισορρόπησης φόρτου** που θα μειώνει τις συχνές αλλαγές προτεραιότητας μεταξύ των εργασιών.

3. **Βελτίωση του Round Robin (RR)**

- Η εφαρμογή ενός **δυναμικού χρονικού κβάντου (Dynamic Time Quantum)**, που θα προσαρμόζεται με βάση το μέγεθος των εργασιών, θα μπορούσε να βελτιώσει σημαντικά την απόδοση του Round Robin.
- Μια **υβριδική προσέγγιση** που θα συνδυάζει Round Robin με Shortest Job Next για συγκεκριμένες κατηγορίες εργασιών μπορεί να αυξήσει την αποδοτικότητα του αλγορίθμου.

Διασύνδεση με Άλλες Πλατφόρμες και Εφαρμογές

Η εφαρμογή μπορεί να γίνει ακόμη πιο λειτουργική μέσω της **διασύνδεσής της με άλλες δημοφιλείς πλατφόρμες**. Ορισμένες από τις πιθανές ενσωματώσεις περιλαμβάνουν:

- **Συγχρονισμός με ημερολόγια** όπως Google Calendar ή Outlook, ώστε οι εργασίες να εμφανίζονται αυτόματα στα ημερολόγια των χρηστών.
- **Σύνδεση με εφαρμογές διαχείρισης έργων** όπως Trello, Asana και Jira, ώστε οι εργασίες να μπορούν να εισάγονται και να ενημερώνονται αυτόματα από διαφορετικές πλατφόρμες.
- **Υποστήριξη ειδοποιήσεων και υπενθυμίσεων** μέσω e-mail ή μέσω εφαρμογών επικοινωνίας, όπως το Slack ή το Microsoft Teams.

Υποστήριξη Πολλαπλών Χρηστών και Ομαδική Συνεργασία

Για να καλύψει μεγαλύτερες ανάγκες, η εφαρμογή μπορεί να αποκτήσει **δυνατότητες διαμοιρασμού εργασιών** και συνεργασίας σε ομάδες. Ορισμένες από τις βελτιώσεις σε αυτήν την κατεύθυνση περιλαμβάνουν:

- **Δημιουργία κοινόχρηστων προγραμμάτων εργασιών**, όπου πολλοί χρήστες θα μπορούν να δουν και να επεξεργαστούν τις ίδιες εργασίες.

- **Εκχώρηση ρόλων και δικαιωμάτων πρόσβασης**, ώστε να μπορούν να οριστούν υπεύθυνοι για κάθε εργασία.
- **Ειδοποιήσεις για ενημερώσεις σε ομαδικά προγράμματα εργασιών**, για καλύτερο συγχρονισμό μεταξύ μελών μιας ομάδας.

Εικονικός Βοηθός (Virtual Assistant)

Η εφαρμογή έχει τη δυνατότητα να εξελιχθεί σε ένα **πλήρως ευφυές εργαλείο διαχείρισης εργασιών**, με ενσωμάτωση ενός **Εικονικού Βοηθού (Virtual Assistant)** που θα λειτουργεί ως **έξυπνος σύμβουλος** για τη βελτιστοποίηση του χρονοπρογραμματισμού και της παραγωγικότητας των χρηστών.

Ενσωμάτωση Εικονικού Βοηθού (Virtual Assistant)

Η προσθήκη ενός **AI-powered Virtual Assistant** θα επιτρέψει στην εφαρμογή να λειτουργεί **διαδραστικά**, βοηθώντας τους χρήστες να οργανώνουν και να προγραμματίζουν τις εργασίες τους πιο αποδοτικά. Ο εικονικός βοηθός θα μπορούσε να λειτουργήσει με τους εξής τρόπους:

1. Φωνητικές Εντολές και Φυσική Γλώσσα

- Οι χρήστες θα μπορούν να **προσθέτουν, διαγράφουν ή ενημερώνουν εργασίες** χρησιμοποιώντας φωνητικές εντολές, αντί να τις καταχωρούν χειροκίνητα.
- **Φυσική γλώσσα (NLP)** για δημιουργία εργασιών: π.χ. ο χρήστης μπορεί να πει *"Πρόσθεσε μια συνάντηση αύριο στις 10 π.μ. για το Project X"*, και η εργασία θα προγραμματιστεί αυτόματα.
- **Αυτόματη υπενθύμιση εργασιών μέσω φωνητικών ειδοποιήσεων**, ώστε να προειδοποιεί τον χρήστη για επικείμενες προθεσμίες.

2. Προσωπικοποιημένες Προτάσεις και Προγραμματισμός

Ο Εικονικός Βοηθός θα μπορεί να αναλύει τα μοτίβα εργασίας του χρήστη και να προτείνει βελτιστοποιημένες στρατηγικές διαχείρισης χρόνου:

- **Ανάλυση προτεραιοτήτων και αυτόματη πρόταση του κατάλληλου αλγορίθμου χρονοπρογραμματισμού** με βάση τις ανάγκες του χρήστη (SJN για γρήγορες εργασίες, LLF για προθεσμίες, RR για ισορροπημένη κατανομή).

- **Αυτόματη αναδιοργάνωση εργασιών** εάν εντοπιστούν επικαλύψεις ή μη ρεαλιστικές προγραμματισμένες ώρες.
- **Ειδοποιήσεις για υπερφόρτωση εργασιών**, προτείνοντας τρόπους κατανομής σε διαφορετικές ημέρες.

3. Δυνατότητα Επικοινωνίας και Υπενθυμίσεων

- Ο βοηθός θα μπορεί να **υπενθυμίζει σημαντικές εργασίες** μέσω φωνητικών ή γραπτών ειδοποιήσεων.
- Δυνατότητα αποστολής **ειδοποιήσεων μέσω e-mail, SMS ή push notifications** στο κινητό του χρήστη.
- **Διαδραστικός διάλογος**: Ο χρήστης μπορεί να ρωτήσει τον βοηθό *"Ποιες είναι οι πιο επείγουσες εργασίες για σήμερα;"*, και ο βοηθός θα εμφανίσει τις σχετικές πληροφορίες.

4. Προσαρμογή και Εξατομίκευση

- Ο εικονικός βοηθός μπορεί να μάθει **τις συνήθειες και τις προτιμήσεις** του χρήστη και να προσαρμόσει τις ειδοποιήσεις και τις προτάσεις του.
- **Δυνατότητα επιλογής φωνής και στυλ επικοινωνίας**, ώστε να είναι πιο φιλικός προς τον χρήστη.

Συνδυασμός με Άλλες Λειτουργίες της Εφαρμογής

Η προσθήκη του εικονικού βοηθού μπορεί να λειτουργήσει **σε συνδυασμό με τις υπόλοιπες βελτιώσεις**, όπως:

✓ **Recommendation System**, ώστε ο βοηθός να προτείνει εξατομικευμένα προγράμματα εργασιών.

✓ **Βελτίωση των αλγορίθμων χρονοπρογραμματισμού**, ώστε να προσαρμόζονται δυναμικά στις ανάγκες του χρήστη.

✓ **Διασύνδεση με εξωτερικές πλατφόρμες** (Google Calendar, Outlook, Trello) για συγχρονισμό εργασιών.

Η ενσωμάτωση ενός **AI-powered Virtual Assistant** θα μετατρέψει την εφαρμογή από ένα απλό εργαλείο διαχείρισης εργασιών σε έναν **έξυπνο προσωπικό βοηθό**, βελτιώνοντας την παραγωγικότητα και την εμπειρία χρήστη.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. AlHeyasat, O., & Herzallah, R. (2010). Estimation of Quantum Time Length for Round-robin Scheduling Algorithm using Neural Networks. IJCCI (ICFC-ICNC), 253–257. https://www.scitepress.org/papers/2010/30580/30580.pdf

2. 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.

https://link.springer.com/chapter/10.1007/11892960_53

3. Alepis, E. and Virvou, M., 2014. Object-oriented user interfaces for personalized mobile learning. Springer. ISBN: 978-3-642-53850-6

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. ISBN: 0-7695-2632-2

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., Tsichrintzi, 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. Attiya, I., Abd Elaziz, M., Abualigah, L., Nguyen, T. N., & Abd El-Latif, A. A. (2022). An improved hybrid swarm intelligence for scheduling iot application tasks in the cloud. IEEE Transactions on Industrial Informatics, 18(9), 6264–6272.

9. Bailey, J., & Field, J. (1985). Personnel scheduling with flexshift models. Journal of Operations Management, 5(3), 327–338. https://doi.org/10.1016/0272-6963(85)90017-8

10. Becker, T., & Schüle, T. (2020). Evaluating Dynamic Task Scheduling with Priorities and Adaptive Aging in a Task-Based Runtime System (pp. 17–31). https://doi.org/10.1007/978-3-030-52794-5_2

11. Caranto, H. S., Olivete, W. C. L., Fernandez, J. V. D., Cabiara, C. A. R., Baquirin, R. B. M., Bayani, E. F. G., & Fronda, R. J. D. (2020). Integrating User-Defined Priority Tasks in a Shortest Job First Round Robin (SJFRR) Scheduling Algorithm. Proceedings of 2020 6th International Conference on Computing and Data Engineering, 9–13. https://doi.org/10.1145/3379247.3379291

12. 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.

13. 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.

14. Chrysafiadi, K. and Virvou, M., 2015. Advances in personalized web-based education. Springer International Publishing. ISBN: 978-3-319-12894-8

15. Chrysafiadi, K., Papadimitriou, S. and Virvou, M., 2022. Cognitive-based adaptive scenarios in educational games using fuzzy reasoning. Knowledge-Based Systems, 250, p.109111.

16. 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.

17. 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.

18. 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.

19. Damuut, L. P., & Dung, P. B. (2019). Comparative Analysis of FCFS, SJN & RR Job Scheduling Algorithms. International Journal of Computer Science & Information Technology (IJCSIT) Volume, 11, 45–51.

20. Datta, L. (2015). Modified RR Algorithm with Dynamic Time Quantum for Externally Prioritized Tasks. International Journal on Recent and Innovation Trends in Computing and Communication, 3(1), 217–221. https://doi.org/10.17762/ijritcc2321-8169.150145

21. Faulring, A., & Myers, B. A. (2006). Availability bars for calendar scheduling. CHI '06 Extended Abstracts on Human Factors in Computing Systems, 760–765. https://doi.org/10.1145/1125451.1125603

22. Guo, M., Guan, Q., Chen, W., Ji, F., & Peng, Z. (2022). Delay-Optimal Scheduling of VMs in a Queueing Cloud Computing System with Heterogeneous Workloads. IEEE Transactions on Services Computing, 15(1), 110–123. https://doi.org/10.1109/TSC.2019.2920954

23. Hatzilygeroudis, I. K., Tsihrintzis, G. A., & Jain, L. C. (Eds.). (2023). Fusion of Machine Learning Paradigms: Theory and Applications (Vol. 236). Springer Nature.

24. Hildebrandt, J., Golatowski, F., & Timmermann, D. (1999). Scheduling coprocessor for enhanced least-laxity-first scheduling in hard real-time systems. Proceedings of 11th Euromicro Conference on Real-Time Systems. Euromicro RTS'99, 208–215.

25. Hosseinzadeh, M., Ghafour, M. Y., Hama, H. K., Vo, B., & Khoshnevis, A. (2020). Multi-Objective Task and Workflow Scheduling Approaches in Cloud Computing: a Comprehensive Review. Journal of Grid Computing, 18(3), 327–356. https://doi.org/10.1007/s10723-020-09533-z

26. Hwang, E., Kim, J.-S., & Choi, Y. (2021). Achieving Fairness-Aware Two-Level Scheduling for Heterogeneous Distributed Systems. IEEE Transactions on Services Computing, 14(3), 639–653. https://doi.org/10.1109/TSC.2018.2836444

27. Ibanez, M., Clark, J. R., Huckman, R. S., & Staats, B. R. (2015). Discretionary Task Ordering: Queue Management in Radiological Services. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.2677200

28. Iqbal, M., Ullah, Z., Khan, I. A., Aslam, S., Shaheer, H., Humayon, M., Salahuddin, M. A., & Mehmood, A. (2023). Optimizing Task Execution: The Impact of Dynamic Time Quantum and Priorities on Round Robin Scheduling. Future Internet, 15(3), 104. https://doi.org/10.3390/fi15030104

29. Jia, L., Li, K., & Shi, X. (2021). Cloud Computing Task Scheduling Model Based on Improved Whale Optimization Algorithm. Wireless Communications and Mobile Computing, 2021(1). https://doi.org/10.1155/2021/4888154

30. 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.

31. 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.

32. 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.

33. 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.

34. Kaur, N., Kumar, A., & Kumar, R. (2021). A systematic review on task scheduling in Fog computing: Taxonomy, tools, challenges, and future directions. Concurrency and Computation: Practice and Experience, 33(21). https://doi.org/10.1002/cpe.6432

35. Khan, Z. A., Aziz, I. A., Osman, N. A. B., & Ullah, I. (2023). A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues, Challenges, and Future Directions. IEEE Access, 11, 143417–143445. https://doi.org/10.1109/ACCESS.2023.3343877

36. Kondo, D., Chien, A. A., & Casanova, H. (2007). Scheduling Task Parallel Applications for Rapid Turnaround on Enterprise Desktop Grids. Journal of Grid Computing, 5(4), 379–405. https://doi.org/10.1007/s10723-007-9063-y

37. 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

38. Kruk, G., Alves, O., Molinari, L., & Roux, E. (2017). Best practices for efficient development of JavaFX applications. Proceedings of the 16th International Conference on Accelerator and Large Experimental Control Systems, 1078–1083.

39. 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.

40. 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).

41. 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.

42. Lou, Y., Chen, J., Zhang, L., Hao, D., & Zhang, L. (2019). History-driven build failure fixing: how far are we? Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, 43–54. https://doi.org/10.1145/3293882.3330578

43. Mackinlay, J. D., Robertson, G. G., & DeLine, R. (1994). Developing calendar visualizers for the information visualizer. Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology - UIST ’94, 109–118. https://doi.org/10.1145/192426.192470

44. Malakooti, B. (2013). Scheduling and Sequencing. Operations and Production Systems with Multiple Objectives, Wiley.

45. Miranda, S., Baker, C., Woodbridge, K., & Griffiths, H. (2006). Knowledge-based resource management for multifunction radar: a look at scheduling and task prioritization. IEEE Signal Processing Magazine, 23(1), 66–76. https://doi.org/10.1109/MSP.2006.1593338

46. Mishra, N. (2017). Statistical Methods for Improving Dynamic Scheduling and Resource Usage in Computing Systems. The University of Chicago.

47. Monteiro, P., Machado, R. J., Kazman, R., Lima, A., Simões, C., & Ribeiro, P. (2013). Mapping CMMI and RUP Process Frameworks for the Context of Elaborating Software Project Proposals (pp. 191–214). https://doi.org/10.1007/978-3-642-35702-2_12

48. 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.

49. 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.

50. Oh, S.-H., & Yang, S.-M. (1998). A modified least-laxity-first scheduling algorithm for real-time tasks. Proceedings Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No. 98EX236), 31–36.

51. Panagoulias, 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.

52. Panagoulias, 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.

53. Panagoulias, 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. Panagoulias, 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.

55. Panagoulias, 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.

56. 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.

57. 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.

58. 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.

59. 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.

60. Paprocka, I., & Skołud, B. (2017). A hybrid multi-objective immune algorithm for predictive and reactive scheduling. Journal of Scheduling, 20(2), 165–182. https://doi.org/10.1007/s10951-016-0494-9

61. Pareto, L., & Boquist, U. (2006). A quality model for design documentation in model-centric projects. Proceedings of the 3rd International Workshop on Software Quality Assurance, 30–37. https://doi.org/10.1145/1188895.1188905

62. 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.

63. Perret, Q., Charlemagne, G., Sotiriadis, S., & Bessis, N. (2013). A Deadline Scheduler for Jobs in Distributed Systems. 2013 27th International Conference on Advanced Information Networking and Applications Workshops, 757–764. https://doi.org/10.1109/WAINA.2013.194

64. 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

65. Prakash, M. (2022). Build Automation Tools for Software Development. Embedded Systems and Applications, 73–89. https://doi.org/10.5121/csit.2022.120608

66. Praveen, S. P., Rao, K. T., & Janakiramaiah, B. (2018). Effective Allocation of Resources and Task Scheduling in Cloud Environment using Social Group Optimization. Arabian Journal for Science and Engineering, 43(8), 4265–4272. https://doi.org/10.1007/s13369-017-2926-z

67. Prihartono, W., & Utami, S. F. (2023). Rational Unified Process as a Software Development Method in Decision Support Systems Design with Simple Additive Weighting in Determining Priority Rehabilitations of Classrooms. JURNAL SISFOTEK GLOBAL, 13(2), 121. https://doi.org/10.38101/sisfotek.v13i2.9678

68. Pupa, A., Van Dijk, W., & Secchi, C. (2021). A human-centered dynamic scheduling architecture for collaborative application. IEEE Robotics and Automation Letters, 6(3), 4736–4743.

69. Pupa, A., Van Dijk, W., Brekelmans, C., & Secchi, C. (2022). A resilient and effective task scheduling approach for industrial human-robot collaboration. Sensors, 22(13), 4901.

70. Richards, R., & Stottler, R. (2019). Complex Project Scheduling Lessons Learned from NASA, Boeing, General Dynamics and Others. 2019 IEEE Aerospace Conference, 1–9. https://doi.org/10.1109/AERO.2019.8741996

71. Sahkhar, L., Balabantaray, B. K., & Yadav, S. S. (2022). Efficient Cloudlet Allocation to Virtual Machine to Impact Cloud System Performance. International Journal of Information System Modeling and Design (IJISMD), 13(6), 1–21.

72. Sasmita Kumari Nayak. (2023). Nature inspired algorithms in dynamic task scheduling: A review. World Journal of Advanced Research and Reviews, 20(3), 829–833. https://doi.org/10.30574/wjarr.2023.20.3.2531

73. Savvopoulos, A. and Vivrou, 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.

74. 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.

75. Shyalika, C., Silva, T., & Karunananda, A. (2020). Reinforcement Learning in Dynamic Task Scheduling: A Review. SN Computer Science, 1(6), 306. https://doi.org/10.1007/s42979-020-00326-5

76. Sinnen, O., Sousa, L. A., & Sandnes, F. E. (2006). Toward a realistic task scheduling model. IEEE Transactions on Parallel and Distributed Systems, 17(3), 263–275. https://doi.org/10.1109/TPDS.2006.40

77. 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.

78. Stan, R.-G., Băjenaru, L., Negru, C., & Pop, F. (2021). Evaluation of Task Scheduling Algorithms in Heterogeneous Computing Environments. Sensors, 21(17), 5906. https://doi.org/10.3390/s21175906

79. 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.

80. Stathopoulou, I.O. and Tsihrintzis, G.A., 2010. Visual affect recognition (Vol. 214). IOS Press.

81. 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.

82. Stathopoulou, I.O. and Tsihrintzis, G.A., 2011. Appearance-based face detection with artificial neural networks. Intelligent Decision Technologies, 5(2), pp.101-111.

83. 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.

84. Sulaiman, M., Halim, Z., Lebbah, M., Waqas, M., & Tu, S. (2021). An Evolutionary Computing-Based Efficient Hybrid Task Scheduling Approach for Heterogeneous Computing Environment. Journal of Grid Computing, 19(1), 11. https://doi.org/10.1007/s10723-021-09552-4

85. 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.

86. 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.

87. 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.

88. 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.

89. 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.

90. Tsihrintzis, G. A., Virvou, M., & Jain, L. C. (2022). Advances in Machine Learning/Deep Learning-based Technologies. Springer International Publishing.

91. Tsihrintzis, G.A. and Virvou, M. eds., 2010. Multimedia Services in Intelligent Environments: Software Development Challenges and Solutions (Vol. 2). Springer Science & Business Media.

92. Tsihrintzis, G.A., Virvou, M. and Phillips-Wren, G., 2019. Surveys in artificial intelligence-based technologies. Intelligent Decision Technologies, 13(4), pp.393-394.

93. 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.

94. 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.

95. 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.

96. 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.

97. 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.

98. 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.

99. 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.

100. 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.

101. Varanasi, B. (2015). Introducing Gradle. Apress. https://doi.org/10.1007/978-1-4842-1031-4

102. 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.

103. Virvou, M. and Alepis, E., 2005. Mobile educational features in authoring tools for personalised tutoring. Computers & Education, 44(1), pp.53-68.

104. Virvou, M. and Du Boulay, B., 1999. Human plausible reasoning for intelligent help. User modeling and user-adapted interaction, 9, pp.321-375.

105. 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).

106. Virvou, M. and Kabassi, K., 2002. Reasoning about users' actions in a graphical user interface. Human-Computer Interaction, 17(4), pp.369-398.

107. 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.

108. 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.

109. 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.

110. 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.

111. 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.

112. 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.

113. 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.

114. 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.

115. 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.

116. 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.

117. 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.

118. Virvou, M., 1999. Automatic reasoning and help about human errors in using an operating system. Interacting with Computers, 11(5), pp.545-573

119. 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).

120. 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.

121. 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

122. 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.

123. 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.

124. 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).

125. 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.

126. 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.

127. 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.

128. 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.

129. 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.

130. Virvou, M., Tsihrintzis, G.A. and Tsichrintzi, E.A., 2024. VIRTSI: A novel trust dynamics model enhancing Artificial Intelligence collaboration with human users–Insights from a ChatGPT evaluation study. Information Sciences, 675, p.120759.

131. 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.

132. 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.

133. Virvou, M., Tsihrintzis, G.A., Sotiropoulos, D.N., Chrysafiadi, K., Sakkopoulos, E. and Tsichrintzi, 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.

134. Vos, J., Gao, W., Chin, S., Iverson, D., & Weaver, J. (2014). Getting a Jump Start in JavaFX. In Pro JavaFX 8 (pp. 1–30). Apress. https://doi.org/10.1007/978-1-4302-6575-7_1

135. Walia, N. K., Kaur, N., Alowaidi, M., Bhatia, K. S., Mishra, S., Sharma, N. K., Sharma, S. K., & Kaur, H. (2021). An Energy-Efficient Hybrid Scheduling Algorithm for Task Scheduling in the Cloud Computing Environments. IEEE Access, 9, 117325–117337. https://doi.org/10.1109/ACCESS.2021.3105727

136. Wan, L., & Bard, J. F. (2007). Weekly staff scheduling with workstation group restrictions. Journal of the Operational Research Society, 58(8), 1030–1046. https://doi.org/10.1057/palgrave.jors.2602215

137. Xu, Y., Pan, F., & Tong, L. (2016). Dynamic Scheduling for Charging Electric Vehicles: A Priority Rule. IEEE Transactions on Automatic Control, 61(12), 4094–4099. https://doi.org/10.1109/TAC.2016.2541305

138. Zhang, M., Li, C., Shang, Y., Huang, H., Zhu, W., & Liu, Y. (2022). A task scheduling model integrating micro-breaks for optimisation of job-cycle time in human-robot collaborative assembly cells. International Journal of Production Research, 60(15), 4766–4777.

139. Zouaoui, S., Boussaid, L., & Mtibaa, A. (2019). Improved time quantum length estimation for round robin scheduling algorithm using neural network. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 7(2). https://doi.org/10.11591/ijeei.v7i2.464

140. Çavdar, D., Birke, R., Chen, L. Y., & Alagöz, F. (2015). A simulation framework for priority scheduling on heterogeneous clusters. Future Generation Computer Systems, 52, 37–48. https://doi.org/10.1016/j.future.2015.04.008

ΠΑΡΑΡΤΗΜΑΤΑ

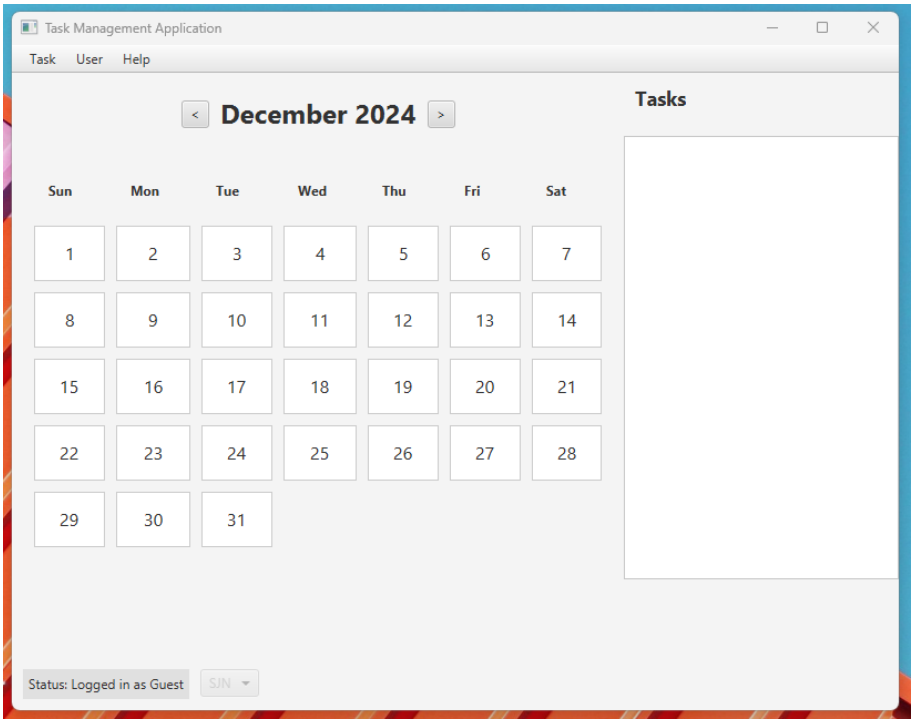
I. Εγχειρίδιο Χρήστη

Η εφαρμογή διαχείρισης εργασιών έχει σχεδιαστεί για να σας βοηθήσει να οργανώνετε, να προγραμματίζετε και να παρακολουθείτε τις εργασίες σας εύκολα και αποδοτικά. Το εγχειρίδιο αυτό περιγράφει τη λειτουργικότητα και τη χρήση της εφαρμογής για μη τεχνικούς χρήστες. Μέσω

μιας διαισθητικής διεπαφής, μπορείτε να προσθέσετε εργασίες, να τις τροποποιήσετε, να διαχειριστείτε προθεσμίες και να παρακολουθείτε την εξέλιξή τους.

Ξεκινώντας με την Εφαρμογή

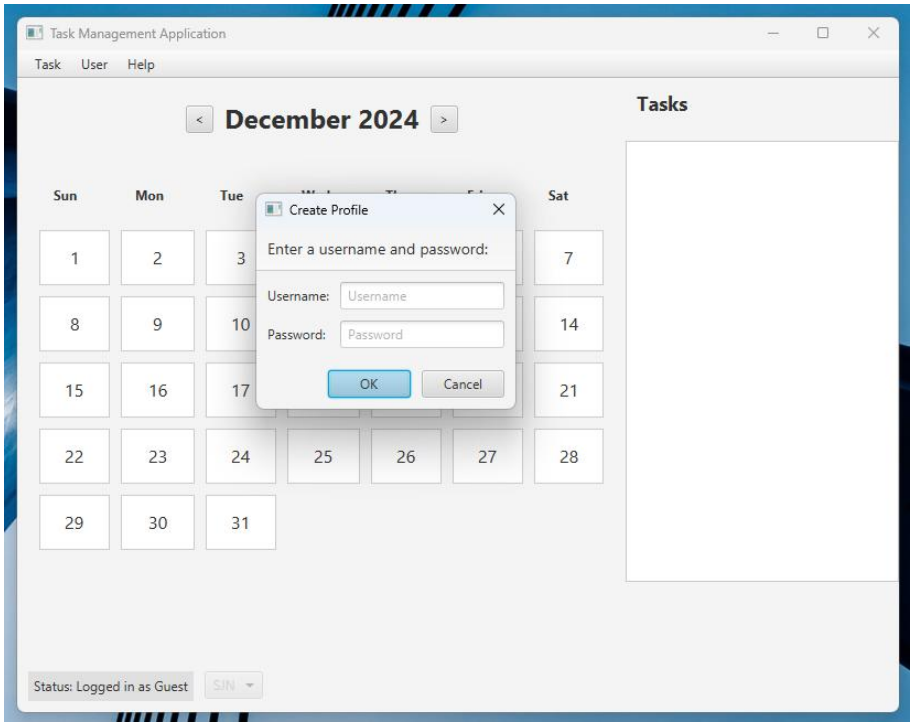
Όταν ξεκινάτε την εφαρμογή, θα εμφανιστεί η κεντρική οθόνη που περιλαμβάνει μια προβολή ημερολογίου, μια λίστα εργασιών και τη γραμμή μενού. Στην πρώτη χρήση, θα είστε συνδεδεμένοι ως επισκέπτης. Μπορείτε να δημιουργήσετε προφίλ ή να συνδεθείτε με έναν υπάρχοντα λογαριασμό μέσω του μενού User.



Εικόνα 32. Αρχικό παράθυρο της εφαρμογής.

Δημιουργία Προφίλ Χρήστη

Για να αποκτήσετε πλήρη πρόσβαση στις λειτουργίες της εφαρμογής, είναι απαραίτητο να δημιουργήσετε ένα προσωπικό προφίλ. Επιλέξτε Create Profile από το μενού User. Στη φόρμα που εμφανίζεται, εισάγετε το όνομα χρήστη και τον κωδικό πρόσβασής σας. Βεβαιωθείτε ότι τα στοιχεία που εισάγετε είναι σωστά, καθώς θα χρειαστούν για τη σύνδεσή σας. Αν το όνομα χρήστη είναι ήδη κατειλημμένο, η εφαρμογή θα σας ζητήσει να επιλέξετε διαφορετικό. Μετά τη δημιουργία του προφίλ, μπορείτε να συνδεθείτε μέσω της επιλογής Login.



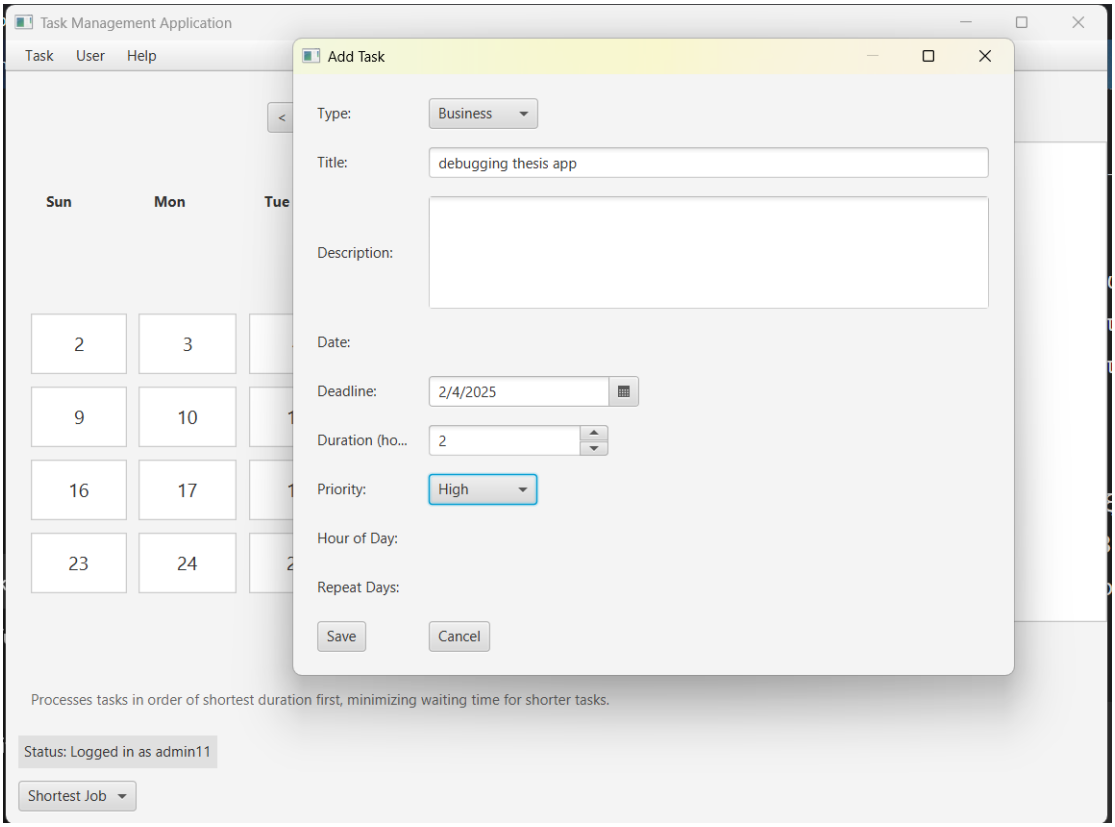
Εικόνα 33. Modal Box – φόρμα για τη δημιουργία νέου προφίλ χρήστη.

Σύνδεση και Αποσύνδεση

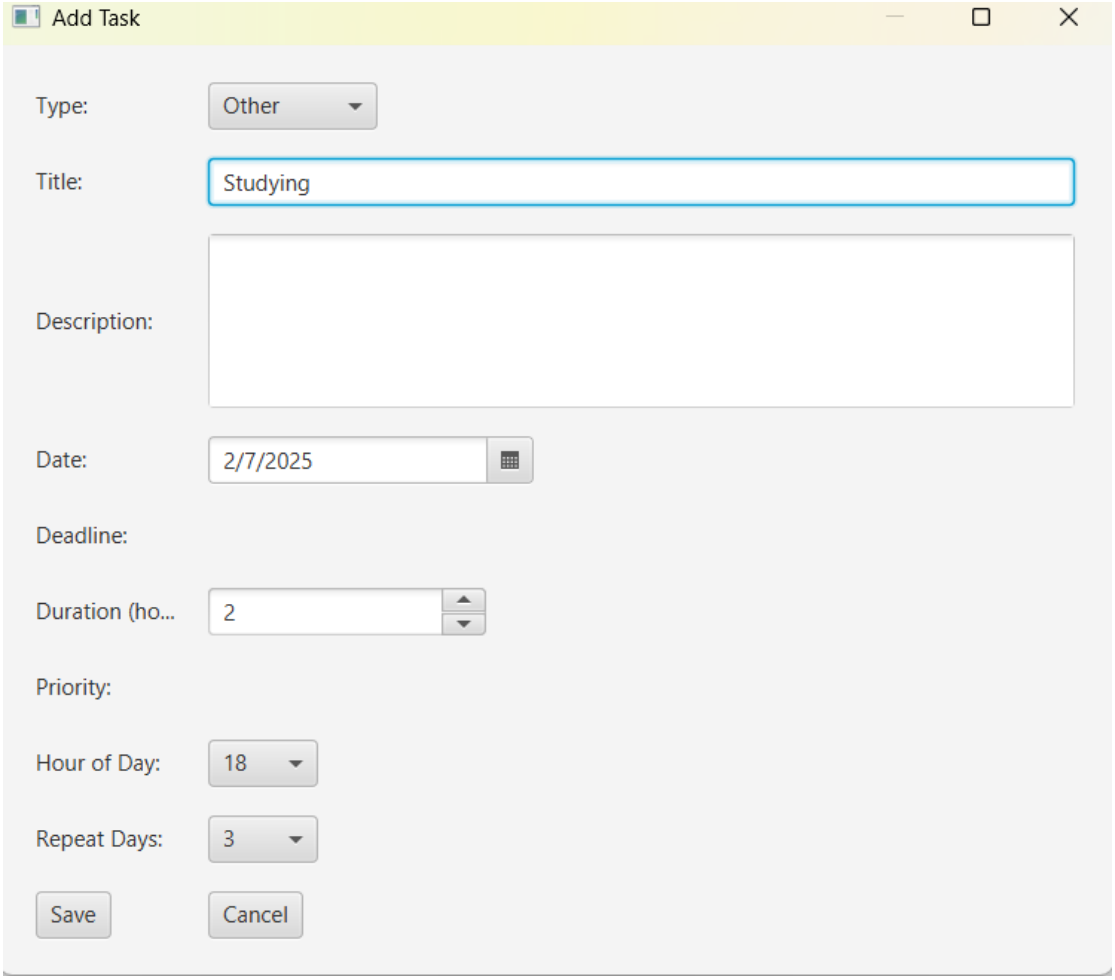
Η σύνδεση γίνεται μέσω της επιλογής Login από το μενού User. Στην οθόνη που εμφανίζεται, εισάγετε το όνομα χρήστη και τον κωδικό πρόσβασης που έχετε καταχωρίσει κατά τη δημιουργία του προφίλ σας. Με επιτυχημένη σύνδεση, η εφαρμογή θα σας χαιρετίσει με το όνομά σας στη γραμμή κατάστασης. Αντίθετα, μπορείτε να αποσυνδεθείτε επιλέγοντας Logout από το ίδιο μενού. Με την αποσύνδεση, η εφαρμογή επιστρέφει σε κατάσταση επισκέπτη, περιορίζοντας την πρόσβαση σε λειτουργίες όπως η προσθήκη ή επεξεργασία εργασιών.

Προσθήκη Νέας Εργασίας

Για να προσθέσετε μια νέα εργασία, επιλέξτε την επιλογή New Task από το μενού File. Θα εμφανιστεί ένα παράθυρο όπου μπορείτε να εισάγετε τον τίτλο, την περιγραφή, την προθεσμία, τη διάρκεια και την προτεραιότητα της εργασίας. Είναι σημαντικό να συμπληρώσετε όλα τα πεδία ώστε η εργασία να αποθηκευτεί σωστά. Μετά την εισαγωγή των στοιχείων, πατήστε Save για να καταχωρηθεί η εργασία σας. Η νέα εργασία θα εμφανιστεί στη λίστα εργασιών και στο ημερολόγιο, αν σχετίζεται με την ημερομηνία που έχετε επιλέξει.



Εικόνα 34. Modal Box – φόρμα για τη δημιουργία νέου business task.



Εικόνα 35. Modal Box – φόρμα για τη δημιουργία νέου other task.

Διαχείριση Εργασιών

Οι εργασίες που δημιουργείτε μπορούν να προβληθούν και να τροποποιηθούν μέσω της προβολής My Tasks από το μενού File. Εδώ μπορείτε να δείτε όλες τις εργασίες σας σε πίνακα, οργανωμένες με βάση την προθεσμία, την προτεραιότητα ή άλλες πληροφορίες. Για να επεξεργαστείτε μια εργασία, επιλέξτε την από τον πίνακα και πατήστε Edit Task. Αντίστοιχα, αν θέλετε να διαγράψετε μια εργασία, επιλέξτε την και πατήστε Delete Task. Οι αλλαγές ενημερώνονται αυτόματα στο ημερολόγιο και στη λίστα εργασιών.

Εργασίες διαφορετικού τύπου

Στην εφαρμογή, κατά τη δημιουργία μίας εργασίας ο χρήστης έχει τη δυνατότητα να επιλέξει αν η εργασία θα έχει σχέση με Business ή με κάτι προσωπικό. Εφόσον έχει σχέση με Business, η εργασία θα εντάσσεται στο παραδοσιακό οκτάωρο 9-5 (καθημερινές) και θα ταξινομείται με τον επιλεγμένο αλγόριθμο χρονολόγησης.

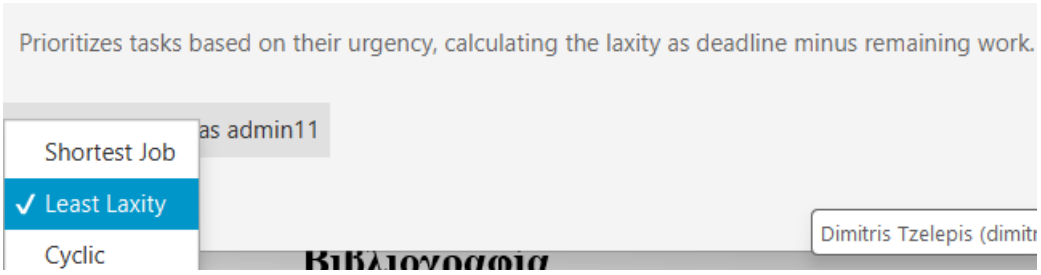
Στην περίπτωση που ο χρήστης επιλέξει την κατηγορία other, θα έχει τη δυνατότητα να προγραμματίσει ο ίδιος την ώρα και την ημερομηνία που θα προγραμματιστεί η εργασία καθώς και το εάν είναι επαναλαμβανόμενη ή όχι.

Χρήση του Ημερολογίου

Το ημερολόγιο σας επιτρέπει να παρακολουθείτε τις εργασίες σας με βάση τις ημερομηνίες. Το ημερολόγιο εμφανίζει τις ημέρες του τρέχοντος μήνα και σας επιτρέπει να μετακινείστε σε προηγούμενους ή επόμενους μήνες μέσω των κουμπιών πλοήγησης < και >. Εργασίες που σχετίζονται με συγκεκριμένες ημερομηνίες εμφανίζονται με ειδική σήμανση. Μπορείτε να κάνετε κλικ σε οποιαδήποτε ημέρα για να δείτε τις σχετικές εργασίες στη λίστα εργασιών που εμφανίζεται στα δεξιά.

Επιλογή Αλγορίθμου Προγραμματισμού

Η εφαρμογή υποστηρίζει διαφορετικούς αλγορίθμους προγραμματισμού για την καλύτερη κατανομή των εργασιών σας. Μέσα από το πλαίσιο επιλογής Scheduling Algorithm που βρίσκεται στην κάτω δεξιά γωνία της κεντρικής οθόνης, μπορείτε να επιλέξετε τον αλγόριθμο που προτιμάτε, όπως SJN, LLF, ή RR. Αφού συνδεθείτε, μπορείτε να αλλάξετε τον αλγόριθμο και να δείτε τις αλλαγές να εφαρμόζονται αυτόματα στο ημερολόγιο και στη λίστα εργασιών. Αυτός ο μηχανισμός σας βοηθά να βρείτε τη μέθοδο που ταιριάζει καλύτερα στις ανάγκες σας.



Εικόνα 36. Dropdown menu για την επιλογή αλγορίθμου χρονολόγησης.

Προβολή και Διαχείριση Εργασιών μέσω της Λίστας

Η λίστα εργασιών εμφανίζεται στη δεξιά πλευρά της κεντρικής οθόνης. Εδώ μπορείτε να δείτε τις εργασίες που σχετίζονται με την επιλεγμένη ημέρα στο ημερολόγιο. Η λίστα εμφανίζει βασικές πληροφορίες για κάθε εργασία, όπως τον τίτλο, την ώρα έναρξης και λήξης, καθώς και την προτεραιότητά της. Κάνοντας κλικ σε μια εργασία, μπορείτε να δείτε περισσότερες λεπτομέρειες, ενώ με δεξί κλικ έχετε τη δυνατότητα να την επεξεργαστείτε ή να τη διαγράψετε. Αυτός ο τρόπος διαχείρισης σας επιτρέπει να προσαρμόζετε εύκολα το πρόγραμμά σας.

Ρύθμιση Προτεραιοτήτων στις Εργασίες

Κατά τη δημιουργία ή επεξεργασία μιας εργασίας, μπορείτε να ορίσετε την προτεραιότητά της ως Υψηλή, Μεσαία, ή Χαμηλή. Η επιλογή αυτή καθορίζει τη σειρά με την οποία οι εργασίες

σας θα οργανωθούν και θα εκτελεστούν, ανάλογα με τον επιλεγμένο αλγόριθμο προγραμματισμού. Οι εργασίες με υψηλή προτεραιότητα προγραμματίζονται πρώτες, ενώ αυτές με χαμηλότερη μπορούν να μετακινηθούν προς το τέλος του προγράμματος. Με αυτόν τον τρόπο, διασφαλίζετε ότι οι πιο σημαντικές εργασίες σας λαμβάνουν τη δέουσα προσοχή.

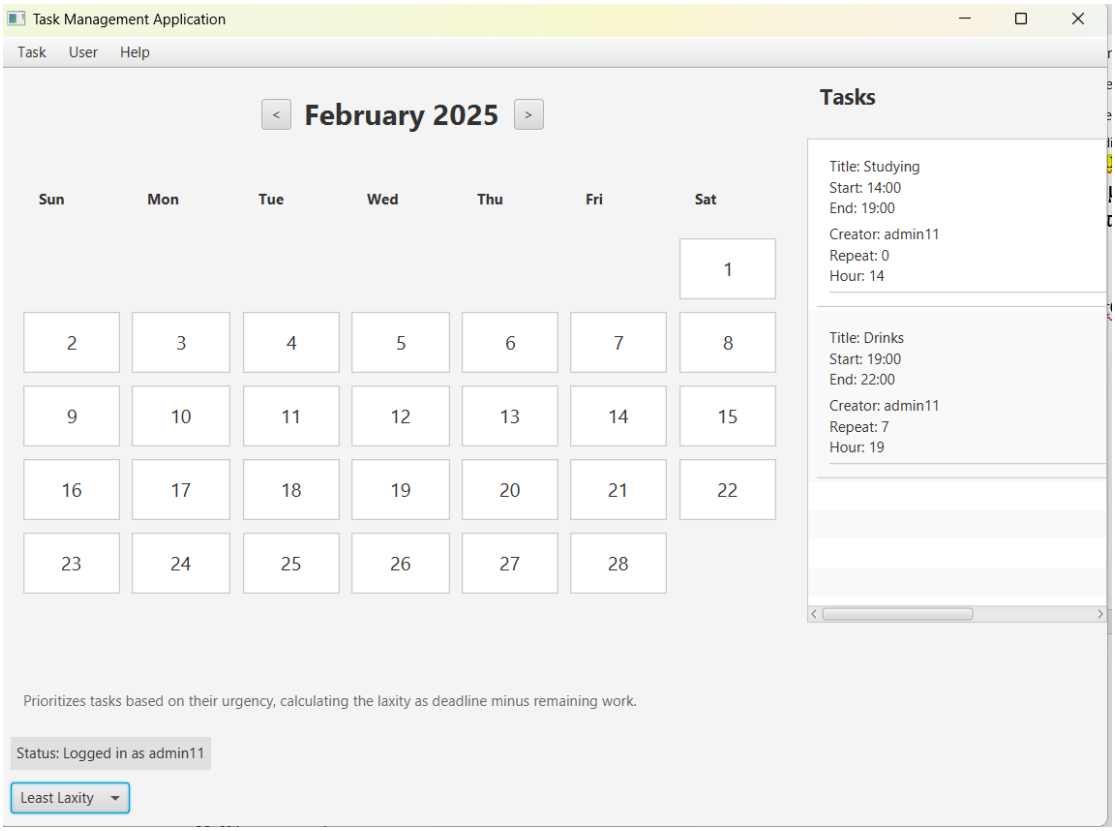
Χρήση της Γραμμής Κατάστασης

Η γραμμή κατάστασης, που βρίσκεται στο κάτω μέρος της οθόνης, σας ενημερώνει για την τρέχουσα κατάσταση σύνδεσης και για άλλες βασικές πληροφορίες. Όταν είστε συνδεδεμένοι, εμφανίζει το όνομα χρήστη σας, ενώ σε κατάσταση επισκέπτη αναφέρει ότι είστε συνδεδεμένοι ως "Guest". Επιπλέον, εμφανίζει ειδοποιήσεις για αλλαγές, όπως την επιτυχημένη δημιουργία εργασιών ή την αλλαγή του αλγορίθμου προγραμματισμού. Χρησιμοποιήστε τη γραμμή κατάστασης για να παρακολουθείτε την πρόοδο σας κατά τη χρήση της εφαρμογής.

Έξοδος από την Εφαρμογή

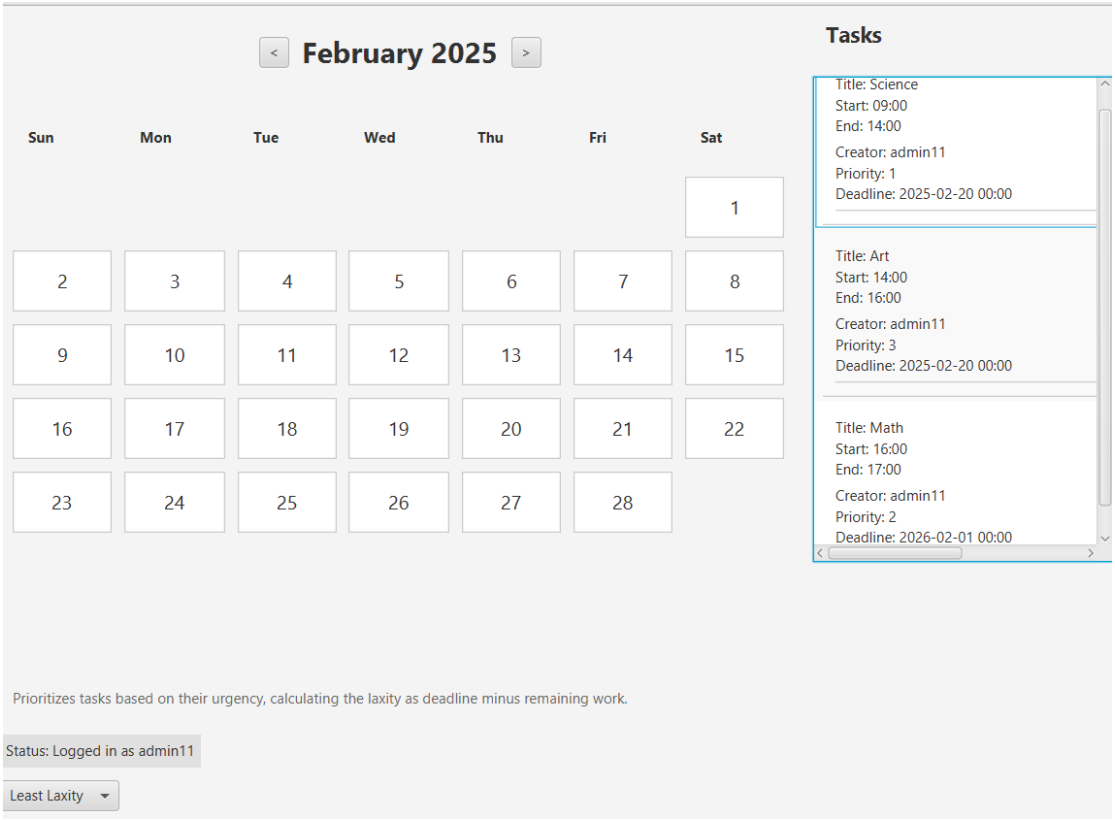
Για να κλείσετε την εφαρμογή, επιλέξτε Exit από το μενού File. Βεβαιωθείτε ότι έχετε αποθηκεύσει όλες τις αλλαγές σας, καθώς τυχόν μη αποθηκευμένα δεδομένα θα χαθούν. Η έξοδος ολοκληρώνεται με ασφάλεια και η εφαρμογή κλείνει, διατηρώντας τις πληροφορίες σας έτοιμες για την επόμενη εκκίνηση. Σε περίπτωση που αντιμετωπίσετε προβλήματα κατά την έξοδο, βεβαιωθείτε ότι έχετε αποθηκεύσει τις εργασίες σας ή επικοινωνήστε με τον διαχειριστή της εφαρμογής για περαιτέρω υποστήριξη.

ΑΠΛΟ ΠΑΡΑΔΕΙΓΜΑ ΠΡΟΓΡΑΜΜΑΤΟΣ



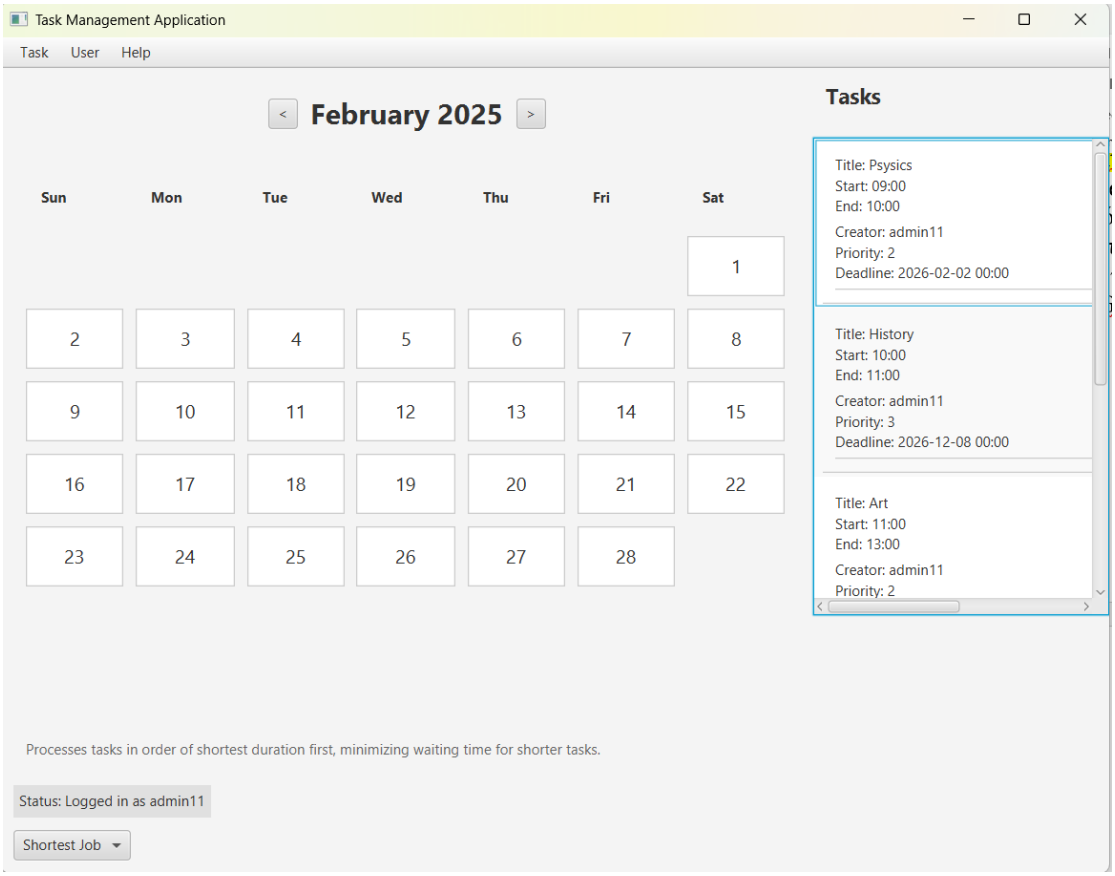
Εικόνα 36. Dropdown menu για την επιλογή αλγορίθμου χρονολόγησης.

Όπως μπορούμε να δούμε στο συγκεκριμένο παράδειγμα, έχουμε προγραμματίσει 2 other tasks. Το Studying το οποίο θα τρέξει το συγκεκριμένο Σάββατο και το Drinks που θα τρέχει κάθε Σάββατο (κάθε 7 ημέρες) την ίδια ώρα.



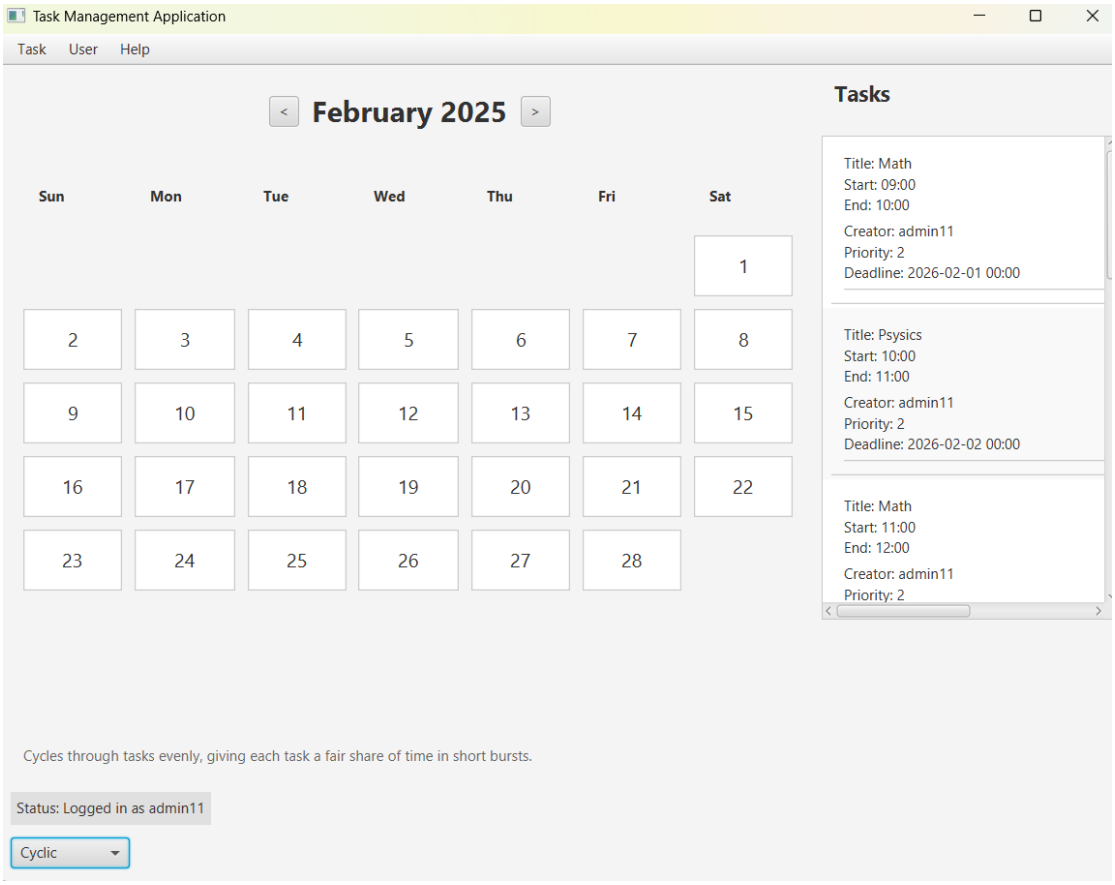
Εικόνα 37. Παράδειγμα προγραμματισμού tasks με αλγόριθμο Least Laxity First.

Χρησιμοποιώντας τον αλγόριθμο χρονολόγησης Least Laxity First παρατηρούμε ότι τα παραπάνω task γίνονται scheduled βάση της διορίας τους. Τα tasks Science και Art προηγούνται από το Task Math διότι η διορία τους είναι πιο κοντά. Εφόσον τα tasks Science και Art έχουν την ίδια διορία, τότε προηγείται το task με την υψηλότερη προτεραιότητα, δηλαδή το task Science.



Εικόνα 38. Παράδειγμα προγραμματισμού tasks με αλγόριθμο Shortest Job Next.

Χρησιμοποιώντας τον αλγόριθμο χρονολόγησης Shortest Job Next παρατηρούμε ότι τα παραπάνω tasks γίνονται scheduled με βάση τη διάρκειά τους. Άρα προηγούνται τα tasks Physics και History. Εφόσον τα tasks Physics και History έχουν την ίδια διορία, τότε προηγείται το task με την υψηλότερη προτεραιότητα, δηλαδή το Task Physic



Εικόνα 38. Παράδειγμα προγραμματισμού tasks με αλγόριθμο Round Robin.

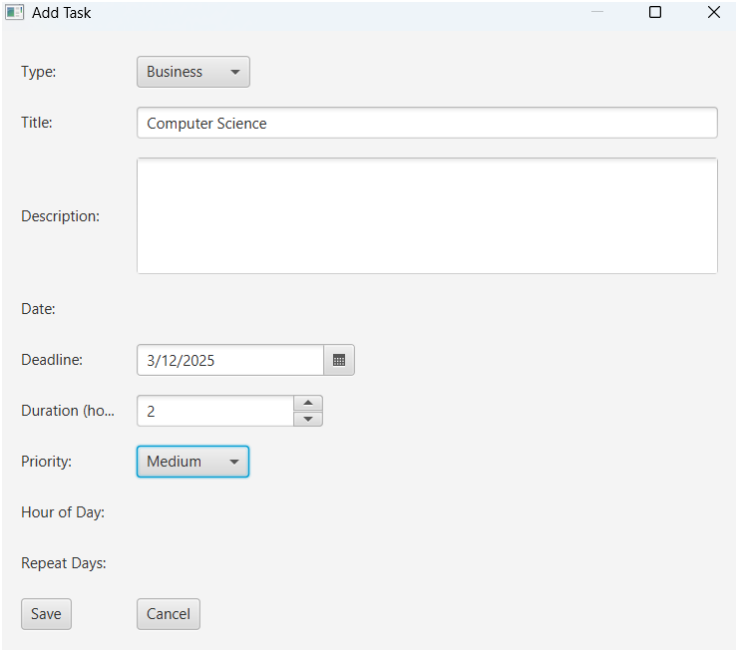
Χρησιμοποιώντας τον αλγόριθμο χρονολόγησης Round Robin παρατηρούμε ότι τα παραπάνω tasks εναλλάσσονται για να δώσουν την ευκαιρία στο χρήστη να εργάζεται παράλληλα σε πολλά tasks.

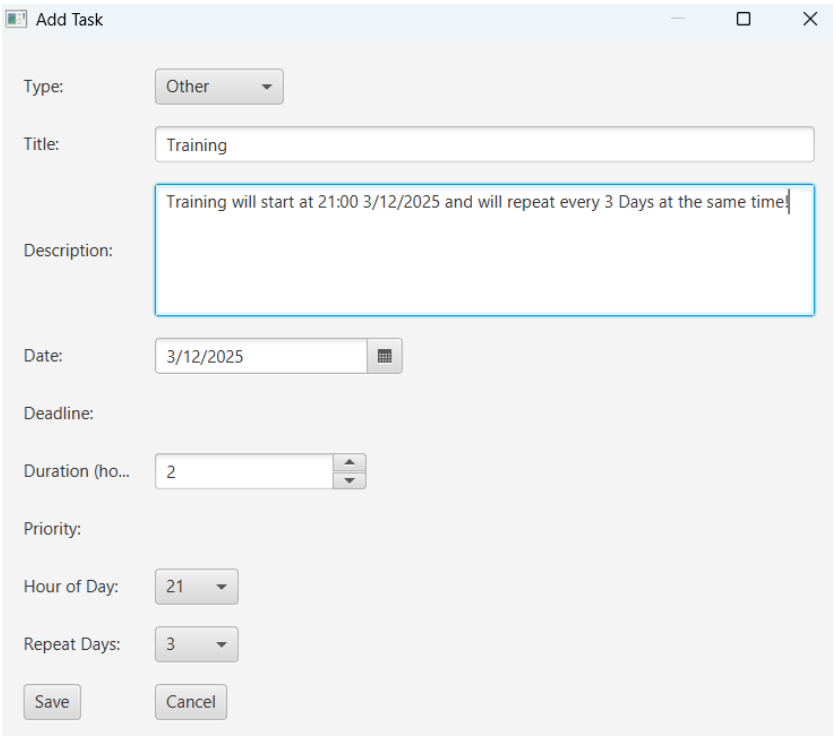
ΣΥΝΘΕΤΟ ΠΑΡΑΔΕΙΓΜΑ ΠΡΟΓΡΑΜΜΑΤΟΣ

Οργάνωση του Πανεπιστημιακού Προγράμματος με την Εφαρμογή Διαχείρισης Εργασιών

Ως φοιτητής πανεπιστημίου, η σωστή διαχείριση του χρόνου είναι κρίσιμη για την επιτυχία στις σπουδές αλλά και για τη διατήρηση μιας ισορροπημένης καθημερινότητας. Η εφαρμογή διαχείρισης εργασιών που χρησιμοποιείται παρέχει ένα εύχρηστο περιβάλλον που επιτρέπει την **οργάνωση ακαδημαϊκών και προσωπικών υποχρεώσεων** με ευελιξία.

Προσθήκη Νέων Εργασιών (Tasks)





Μέσα από τη λειτουργία **"Add Task"**, ο φοιτητής μπορεί να προσθέσει **δύο βασικές κατηγορίες εργασιών**:

- **Business Tasks**: Ακαδημαϊκές δραστηριότητες όπως διαλέξεις, μελέτη, παραδόσεις εργασιών, εξετάσεις και σεμινάρια.
- **Other Tasks**: Προσωπικές δραστηριότητες, όπως προπονήσεις, κοινωνικές εκδηλώσεις, χόμπι και άλλες καθημερινές υποχρεώσεις.

Κατά τη δημιουργία μιας εργασίας, ο φοιτητής μπορεί να ορίσει:

- **Τίτλο και Περιγραφή** για σαφή αναφορά στην εργασία.

- **Deadline** για να παρακολουθεί τις ημερομηνίες λήξης εργασιών και εξετάσεων.
- **Διάρκεια** σε ώρες, ώστε να προγραμματίζει τον χρόνο που θα αφιερώσει.
- **Ώρα της ημέρας**, εάν η εργασία είναι προγραμματισμένη για συγκεκριμένη χρονική στιγμή.
- **Προτεραιότητα (Low, Medium, High)** για να ταξινομεί τις πιο σημαντικές εργασίες.
- **Επανάληψη** (Repeat Days) αν μια εργασία πρέπει να γίνεται περιοδικά, π.χ. κάθε εβδομάδα.

Υπάρχουσες Καταχωρημένες Εργασίες

My Tasks

Title	Type	Description	Deadline	Priority	Duration	Date	Hour	Repeat every (days)
Math	Business		2026-02-01 00:00	2	3			
Psysics	Business		2026-02-02 00:00	2	1			
Gym	Other				1	2025-02-12	19	3
Chemistry	Business		2026-02-04 00:00	1	3			
Music	Business		2026-02-09 00:00	3	4			
English	Business	Speaking	2026-02-03 00:00	1	4			
Science	Business		2026-01-20 00:00	1	5			
History	Business		2026-12-08 00:00	1	1			
Chess	Other				1	2025-02-12	19	7
Swimming	Other				1	2025-02-12	19	14
Studying	Other				5	2025-02-08	14	0
Computer Science	Business		2026-02-10 00:00	1	4			
Analytics	Business		2025-03-22 00:00	2	3			
Big Project	Business		2025-03-20 00:00	2	5			
Java	Business		2025-03-22 00:00	1	4			
Python	Business		2025-03-27 00:00	3	3			

Ο φοιτητής έχει ήδη προσθέσει **αρκετές ακαδημαϊκές και προσωπικές υποχρεώσεις**, διαχωρισμένες σε κατηγορίες. Μερικές από αυτές περιλαμβάνουν:

Ακαδημαϊκές Υποχρεώσεις (Business Tasks)

- **Math (01/02/2026, Προτεραιότητα: 2, Διάρκεια: 3 ώρες)**
- **Physics (02/02/2026, Προτεραιότητα: 2, Διάρκεια: 1 ώρα)**
- **Chemistry (04/02/2026, Προτεραιότητα: 1, Διάρκεια: 3 ώρες)**
- **Music (09/02/2026, Προτεραιότητα: 3, Διάρκεια: 4 ώρες)**
- **English - Speaking (03/02/2026, Προτεραιότητα: 1, Διάρκεια: 4 ώρες)**
- **Science (20/01/2026, Προτεραιότητα: 1, Διάρκεια: 5 ώρες)**
- **History (08/12/2026, Προτεραιότητα: 1, Διάρκεια: 1 ώρα)**
- **Computer Science (10/02/2026, Προτεραιότητα: 1, Διάρκεια: 4 ώρες)**
- **Analytics (22/03/2025, Προτεραιότητα: 2, Διάρκεια: 3 ώρες)**

- **Big Project (20/03/2025, Προτεραιότητα: 3, Διάρκεια: 8 ώρες)**
- **Java (22/03/2025, Προτεραιότητα: 1, Διάρκεια: 4 ώρες)**
- **Python (27/03/2025, Προτεραιότητα: 3, Διάρκεια: 3 ώρες)**
- **DevOps (28/03/2025, Προτεραιότητα: 2, Διάρκεια: 5 ώρες)**
- **Seminar Part A (28/03/2025, Προτεραιότητα: 2, Διάρκεια: 20 ώρες)**
- **Seminar Part B (31/03/2025, Προτεραιότητα: 2, Διάρκεια: 20 ώρες)**

Προσωπικές Υποχρεώσεις (Other Tasks)

- Gym (Επαναλαμβάνεται κάθε 3 μέρες, Διάρκεια: 1 ώρα στις 19:00)

- Chess (Επαναλαμβάνεται κάθε 7 μέρες, Διάρκεια: 2 ώρες στις 19:00)

- Swimming (Επαναλαμβάνεται κάθε 14 μέρες, Διάρκεια: 1 ώρα στις 19:00)

- Studying (08/02/2025, Προτεραιότητα: 1, Διάρκεια: 5 ώρες στις 14:00)

- Drinks (15/03/2025, Προτεραιότητα: 2, Διάρκεια: 2 ώρες στις 22:00)

Ο φοιτητής έχει διαμορφώσει ένα **ισορροπημένο πρόγραμμα** μεταξύ σπουδών και προσωπικής ζωής, με προγραμματισμένες ώρες μελέτης, προπονήσεων και κοινωνικών δραστηριοτήτων.

Επιλογή Αλγορίθμου Χρονοπρογραμματισμού

Η εφαρμογή παρέχει τρεις διαφορετικούς αλγορίθμους διαχείρισης εργασιών, καθένας με συγκεκριμένα **πλεονεκτήματα και μειονεκτήματα**. Ο φοιτητής μπορεί να επιλέξει τον κατάλληλο αλγόριθμο ανάλογα με τις ανάγκες του:

1. Shortest Job Next (SJN) – Προτεραιότητα σε σύντομες εργασίες

✓ Πλεονεκτήματα:

- Εξαιρετικά χρήσιμος για γρήγορη ολοκλήρωση μικρών εργασιών.
- Βοηθά τον φοιτητή να ολοκληρώσει πρώτα τις εργασίες μικρής διάρκειας, μειώνοντας το συνολικό χρόνο αναμονής.

✓ Χρήσιμο για:

- Τακτοποίηση σύντομων ακαδημαϊκών εργασιών όπως **Physics (1 ώρα), History (1 ώρα), Chess (2 ώρες)**.
- Οργάνωση προσωπικών δραστηριοτήτων που δεν απαιτούν πολύ χρόνο, όπως το **Gym (1 ώρα) ή Swimming (1 ώρα)**.

✕ Μειονεκτήματα:

- Εργασίες μεγάλης διάρκειας (όπως το "Big Project" ή το "Seminar Part A") μπορεί να καθυστερούν συνεχώς, καθώς οι μικρότερες εργασίες θα προηγούνται.

- Αν ο φοιτητής έχει πολλές μικρές εργασίες, ίσως παραμελήσει τις σημαντικές και απαιτητικές υποχρεώσεις.
- Μεγάλες εργασίες όπως **Big Project (8 ώρες) ή Seminar Part A (20 ώρες)** μπορεί να παραμεληθούν, καθυστερώντας την πρόοδό τους.
- Αν ο φοιτητής έχει μεγάλες εργασίες με επείγουσες προθεσμίες, αυτός ο αλγόριθμος ίσως δεν είναι η καλύτερη επιλογή.

✅ **Πότε να το χρησιμοποιήσει:**

- Για εβδομάδες με μικρές, αλλά πολλές εργασίες.
- Όταν θέλει να «καθαρίσει» γρήγορα το πρόγραμμά του από εύκολα tasks.

---

2. Least Laxity First (LLF) – Προτεραιότητα στις εργασίες που πλησιάζει η προθεσμία

✓ **Πλεονεκτήματα:**

- Ιδανικός για την αποφυγή καθυστερήσεων και missed deadlines.
- Βοηθά στη σωστή διαχείριση εργασιών με σφιχτές προθεσμίες, όπως εξετάσεις και παραδόσεις εργασιών.

Χρήσιμο για:

- Εργασίες με αυστηρά deadlines όπως **Python (27/03/2025), DevOps (28/03/2025), Seminar Part B (31/03/2025)**.
- Αποφυγή εκπρόθεσμων εργασιών, καθώς κάθε φορά η εργασία που έχει τη μικρότερη διαθέσιμη χαλαρότητα προγραμματίζεται πρώτη.

✕ **Μειονεκτήματα:**

- Μπορεί να οδηγήσει σε συχνές αλλαγές προγραμματισμού, καθώς οι εργασίες με μικρότερο διαθέσιμο χρόνο θα προτιμώνται συνεχώς.
- Οι εργασίες που έχουν αρκετό περιθώριο μέχρι την προθεσμία τους μπορεί να αγνοηθούν προσωρινά, κάτι που ίσως προκαλέσει συσσώρευση υποχρεώσεων στο μέλλον.
- Συνεχείς αλλαγές προτεραιότητας μπορεί να οδηγήσουν σε μη ομαλή ροή εργασίας.
- Αν μια εργασία έχει ευέλικτη προθεσμία, ίσως μείνει πίσω λόγω άλλων πιο πιεστικών εργασιών.

✅ **Πότε να το χρησιμοποιήσει:**

- Κατά την περίοδο εξετάσεων, ώστε να δίνει προτεραιότητα στα πιο πιεστικά μαθήματα.
- Όταν έχει σημαντικά deadlines, όπως παραδόσεις εργασιών.

---

3. Round Robin (RR) – Ισορροπημένος καταμερισμός χρόνου

✓ **Πλεονεκτήματα:**

- Διανέμει τον διαθέσιμο χρόνο ισότιμα σε όλες τις εργασίες, αποτρέποντας την υπερβολική εστίαση σε μία μόνο υποχρέωση.
- Ιδανικός για multitasking, καθώς καμία εργασία δεν παραμένει ανεκτέλεστη για μεγάλο χρονικό διάστημα.

Χρήσιμο για:

- Μεγάλες εργασίες που δεν μπορούν να ολοκληρωθούν σε μία συνεδρία, όπως το **Big Project (8 ώρες), Seminar Part A (20 ώρες)**.
- Φοιτητές που θέλουν να ασχολούνται με πολλές εργασίες ταυτόχρονα, διατηρώντας έναν ισορροπημένο ρυθμό προόδου.
- Οργάνωση μελέτης, καθώς επιτρέπει στον φοιτητή να μοιράζει τον χρόνο του σε διαφορετικά αντικείμενα, π.χ. **Math – 3 ώρες, Computer Science – 4 ώρες, Chemistry – 3 ώρες**.

✕ Μειονεκτήματα:

- Οι εργασίες μεγάλης διάρκειας ενδέχεται να χρειαστούν αρκετούς κύκλους για να ολοκληρωθούν, καθώς καταλαμβάνουν μικρό μέρος του διαθέσιμου χρόνου σε κάθε βήμα.
- Δεν δίνει προτεραιότητα σε επείγουσες εργασίες, κάτι που μπορεί να είναι μειονέκτημα αν υπάρχουν πολλές προθεσμίες.

✅ Πότε να το χρησιμοποιήσει:

- Όταν θέλει να διαχειριστεί όλες τις υποχρεώσεις του εξίσου, χωρίς να αφήνει καμία πίσω.
- Σε περιόδους όπου οι προθεσμίες δεν είναι τόσο πιεστικές, αλλά υπάρχει ανάγκη για ισορροπημένη διαχείριση των εργασιών.

Συμπέρασμα

Ο φοιτητής έχει την ευκαιρία να επιλέξει τον κατάλληλο αλγόριθμο ανάλογα με τις ανάγκες του κάθε εβδομάδα.

- SJN όταν θέλει να απαλλαγεί γρήγορα από μικρές εργασίες.
- LLF όταν υπάρχουν προθεσμίες που πρέπει να τηρηθούν αυστηρά.
- RR όταν θέλει να διαχειριστεί μεγάλες εργασίες με ισορροπημένο τρόπο.

Η σωστή χρήση των αλγορίθμων βοηθά τον φοιτητή να είναι πιο παραγωγικός και οργανωμένος, αποφεύγοντας την αναβλητικότητα και τη συσσώρευση εργασιών.

Ταξινόμηση των εργασιών με τον αλγόριθμο Shortest Job Next (SJN)

Ο αλγόριθμος **Shortest Job Next** δίνει προτεραιότητα στις εργασίες με τη μικρότερη διάρκεια. Αν δύο εργασίες έχουν την ίδια διάρκεια, τότε προτιμά αυτή με τη **μεγαλύτερη προτεραιότητα (1 είναι η υψηλότερη προτεραιότητα)**.

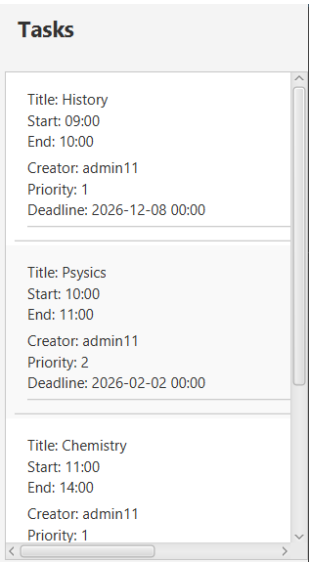
Ταξινομημένες εργασίες:

1. **History (08/12/2026, Προτεραιότητα: 1, Διάρκεια: 1 ώρα)**
2. **Physics (02/02/2026, Προτεραιότητα: 2, Διάρκεια: 1 ώρα)**
3. **Chemistry (04/02/2026, Προτεραιότητα: 1, Διάρκεια: 3 ώρες)**
4. **Math (01/02/2026, Προτεραιότητα: 2, Διάρκεια: 3 ώρες)**
5. **Analytics (22/03/2025, Προτεραιότητα: 2, Διάρκεια: 3 ώρες)**
6. **Python (27/03/2025, Προτεραιότητα: 3, Διάρκεια: 3 ώρες)**
7. **English - Speaking (03/02/2026, Προτεραιότητα: 1, Διάρκεια: 4 ώρες)**

8. **Computer Science (10/02/2026, Προτεραιότητα: 1, Διάρκεια: 4 ώρες)**
9. **Java (22/03/2025, Προτεραιότητα: 1, Διάρκεια: 4 ώρες)**
10. **Music (09/02/2026, Προτεραιότητα: 3, Διάρκεια: 4 ώρες)**
11. **Science (20/01/2026, Προτεραιότητα: 1, Διάρκεια: 5 ώρες)**
12. **DevOps (28/03/2025, Προτεραιότητα: 2, Διάρκεια: 5 ώρες)**
13. **Big Project (20/03/2025, Προτεραιότητα: 3, Διάρκεια: 8 ώρες)**
14. **Seminar Part A (28/03/2025, Προτεραιότητα: 2, Διάρκεια: 20 ώρες)**
15. **Seminar Part B (31/03/2025, Προτεραιότητα: 2, Διάρκεια: 20 ώρες)**

Παρατηρήσεις:

- Οι εργασίες **History** και **Physics** είχαν τη μικρότερη διάρκεια (1 ώρα), αλλά η **History** προηγήθηκε επειδή έχει υψηλότερη προτεραιότητα (1 έναντι 2).



- Οι εργασίες **Music, English - Speaking, Computer Science, Java** είχαν την ίδια διάρκεια (4 ώρες), οπότε ταξινομήθηκαν με βάση την προτεραιότητα.
- Οι μεγαλύτερες εργασίες, όπως **Big Project (8 ώρες) και τα δύο Σεμινάρια (20 ώρες)**, τοποθετήθηκαν στο τέλος.

Με αυτήν την ταξινόμηση, ο φοιτητής μπορεί να ολοκληρώσει πρώτα τις μικρότερες εργασίες και να μειώσει τον συνολικό χρόνο αναμονής, **βελτιστοποιώντας τον φόρτο εργασίας του**. Όπως βλέπουμε στις εικόνες, η ταξινόμηση έγινε όπως περιέγραψα.

Ταξινόμηση των εργασιών με τον αλγόριθμο Least Laxity First (LLF)

Ο αλγόριθμος **Least Laxity First (LLF)** προγραμματίζει πρώτα τις εργασίες με τη μικρότερη "χαλαρότητα" (**laxity**), που υπολογίζεται ως εξής:

Laxity = (Deadline−Σημερινή Ημερομηνία) - Διάρκεια

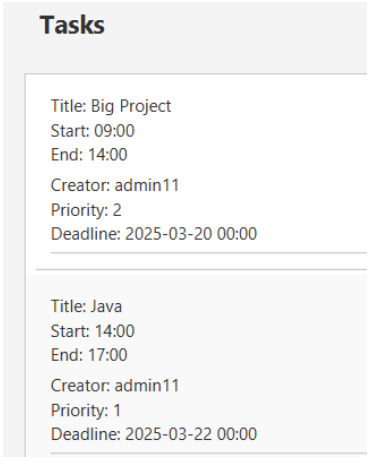
Αν δύο εργασίες έχουν την ίδια χαλαρότητα, τότε δίνεται προτεραιότητα στην εργασία με τη **μεγαλύτερη προτεραιότητα** (**1 είναι η υψηλότερη προτεραιότητα**).

Ταξινομημένες εργασίες με LLF:

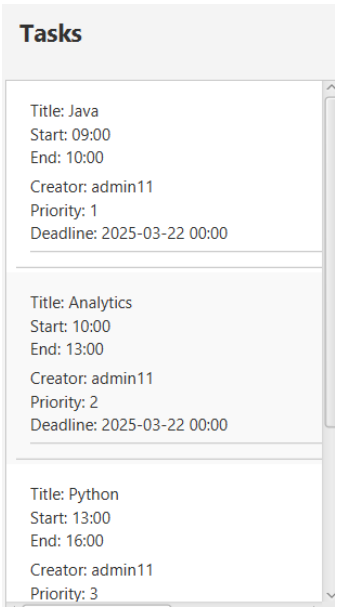
1. **Big Project (20/03/2025, Προτεραιότητα: 3, Διάρκεια: 8 ώρες, Laxity: 70)**
2. **Java (22/03/2025, Προτεραιότητα: 1, Διάρκεια: 4 ώρες, Laxity: 80)**
3. **Analytics (22/03/2025, Προτεραιότητα: 2, Διάρκεια: 3 ώρες, Laxity: 80)**
4. **Python (27/03/2025, Προτεραιότητα: 3, Διάρκεια: 3 ώρες, Laxity: 85)**
5. **Seminar Part A (28/03/2025, Προτεραιότητα: 2, Διάρκεια: 20 ώρες, Laxity: 86)**
6. **DevOps (28/03/2025, Προτεραιότητα: 2, Διάρκεια: 5 ώρες, Laxity: 86)**
7. **Seminar Part B (31/03/2025, Προτεραιότητα: 2, Διάρκεια: 20 ώρες, Laxity: 89)**
8. **Science (20/01/2026, Προτεραιότητα: 1, Διάρκεια: 5 ώρες, Laxity: 393)**
9. **Math (01/02/2026, Προτεραιότητα: 2, Διάρκεια: 3 ώρες, Laxity: 395)**
10. **Physics (02/02/2026, Προτεραιότητα: 2, Διάρκεια: 1 ώρα, Laxity: 396)**
11. **English - Speaking (03/02/2026, Προτεραιότητα: 1, Διάρκεια: 4 ώρες, Laxity: 397)**
12. **Chemistry (04/02/2026, Προτεραιότητα: 1, Διάρκεια: 3 ώρες, Laxity: 398)**
13. **Computer Science (10/02/2026, Προτεραιότητα: 1, Διάρκεια: 4 ώρες, Laxity: 404)**
14. **Music (09/02/2026, Προτεραιότητα: 3, Διάρκεια: 4 ώρες, Laxity: 403)**
15. **History (08/12/2026, Προτεραιότητα: 1, Διάρκεια: 1 ώρα, Laxity: 707)**

Παρατηρήσεις:

- Οι εργασίες με τα **πιο πιεστικά deadlines** (Big Project, Seminar Part A, Seminar Part B) εμφανίζονται πρώτες, καθώς έχουν τη **χαμηλότερη χαλαρότητα (laxity)**.



(Σημείωση: Η εργασία Java έχει διάρκεια 4 ωρών και δε «χωράει» στο οκτάωρο 9-5. Οπότε θα συνεχίσει την επόμενη ημέρα)



- Όταν δύο εργασίες έχουν **ίδια χαλαρότητα**, τότε ταξινομούνται με βάση την **προτεραιότητα** (π.χ. Seminar Part A και DevOps έχουν ίδια χαλαρότητα 86, αλλά το Seminar Part A έχει υψηλότερη προτεραιότητα).
- Οι εργασίες που έχουν **πολύ μακρινή προθεσμία**, όπως το **History (08/12/2026, Laxity: 707)**, τοποθετούνται τελευταίες, καθώς υπάρχει αρκετός χρόνος για την ολοκλήρωσή τους.

Ο αλγόριθμος **LLF είναι ιδανικός για εβδομάδες όπου υπάρχουν πολλές επείγουσες εργασίες**, καθώς εξασφαλίζει ότι οι εργασίες που κινδυνεύουν να μην ολοκληρωθούν εγκαίρως προγραμματίζονται πρώτες.

Ταξινόμηση των Εργασιών με τον Αλγόριθμο Round Robin (RR) – Εναλλαγή Ανά 1 Ώρα

Ο **Round Robin (RR)** προγραμματίζει τις εργασίες κυκλικά, δίνοντας σε κάθε εργασία **ίση κατανομή χρόνου**. Σε αυτήν την περίπτωση, κάθε εργασία λαμβάνει **1 ώρα εκτέλεσης ανά κύκλο** μέχρι να ολοκληρωθεί.

Διαδικασία Χρονοπρογραμματισμού με RR (Quantum = 1 ώρα)

Ο αλγόριθμος θα λειτουργήσει ως εξής:

1. **Ξεκινάει από την πρώτη εργασία** και της αποδίδει 1 ώρα εκτέλεσης.
2. **Μετακινείται στην επόμενη εργασία**, αποδίδοντάς της επίσης 1 ώρα.
3. **Συνεχίζει κυκλικά**, επαναλαμβάνοντας τη διαδικασία, έως ότου όλες οι εργασίες ολοκληρωθούν.

Ταξινομημένες Εργασίες με RR (Ανά 1 ώρα)

1ος Κύκλος (Όλες λαμβάνουν 1 ώρα εκτέλεσης)

- **Big Project** *(Απομένουν: 7 ώρες)*
- **Java** *(Απομένουν: 3 ώρες)*
- **Analytics** *(Απομένουν: 2 ώρες)*
- **Python** *(Απομένουν: 2 ώρες)*
- **Seminar Part A** *(Απομένουν: 19 ώρες)*
- **DevOps** *(Απομένουν: 4 ώρες)*
- **Seminar Part B** *(Απομένουν: 19 ώρες)*
- **Science** *(Απομένουν: 4 ώρες)*
- **Math** *(Απομένουν: 2 ώρες)*
- **Physics** *(Ολοκληρώθηκε ✅)*
- **English - Speaking** *(Απομένουν: 3 ώρες)*
- **Chemistry** *(Απομένουν: 2 ώρες)*
- **Computer Science** *(Απομένουν: 3 ώρες)*
- **Music** *(Απομένουν: 3 ώρες)*
- **History** *(Ολοκληρώθηκε ✅)*

2ος Κύκλος

- **Big Project** *(Απομένουν: 6 ώρες)*
- **Java** *(Απομένουν: 2 ώρες)*
- **Analytics** *(Απομένουν: 1 ώρα)*
- **Python** *(Απομένουν: 1 ώρα)*
- **Seminar Part A** *(Απομένουν: 18 ώρες)*

- **DevOps** *(Απομένουν: 3 ώρες)*
- **Seminar Part B** *(Απομένουν: 18 ώρες)*
- **Science** *(Απομένουν: 3 ώρες)*
- **Math** *(Απομένουν: 1 ώρα)*
- **English - Speaking** *(Απομένουν: 2 ώρες)*
- **Chemistry** *(Απομένουν: 1 ώρα)*
- **Computer Science** *(Απομένουν: 2 ώρες)*
- **Music** *(Απομένουν: 2 ώρες)*

3ος Κύκλος

- **Big Project** *(Απομένουν: 5 ώρες)*
- **Java** *(Απομένουν: 1 ώρα)*
- **Analytics** *(Ολοκληρώθηκε ✅)*
- **Python** *(Ολοκληρώθηκε ✅)*
- **Seminar Part A** *(Απομένουν: 17 ώρες)*
- **DevOps** *(Απομένουν: 2 ώρες)*
- **Seminar Part B** *(Απομένουν: 17 ώρες)*
- **Science** *(Απομένουν: 2 ώρες)*
- **Math** *(Ολοκληρώθηκε ✅)*
- **English - Speaking** *(Απομένουν: 1 ώρα)*
- **Chemistry** *(Ολοκληρώθηκε ✅)*
- **Computer Science** *(Απομένουν: 1 ώρα)*
- **Music** *(Απομένουν: 1 ώρα)*

Τελικοί Κύκλοι

Η διαδικασία συνεχίζεται με εναλλαγή **1 ώρας ανά εργασία**, μέχρι όλες οι εργασίες να ολοκληρωθούν. Οι **μεγάλες εργασίες**, όπως το **Seminar Part A & B (20 ώρες) και το Big Project (8 ώρες)**, χρειάζονται **πολλαπλούς κύκλους** για να ολοκληρωθούν.

Παρατηρήσεις

- **Οι μικρές εργασίες (1-3 ώρες) ολοκληρώνονται γρήγορα, μέσα σε λίγους κύκλους.**
- **Οι μεγάλες εργασίες (Big Project, Seminar Part A & B) μοιράζονται σε πολλούς κύκλους, αλλά δεν παραμελούνται, καθώς κάθε εργασία παίρνει ίσο χρόνο σε κάθε γύρο.**

- **Round Robin είναι εξαιρετικός για εργασίες που απαιτούν multitasking**, επειδή **καμία εργασία δεν αφήνεται ανεπεξέργαστη για πολύ καιρό**.

Ο αλγόριθμος **Round Robin είναι κατάλληλος για φοιτητές που θέλουν να προοδεύουν σταδιακά σε όλες τις εργασίες τους**, αντί να εστιάζουν σε μία εργασία κάθε φορά!