ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Σχεδιασμός και Ανάπτυξη Ολοκληρωμένης Διαδικτυακής Πλατφόρμας Ηλεκτρονικών Κρατήσεων και Αξιολόγησης Κινηματογραφικών Υπηρεσιών Design and Development of an Integrated Web Platform for Online Booking and Rating of Cinematic Services
Ονοματεπώνυμο Φοιτητή	Γκότσης Αθανάσιος
Πατρώνυμο	Ιωάννης
Αριθμός Μητρώου	Π / 21020
Επιβλέπουσα	Βίρβου Μαρία, Καθηγήτρια - Κοσμήτορας Σχολής Τεχνολογιών Πληροφορικής και Επικοινωνιών

Ημερομηνία Παράδοσης 9/2026

Copyright ©

Περίληψη

Η παρούσα εργασία παρουσιάζει τη μελέτη, τον σχεδιασμό και την ανάπτυξη του **CINEHUB**, μιας ολοκληρωμένης και ασφαλούς διαδικτυακής πλατφόρμας για τη διαχείριση κινηματογραφικών υπηρεσιών. Το σύστημα στοχεύει στην επίλυση του προβλήματος του κατακερματισμού της πληροφορίας, ενοποιώντας σε μία υποδομή την ενημέρωση για ταινίες, την ηλεκτρονική κράτηση εισιτηρίων και την κοινωνική αλληλεπίδραση μέσω αξιολογήσεων.

Αρχιτεκτονικά, η πλατφόρμα ακολουθεί το μοντέλο **Client-Server (N-Tier)**. Το επίπεδο παρουσίασης (Frontend) αναπτύχθηκε με τη βιβλιοθήκη **React.js**, προσφέροντας μια αποκρινόμενη εμπειρία Single Page Application (SPA). Το επίπεδο επιχειρησιακής λογικής (Backend) βασίζεται στο πλαίσιο **Spring Boot** (Java), ενώ η διαχείριση των δεδομένων γίνεται μέσω του **Spring Data JPA** σε σχεσιακή βάση δεδομένων. Η ασφάλεια του συστήματος διασφαλίζεται μέσω του **Spring Security** και της χρήσης **JSON Web Tokens (JWT)** για stateless αυθεντικοποίηση και έλεγχο πρόσβασης βάσει ρόλων (RBAC).

Στα κύρια λειτουργικά χαρακτηριστικά περιλαμβάνονται:

- **Για τον πελάτη:** Δυναμική αναζήτηση ταινιών (και φωνητική), διαδραστική επιλογή θέσεων με έξυπνους αλγορίθμους, αγορά συνοδευτικών προϊόντων και αυτόματη έκδοση ψηφιακού εισιτηρίου (PDF με QR Code).
- **Για τον διαχειριστή:** Εργαλεία μαζικού προγραμματισμού προβολών (Bulk Scheduling) με αυτόματο έλεγχο επικαλύψεων, δυναμική χαρτογράφηση αιθουσών και ζωντανή εποπτεία κρατήσεων.

Τέλος, η εργασία εξετάζει τη σύνδεση με τη διεθνή βιβλιογραφία στους τομείς της **Τεχνητής Νοημοσύνης (AI)** και της **Αλληλεπίδρασης Ανθρώπου-Υπολογιστή (HCI)**. Προτείνονται μελλοντικές επεκτάσεις όπως η ενσωμάτωση συστημάτων συστάσεων (Recommender Systems), η δημιουργία αυτόνομων εφαρμογών για κινητά και η διασύνδεση με πραγματικές πύλες πληρωμών.

Λέξεις Κλειδιά: Διαδικτυακή πλατφόρμα , Ηλεκτρονική κράτηση εισιτηρίων , Κοινωνική αλληλεπίδραση , Full-Stack development , Αρχιτεκτονική RESTful API , Single Page Applications , Αρχιτεκτονική Client-Server , Ασφάλεια δεδομένων , Εμπειρία χρήστη , Διαδραστική επιλογή θέσεων , Ψηφιακό εισιτήριο , Κώδικας QR , Μαζικός προγραμματισμός , Πίνακες ελέγχου , Έλεγχος πρόσβασης βάσει ρόλων , React.js , Virtual DOM , Spring Boot , Java , JSON Web Tokens , Μηχανική μάθηση , Αλληλεπίδραση ανθρώπου-υπολογιστή , Ιδιωτικότητα δεδομένων , Ανάλυση απαιτήσεων , Διαγράμματα UML , Σχεσιακές βάσεις δεδομένων , Εξατομίκευση , Μοντελοποίηση χρηστών , Ηθική της AI , Μεγάλα γλωσσικά μοντέλα , Φωνητική αναζήτηση.

Abstract

This thesis presents the study, design and development of CINEHUB, an integrated and secure web platform for the management of cinema services. The system addresses the fragmentation of information that characterises the current landscape, unifying movie discovery, online ticket booking and social interaction through ratings and reviews within a single infrastructure.

Architecturally, the platform follows the Client-Server (N-Tier) model. The presentation layer (Frontend) was developed with the React.js library, delivering a responsive Single Page Application (SPA) experience. The business logic layer (Backend) is built on the Spring Boot (Java) framework, while data management is handled through Spring Data JPA over a relational database. System security is enforced by Spring Security and JSON Web Tokens (JWT), providing stateless authentication and Role-Based Access Control (RBAC).

The main functional characteristics include, for the customer: dynamic movie search (including voice search), interactive seat selection driven by heuristic algorithms, purchase of concession products and automatic issuing of a digital ticket (PDF with QR Code). For the administrator: bulk scheduling of screenings with automatic conflict detection, dynamic hall mapping and live monitoring of bookings.

Finally, the thesis relates the implementation to the international literature in the fields of Artificial Intelligence (AI) and Human-Computer Interaction (HCI). Future extensions are proposed, such as the integration of recommender systems, the development of native mobile applications and the connection to real payment gateways.

Keywords: Web platform, Online ticket booking, Social interaction, Full-Stack development, RESTful API architecture, Single Page Applications, Client-Server architecture, Data security, User experience, Interactive seat selection, Digital ticket, QR Code, Bulk scheduling, Dashboards, Role-Based Access Control, React.js, Virtual DOM, Spring Boot, Java, JSON Web Tokens, Machine learning, Human-computer interaction, Data privacy, Requirements analysis, UML diagrams, Relational databases, Personalisation, User modelling, AI ethics, Large language models, Voice search.

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

1. ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγικές Έννοιες και Αντικείμενο της Πτυχιακής Εργασίας

Στη σύγχρονη εποχή, η ραγδαία και διαρκής εξέλιξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών (ΤΠΕ) έχει αναδιαμορφώσει ριζικά τον τρόπο με τον οποίο οι άνθρωποι αλληλεπιδρούν με τα υπολογιστικά συστήματα, καταναλώνουν ψηφιακές υπηρεσίες και διεκπεραιώνουν τις καθημερινές τους συναλλαγές. Ο ψηφιακός μετασχηματισμός έχει αγγίξει σχεδόν κάθε πτυχή της ανθρώπινης δραστηριότητας, και η βιομηχανία της ψυχαγωγίας, ειδικότερα ο τομέας του κινηματογράφου, δεν αποτελεί εξαίρεση. Ιστορικά, η μετάβαση από τα παραδοσιακά, φυσικά εκδοτήρια εισιτηρίων (box offices) στα πρώτα ψηφιακά συστήματα ηλεκτρονικών κρατήσεων (e-ticketing) αποτέλεσε ένα σημαντικό τεχνολογικό άλμα, το οποίο εκμηδένισε γεωγραφικούς και χρονικούς περιορισμούς.

Ωστόσο, η σημερινή εποχή της πληροφορίας και των έξυπνων συστημάτων απαιτεί κάτι πολύ περισσότερο από την απλή ψηφιοποίηση μιας συναλλαγής. Η αφθονία της διαθέσιμης πληροφορίας καθιστά επιτακτική την ανάγκη για σύγχρονα, ολοκληρωμένα πληροφοριακά

συστήματα (Information Systems) τα οποία δεν λειτουργούν απλώς ως παθητικές βάσεις δεδομένων, αλλά υποστηρίζουν ενεργά τον χρήστη. Τα συστήματα αυτά καλούνται να παρέχουν εξατομικευμένη πληροφόρηση, να υποβοηθούν στη λήψη αποφάσεων μέσω έξυπνων μηχανισμών και να προσφέρουν διαδραστικές, φιλικές προς τον χρήστη διεπαφές (User Interfaces - UIs). Η ενσωμάτωση στοιχείων Τεχνητής Νοημοσύνης (Artificial Intelligence - AI) και μηχανισμών προσαρμογής στο λογισμικό έχει καταστεί κεντρικός πυλώνας της σύγχρονης Τεχνολογίας Λογισμικού (Software Engineering).

Η παρούσα πτυχιακή εργασία εστιάζει ακριβώς σε αυτό το σύγχρονο επιστημονικό πεδίο. Αντικείμενό της είναι η μελέτη, ο σχεδιασμός και η ανάπτυξη μιας ολοκληρωμένης, επεκτάσιμης και ασφαλούς διαδικτυακής πλατφόρμας με την ονομασία **CINEHUB**. Η πλατφόρμα αυτή φιλοδοξεί να γεφυρώσει το χάσμα μεταξύ της απλής παροχής πληροφοριών, της άμεσης ηλεκτρονικής συναλλαγής (κράτηση εισιτηρίων και προϊόντων) και της κοινωνικής αλληλεπίδρασης μέσω μηχανισμών αξιολόγησης (ratings και reviews).

Η σημασία της εκπόνησης μιας τέτοιας εργασίας στο πλαίσιο της επιστήμης της Πληροφορικής είναι πολυδιάστατη. Αρχικά, αποδεικνύει την ικανότητα σύνθεσης ετερογενών τεχνολογιών αιχμής (Full-Stack Development) για την επίλυση πραγματικών (real-world) προβλημάτων. Απαιτείται η βαθιά κατανόηση προηγμένων αρχιτεκτονικών λογισμικού, όπως είναι ο σχεδιασμός ασφαλών Διεπαφών Προγραμματισμού Εφαρμογών (RESTful APIs), η διαχείριση και κανονικοποίηση σχεσιακών βάσεων δεδομένων, η εφαρμογή πρωτοκόλλων αυθεντικοποίησης δίχως κατάσταση (stateless authentication) και η υλοποίηση αποκρινόμενων διεπαφών χρήστη (Responsive UI) με χρήση της φιλοσοφίας των Single Page Applications (SPAs). Παράλληλα, αναδεικνύει τις σύγχρονες ανάγκες για κατανεμημένα συστήματα (Client-Server architecture) που μπορούν να διαχειριστούν ταυτόχρονα αιτήματα (concurrency), εξασφαλίζοντας υψηλή διαθεσιμότητα, συνέπεια δεδομένων και ασφάλεια.

1.2 Η Ανάγκη για Σύγχρονα Ολοκληρωμένα Πληροφοριακά Συστήματα

Η ανάλυση του τρέχοντος τοπίου στη βιομηχανία των διαδικτυακών συστημάτων ψυχαγωγίας και κινηματογράφου αναδεικνύει σημαντικές ελλείψεις και φαινόμενα κατακερματισμού. Σήμερα, οι χρήστες έρχονται συχνά αντιμέτωποι με πολλαπλές, αποκομμένες μεταξύ τους πλατφόρμες προκειμένου να ολοκληρώσουν μια απλή εμπειρία εξόδου.

Πιο συγκεκριμένα, η τρέχουσα κατάσταση διαμορφώνεται ως εξής:

1. **Κατακερματισμός της Πληροφορίας:** Ένας χρήστης που επιθυμεί να παρακολουθήσει μια ταινία πρέπει συνήθως να επισκεφθεί εξειδικευμένες πλατφόρμες αξιολόγησης (π.χ.

IMDb, Letterboxd) για να διαβάσει κριτικές, στη συνέχεια να μεταβεί σε ιστοσελίδες αναπαραγωγής βίντεο για να δει το trailer, και τελικά να μεταβεί στο ανεξάρτητο σύστημα του εκάστοτε κινηματογράφου για να ολοκληρώσει την κράτηση. Αυτός ο διαχωρισμός αυξάνει το γνωστικό φορτίο του χρήστη και υποβαθμίζει τη συνολική εμπειρία αλληλεπίδρασης.

2. **Απουσία Ενιαίου Οικοσυστήματος Ανατροφοδότησης:** Μετά την ολοκλήρωση της προβολής, ελάχιστα συστήματα ηλεκτρονικών κρατήσεων ενθαρρύνουν τον χρήστη να επιστρέψει στην πλατφόρμα για να καταχωρήσει την άποψή του. Η απουσία αυτού του βρόχου ανατροφοδότησης (feedback loop) στερεί από τα συστήματα πολύτιμα δεδομένα που θα μπορούσαν να χρησιμοποιηθούν για την ανάπτυξη αλγορίθμων Μηχανικής Μάθησης και συστημάτων προσωποποιημένων συστάσεων (Recommender Systems) στο μέλλον.
3. **Διαχειριστικός Φόρτος (Administrative Overhead):** Από την πλευρά των διαχειριστών των κινηματογράφων (Cinema Managers), τα υπάρχοντα συστήματα back-office συχνά πάσχουν από έλλειψη αυτοματοποίησης. Η χειροκίνητη καταχώρηση πολλαπλών προβολών (επαναλαμβανόμενα γεγονότα) απαιτεί σημαντικό χρόνο και ενέχει τον κίνδυνο ανθρώπινου λάθους, όπως η επικάλυψη ωραρίων στην ίδια αίθουσα.
4. **Ασφάλεια και Ιδιωτικότητα:** Στην εποχή του GDPR (Γενικός Κανονισμός για την Προστασία Δεδομένων), η ασφαλής διαχείριση προσωπικών δεδομένων, στοιχείων πληρωμής και ψηφιακών εισιτηρίων απαιτεί συστήματα σχεδιασμένα με την αρχή "Security by Design".

Η ανάγκη λοιπόν δημιουργίας ενός ενοποιημένου συστήματος που θα καλύπτει όλες τις παραπάνω αδυναμίες, εξυπηρετώντας ταυτόχρονα τον πελάτη και τον διαχειριστή κάτω από μία κοινή και ασφαλή υποδομή, καθιστά την υλοποίηση του συστήματος CINEHUB απολύτως επίκαιρη και αναγκαία.

1.3 Σκοπός και Ερευνητικοί Στόχοι της Εργασίας

Ο κεντρικός σκοπός της πτυχιακής εργασίας είναι ο σχεδιασμός, η αρχιτεκτονική δόμηση και η προγραμματιστική υλοποίηση της πλατφόρμας CINEHUB, η οποία φιλοδοξεί να αποτελέσει ένα πρότυπο λογισμικό ενοποίησης υπηρεσιών ψυχαγωγίας. Το σύστημα σχεδιάστηκε για να λειτουργήσει ως ένας κεντρικός κόμβος, παρέχοντας υψηλής ποιότητας εμπειρία χρήστη (User Experience) και ισχυρά εργαλεία διαχείρισης.

Για την επιτυχή εκπλήρωση του σκοπού αυτού, η ανάπτυξη του συστήματος βασίστηκε σε σαφείς και μετρήσιμους ερευνητικούς-υλοποιητικούς στόχους, οι οποίοι κατηγοριοποιούνται με βάση τους ρόλους των χρηστών (Actors) της πλατφόρμας:

Άξονας 1ος: Στόχοι αναφορικά με τον Τελικό Χρήστη (Customer)

- Δημιουργία ενός εύχρηστου και δυναμικού περιβάλλοντος (Browse Environment) για την αναζήτηση και περιήγηση στον κατάλογο ταινιών, με υποστήριξη φιλτραρίσματος βάσει κατηγοριών και ενσωματωμένη αναπαραγωγή πολυμέσων (trailers).
- Σχεδιασμός ενός πλήρως διαδραστικού συστήματος επιλογής θέσεων (Interactive Seat Mapping), όπου ο χρήστης βλέπει τη διαθεσιμότητα σε πραγματικό χρόνο και μπορεί να επιλέξει τη θέση του ή να χρησιμοποιήσει έξυπνους αλγόριθμους για την αυτόματη πρόταση των καλύτερων κεντρικών θέσεων.
- Ολοκλήρωση της ροής κράτησης με την επιλογή συνοδευτικών προϊόντων κυλικείου (snacks) και την προσομοίωση ενός ασφαλούς περιβάλλοντος ελέγχου συναλλαγών (checkout).
- Αυτοματοποιημένη παραγωγή του ψηφιακού εισιτηρίου (E-Ticket) σε μορφή PDF, το οποίο φέρει μοναδικό κώδικα QR για την ασφαλή ταυτοποίηση του πελάτη κατά την προσέλευσή του στο φυσικό χώρο.
- Ανάπτυξη υποσυστήματος «Ψηφιακού Πορτοφολιού» (My Tickets) και διαχείρισης κριτικών, όπου ο χρήστης μπορεί να αξιολογεί τις ταινίες που έχει ήδη παρακολουθήσει.

Άξονας 2ος: Στόχοι αναφορικά με τον Διαχειριστή Κινηματογράφου (Cinema Manager)

- Ανάπτυξη εργαλείων δυναμικής χαρτογράφησης αιθουσών (Hall Configuration), επιτρέποντας τον καθορισμό διαστάσεων (σειρές και στήλες) για την αυτόματη παραγωγή του πλέγματος των θέσεων.
- Υλοποίηση έξυπνων μηχανισμών προγραμματισμού προβολών (Scheduling). Ειδικότερα, η ανάπτυξη αλγορίθμου Μαζικού Προγραμματισμού (Bulk/Recurring Scheduling) που επιτρέπει στον διαχειριστή να ορίζει προβολές για μεγάλα χρονικά διαστήματα με μία μόνο ενέργεια, εξοικονομώντας πόρους και αποτρέποντας επικαλύψεις ωραρίων.
- Δημιουργία πινάκων ελέγχου (Dashboards) για την οπτικοποίηση των εσόδων και τη ζωντανή παρακολούθηση (Live View) των κρατήσεων ανά προβολή.

Άξονας 3ος: Στόχοι αναφορικά με τον Διαχειριστή Συστήματος (System Admin)

- Υλοποίηση Κέντρου Ελέγχου (Command Center) για την ολική εποπτεία της υγείας της πλατφόρμας.
- Ενσωμάτωση εργαλείων διαχείρισης χρηστών (User Management) και ελέγχου προσβάσεων βάσει ρόλων (Role-Based Access Control).
- Δημιουργία μηχανισμών ελέγχου και εποπτείας περιεχομένου (Content Moderation), ώστε οι διαχειριστές να μπορούν να φιλτράρουν ή να αφαιρούν κακόβουλες κριτικές.

1.4 Σύντομη Περιγραφή Μεθοδολογίας και Αρχιτεκτονικής

Η σύλληψη και η υλοποίηση του συστήματος CINEHUB ακολούθησε τις βέλτιστες πρακτικές της σύγχρονης Τεχνολογίας Λογισμικού. Επιλέχθηκε η αρχιτεκτονική Πελάτη-Εξυπηρετητή (Client-Server Architecture) για τον αυστηρό διαχωρισμό του επιπέδου παρουσίασης από το επίπεδο επιχειρησιακής λογικής και διαχείρισης δεδομένων. Αυτός ο διαχωρισμός (Separation of Concerns) διασφαλίζει την επεκτασιμότητα (scalability) και τη συντηρησιμότητα (maintainability) του κώδικα.

- **Επίπεδο Παρουσίασης (Frontend):** Η οπτική διεπαφή του συστήματος αναπτύχθηκε εξ ολοκλήρου με τη βιβλιοθήκη JavaScript **React.js**. Η επιλογή της React έγινε λόγω της ιδιότητάς της να διαχειρίζεται αποδοτικά το Document Object Model (DOM) μέσω του Virtual DOM, επιτρέποντας τη δημιουργία Single Page Applications (SPAs). Τα SPAs προσφέρουν μια εμπειρία χρήσης που προσομοιάζει αυτή των εγγενών (native) εφαρμογών, καθώς η σελίδα δεν ανανεώνεται ολόκληρη σε κάθε κλικ, αλλά τα δεδομένα φορτώνονται ασύγχρονα.
- **Επίπεδο Επιχειρησιακής Λογικής (Backend):** Το υποσύστημα του εξυπηρετητή (server-side) αναπτύχθηκε με το πλαίσιο εργασίας **Spring Boot**, βασισμένο στη γλώσσα προγραμματισμού Java. Το Spring Boot αποτελεί το βιομηχανικό πρότυπο για την ανάπτυξη εταιρικών (enterprise) εφαρμογών, προσφέροντας απαράμιλλη αξιοπιστία, ευκολία στη δημιουργία RESTful Web Services και αυτόματη διαχείριση εξαρτήσεων (Dependency Injection). Το επίπεδο δεδομένων διαχειρίζεται μέσω του Spring Data JPA, το οποίο επικοινωνεί με τη σχεσιακή βάση δεδομένων.
- **Ασφάλεια (Security):** Η επικοινωνία μεταξύ Frontend και Backend θωρακίζεται μέσω του **Spring Security** και της χρήσης **JSON Web Tokens (JWT)**. Ο μηχανισμός αυτός επιτρέπει την αυθεντικοποίηση (authentication) και την εξουσιοδότηση (authorization) των χρηστών χωρίς την ανάγκη διατήρησης συνεδρίας στον εξυπηρετητή (stateless), αυξάνοντας δραματικά την ασφάλεια έναντι κακόβουλων επιθέσεων.

1.5 Διάρθρωση της Πτυχιακής Εργασίας

Για τη βέλτιστη παρουσίαση του θεωρητικού υπόβαθρου, της ανάλυσης απαιτήσεων και της τεχνικής τεκμηρίωσης, η παρούσα πτυχιακή εργασία διαρθρώνεται σε επτά (7) διακριτά κεφάλαια. Πιο συγκεκριμένα:

- Στο **Κεφάλαιο 1**, το οποίο αποτελεί την τρέχουσα ενότητα, πραγματοποιείται μια εισαγωγή στο αντικείμενο, παρουσιάζεται η ανάγκη για σύγχρονα συστήματα στο χώρο του θεάματος, και καθορίζονται σαφώς οι σκοποί, οι στόχοι και η γενική αρχιτεκτονική προσέγγιση του έργου.
- Στο **Κεφάλαιο 2 (Ανασκόπηση Πεδίου)**, γίνεται μια ενδελεχής μελέτη της υπάρχουσας βιβλιογραφίας. Αναλύεται η συμβολή της Τεχνητής Νοημοσύνης (AI) στη βελτίωση της Αλληλεπίδρασης Ανθρώπου-Υπολογιστή (HCI), η σημασία της μηχανικής μάθησης στα σύγχρονα πληροφοριακά συστήματα, και οι ηθικές προκλήσεις σχετικά με την ιδιωτικότητα. Παρουσιάζεται το τρέχον τοπίο και οι ελλείψεις που η εργασία έρχεται να καλύψει.
- Στο **Κεφάλαιο 3 (Ανάλυση και Σχεδιασμός)**, περιγράφεται αναλυτικά ο κύκλος ζωής της ανάπτυξης του λογισμικού. Καταγράφονται οι Λειτουργικές και Μη Λειτουργικές απαιτήσεις, ενώ παρατίθενται τα μοντέλα της Ενοποιημένης Γλώσσας Μοντελοποίησης (UML) για τις περιπτώσεις χρήσης και τον σχεδιασμό των κλάσεων και της βάσης δεδομένων.
- Στο **Κεφάλαιο 4 (Υλοποίηση Εφαρμογής)**, αναλύεται σε βάθος η θεωρία πίσω από τα εργαλεία ανάπτυξης. Εξηγείται η φιλοσοφία του Spring Boot, οι μηχανισμοί ασφαλείας του JWT, ο τρόπος λειτουργίας του Virtual DOM της React, και τα προβλήματα που προέκυψαν κατά την ανάπτυξη και πώς αυτά επιλύθηκαν.
- Στο **Κεφάλαιο 5 (Λειτουργικότητα της Εφαρμογής)**, η εστίαση στρέφεται στην οπτική του προγραμματιστή (Developer Perspective). Αναλύεται με λεπτομέρεια η μαθηματική και λογική ροή των αλγορίθμων που αναπτύχθηκαν, όπως η έξυπνη επιλογή θέσεων (smart seat selection) και ο μηχανισμός μαζικού προγραμματισμού (bulk scheduling).
- Στο **Κεφάλαιο 6 (Εγχειρίδιο Χρήστη)**, παρουσιάζεται η λειτουργικότητα της πλατφόρμας από την οπτική του τελικού χρήστη. Με τη βοήθεια στιγμιότυπων οθόνης (screenshots), καθοδηγείται ο αναγνώστης στις δυνατότητες του συστήματος, τόσο από την πλευρά του πελάτη όσο και από την πλευρά του διαχειριστή.

- Στο **Κεφάλαιο 7 (Συμπεράσματα και Μελλοντικές Επεκτάσεις)**, συνοψίζονται τα αποτελέσματα και η επιτυχία των στόχων του συστήματος CINEHUB. Τέλος, προτείνονται ρεαλιστικές κατευθύνσεις για τη μελλοντική επέκταση της πλατφόρμας, όπως η ενσωμάτωση αλγορίθμων Collaborative Filtering για προσωποποιημένες συστάσεις και η δημιουργία Native Mobile εφαρμογής.

2. ΑΝΑΣΚΟΠΗΣΗ ΠΕΔΙΟΥ

2.1 Επιστημονικές Εργασίες σε Παρόμοιο Πεδίο Εφαρμογής

Η ανάπτυξη σύγχρονων πληροφοριακών συστημάτων, όπως η πλατφόρμα CINEHUB, δεν αποτελεί μια μεμονωμένη τεχνολογική προσπάθεια, αλλά βασίζεται σε ένα ευρύ και διαρκώς εξελισσόμενο φάσμα επιστημονικών ερευνών. Οι έρευνες αυτές συνδυάζουν τις αρχές της Αλληλεπίδρασης Ανθρώπου-Υπολογιστή (HCI) με τις προηγμένες δυνατότητες της Τεχνητής Νοημοσύνης (AI), την αυτοματοποίηση και την εξατομίκευση [1, 3-7, 9].

2.1.1 Αλληλεπίδραση Ανθρώπου-Υπολογιστή (HCI) και Σύγχρονες Διεπαφές

Η εξέλιξη της αλληλεπίδρασης ανθρώπου-υπολογιστή (HCI) διαδραματίζει πλέον καίριο ρόλο στην κατασκευή σύγχρονων διεπαφών χρήστη (User Interfaces), προσφέροντας τη δυνατότητα δυναμικής προσαρμογής και βελτίωσης της Εμπειρίας Χρήστη (UX) [14]. Στο πλαίσιο του CINEHUB, οι θεωρητικές αρχές της HCI εφαρμόζονται πρακτικά μέσω της χρήσης τεχνολογιών αιχμής στο Frontend (React.js), μειώνοντας δραματικά το γνωστικό φορτίο (cognitive load) του χρήστη χάρη στην αρχιτεκτονική Single Page Application (SPA) [32, 33, 34]. Επιπλέον, η προσβασιμότητα (accessibility) και η φιλικότητα του συστήματος ενισχύονται σημαντικά μέσω της ενσωμάτωσης λειτουργιών Φωνητικής Αναζήτησης (Voice Search) με χρήση του Web Speech API, καταργώντας τα παραδοσιακά εμπόδια πληκτρολόγησης [17, 18, 20].

2.1.2 Ευρετικοί Αλγόριθμοι, Μηχανική Μάθηση και Συστήματα Συστάσεων

Η εξέλιξη της αλληλεπίδρασης ανθρώπου-υπολογιστή οδηγεί σε ριζικά βελτιωμένες εμπειρίες. Η τεχνολογία λογισμικού ενισχυμένη με AI (AI-empowered software engineering) αντιπροσωπεύει μια μεγάλη αλλαγή στις μεθοδολογίες ανάπτυξης, τονίζοντας την ένταξη μηχανισμών υποστήριξης. [2, 10]

Αν και τα πλήρη συστήματα συστάσεων (Recommender Systems) βασίζονται συχνά σε πολύπλοκα μοντέλα Συνεργατικού Φιλτραρίσματος (Collaborative Filtering), το CINEHUB θέτει γερά τα θεμέλια για τέτοιες μελλοντικές επεκτάσεις μέσω της συστηματικής καταγραφής αξιολογήσεων (ratings). Πέραν τούτου, το σύστημα ήδη ενσωματώνει Ευρετικούς Αλγορίθμους Υποστήριξης Λήψης Αποφάσεων (Decision Support Heuristics). Χαρακτηριστικό παράδειγμα αποτελεί ο αλγόριθμος "Smart Seat Selection" (Έξυπνη Επιλογή Θέσεων), ο οποίος αναλύει γεωμετρικά και στατιστικά το πλάνο της αίθουσας για να προτείνει αυτόματα τις βέλτιστες κεντρικές θέσεις στον πελάτη. Αυτή η αυτοματοποιημένη συλλογιστική συμβάλλει στη διευκόλυνση του χρήστη, προσομοιάζοντας τη λογική καθοδήγησης ενός ανθρώπινου εξυπηρετητή [11, 28, 29].

2.1.3 Αξιολόγηση Χρηστών και Βρόχοι Ανατροφοδότησης (Feedback Loops)

Η κατανόηση της ανταπόκρισης και της ικανοποίησης του χρήστη αποτελεί θεμελιώδη στόχο των σύγχρονων διαδραστικών συστημάτων [31, 42]. Η συστηματική μοντελοποίηση των χρηστών μέσω της ανάλυσης των ενεργειών και των προτιμήσεών τους επιτρέπει τη δημιουργία λογισμικού που προσαρμόζεται στις ανάγκες του πελάτη [19, 21, 22, 25, 26, 27, 30, 35, 36]. Στην πλατφόρμα CINEHUB, η ανάγκη αυτή καλύπτεται μέσω ενός άμεσου βρόχου ανατροφοδότησης (feedback loop) που ενεργοποιείται μετά την ολοκλήρωση της προβολής. Μέσα από το ψηφιακό πορτοφόλι (My Tickets), ο χρήστης καλείται να παράσχει ποσοτική βαθμολογία και ποιοτική γραπτή κριτική [12, 13]. Τα δεδομένα αυτά τροφοδοτούν το Κέντρο Ελέγχου (Command Center) του διαχειριστή, επιτρέποντας την εποπτεία της «υγείας» της πλατφόρμας και τον ποιοτικό έλεγχο (moderation) σε πραγματικό χρόνο. Με τον τρόπο αυτό, η εφαρμογή μετατρέπεται από ένα στατικό σύστημα συναλλαγών σε μια δυναμική κοινότητα, όπου η γνώμη του χρήστη επηρεάζει άμεσα τη δημοτικότητα και την προβολή των κινηματογραφικών υπηρεσιών.

2.1.4 Ιδιωτικότητα, Ασφάλεια Δεδομένων και GDPR

Στη σύγχρονη εποχή των κατανεμημένων συστημάτων, οι εφαρμογές συλλέγουν και επεξεργάζονται σημαντικό όγκο προσωπικών δεδομένων. Η ηθική χρήση αυτών των τεχνολογιών και η αυστηρή συμμόρφωση με κανονισμούς προστασίας δεδομένων (όπως ο GDPR) αποτελούν απαραίτητη προϋπόθεση για την προστασία των ευαίσθητων πληροφοριών των χρηστών [40, 41]. Στην αρχιτεκτονική υλοποίηση του CINEHUB, η ιδιωτικότητα διασφαλίζεται δομικά με την αρχή "Security by Design". Αντί να βασίζεται σε παραδοσιακές συνεδρίες (sessions) που εγκυμονούν κινδύνους υποκλοπής, το σύστημα υιοθετεί την τεχνολογία JSON Web Tokens (JWT) για stateless αυθεντικοποίηση. Παράλληλα, εφαρμόζεται αυστηρός Έλεγχος Πρόσβασης Βάσει Ρόλων (Role-Based Access Control - RBAC), εξασφαλίζοντας ότι κρίσιμα δεδομένα είναι απολύτως θωρακισμένα και προσβάσιμα μόνο από τις εξουσιοδοτημένες οντότητες του συστήματος.

2.2 Παρούσα Κατάσταση

Στη σημερινή αγορά, τα συστήματα διαχείρισης κινηματογράφων και ηλεκτρονικών κρατήσεων εστιάζουν σχεδόν αποκλειστικά στη διεκπεραίωση της οικονομικής συναλλαγής [8, 38]. Οι περισσότερες εφαρμογές προσφέρουν μια απλή προβολή του καταλόγου ταινιών, βασική επιλογή θέσεων σε ψηφιακό πλάνο και την έκδοση ηλεκτρονικού εισιτηρίου (e-ticket). Παρόλο που υπάρχουν παγκόσμιες πλατφόρμες συλλογής πληροφοριών και κριτικών, όπως το IMDb ή το Letterboxd, αυτές παραμένουν συχνά αποκομμένες από την άμεση διαδικασία κράτησης και την τοπική διαχείριση του εκάστοτε κινηματογράφου. Από την πλευρά των διαχειριστών (back-office), τα εργαλεία συχνά απαιτούν επαναλαμβανόμενη, χειροκίνητη καταχώρηση δεδομένων, αυξάνοντας δραματικά το διαχειριστικό κόστος και την πιθανότητα ανθρώπινου λάθους.

2.3 Υπάρχουσες Ελλείψεις

Με βάση την ανάλυση του επιστημονικού πεδίου και της αγοράς, εντοπίζονται οι εξής σημαντικές ελλείψεις στα τρέχοντα πληροφοριακά συστήματα:

- **Κατακερματισμός Πληροφορίας και Κράτησης:** Ο χρήστης αναγκάζεται να μεταβαίνει σε πολλαπλές εφαρμογές για να δει trailers, να διαβάσει αξιολογήσεις και, τελικά, να κλείσει το εισιτήριό του.
- **Απουσία Ενιαίου Οικοσυστήματος Αξιολόγησης:** Η διαδικασία της κριτικής σπάνια συνδέεται άμεσα με το επαληθευμένο ιστορικό κρατήσεων του χρήστη, μειώνοντας την αξιοπιστία των δεδομένων για μελλοντικά συστήματα συστάσεων.
- **Διαχειριστικός Φόρτος (Administrative Overhead):** Απουσία έξυπνων μηχανισμών αυτοματοποίησης προγραμματισμού (όπως το Bulk Scheduling), οι οποίοι θα προστατεύουν το σύστημα από επικαλύψεις ωραρίων και θα εξοικονομούν πόρους.
- **Παθητικές Διεπαφές Χρήστη (Passive UIs):** Έλλειψη δυναμικών, βοηθητικών λειτουργιών, όπως η φωνητική αναζήτηση ή η έξυπνη πρόταση θέσεων, που διευκολύνουν την πλοήγηση και βελτιώνουν δραστικά την εμπειρία.

2.4 Στόχος της Πτυχιακής Εργασίας

Η παρούσα πτυχιακή εργασία στοχεύει στην κάλυψη των παραπάνω ελλείψεων μέσω της ολοκληρωμένης πλατφόρμας CINEHUB. Οι βασικοί στόχοι περιλαμβάνουν την πλήρη ενοποίηση της πληροφορίας (trailers, reviews) με την άμεση αγορά εισιτηρίου μέσα σε ένα ταχύτατο περιβάλλον Single Page Application. Επιπρόσθετα, αποσκοπεί στην υλοποίηση ευφυών μηχανισμών υποβοήθησης, όπως η φωνητική αναζήτηση και ο αλγόριθμος "Smart Seat Selection" για τους πελάτες. Ταυτόχρονα, παρέχει καινοτόμα εργαλεία για τους διαχειριστές, όπως το "Command Center" και δυνατότητες Μαζικού Προγραμματισμού (Bulk Scheduling) με ζωντανή εποπτεία, γεφυρώνοντας επιτυχώς το χάσμα μεταξύ της θεωρίας της Τεχνητής Νοημοσύνης και της εφαρμοσμένης μηχανικής λογισμικού.

3. ΑΝΑΛΥΣΗ ΚΑΙ ΣΧΕΔΙΑΣΜΟΣ ΣΥΣΤΗΜΑΤΟΣ

3.1 Εισαγωγή στον Κύκλο Ζωής Ανάπτυξης Λογισμικού και Μεθοδολογία Ανάπτυξης

Η επιτυχής υλοποίηση ενός σύγχρονου και πολύπλοκου πληροφοριακού συστήματος, όπως η πλατφόρμα CINEHUB, προϋποθέτει την αυστηρή τήρηση των αρχών της Τεχνολογίας Λογισμικού (Software Engineering). Η φάση της ανάλυσης και του σχεδιασμού αποτελεί το κρισιμότερο στάδιο στον Κύκλο Ζωής Ανάπτυξης Λογισμικού (Software Development Life Cycle - SDLC), καθώς καθορίζει τα θεμέλια πάνω στα οποία θα οικοδομηθεί η εφαρμογή. Σκοπός αυτού του σταδίου είναι η πλήρης κατανόηση του προβλήματος, η συλλογή και καταγραφή των απαιτήσεων των χρηστών, και η μετατροπή αυτών σε συγκεκριμένες τεχνικές προδιαγραφές και αρχιτεκτονικά μοντέλα.

Για την ανάπτυξη του συστήματος CINEHUB υιοθετήθηκε μια Ευέλικτη (Agile) μεθοδολογία με Επαναληπτικό (Iterative) χαρακτήρα. Σε αντίθεση με τα παραδοσιακά γραμμικά μοντέλα (όπως το μοντέλο του Καταρράκτη), η Agile προσέγγιση επέτρεψε τη σταδιακή υλοποίηση των λειτουργικοτήτων σε διακριτούς κύκλους. Η επιλογή αυτή κρίθηκε αναγκαία για τους εξής λόγους:

- **Σταδιακή Επαλήθευση απαιτήσεων:** Επέτρεψε την άμεση οπτικοποίηση κρίσιμων υποσυστημάτων, όπως το διαδραστικό πλάνο θέσεων (Seat Map), και την προσαρμογή τους πριν την ολοκλήρωση του Backend.
- **Διαχείριση Πολυπλοκότητας:** Η εφαρμογή διαχωρίστηκε σε αυτόνομες ενότητες (Αυθεντικοποίηση , Διαχείριση Κινηματογράφου , Κρατήσεις), οι οποίες αναπτύχθηκαν και ελέγχθηκαν ανεξάρτητα.
- **Μείωση Ρίσκου:** Η διαδικασία αυτή ελαχιστοποιεί τον κίνδυνο δομικών αστοχιών κατά τη φάση της συγγραφής του κώδικα και διασφαλίζει ότι το τελικό προϊόν θα ικανοποιεί πλήρως τις ανάγκες του τελικού χρήστη και της επιχείρησης.

3.2 Ανάλυση Απαιτήσεων Συστήματος (Requirements Engineering)

Η μηχανική απαιτήσεων αποτελεί τη συστηματική διαδικασία καταγραφής, τεκμηρίωσης και διαχείρισης των αναγκών που πρέπει να καλύπτει το λογισμικό [43]. Οι απαιτήσεις για την πλατφόρμα CINEHUB διαχωρίζονται σε δύο μεγάλες κατηγορίες: τις Λειτουργικές Απαιτήσεις

(Functional Requirements), οι οποίες περιγράφουν τις ακριβείς ενέργειες και συμπεριφορές του συστήματος, και τις Μη Λειτουργικές Απαιτήσεις (Non-Functional Requirements), οι οποίες ορίζουν τα ποιοτικά χαρακτηριστικά, όπως η ασφάλεια, η απόδοση και η επεκτασιμότητα.

3.2.1 Λειτουργικές Απαιτήσεις ανά Ρόλο Χρήστη

Για την εξασφάλιση της μέγιστης ασφάλειας και οργάνωσης, το σύστημα σχεδιάστηκε με βάση τον Έλεγχο Πρόσβασης βάσει Ρόλων (Role-Based Access Control - RBAC). Οι λειτουργικές απαιτήσεις δομούνται γύρω από τέσσερις διακριτούς ρόλους (Actors):

1. Απλός Χρήστης / Πελάτης (Role: Customer)

- **Ταυτοποίηση:** Ασφαλής εγγραφή και σύνδεση στο σύστημα.
- **Αναζήτηση & Περιήγηση:** Αναζήτηση ταινιών μέσω φίλτρων (είδος, τίτλος) και υποστήριξη φωνητικής αναζήτησης (voice search) για βελτιωμένη προσβασιμότητα. Προβολή αναλυτικών μεταδεδομένων της ταινίας και αναπαραγωγή διαφημιστικών βίντεο (trailers).
- **Κράτηση & Επιλογή Θέσεων:** Προβολή διαθέσιμων προβολών (screenings) ανά ημερομηνία και κινηματογράφο. Διαδραστική επιλογή θέσεων πάνω σε ένα δυναμικά παραγόμενο πλάνο αίθουσας (seat grid).
- **Αγορά Συνοδευτικών & Πληρωμή:** Δυνατότητα προσθήκης προϊόντων κυλικείου (snacks) στο καλάθι και προσομοίωση ελέγχου στοιχείων πιστωτικής κάρτας (checkout) για την ολοκλήρωση της συναλλαγής.
- **Πορτοφόλι & Αξιολογήσεις:** Προβολή ψηφιακών εισιτηρίων (με παραγωγή QR Code και εξαγωγή σε PDF). Υποβολή βαθμολογίας και γραπτής κριτικής (review) για τις ταινίες του παρελθόντος.

2. Διαχειριστής Κινηματογράφου (Role: Cinema Manager)

- **Διαχείριση Υποδομών:** Δημιουργία και επεξεργασία του προφίλ του κινηματογράφου. Καταχώρηση αιθουσών (Halls) με ακριβή ορισμό των διαστάσεών τους (αριθμός σειρών και στηλών).
- **Προγραμματισμός Προβολών:** Ανάθεση ταινιών σε αίθουσες με καθορισμό τιμής και ώρας. Δυνατότητα μαζικού προγραμματισμού (Bulk Scheduling) για επαναλαμβανόμενες προβολές εντός ενός χρονικού εύρους, με αυτόματο έλεγχο επικαλύψεων.

- **Εποπτεία & Στατιστικά:** Παρακολούθηση της κατάστασης των κρατήσεων σε πραγματικό χρόνο (Live View Inspection) ανά προβολή και οπτικοποίηση των εσόδων (Revenue) της τελευταίας εβδομάδας μέσω διαγραμμάτων.

3. Διαχειριστής Ταινιών (Role: Movie Manager)

- **Διαχείριση Περιεχομένου:** Προσθήκη νέων κινηματογραφικών παραγωγών στο σύστημα, εισάγοντας μεταδεδομένα όπως τίτλο, σύνοψη, διάρκεια, URL αφίσας (poster), URL trailer και βασική βαθμολογία (IMDb rating).
- **Προεπισκόπηση:** Ζωντανή προεπισκόπηση (live preview) της κάρτας της ταινίας όπως αυτή θα εμφανίζεται στους τελικούς χρήστες, πριν την οριστική δημοσίευσή της.

4. Κεντρικός Διαχειριστής Συστήματος (Role: System Admin)

- **Διαχείριση Χρηστών:** Προβολή όλων των εγγεγραμμένων μελών, με δυνατότητα αλλαγής των δικαιωμάτων τους (εκχώρηση νέων ρόλων) ή οριστικής διαγραφής τους από το σύστημα.
- **Κέντρο Ελέγχου (Command Center):** Εποπτεία των συνολικών μετρικών της πλατφόρμας (συνολικές κριτικές, μέσος όρος βαθμολογιών) και μεσολάβηση (moderation) περιεχομένου με δυνατότητα αφαίρεσης κακόβουλων ή προσβλητικών κριτικών.

3.2.2 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις διασφαλίζουν ότι το σύστημα δεν εκτελεί απλώς τις λειτουργίες του, αλλά τις εκτελεί σωστά, γρήγορα και με ασφάλεια.

- **Ασφάλεια και Ιδιωτικότητα (Security & Privacy):** Το σύστημα πρέπει να διασφαλίζει την κρυπτογράφηση των κωδικών πρόσβασης στη βάση δεδομένων (π.χ. με χρήση αλγορίθμων hashing). Η αυθεντικοποίηση και η εξουσιοδότηση πρέπει να υλοποιούνται μέσω ασφαλών πρωτοκόλλων, όπως τα JSON Web Tokens (JWT), αποτρέποντας την υποκλοπή συνεδριών.
- **Αποκριτικότητα Διεπαφής (Responsiveness & UX):** Το περιβάλλον χρήστη (UI) πρέπει να προσαρμόζεται δυναμικά (responsive design) σε διάφορες αναλύσεις οθονών, από επιτραπέζιους υπολογιστές έως κινητές συσκευές, προσφέροντας ομαλή πλοήγηση χωρίς καθυστερήσεις στην επανασχεδίαση του DOM.
- **Επεκτασιμότητα (Scalability):** Η αρχιτεκτονική του συστήματος (backend) πρέπει να υποστηρίζει την ταυτόχρονη εξυπηρέτηση χιλιάδων χρηστών, χρησιμοποιώντας

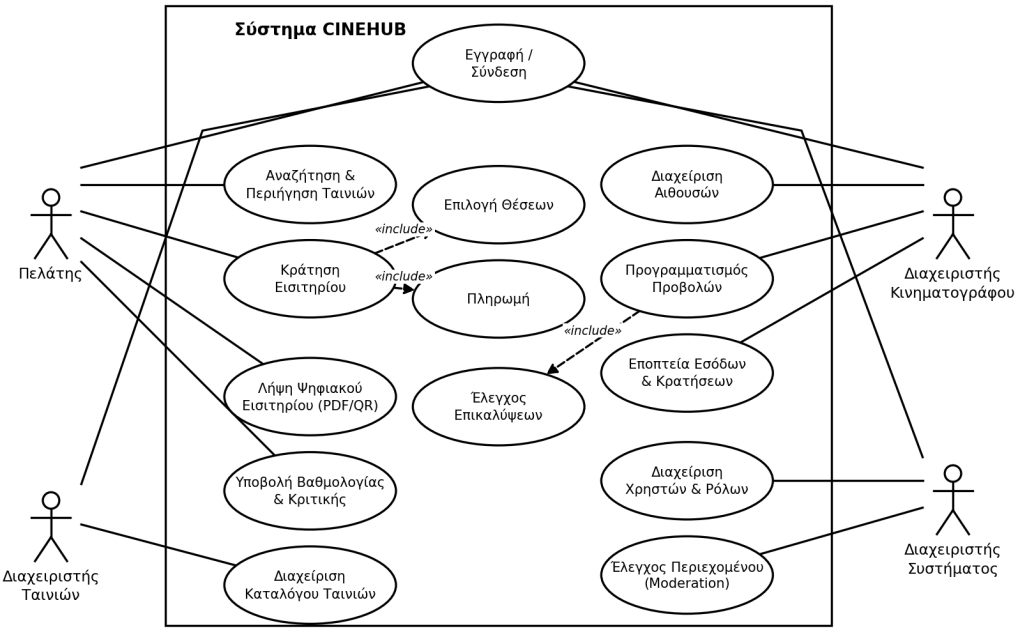
αποδοτικές δομές δεδομένων και αποφεύγοντας τα σημεία συμφόρησης (bottlenecks) κατά την ταυτόχρονη κράτηση της ίδιας θέσης (concurrency handling).

3.3 Σχεδιασμός με Διαγράμματα UML

Η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language - UML) αποτελεί το βιομηχανικό πρότυπο για την οπτικοποίηση, τον προσδιορισμό και την κατασκευή των τεχνημάτων ενός συστήματος λογισμικού. Στο πλαίσιο του CINEHUB, χρησιμοποιήθηκαν διαγράμματα UML για την αφαίρεση (abstraction) της πολυπλοκότητας και την ευκολότερη κατανόηση των αλληλεπιδράσεων.

3.3.1 Μοντελοποίηση Περιπτώσεων Χρήσης (Use Case Model)

Το Διάγραμμα Περιπτώσεων Χρήσης αποτυπώνει τις εξωτερικές αλληλεπιδράσεις του συστήματος. Κάθε «Περίπτωση Χρήσης» (Use Case) αντιπροσωπεύει μια ολοκληρωμένη ροή γεγονότων.



Εικόνα 3.1: Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagram) της πλατφόρμας CINEHUB.

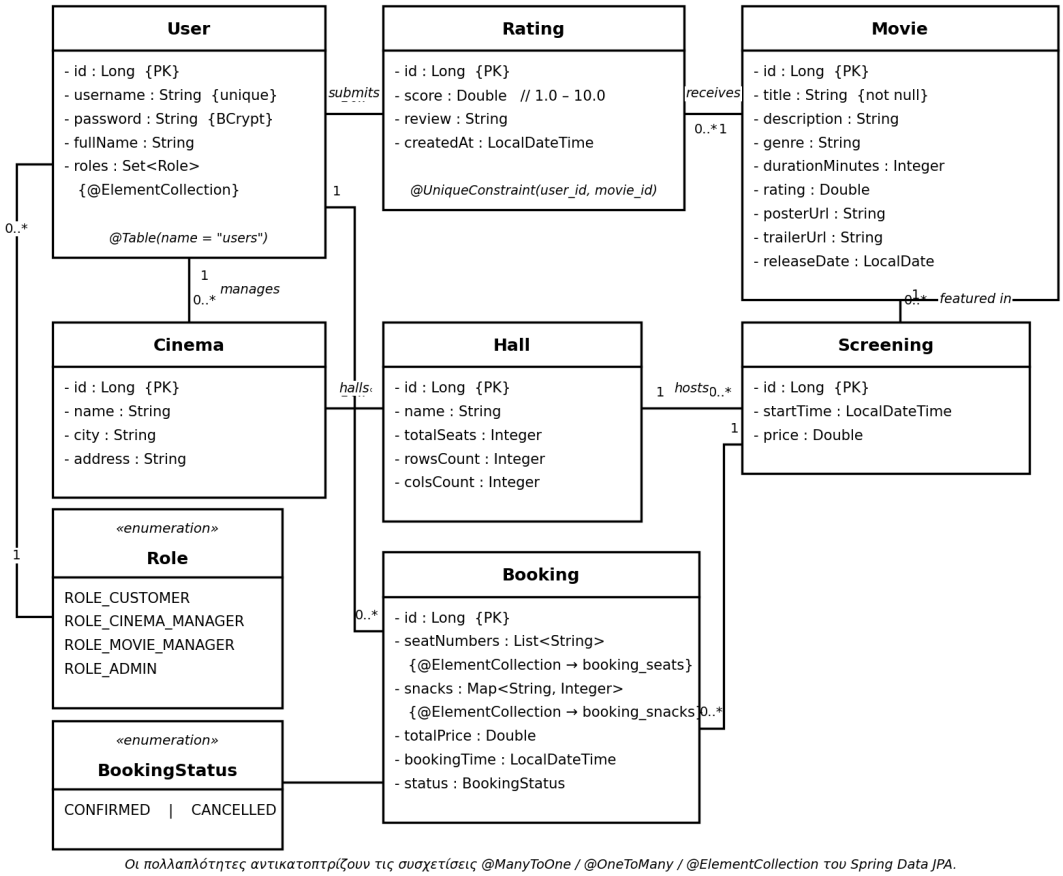
Όπως διακρίνεται και στο παραπάνω διάγραμμα:

- Ο **Πελάτης** (Primary Actor) αλληλεπιδρά με το σύστημα εκτελώντας περιπτώσεις όπως: "Αναζήτηση Ταινίας", "Κράτηση Εισιτηρίου" (η οποία περιλαμβάνει ως υπο-περίπτωση την "Επιλογή Θέσης" και την "Πληρωμή") και "Αξιολόγηση Ταινίας".

- Ο **Διαχειριστής Κινηματογράφου** εκτελεί περιπτώσεις όπως: "Δημιουργία Αίθουσας", "Προγραμματισμός Προβολής" και "Επισκόπηση Εσόδων".
- Ο **Διαχειριστής Συστήματος (Admin)** έχει την ευθύνη για την περίπτωση "Διαχείριση Χρηστών" και "Έλεγχος Περιεχομένου (Moderation)".

3.3.2 Σχεδιασμός Σχεσιακής Βάσης Δεδομένων και Μοντέλο Κλάσεων

Η ορθή αποθήκευση και συσχέτιση των δεδομένων είναι ο πυρήνας κάθε κατανεμημένου συστήματος. Ο σχεδιασμός της βάσης δεδομένων του CINEHUB ακολουθεί το μοντέλο Οντοτήτων-Συσχετίσεων (Entity-Relationship Model), το οποίο αντικατοπτρίζεται πλήρως στον αντικειμενοστρεφή σχεδιασμό των κλάσεων του Backend. Η ολοκληρωμένη δομή των οντοτήτων, τα επιμέρους πεδία τους καθώς και οι μεταξύ τους πρωτεύουσες και ξένες συσχετίσεις αποτυπώνονται αναλυτικά στο παρακάτω διάγραμμα (Εικόνα 3.2):



Εικόνα 3.2: ***Αναλυτικό Διάγραμμα Κλάσεων UML (Entities) και Συσχετίσεων της πλατφόρμας CINEHUB.***

Οι κύριες οντότητες (Entities) και οι συσχετίσεις τους, όπως αυτές απεικονίζονται σχηματικά παραπάνω, αναλύονται λεπτομερώς ως εξής:

1. **Οντότητα User:** Αποτελεί τον κεντρικό άξονα πρόσβασης. Περιλαμβάνει πεδία όπως Username, Password (κρυπτογραφημένο), Full Name και μια συλλογή (Set) από Ρόλους (Roles).
2. **Οντότητα Movie:** Περιέχει τα στατικά μεταδεδομένα του κινηματογραφικού έργου (Title, Description, Genre, Rating, Duration, PosterUrl, TrailerUrl). Αποτελεί ανεξάρτητη οντότητα η οποία συσχετίζεται εμμέσως με τις προβολές.
3. **Οντότητες Cinema & Hall:** Η οντότητα Cinema (όνομα, πόλη, διεύθυνση) ανήκει σε έναν χρήστη με ρόλο Cinema Manager. Ένας κινηματογράφος διατηρεί σχέση 'ένα-προς-πολλά' (One-to-Many) με τις αίθουσες (Halls). Κάθε αίθουσα (Hall) ορίζεται από το όνομά της, τον αριθμό σειρών (RowsCount) και στηλών (ColsCount), παράγοντας τη συνολική χωρητικότητα.
4. **Οντότητα Screening:** Αποτελεί την οντότητα-γέφυρα μεταξύ του χρόνου, του χώρου και του περιεχομένου. Ένα Screening συνδέει (Many-to-One) μια ταινία (Movie) με μια συγκεκριμένη αίθουσα (Hall), σε μια αυστηρά καθορισμένη ημερομηνία και ώρα (StartTime), ορίζοντας παράλληλα την τιμή του εισιτηρίου (Price).
5. **Οντότητα Booking:** Αναπαριστά την τελική συναλλαγή. Συσχετίζεται (Many-to-One) με ένα συγκεκριμένο Screening. Αποθηκεύει τον χρήστη που έκανε την κράτηση, μια λίστα από δεσμευμένες θέσεις (SeatNumbers, π.χ. ["A1", "A2"]), τον αριθμό των snacks και το συνολικό κόστος της κράτησης. Η οντότητα αυτή είναι κρίσιμη για τον έλεγχο των διαθέσιμων θέσεων.
6. **Οντότητα Rating:** Συνδέει έναν Χρήστη με μια Ταινία, αποθηκεύοντας τη βαθμολογία (Score από 1-10) και το κείμενο της κριτικής (Review).

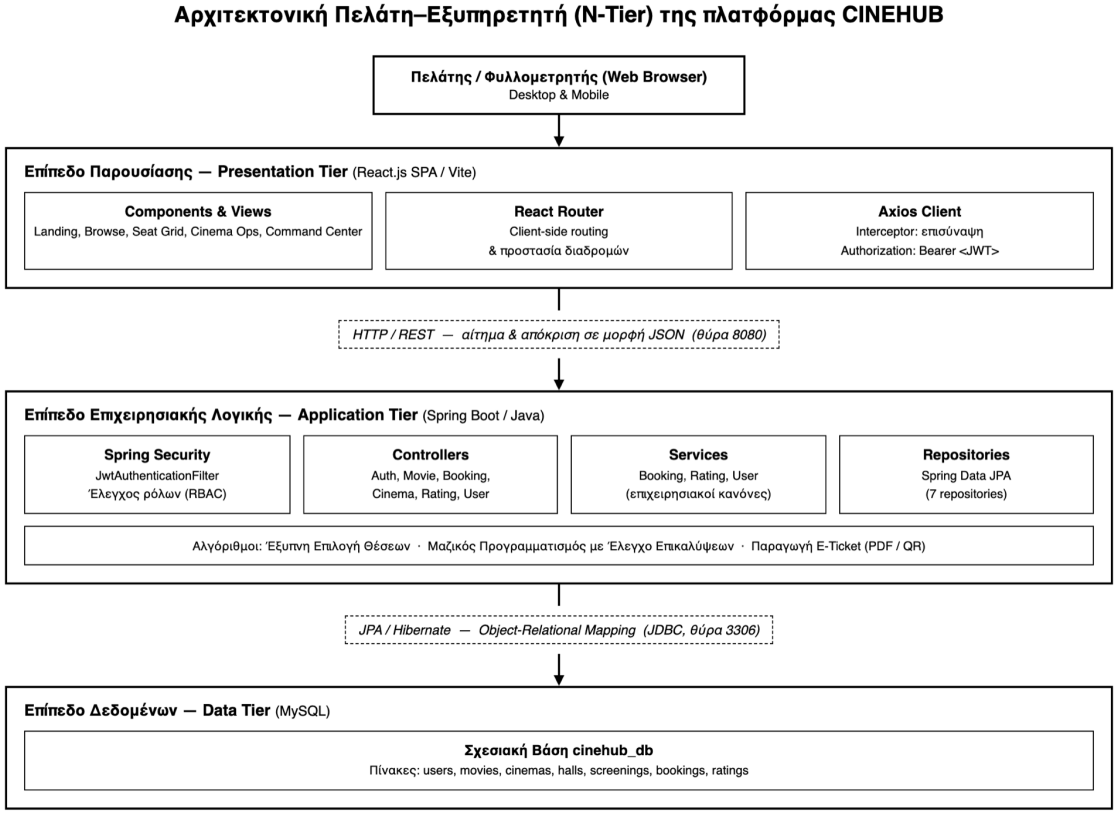
3.4 Αρχιτεκτονική Συστήματος (System Architecture)

Η αρχιτεκτονική που επιλέχθηκε για το CINEHUB είναι αυτή του μοντέλου Πελάτη-Εξυπηρετητή (Client-Server Architecture), κατανεμημένη σε πολλαπλά λογικά επίπεδα (N-Tier Architecture). Αυτή η αποσύνδεση επιτρέπει σε κάθε επίπεδο να αναπτύσσεται, να συντηρείται και να επεκτείνεται ανεξάρτητα.

1. **Επίπεδο Παρουσίασης (Frontend - Presentation Layer):** Υλοποιημένο ως Single Page Application (SPA) με τη χρήση της τεχνολογίας React. Το επίπεδο αυτό τρέχει εξ

ολοκλήρου στον φυλλομετρητή (browser) του τελικού χρήστη. Επικοινωνεί με το Backend αποκλειστικά μέσω ασύγχρονων HTTP κλήσεων (AJAX/Axios), ανταλλάσσοντας δεδομένα σε μορφή JSON (JavaScript Object Notation). Αυτό μειώνει δραματικά τον φόρτο δικτύου, καθώς μεταφέρονται μόνο τα απαραίτητα δεδομένα και όχι ολόκληρες σελίδες HTML.

2. **Επίπεδο Επιχειρησιακής Λογικής (Backend - Business Logic Layer):** Υλοποιημένο με το πλαίσιο Spring Boot (Java), το οποίο εκθέτει ένα σύνολο από RESTful Web Services (APIs). Ακολουθεί το πρότυπο Model-View-Controller (MVC) στο επίπεδο του εξυπηρετητή.
    - Οι **Controllers** (π.χ. CinemaController, MovieController) δέχονται τα HTTP αιτήματα από το Frontend, επικυρώνουν τα δεδομένα εισόδου (Data Transfer Objects - DTOs) και δρομολογούν την εκτέλεση.
    - Τα **Services** ενθυλακώνουν (encapsulate) την πολύπλοκη επιχειρησιακή λογική, όπως τον έλεγχο διαθεσιμότητας μιας αίθουσας ή τον υπολογισμό του κόστους με snacks.
3. **Επίπεδο Δεδομένων (Data Access Layer):** Το επίπεδο αυτό υλοποιείται μέσω του Spring Data JPA (Java Persistence API), το οποίο λειτουργεί ως ένα επίπεδο αφαίρεσης Object-Relational Mapping (ORM). Αυτό σημαίνει ότι ο κώδικας της Java επικοινωνεί με τη σχεσιακή βάση δεδομένων (RDBMS) χειριζόμενος άμεσα τα αντικείμενα των κλάσεων (Entities), χωρίς την ανάγκη συγγραφής εκτενών ακατέργαστων ερωτημάτων SQL (raw SQL queries). Η προσέγγιση αυτή αυξάνει την ασφάλεια (αποτροπή SQL Injection) και την ταχύτητα ανάπτυξης.



Εικόνα 3.3: Η αρχιτεκτονική Πελάτη–Εξυπηρετητή (N-Tier) της πλατφόρμας CINEHUB.

4. ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΟ ΥΠΟΒΑΘΡΟ

4.1 Εισαγωγή στην Αρχιτεκτονική της Υλοποίησης

Το στάδιο της υλοποίησης αποτελεί τη μετάφραση των αρχιτεκτονικών μοντέλων και των απαιτήσεων σε εκτελέσιμο λογισμικό. Για την πλατφόρμα CINEHUB, η διαδικασία ανάπτυξης (development) ακολούθησε τις σύγχρονες πρακτικές του Full-Stack Development. Βασικός γνώμονας κατά την επιλογή των εργαλείων και τη συγγραφή του πηγαίου κώδικα ήταν η δημιουργία ενός συστήματος επεκτάσιμου (scalable), ασφαλούς (secure), και συντηρήσιμου (maintainable). Η εφαρμογή αναπτύχθηκε βάσει του διαχωρισμού των επιπέδων (N-Tier Architecture), επιτρέποντας στο περιβάλλον διεπαφής (Frontend) να είναι πλήρως αποσυνδεδεμένο από το περιβάλλον επιχειρησιακής λογικής (Backend), επικοινωνώντας αποκλειστικά μέσω ασύγχρονων διεπαφών προγραμματισμού (REST API).

Η δομή των πακέτων (packages) του Backend και του Frontend

cinehub-backend — Spring Boot / Maven

```
src/main/java/com/thanosgk/cinehub/
├── CinehubApplication.java      σημείο εισόδου
│
├── config/                      ρυθμίσεις πλαισίου
│   ├── SecurityConfig.java
│   └── GlobalExceptionHandler.java
│
├── security/                    αυθεντικοποίηση JWT
│   ├── JwtProvider.java
│   ├── JwtAuthenticationFilter.java
│   └── UserDetailsServiceImpl.java
│
├── controller/                  REST άκρα (endpoints)
│   ├── AuthController.java
│   ├── MovieController.java
│   ├── CinemaController.java
│   ├── BookingController.java
│   ├── RatingController.java
│   └── UserController.java
│
├── service/                     επιχειρησιακή λογική
│   ├── BookingService.java
│   ├── RatingService.java
│   └── UserService.java
│
├── repository/                  Spring Data JPA
│   ├── UserRepository.java
│   ├── MovieRepository.java
│   ├── CinemaRepository.java
│   ├── HallRepository.java
│   ├── ScreeningRepository.java
│   ├── BookingRepository.java
│   └── RatingRepository.java
│
├── entity/                      οντότητες (ORM)
│   ├── User.java        ├── Movie.java
│   ├── Cinema.java      ├── Hall.java
│   ├── Screening.java   ├── Booking.java
│   ├── Rating.java      ├── Role.java
│   └── BookingStatus.java
│
└── dto/                         αντικείμενα μεταφοράς
    ├── AuthRequest.java
    ├── BulkScheduleRequest.java
    ├── MyBookingResponse.java
    └── Dtos.java

src/main/resources/
└── application.properties       σύνδεση MySQL, JWT
pom.xml                          εξαρτήσεις Maven
```

cinehub-frontend — React.js / Vite

```
src/
├── main.jsx                     σημείο εισόδου
├── App.jsx                      δρομολόγηση (Router)
│
├── pages/                       οθόνες εφαρμογής
│   ├── Landing.jsx
│   ├── Login.jsx
│   ├── Signup.jsx
│   ├── Dashboard.jsx
│   │     CommandCenterView
│   │     UserManagerView
│   │     MovieManagerView
│   │     CinemaManagerView
│   │       (Halls / Planner /
│   │       Revenue / Live View)
│   │     CustomerBrowser
│   │     MovieDetailsView
│   │     BookingView
│   │       (Seats / Snacks / Pay)
│   │     MyTicketsView
│   ├── MovieManager.jsx
│   ├── CinemaManager.jsx
│   └── Customer.jsx
│
├── components/
│   └── Layout.jsx               κοινό κέλυφος
│
├── services/
│   └── api.js                   Axios + JWT
│                                interceptor
│
├── styles/
│   └── tokens/                  design tokens
│
├── assets/
│
├── Landing.css
├── Login.css
├── Dashboard.css
├── App.css
└── index.css

index.html
vite.config.js
package.json                     εξαρτήσεις npm
```

Εικόνα 4.1: Η δομή των πακέτων (packages) του Backend και του Frontend στο περιβάλλον ανάπτυξης.

4.2 Τεχνολογικό Υπόβαθρο και Εργαλεία Ανάπτυξης

4.2.1 Το Επίπεδο Παρουσίασης (Frontend): React.js

Το οπτικό περιβάλλον αλληλεπίδρασης του χρήστη αναπτύχθηκε εξ ολοκλήρου με τη χρήση της βιβλιοθήκης **React.js**. Η React εισάγει τη φιλοσοφία της "δημιουργίας διεπαφών βάσει συστατικών μερών" (Component-based architecture). Το κυριότερο πλεονέκτημα της React που την κατέστησε ιδανική για το CINEHUB είναι η χρήση του Εικονικού Μοντέλου Αντικειμένων Εγγράφου (Virtual DOM). Σε αντίθεση με τις παραδοσιακές ιστοσελίδες, η React διατηρεί ένα ελαφρύ αντίγραφο του DOM στη μνήμη. Όταν η κατάσταση (state) της εφαρμογής αλλάζει –για παράδειγμα, όταν ο χρήστης επιλέγει μια θέση στο πλάνο του κινηματογράφου– η React υπολογίζει τις ελάχιστες δυνατές αλλαγές (diffing algorithm) και ενημερώνει μόνο το συγκεκριμένο τμήμα της πραγματικής οθόνης.

Για τη βέλτιστη διαχείριση του κύκλου ζωής των συστατικών (components) και της πολύπλοκης ροής δεδομένων της πλατφόρμας, υιοθετήθηκε η αρχιτεκτονική των Συναρτήσεων Άγκιστρων (React Hooks). Η χρήση τους επέτρεψε την επίλυση κρίσιμων λειτουργικών απαιτήσεων του CINEHUB ως εξής:

- **Διαχείριση Τοπικής Κατάστασης (State Management) με το useState:** Η συνάρτηση αυτή υπήρξε καθοριστική για τη διαχείριση της παροδικής κατάστασης (transient state) του χρήστη. Για παράδειγμα, στη διαδικασία κράτησης, το useState διατηρεί τοπικά στη μνήμη του φυλλομετρητή τις επιλεγμένες θέσεις και τα προϊόντα του κυλικείου. Αυτό εξασφαλίζει ότι ο χρήστης μπορεί να πλοηγηθεί ομαλά στα βήματα της κράτησης (Επιλογή Θέσεων → Snacks → Πληρωμή) χωρίς απώλεια δεδομένων και χωρίς την ανάγκη συνεχούς, κοστοβόρου συγχρονισμού με τον εξυπηρετητή μέχρι την τελική επιβεβαίωση της συναλλαγής.
- **Συγχρονισμός Δεδομένων και Ασύγχρονες Κλήσεις με το useEffect:** Αξιοποιήθηκε στρατηγικά για τη διαχείριση των παρενεργειών (side-effects) του συστήματος, και συγκεκριμένα για τη διασύνδεση του Frontend με το RESTful API του Backend. Μέσω του useEffect, το σύστημα "ενυδατώνεται" (hydrates) με πραγματικά δεδομένα από τη βάση (όπως ο κατάλογος ταινιών και οι διαθέσιμες προβολές) ακριβώς τη στιγμή που το συστατικό "προσαρτάται" (mounts) στο DOM.

- **Άμεσος Χειρισμός DOM με το useRef:** Σε περιπτώσεις όπου ο τυπικός κύκλος επανασχεδίασης (re-render) της React δεν επαρκούσε ή προκαλούσε πτώση επιδόσεων, χρησιμοποιήθηκε το useRef. Χαρακτηριστικό παράδειγμα αποτελεί η υλοποίηση του διαδραστικού καρουζέλ (carousel) για την επιλογή ημερομηνιών στην καρτέλα της ταινίας, όπου η άμεση αναφορά στο DOM προσέφερε μια ομαλή, "native-like" εμπειρία οριζόντιας κύλισης.

Επιπλέον, χρησιμοποιήθηκαν βιβλιοθήκες όπως το Axios για την αποστολή ασύγχρονων HTTP αιτημάτων, το React Router για τη δρομολόγηση χωρίς ανανέωση της σελίδας (SPA routing), και ειδικές βιβλιοθήκες όπως η jsPDF και το react-qr-code για την επιτόπια, πελατοκεντρική (client-side) παραγωγή των ψηφιακών εισιτηρίων.

4.2.2 Το Επίπεδο Επιχειρησιακής Λογικής (Backend): Spring Boot

Η "ραχοκοκαλιά" του συστήματος βασίστηκε στο **Spring Boot**, ένα ισχυρό πλαίσιο εργασίας της γλώσσας Java. Το Spring Boot διευκολύνει τη δημιουργία αυτόνομων (stand-alone) εφαρμογών βιομηχανικών προδιαγραφών (production-grade).

Η αρχιτεκτονική του Spring βασίζεται σε δύο θεμελιώδεις αρχές της Μηχανικής Λογισμικού: την Αντιστροφή Ελέγχου (Inversion of Control - IoC) και την Έγχυση Εξαρτήσεων (Dependency Injection - DI). Αυτό σημαίνει ότι ο προγραμματιστής δεν δημιουργεί χειροκίνητα αντικείμενα (instances) των κλάσεων μέσω της εντολής new, αλλά το ίδιο το πλαίσιο (Spring Container) αναλαμβάνει να "ενέσει" τα απαραίτητα αντικείμενα (Beans) εκεί που χρειάζονται. Για παράδειγμα, στα αρχεία των ελεγκτών (Controllers) του CINEHUB, χρησιμοποιήθηκε η τεχνική Constructor Injection (μέσω της σήμανσης @RequiredArgsConstructor του Lombok) για την ασφαλή και άμεση παροχή των Repositories και Services.

4.2.3 Το Επίπεδο Δεδομένων (Data Access): Spring Data JPA

Η επικοινωνία με τη σχεσιακή βάση δεδομένων υλοποιήθηκε μέσω του **Spring Data JPA** (Java Persistence API), το οποίο προσφέρει ένα επίπεδο αφαίρεσης Αντικειμενοστρεφούς-Σχεσιακής Χαρτογράφησης (Object-Relational Mapping - ORM). Με το JPA, κάθε πίνακας της βάσης αντιστοιχίζεται (mapped) σε μία κλάση της Java (Entity). Οι βιβλιοθήκες αυτές αναλαμβάνουν την αυτόματη μετατροπή των SQL ερωτημάτων, προστατεύοντας παράλληλα την εφαρμογή από κενά ασφαλείας, όπως οι επιθέσεις "SQL Injection".

4.3 Υλοποίηση Backend: Ανάλυση Ελεγκτών Συστήματος (Controllers)

Το επίπεδο των Controllers αποτελεί το σημείο εισόδου (entry point) για την επικοινωνία μεταξύ Frontend και Backend. Στο CINEHUB αναπτύχθηκαν πολλαπλοί ελεγκτές, καθένας υπεύθυνος για έναν διακριτό τομέα επιχειρησιακής λογικής, τηρώντας την αρχή της Ενιαίας Ευθύνης (Single Responsibility Principle).

4.3.1 Ελεγκτής Αυθεντικοποίησης (AuthController)

Ο AuthController αναλαμβάνει την κρίσιμη λειτουργία της εισόδου (/login) και εγγραφής (/register) των χρηστών. Κατά την εγγραφή, το σύστημα ελέγχει αρχικά την ύπαρξη του ονόματος χρήστη (username) για την αποτροπή διπλότυπων εγγραφών. Εν συνεχεία, ο κωδικός πρόσβασης δεν αποθηκεύεται σε απλή μορφή (plain text), αλλά κρυπτογραφείται μέσω του PasswordEncoder (συγκεκριμένα με τον αλγόριθμο BCrypt), διασφαλίζοντας την ιδιωτικότητα. Ανάλογα με τα αιτούμενα δικαιώματα, το σύστημα αποδίδει τον κατάλληλο ρόλο (π.χ. ROLE_CUSTOMER, ROLE_CINEMA_MANAGER).

Κατά τη διαδικασία εισόδου, το AuthenticationManager επαληθεύει τα διαπιστευτήρια και, εφόσον είναι ορθά, η κλάση JwtProvider παράγει ένα μοναδικό JSON Web Token. Το token αυτό, μαζί με το όνομα χρήστη και τους ρόλους του, επιστρέφεται στο Frontend ενθυλακωμένο στο αντικείμενο AuthResponse.

4.3.2 Ελεγκτής Διαχείρισης Κινηματογράφου (CinemaController)

Ο CinemaController ενσωματώνει προηγμένη επιχειρησιακή λογική, καθώς διαχειρίζεται τα φυσικά περιουσιακά στοιχεία των αιθουσών και τα ωράρια των προβολών. Μια κρίσιμη προγραμματιστική απόφαση ήταν η υλοποίηση του μηχανισμού εξουσιοδότησης και ιδιοκτησίας. Πριν εκτελεστεί οποιαδήποτε μεταβολή (όπως επεξεργασία κινηματογράφου ή διαγραφή αίθουσας), το σύστημα εκτελεί τον εσωτερικό έλεγχο getOwnedCinema. Αυτός ο έλεγχος διασφαλίζει ότι μόνο ο νόμιμος ιδιοκτήτης-διαχειριστής του κινηματογράφου ή ο κεντρικός διαχειριστής (Admin) μπορεί να επηρεάσει τα δεδομένα, προστατεύοντας το σύστημα από αυθαίρετες τροποποιήσεις (IDOR - Insecure Direct Object Reference).

Μία από τις πιο πολύπλοκες λειτουργίες αυτού του ελεγκτή είναι ο μηχανισμός προγραμματισμού (scheduleScreening). Το Backend λαμβάνει την ταινία, την αίθουσα, τον χρόνο έναρξης και την τιμή. Ωστόσο, το σύστημα δεν αποθηκεύει τυφλά την εγγραφή. Υπολογίζει δυναμικά τον χρόνο λήξης (προσθέτοντας τη διάρκεια της ταινίας συν 15 λεπτά για καθαρισμό της αίθουσας) και εκτελεί έλεγχο επικαλύψεων (conflict validation) με τις ήδη υπάρχουσες προβολές στην ίδια αίθουσα (Checking Overlaps).

4.3.3 Ελεγκτής Κρατήσεων (BookingController)

Ο BookingController διαχειρίζεται τον πυρήνα των συναλλαγών της πλατφόρμας. Για την ασφαλή ταυτοποίηση του πελάτη που εκτελεί την κράτηση, το Backend δεν εξαρτάται από παραμέτρους του Frontend, αλλά αντλεί την ταυτότητα του χρήστη άμεσα από το SecurityContextHolder του Spring Security.

Επιπλέον, αναλαμβάνει τη σύνδεση μεταξύ βάσης δεδομένων και Frontend μέσω των Data Transfer Objects (DTOs). Για παράδειγμα, όταν ο χρήστης αιτείται το ιστορικό των εισιτηρίων του (/mine), η οντότητα Booking –η οποία περιέχει ευαίσθητα δεδομένα συσχετίσεων– μετατρέπεται μέσω της συνάρτησης mapToDto στο αντικείμενο MyBookingResponse. Το αντικείμενο αυτό περιέχει μόνο τα δεδομένα που ενδιαφέρουν τον πελάτη: το αναγνωριστικό της κράτησης, τον τίτλο της ταινίας, την ημερομηνία, τις θέσεις και τα συνοδευτικά snacks, προστατεύοντας την αρχιτεκτονική του Backend (data hiding).

4.3.4 Ελεγκτές Αξιολογήσεων (RatingController) & Χρηστών (UserController)

Το σύστημα ενθαρρύνει τη διαδραστικότητα μέσω του RatingController. Επιτρέπει στους χρήστες να βαθμολογούν ταινίες, με το Backend να ελέγχει αν ο χρήστης έχει ήδη υποβάλει κριτική (checkRatingStatus). Επιπλέον, παρέχει ένα endpoint (/all) προσβάσιμο μόνο σε διαχειριστές (@PreAuthorize("hasRole('ADMIN')")), όπου ανακτώνται όλες οι κριτικές της πλατφόρμας για σκοπούς ελέγχου (moderation) και ενδεχόμενης διαγραφής προσβλητικού περιεχομένου. Αντίστοιχα, ο UserController λειτουργεί ως το εργαλείο διαχείρισης συστήματος. Επιτρέπει στους Administrators να ανακτούν τη λίστα των εγγεγραμμένων χρηστών (εξαιρουμένων των κωδικών πρόσβασης), να ενημερώνουν δυναμικά τα δικαιώματά τους (updateRoles) ή να προχωρούν σε διαγραφές.

4.4 Επίπεδο Επιχειρησιακής Λογικής (Service Layer) και Αλγοριθμική Διαχείριση

Το Επίπεδο Επιχειρησιακής Λογικής (Business Logic Layer) αποτελεί τον ενδιάμεσο κρίκο επικοινωνίας μεταξύ των Ελεγκτών (Controllers) και του Επιπέδου Πρόσβασης Δεδομένων (Repositories). Στο πλαίσιο του Spring Boot, το επίπεδο αυτό υλοποιείται μέσω κλάσεων που φέρουν τη σήμανση @Service. Η χρήση των Services διασφαλίζει τον διαχωρισμό των αρμοδιοτήτων (Separation of Concerns), διατηρώντας τους Controllers "λεπτούς" (thin controllers) ώστε να ασχολούνται αποκλειστικά με τη δρομολόγηση των HTTP αιτημάτων, ενώ η βαριά υπολογιστική λογική εκτελείται στο παρασκήνιο.

Επιπλέον, στα Services γίνεται η διαχείριση των συναλλαγών της βάσης δεδομένων (Transaction Management) μέσω της σήμανσης @Transactional, η οποία εγγυάται την ατομικότητα (atomicity) των λειτουργιών (π.χ. αν αποτύχει η πληρωμή, ακυρώνεται και η δέσμευση της θέσης).

4.4.1 Υποσύστημα Κρατήσεων και Υπολογισμού Κόστους (BookingService)

Ο πυρήνας των συναλλαγών της πλατφόρμας ελέγχεται από το BookingService. Όταν ένας χρήστης υποβάλλει ένα αίτημα κράτησης, η επιχειρησιακή λογική ακολουθεί μια αυστηρή ακολουθία επικυρώσεων:

1. **Ταυτοποίηση Χρήστη:** Αντλείται το username μέσω του SecurityContextHolder, διασφαλίζοντας ότι η κράτηση θα χρεωθεί στον αυθεντικοποιημένο χρήστη και αποτρέποντας απόπειρες πλαστογράφησης ταυτότητας.
2. **Έλεγχος Διαθεσιμότητας και Ταυτοχρονισμού (Concurrency):** Το σύστημα ανασύρει την προβολή (Screening) από τη βάση δεδομένων και ελέγχει αν οι αιτούμενες θέσεις (μια λίστα από Strings, π.χ. ["A1", "A2"]) περιλαμβάνονται ήδη στις δεσμευμένες θέσεις προηγούμενων κρατήσεων. Ο έλεγχος αυτός είναι κρίσιμος για την αποτροπή διπλοκρατήσεων (double-booking).
3. **Υπολογισμός Κόστους Εισιτηρίων και Προϊόντων Κυλικείου:** Ο αλγόριθμος τιμολόγησης δεν βασίζεται σε τιμές που αποστέλλει το Frontend (καθώς ο χρήστης θα μπορούσε να τις παραποιήσει). Αντιθέτως, υπολογίζει το κόστος δυναμικά στο Backend:
    - **Βασικό Κόστος:** Πολλαπλασιάζεται το πλήθος των επιλεγμένων θέσεων με την τιμή μονάδας (price) της εκάστοτε προβολής.

- **Κόστος Snacks:** Το σύστημα διατρέχει τη δομή δεδομένων Λεξικού (Map) των snacks, όπου το κλειδί είναι το είδος (π.χ. "popcorn") και η τιμή είναι η ποσότητα. Πολλαπλασιάζει την ποσότητα με τον εσωτερικό τιμοκατάλογο του Backend και προσθέτει το αποτέλεσμα στο συνολικό ποσό.

4. **Οριστικοποίηση:** Η κράτηση λαμβάνει την κατάσταση BookingStatus.CONFIRMED, καταγράφεται η χρονική σφραγίδα (bookingTime) και αποθηκεύεται ατομικά (atomically) στη βάση μέσω του BookingRepository.

4.4.2 Μηχανισμοί Στατιστικής Ανάλυσης και Εσόδων (Revenue Calculation)

Στο πλαίσιο των εργαλείων διαχείρισης, υλοποιήθηκε ένας προηγμένος μηχανισμός υπολογισμού εσόδων (Revenue Analytics) για τη δημιουργία των διαγραμμάτων που βλέπουν οι Διαχειριστές Κινηματογράφων (Cinema Managers).

Η διαδικασία αυτή, που καλείται μέσω του endpoint /api/cinemas/{cinemaId}/stats/revenue, λειτουργεί ως εξής:

- **Καθορισμός Χρονικού Πλαισίου:** Ορίζεται ημερομηνία έναρξης (start date) 6 ημέρες πριν από την τρέχουσα ημερομηνία, δημιουργώντας ένα κυλιόμενο παράθυρο 7 ημερών.
- **Άντληση και Φιλτράρισμα Δεδομένων:** Το σύστημα εκτελεί ερωτήματα στη βάση μέσω του ScreeningRepository για να βρει όλες τις προβολές του συγκεκριμένου κινηματογράφου (findByHall_Cinema_Id).
- **Ροή Επεξεργασίας (Stream API):** Χρησιμοποιώντας τις σύγχρονες δυνατότητες του Java Stream API, ο αλγόριθμος εκτελεί έναν βρόχο (loop) 7 επαναλήψεων (μία για κάθε ημέρα). Σε κάθε επανάληψη:
    1. Φιλτράρει τις προβολές ώστε να ταιριάζουν με την ημερομηνία-στόχο (target date).
    2. Ανασύρει όλες τις κρατήσεις (Booking) που συνδέονται με αυτές τις προβολές.
    3. Φιλτράρει αυστηρά μόνο τις κρατήσεις που έχουν κατάσταση CONFIRMED (αποκλείοντας τις ακυρωμένες).
    4. Αθροίζει το totalPrice (mapToDouble().sum()) για να υπολογίσει τα καθαρά έσοδα της ημέρας.
- **Εξαγωγή Αποτελέσματος:** Το αποτέλεσμα είναι μια δομημένη λίστα (List) από πραγματικούς αριθμούς (Double), η οποία καταναλώνεται απευθείας από τη βιβλιοθήκη γραφημάτων του Frontend.

4.4.3 Επιχειρησιακή Λογική Αξιολογήσεων και Ποιότητας Περιεχομένου

Η αξιοπιστία του συστήματος αξιολογήσεων (Rating System) βασίζεται στην επιχειρησιακή λογική του RatingService. Για την αποτροπή χειραγώγησης των βαθμολογιών (review bombing), έχουν υλοποιηθεί οι εξής αλγοριθμικοί έλεγχοι:

- **Έλεγχος Μοναδικότητας (Check Rating Status):** Πριν την καταχώρηση μιας κριτικής, η μέθοδος hasUserRated ελέγχει εάν ο συγκεκριμένος χρήστης έχει ήδη αξιολογήσει τη συγκεκριμένη ταινία. Αυτό υποστηρίζεται και στο επίπεδο της βάσης δεδομένων με τον περιορισμό UniqueConstraint(columnNames = {"user_id", "movie_id"}).
- **Ενοποίηση Δεδομένων για Διαχειριστές (Admin Moderation):** Η μέθοδος getAllSystemReviews ανασύρει όλες τις κριτικές της πλατφόρμας και τις ταξινομεί χρονολογικά (από τις νεότερες προς τις παλαιότερες). Στη συνέχεια, χαρτογραφεί (maps) τις πολύπλοκες οντότητες της βάσης σε απλά λεξικά (Map), συνδέοντας το όνομα του χρήστη με τον τίτλο της ταινίας και το κείμενο της κριτικής. Αυτή η λογική επιτρέπει στο Κέντρο Ελέγχου (Command Center) να λειτουργεί γρήγορα, χωρίς να φορτώνει ολόκληρα δέντρα αντικειμένων (Object Trees) στη μνήμη του διακομιστή.

4.5 Αρχιτεκτονική Ασφάλειας, Stateless Αυθεντικοποίηση (JWT) και Κρυπτογραφική Θωράκιση (BCrypt)

Η ασφάλεια των δεδομένων, η προστασία της ιδιωτικότητας των χρηστών και η θωράκιση των κρίσιμων συναλλαγών (όπως η κράτηση θέσεων και η διαχείριση εσόδων) αποτέλεσαν κεντρικό πυλώνα κατά τον σχεδιασμό και την ανάπτυξη της πλατφόρμας CINEHUB. Ακολουθώντας την αρχή του "Security by Design", το σύστημα δεν ενσωματώνει την ασφάλεια ως μια εκ των υστέρων προσθήκη, αλλά τη δομεί οργανικά στα χαμηλότερα επίπεδα της αρχιτεκτονικής του.

Στο παρόν κεφάλαιο αναλύεται διεξοδικά, τόσο σε θεωρητικό όσο και σε πρακτικό-υλοποιητικό επίπεδο, το τρίπτυχο της ασφάλειας του CINEHUB: η αυθεντικοποίηση χωρίς κατάσταση (Stateless Authentication) μέσω JSON Web Tokens (JWT), ο συντονισμός των δικαιωμάτων μέσω του AuthenticationManager του Spring Security, και η μη αναστρέψιμη κρυπτογράφηση κωδικών πρόσβασης με τον αλγόριθμο BCrypt.

4.5.1 Μηχανισμός Αυθεντικοποίησης δίχως Κατάσταση (Stateless JWT Authentication)

Στις παραδοσιακές διαδικτυακές εφαρμογές, η αυθεντικοποίηση βασίζεται σε κρατούμενες συνεδρίες (Stateful Sessions). Ο διακομιστής (Server) δημιουργεί μια συνεδρία στη μνήμη του (ή σε κάποια βάση δεδομένων τύπου Redis) και επιστρέφει ένα αναγνωριστικό (Session ID) στον φυλλομετρητή (Browser) μέσω ενός Cookie. Η προσέγγιση αυτή, ωστόσο, παρουσιάζει σημαντικά μειονεκτήματα:

- Προβλήματα Κλιμάκωσης (Scalability Bottlenecks): Όσο αυξάνονται οι ταυτόχρονοι χρήστες, η μνήμη του Server επιβαρύνεται γραμμικά.
- Αρχιτεκτονική Μικροϋπηρεσιών / Κατανεμημένων Συστημάτων: Σε περίπτωση που το σύστημα επεκταθεί σε πολλούς servers πίσω από έναν Load Balancer, απαιτείται πολύπλοκος συγχρονισμός των sessions (Session Replication).
- Τρωτότητα σε Επιθέσεις: Τα cookies είναι ευάλωτα σε διασταυρούμενες πλαστογραφήσεις αιτημάτων (Cross-Site Request Forgery - CSRF).

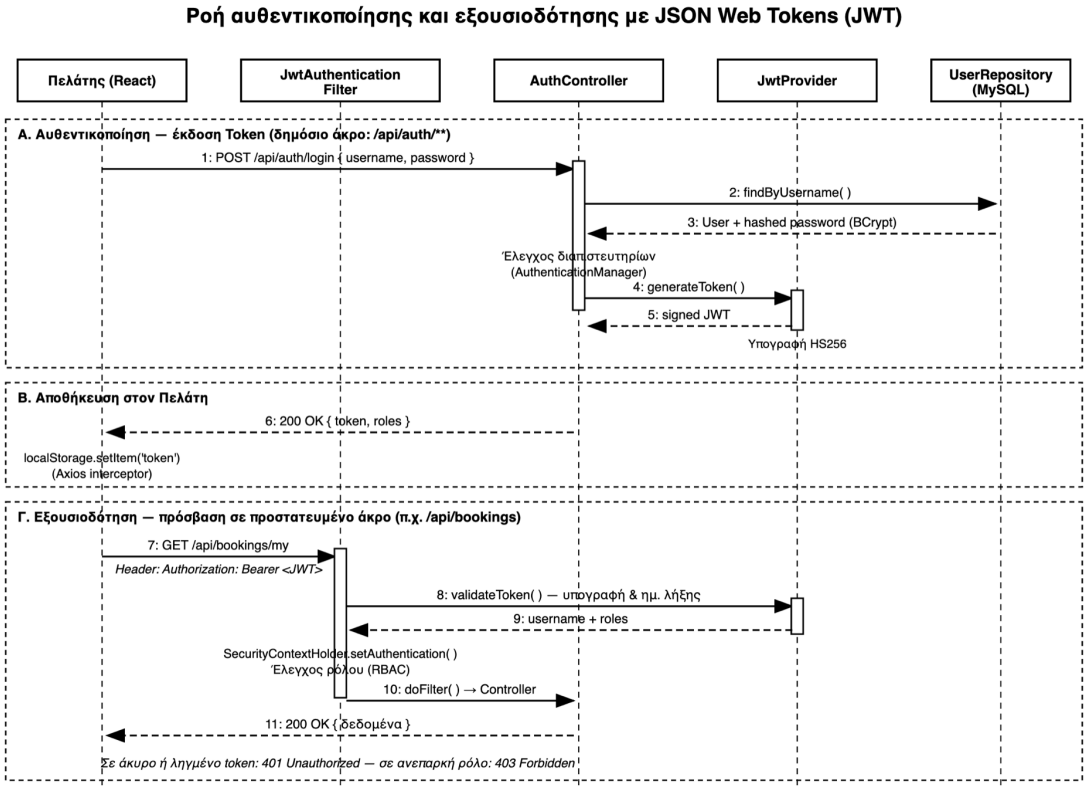
Για την επίλυση των ανωτέρω προβλημάτων, η πλατφόρμα CINEHUB υιοθετεί το διεθνές πρότυπο JSON Web Token (JWT - RFC 7519). Η αυθεντικοποίηση με JWT είναι πλήρως stateless, πράγμα που σημαίνει ότι ο διακομιστής δεν αποθηκεύει καμία πληροφορία για τους συνδεδεμένους χρήστες στη μνήμη του. Κάθε Token είναι μια αυτόνομη (self-contained) κρυπτογραφημένη συμβολοσειρά που μεταφέρει την ταυτότητα και τα δικαιώματα του χρήστη.

Δομικά, ένα JWT Token αποτελείται από τρία διακριτά μέρη που διαχωρίζονται με τελείες:

1. Header (Κεφαλίδα): Καθορίζει τον τύπο του token (JWT) και τον αλγόριθμο ψηφιακής υπογραφής που χρησιμοποιήθηκε (π.χ., HMAC SHA256).
2. Payload (Ωφέλιμο Φορτίο): Περιέχει τα λεγόμενα Claims, δηλαδή τα δεδομένα του χρήστη. Στο CINEHUB, το Payload μεταφέρει το username και το σύνολο των δικαιωμάτων πρόσβασης (roles), όπως αυτά αντλούνται από την οντότητα User (π.χ. ROLE_CUSTOMER, ROLE_CINEMA_MANAGER).
3. Signature (Ψηφιακή Υπογραφή): Παράγεται από τον συνδυασμό του κωδικοποιημένου Header, του κωδικοποιημένου Payload και ενός μυστικού κλειδιού (Secret Key) που γνωρίζει αποκλειστικά το Backend. Η υπογραφή διασφαλίζει την ακεραιότητα του token. Εάν ένας κακόβουλος χρήστης προσπαθήσει να τροποποιήσει το Payload (π.χ. να αλλάξει τον ρόλο του από CUSTOMER σε ADMIN), η υπογραφή καθίσταται αυτόματα άκυρη και το Backend απορρίπτει το αίτημα.

Η Πρακτική Ροή στην Εφαρμογή: Όπως υλοποιείται στον AuthController.java, κατά τη διαδικασία του login, αφού επαληθευτούν τα διαπιστευτήρια, καλείται η κλάση jwtProvider.generateToken(authentication). Το παραχθέν token επιστρέφεται στο Frontend μέσα στο αντικείμενο Dtos.AuthResponse. Η React λαμβάνει το token, το αποθηκεύει με ασφάλεια στο LocalStorage της συσκευής και, μέσω της βιβλιοθήκης Axios, το επισυνάπτει αυτόματα στο HTTP Header κάθε επόμενου αιτήματος με τη μορφή: "Authorization: Bearer <token_string>".

Στο Backend, ένα custom φίλτρο ασφαλείας (Security Filter Chain) υποκλέπτει το αίτημα, επαληθεύει την υπογραφή του token χρησιμοποιώντας το Secret Key, εξάγει το username και τα roles, και τροφοδοτεί το SecurityContextHolder του Spring Security, επιτρέποντας ή απαγορεύοντας την πρόσβαση στα endpoints.



Εικόνα 4.2: Ροή αυθεντικοποίησης και εξουσιοδότησης με JSON Web Tokens (JWT).

4.5.2 Ο Ρόλος του AuthenticationManager στο Spring Security

Ο AuthenticationManager αποτελεί τον κεντρικό "ενορχηστρωτή" της διαδικασίας ταυτοποίησης στο οικοσύστημα του Spring Security. Δεν εκτελεί ο ίδιος την επαλήθευση, αλλά αναθέτει τη διασταύρωση των στοιχείων σε έναν ή περισσότερους AuthenticationProviders.

Στον πηγαίο κώδικα του AuthController.java, η διαδικασία της εισόδου ξεκινά με την εξής εντολή:

Authentication authentication = authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(request.getUsername(), request.getPassword()));

Η λειτουργία αυτή αναλύεται στα εξής βήματα:

1. Δημιουργία unauthenticated Token: Τα ακατέργαστα στοιχεία που εισήγαγε ο χρήστης στη φόρμα Login.jsx (username και password) ενθυλακώνονται στο αντικείμενο UsernamePasswordAuthenticationToken. Σε αυτό το στάδιο, το αντικείμενο φέρει τη σήμανση ότι δεν έχει ακόμα αυθεντικοποιηθεί (authenticated = false).
2. Ανάθεση και Επαλήθευση: Ο AuthenticationManager λαμβάνει αυτό το αντικείμενο και καλεί τον κατάλληλο Provider (συνήθως τον DaoAuthenticationProvider). Ο Provider χρησιμοποιεί ένα custom UserDetailsService το οποίο εκτελεί το ερώτημα userRepository.findByUsername(username).
3. Διασταύρωση Κωδικών: Εάν ο χρήστης βρεθεί, ο Provider λαμβάνει τον κρυπτογραφημένο κωδικό από τη βάση δεδομένων και τον συγκρίνει με το plain-text password που έδωσε ο χρήστης στο login, χρησιμοποιώντας τη μέθοδο matches() του BCrypt.
4. Δημιουργία authenticated Token: Εάν οι κωδικοί ταιριάζουν, ο AuthenticationManager επιστρέφει ένα νέο αντικείμενο Authentication, πλήρως συμπληρωμένο, το οποίο πλέον περιέχει τα Principal δεδομένα του χρήστη και τη λίστα με τα εξουσιοδοτημένα δικαιώματά του (GrantedAuthorities), φέροντας τη σήμανση authenticated = true. Αυτό το αντικείμενο χρησιμοποιείται στη συνέχεια από τον JwtProvider για τη δημιουργία του JWT.

4.5.3 Κρυπτογραφική Θωράκιση Κωδικών Πρόσβασης (BCrypt Hashing)

Μία από τις σημαντικότερες παραβιάσεις της αρχής της ασφάλειας σε πληροφοριακά συστήματα είναι η αποθήκευση κωδικών πρόσβασης σε απλή μορφή (Plain Text) ή η χρήση ξεπερασμένων και γραμμικών αλγορίθμων hashing, όπως ο MD5 και ο SHA-256.

Εάν ένας κακόβουλος χρήστης αποκτήσει πρόσβαση στη βάση δεδομένων (μέσω κάποιου κενού ασφαλείας όπως το SQL Injection), οι απλοί κωδικοί εκτίθενται αμέσως. Ακόμη και οι απλές συναρτήσεις SHA-256 είναι ευάλωτες σε επιθέσεις με προ-υπολογισμένους πίνακες τιμών (Rainbow Tables) ή σε επιθέσεις ωμής βίας (Brute Force), καθώς οι σύγχρονες κάρτες γραφικών (GPUs) μπορούν να υπολογίσουν δισεκατομμύρια SHA-256 hashes το δευτερόλεπτο.

Για τον λόγο αυτό, η πλατφόρμα CINEHUB χρησιμοποιεί αποκλειστικά τον αλγόριθμο BCrypt μέσω της διεπαφής PasswordEncoder του Spring Security. Ο BCrypt είναι μια ισχυρή, μη αναστρέψιμη (one-way) συνάρτηση κατακερματισμού (hashing), η οποία βασίζεται στον κρυπτογραφικό αλγόριθμο Blowfish. Η ανωτερότητα του BCrypt οφείλεται σε δύο βασικά χαρακτηριστικά:

1. Αυτόματη Ενσωμάτωση "Αλατιού" (Cryptographic Salting): Κατά την εγγραφή ενός χρήστη (όπως φαίνεται στη μέθοδο register του AuthController.java), εκτελείται η εντολή: passwordEncoder.encode(request.getPassword()). Ο BCrypt παράγει για κάθε χρήστη ένα τυχαίο string (salt) και το ενσωματώνει στον κωδικό πριν εκτελέσει το hashing. Αυτό σημαίνει ότι αν δύο διαφορετικοί χρήστες επιλέξουν τον ίδιο ακριβώς κωδικό πρόσβασης (π.χ. password123), τα τελικά hashes που θα αποθηκευτούν στον πίνακα users της βάσης δεδομένων θα είναι εντελώς διαφορετικά. Η τεχνική αυτή εξουδετερώνει πλήρως την αποτελεσματικότητα των επιθέσεων με Rainbow Tables.
2. Προσαρμόσιμος Συντελεστής Κόστους (Work Factor / Adaptive Hashing): Ο BCrypt ενσωματώνει έναν παράμετρο κόστους (Log Rounds), ο οποίος καθορίζει πόσες φορές θα επαναληφθεί η κρυπτογραφική διεργασία στο παρασκήνιο (εκθετική αύξηση). Αυτό καθιστά τον αλγόριθμο σκόπιμα "αργό". Ενώ ένας Server μπορεί να επαληθεύσει έναν κωδικό σε λίγα χιλιοστά του δευτερολέπτου (κάτι που δεν επηρεάζει την εμπειρία του χρήστη), μια απόπειρα μαζικής επίθεσης Brute Force καθίσταται υπολογιστικά αδύνατη, καθώς η απαίτηση σε χρόνο και πόρους GPU αυξάνεται σε απαγορευτικά επίπεδα.

Θωράκιση στο επίπεδο των Entity Logs: Η προστασία των κωδικών επεκτείνεται και στο επίπεδο της αποσφαλμάτωσης (logging). Στην οντότητα User.java, το πεδίο password φέρει το annotation του Lombok: @ToString.Exclude private String password;

Η συγκεκριμένη υλοποίηση διασφαλίζει ότι ακόμα και αν ένας προγραμματιστής εκτυπώσει κατά λάθος τα δεδομένα ενός χρήστη στα αρχεία καταγραφής του συστήματος (π.χ. System.out.println(user) ή μέσω κάποιου Logger), το κρυπτογραφημένο password θα εξαιρεθεί αυτόματα, αποτρέποντας την ακούσια διαρροή ευαίσθητων πληροφοριών στα system logs.

4.6 Αρχιτεκτονική Ενιαίας Σελίδας (SPA), Διαχείριση Κατάστασης και Σχεδιασμός Διεπαφής Χρήστη (UI/UX)

Η αποτελεσματικότητα ενός σύγχρονου πληροφοριακού συστήματος δεν κρίνεται μόνο από την ευρωστία του Backend, αλλά και από την εμπειρία που προσφέρει στον τελικό χρήστη. Για την πλατφόρμα CINEHUB, το Επίπεδο Παρουσίασης (Frontend) αναπτύχθηκε εξ ολοκλήρου με τη βιβλιοθήκη React.js, υιοθετώντας την αρχιτεκτονική της Εφαρμογής Ενιαίας Σελίδας (Single Page Application - SPA). Στο παρόν υποκεφάλαιο αναλύονται οι τεχνικές επιλογές για τη δρομολόγηση, τη διαχείριση της κατάστασης (State Management) και τον διαδραστικό σχεδιασμό διεπαφής (UI).

4.6.1 Δρομολόγηση στο Επίπεδο του Πελάτη (Client-Side Routing)

Στις παραδοσιακές διαδικτυακές εφαρμογές, κάθε μετάβαση σε νέα σελίδα απαιτεί την αποστολή αιτήματος στον διακομιστή και την πλήρη επαναφόρτωση του εγγράφου HTML. Στο CINEHUB, η προσέγγιση αυτή αντικαταστάθηκε από τη δρομολόγηση στο επίπεδο του πελάτη (Client-Side Routing) με χρήση της βιβλιοθήκης React Router.

Όπως αποτυπώνεται στο αρχείο App.jsx, το σύστημα χρησιμοποιεί τα συστατικά BrowserRouter και Routes για να παρακολουθεί τις αλλαγές στο URL του φυλλομετρητή. Η αλλαγή σελίδας (π.χ. από το /login στο /dashboard) γίνεται ακαριαία, καθώς η React αναλαμβάνει να καταστρέψει (unmount) τα παλιά συστατικά (components) και να προσαρτήσει (mount) τα νέα στο Εικονικό DOM (Virtual DOM), χωρίς καμία επικοινωνία με τον διακομιστή για λήψη HTML.

Επιπλέον, η δρομολόγηση συνδέθηκε άμεσα με τον Έλεγχο Πρόσβασης Βάσει Ρόλων (RBAC). Με τη χρήση προστατευμένων διαδρομών (Protected Routes), το σύστημα διαβάζει τον ρόλο του συνδεδεμένου χρήστη (π.χ. role === 'cinema_manager') και επιτρέπει την απόδοση (rendering) μόνο των αντίστοιχων συστατικών (όπως το CinemaHalls και CinemaSchedule). Οποιαδήποτε μη εξουσιοδοτημένη πρόσβαση ανακατευθύνεται αυτόματα στην αρχική σελίδα (Catch-all Navigate).

4.6.2 Διαχείριση Κατάστασης και Παρέμβασης (React Hooks)

Η επιχειρησιακή λογική στο Frontend της εφαρμογής είναι εξαιρετικά δυναμική και βασίζεται στη χρήση Συναρτήσεων Άγκιστρων (React Hooks). Τα κυριότερα εργαλεία που αξιοποιήθηκαν είναι τα εξής:

1. Διαχείριση Τοπικής Κατάστασης με το useState: Αποτελεί τον πυρήνα της διαδραστικότητας. Στη διαδικασία κράτησης (BookingInterface), το useState διατηρεί τις επιλεγμένες θέσεις (selectedSeats) και τις ποσότητες των συνοδευτικών προϊόντων (snacks). Η τοπική αυτή αποθήκευση επιτρέπει στον χρήστη να μετακινείται ελεύθερα μεταξύ των βημάτων της κράτησης (Επιλογή Θέσεων -> Κυλικείο -> Πληρωμή) χωρίς να χάνει τα δεδομένα του και χωρίς το Frontend να "βομβαρδίζει" το Backend με ενδιάμεσα, ημιτελή αιτήματα. Παράλληλα, το useState ελέγχει τη διεπαφή (π.χ. activeTab στο Dashboard, εναλλαγή μεταξύ Single και Bulk Mode στον προγραμματισμό).
2. Διαχείριση Παρενεργειών με το useEffect: Χρησιμοποιήθηκε εκτενώς για τον συγχρονισμό του Frontend με εξωτερικά συστήματα. Κατά το φόρτωμα ενός συστατικού (Component Mount), το useEffect αναλαμβάνει να εκτελέσει τις ασύγχρονες HTTP κλήσεις προς το Backend (π.χ. fetchMovies, fetchInitialData) για να αντλήσει τα δεδομένα. Επίσης, χρησιμοποιήθηκε για την παρακολούθηση γεγονότων του παραθύρου (Window Event Listeners), όπως η ανίχνευση κύλισης (scroll progress) στην αρχική σελίδα.
3. Άμεσος Χειρισμός του DOM με το useRef: Μολονότι η React αποθαρρύνει την άμεση επέμβαση στο DOM, σε περιπτώσεις που απαιτούνταν υψηλή απόδοση (performance) χωρίς διαρκείς επανασχεδιάσεις (re-renders), το useRef αποδείχθηκε σωτήριο. Χρησιμοποιήθηκε στον καμβά σωματιδίων (ParticleCanvas) για τη διατήρηση της αναφοράς στο στοιχείο HTML Canvas, καθώς και στην υλοποίηση του οριζόντιου κυλιόμενου μενού ημερομηνιών (Date Selector Carousel), όπου η λειτουργία scrollBy κλήθηκε άμεσα στο DOM node για την επίτευξη ομαλής κύλισης (smooth scroll).

4.6.3 Σχεδιασμός Διεπαφής και Οπτικής Εμπειρίας (UI/UX)

Ο σχεδιασμός του CINEHUB ξέφυγε από τα παραδοσιακά, στατικά βιομηχανικά πρότυπα, ενσωματώνοντας σύγχρονες τεχνικές οπτικής απεικόνισης [15].

Δυναμική Προσαρμογή Χρωμάτων (Ambient Glow): Μία από τις καινοτομίες της εφαρμογής στο επίπεδο του UI είναι η δυναμική προσαρμογή της χρωματικής παλέτας ανάλογα με το περιεχόμενο. Μέσω της React, το σύστημα αναλύει το είδος (Genre) της επιλεγμένης ταινίας (π.χ. Κόκκινο για ταινίες Δράσης, Μπλε για Επιστημονικής Φαντασίας) και με τη βοήθεια του useEffect τροποποιεί δυναμικά τη μεταβλητή CSS (--primary-glow-dynamic) στο ανώτατο επίπεδο (document root). Αυτό δημιουργεί μια μοναδική, εμβυθιστική (immersive) εμπειρία περιήγησης.

Αλγοριθμική Κίνηση με Particle Canvas: Η σελίδα υποδοχής (Landing Page) και η σελίδα εισόδου ενσωματώνουν ένα σύστημα αδρανειακών σωματιδίων (ParticleCanvas). Ο αλγόριθμος, γραμμένος σε καθαρή JavaScript μέσω του Canvas API 2D, δημιουργεί 80 ανεξάρτητα σωματίδια τα οποία κινούνται στον χώρο. Κάθε φορά που δύο σωματίδια πλησιάζουν μεταξύ τους σε απόσταση μικρότερη των 120 pixels, σχεδιάζεται αυτόματα μια γραμμή σύνδεσης. Η ανανέωση της οθόνης γίνεται μέσω της μεθόδου requestAnimationFrame, διασφαλίζοντας απόδοση 60 καρέ ανά δευτερόλεπτο (60 FPS) χωρίς να επιβαρύνεται ο επεξεργαστής της συσκευής.

Διαχείριση Ασύγχρονης Κατάστασης (Loading States & Toast Notifications): Για την αποφυγή σύγχυσης του χρήστη κατά τη διάρκεια ασύγχρονων ενεργειών (όπως η αναμονή επιβεβαίωσης πληρωμής από την τράπεζα), ενσωματώθηκαν σαφείς δείκτες φόρτωσης (loading spinners) και μηνύματα προόδου. Επιπλέον, για τη διαρκή και μη παρεμβατική ενημέρωση του χρήστη, αναπτύχθηκε ένα προσαρμοσμένο σύστημα αναδυόμενων ειδοποιήσεων (ToastContainer), το οποίο διαχειρίζεται τα μηνύματα επιτυχίας ή σφάλματος και τα αφαιρεί αυτόματα από την οθόνη μετά από 3 δευτερόλεπτα.

4.7 Αρχιτεκτονική Επικοινωνίας, Αντικείμενα Μεταφοράς Δεδομένων (DTOs) και Ενθυλάκωση

Κατά τον σχεδιασμό σύγχρονων κατανεμημένων συστημάτων, μία από τις χειρότερες πρακτικές είναι η άμεση έκθεση των οντοτήτων της βάσης δεδομένων (Database Entities) στο επίπεδο της διεπαφής (Frontend) μέσω των REST APIs. Η πρακτική αυτή δημιουργεί στενή σύζευξη (tight coupling) μεταξύ της δομής των δεδομένων και της παρουσίασής τους, ενώ ταυτόχρονα ελλοχεύει σοβαρούς κινδύνους ασφαλείας, καθώς ενδέχεται να διαρρεύσουν ευαίσθητες πληροφορίες (π.χ. κρυπτογραφημένοι κωδικοί) ή μεταδεδομένα της βάσης.

Για την αποτροπή αυτών των φαινομένων, η πλατφόρμα CINEHUB εφαρμόζει το Αρχιτεκτονικό Πρότυπο των Αντικειμένων Μεταφοράς Δεδομένων (Data Transfer Objects - DTOs). Τα DTOs είναι απλές κλάσεις της Java (POJOs) που δεν περιέχουν καμία επιχειρησιακή λογική, αλλά χρησιμεύουν αποκλειστικά για την ενθυλάκωση και μεταφορά δεδομένων μεταξύ των υποσυστημάτων.

Στον πηγαίο κώδικα, η στρατηγική αυτή διακρίνεται ξεκάθαρα στους Ελεγκτές (Controllers):

- Αιτήματα Εισόδου (Request DTOs): Κατά την είσοδο του χρήστη, το σύστημα δεν δέχεται μια ολόκληρη οντότητα User, αλλά το ειδικά διαμορφωμένο "AuthRequest", το οποίο περιέχει αυστηρά μόνο το username και το password. Ομοίως, για τον μαζικό προγραμματισμό αιθουσών χρησιμοποιείται το "BulkScheduleRequest", το οποίο ομαδοποιεί παραμέτρους όπως ημερομηνίες, λίστα ωρών και τιμές, διευκολύνοντας την αποσειριοποίηση (deserialization) του JSON.
- Αιτήματα Εξόδου (Response DTOs) και Επιπεδοποίηση (Flattening): Η σπουδαιότητα των DTOs αναδεικνύεται στην ανάκτηση του ιστορικού κρατήσεων. Η οντότητα Booking στη βάση δεδομένων είναι εξαιρετικά περίπλοκη, καθώς περιέχει συσχετίσεις (@ManyToOne) με τον Χρήστη, την Προβολή, την Ταινία και την Αίθουσα. Αντί να σταλεί όλο αυτό το δέντρο αντικειμένων στο Frontend, ο "BookingController" χρησιμοποιεί τη βοηθητική μέθοδο mapToDto. Η μέθοδος αυτή εξάγει τις απαραίτητες πληροφορίες και δημιουργεί ένα επίπεδο (flat) αντικείμενο "MyBookingResponse", το οποίο περιέχει μόνο τα τελικά δεδομένα (movieTitle, cinemaName, hallName, seats, totalPrice). Η προσέγγιση αυτή μειώνει δραματικά τον όγκο των δεδομένων που μεταφέρονται μέσω του δικτύου (network payload) και βελτιώνει την απόδοση της εφαρμογής.

4.8 Διαχείριση Σφαλμάτων και Συνέπεια Διεπαφών (Exception Handling)

Η ευρωστία ενός RESTful API εξαρτάται σε μεγάλο βαθμό από τον τρόπο που το σύστημα αντιδρά σε μη αναμενόμενες καταστάσεις ή εσφαλμένα δεδομένα εισόδου. Το CINEHUB υιοθετεί μια δομημένη προσέγγιση στη διαχείριση εξαιρέσεων (Exception Handling), διασφαλίζοντας ότι το Frontend λαμβάνει πάντα κατανοητά μηνύματα και τους κατάλληλους κωδικούς κατάστασης HTTP (HTTP Status Codes) [23, 24].

- Έλεγχος Πόρων (404 Not Found): Σε περιπτώσεις όπου ζητείται η επεξεργασία μιας ανύπαρκτης οντότητας (π.χ. προσπάθεια προγραμματισμού σε διαγραμμένη αίθουσα), το σύστημα ρίχνει την εξαίρεση ResponseStatusException(HttpStatus.NOT_FOUND), ενημερώνοντας τον πελάτη ότι ο πόρος δεν είναι διαθέσιμος.
- Έλεγχος Δικαιωμάτων Ιδιοκτησίας (403 Forbidden): Η πλατφόρμα υλοποιεί αυστηρούς ελέγχους μέσω της μεθόδου getOwnedCinema. Αν ένας Διαχειριστής Κινηματογράφου προσπαθήσει να τροποποιήσει μια αίθουσα που ανήκει σε ανταγωνιστικό κινηματογράφο, το Backend το ανιχνεύει, απορρίπτει την ενέργεια και επιστρέφει κωδικό 403 Forbidden, αποτρέποντας επιθέσεις IDOR (Insecure Direct Object Reference).
- Συγκρούσεις Επιχειρησιακής Λογικής (409 Conflict): Στον προγραμματισμό προβολών (CinemaController), όταν το σύστημα ανιχνεύει επικάλυψη ωραρίων στην ίδια αίθουσα (conflict validation), η διαδικασία ανακόπτεται άμεσα. Το Backend επιστρέφει το σφάλμα HttpStatus.CONFLICT μαζί με ένα επεξηγηματικό μήνυμα, το οποίο το Frontend μετατρέπει σε αναδυόμενη ειδοποίηση (Toast Notification) για την καθοδήγηση του διαχειριστή.

4.9 Διαχείριση Συναλλαγών και Ταυτοχρονισμός (Transaction Management & Concurrency)

Η μεγαλύτερη πρόκληση σε ένα σύστημα ηλεκτρονικών κρατήσεων είναι η διαχείριση του Ταυτοχρονισμού (Concurrency). Όταν εκατοντάδες χρήστες προσπαθούν να κλείσουν εισιτήρια για την ίδια δημοφιλή πρεμιέρα, υπάρχει υψηλός κίνδυνος δύο πελάτες να επιλέξουν την ίδια θέση (π.χ. "A1") την ίδια ακριβώς στιγμή (Race Condition).

Για την αποτροπή αυτού του φαινομένου, η αρχιτεκτονική του Spring Boot αξιοποιεί τον μηχανισμό των Συναλλαγών (Transactions).

- Ιδιότητες ACID: Η διαδικασία της κράτησης στο επίπεδο του Service (BookingService) εκτελείται υπό το πρίσμα των ιδιοτήτων ACID (Ατομικότητα, Συνέπεια, Απομόνωση, Ανθεκτικότητα). Ολόκληρη η διαδικασία – από τον έλεγχο της διαθεσιμότητας των θέσεων, τον υπολογισμό του κόστους, έως την τελική εγγραφή στη βάση – αντιμετωπίζεται ως μία ενιαία και αδιαίρετη μονάδα εργασίας (Atomic Operation).
- Αναίρεση (Rollback): Εάν σε οποιοδήποτε σημείο της διαδικασίας προκύψει σφάλμα (π.χ. διακοπή δικτύου, απόρριψη πληρωμής, ή αν η θέση δεσμεύτηκε κλάσματα του δευτερολέπτου νωρίτερα από άλλο νήμα εκτέλεσης), το σύστημα εκτελεί αυτόματη αναίρεση (Rollback). Καμία αλλαγή δεν αποθηκεύεται οριστικά στη βάση δεδομένων, αποτρέποντας τη δημιουργία "ορφανών" ή διπλοκρατημένων εγγραφών και διασφαλίζοντας την απόλυτη ακεραιότητα των δεδομένων.

4.10 Στρατηγική Ελέγχου Ποιότητας και Δοκιμών Λογισμικού (Προτεινόμενο Πλαίσιο)

Η διασφάλιση ποιότητας (Quality Assurance - QA) και ο συστηματικός έλεγχος λογισμικού αποτελούν αναπόσπαστο κομμάτι του Κύκλου Ζωής Ανάπτυξης Λογισμικού (SDLC). Σε ένα σύστημα ηλεκτρονικών κρατήσεων όπως το CINEHUB, όπου διαχειρίζονται ευαίσθητα δεδομένα χρηστών, συναλλαγές και ταυτόχρονες προσβάσεις (concurrency), η απουσία δοκιμών ενδέχεται να οδηγήσει σε κρίσιμες αστοχίες στην παραγωγή, όπως διπλοκρατήσεις ή εσφαλμένους υπολογισμούς εσόδων.

Στο πλαίσιο της παρούσας εργασίας, ο έλεγχος της ορθής λειτουργίας πραγματοποιήθηκε κατά κύριο λόγο μέσω συστηματικών χειροκίνητων λειτουργικών δοκιμών (manual functional testing) των κρίσιμων ροών του συστήματος: εγγραφή και σύνδεση χρήστη, αναζήτηση ταινίας, ολοκλήρωση κράτησης, παραγωγή εισιτηρίου και μαζικός προγραμματισμός προβολών. Παράλληλα, σχεδιάστηκε η προτεινόμενη στρατηγική αυτοματοποιημένου ελέγχου που παρουσιάζεται στη συνέχεια, η οποία βασίζεται στην «Πυραμίδα των Δοκιμών» (Test Pyramid) και δίνει έμφαση στις Μοναδιαίες Δοκιμές (Unit Tests) και στις Δοκιμές Ενοποίησης (Integration Tests). Η πλήρης υλοποίησή της αποτελεί την πρώτη προτεραιότητα για τη μετάβαση του συστήματος σε περιβάλλον παραγωγής.

4.10.1 Προτεινόμενες Μοναδιαίες Δοκιμές (Unit Testing) με JUnit 5 και Mockito

Το πρώτο και θεμελιώδες επίπεδο ελέγχου αφορά τις Μοναδιαίες Δοκιμές (Unit Testing). Στόχος τους είναι η απομόνωση και ο έλεγχος της μικρότερης δυνατής μονάδας κώδικα (συνήθως μιας μεθόδου εντός μιας κλάσης), προκειμένου να επαληθευτεί ότι η επιχειρησιακή λογική λειτουργεί ορθά, ανεξάρτητα από εξωτερικά συστήματα (όπως η βάση δεδομένων ή το δίκτυο).

Στο οικοσύστημα του Spring Boot, το βιομηχανικό πρότυπο για τη συγγραφή αυτών των δοκιμών είναι ο συνδυασμός των βιβλιοθηκών **JUnit 5** και **Mockito**.

Η χρήση της βιβλιοθήκης Mockito είναι κρίσιμη για την απομόνωση των Services. Κατά τον έλεγχο της επιχειρησιακής λογικής (π.χ. στο BookingService), το σύστημα δεν πρέπει να αλληλεπιδρά με την πραγματική σχεσιακή βάση δεδομένων, καθώς κάτι τέτοιο θα προκαλούσε καθυστερήσεις, θα απαιτούσε πολύπλοκο στήσιμο (setup) και θα αλλοίωνε τα πραγματικά δεδομένα. Αντιθέτως, το Mockito επιτρέπει τη δημιουργία "εικονικών αντικειμένων" (Mocks) τα οποία αντικαθιστούν τα πραγματικά Repositories.

Ενδεικτικό σενάριο δοκιμής στο CINEHUB: Για τον έλεγχο της μεθόδου υπολογισμού κόστους κρατήσεων (totalPrice), δημιουργείται ένα Mock του ScreeningRepository. Το Mock ρυθμίζεται (stubbing) ώστε, όταν του ζητηθεί μια προβολή, να επιστρέφει ένα εικονικό αντικείμενο Screening με προκαθορισμένη τιμή εισιτηρίου (π.χ. 10€). Η δοκιμή JUnit καλεί τη μέθοδο κράτησης περνώντας 3 επιλεγμένες θέσεις και 2 προϊόντα κυλικείου. Στο τέλος, μέσω της εντολής assertEquals, το σύστημα επαληθεύει (assert) ότι το τελικό ποσό που υπολόγισε η Java ταυτίζεται με το μαθηματικά αναμενόμενο. Αν ο προγραμματιστής κατά λάθος αλλάξει τον αλγόριθμο στον κώδικα, η δοκιμή θα αποτύχει (fail), ειδοποιώντας τον άμεσα για το σφάλμα (regression).

4.10.2 Προτεινόμενες Δοκιμές Ενοποίησης (Integration Testing) στο Επίπεδο των REST APIs

Ενώ οι μοναδιαίες δοκιμές πιστοποιούν την ορθότητα της μεμονωμένης λογικής, δεν εγγυώνται ότι τα διαφορετικά υποσυστήματα συνεργάζονται σωστά μεταξύ τους. Σε αυτό το σημείο υπεισέρχονται οι Δοκιμές Ενοποίησης (Integration Tests).

Στην πλατφόρμα CINEHUB, οι δοκιμές ενοποίησης προτείνεται να επικεντρωθούν στο επίπεδο των Ελεγκτών (Controllers), ελέγχοντας τη ροή από το HTTP Request μέχρι την αποθήκευση στη βάση δεδομένων και την επιστροφή του HTTP Response.

Η προτεινόμενη υλοποίηση βασίζεται στο annotation @SpringBootTest σε συνδυασμό με το MockMvc. Το MockMvc αποτελεί ένα πανίσχυρο εργαλείο του Spring, καθώς επιτρέπει την προσομοίωση HTTP αιτημάτων (GET, POST, PUT, DELETE) χωρίς την ανάγκη εκκίνησης της πραγματικού web server (της ο Tomcat).

Ενδεικτικό σενάριο δοκιμής στο CINEHUB: Μια χαρακτηριστική δοκιμή ενοποίησης αφορά το Endpoint Μαζικού Προγραμματισμού (/api/cinemas/schedule/bulk). Το MockMvc αποστέλλει ένα εικονικό POST αίτημα με ένα JSON φορτίο (payload) που περιέχει επικαλυπτόμενες ημερομηνίες (conflict scenario). Η δοκιμή διασχίζει ολόκληρο το σύστημα: το JSON αποσειριοποιείται (deserialized) στο DTO BulkScheduleRequest, περνάει από το CinemaController, ελέγχεται από το Service, και απορρίπτεται. Η δοκιμή ολοκληρώνεται επιτυχώς μόνο εάν το MockMvc επιβεβαιώσει ότι το Backend επέστρεψε κωδικό κατάστασης 409 CONFLICT και δεν έγινε καμία εγγραφή στη βάση. Με αυτόν τον τρόπο, επαληθεύεται ολόκληρη η αλυσίδα επικοινωνίας.

4.10.3 Προτεινόμενες Δοκιμές Διεπαφής Χρήστη (Frontend Testing)

Η διασφάλιση ποιότητας εκτείνεται και στο επίπεδο του πελάτη (Frontend). Στο περιβάλλον της React, ο έλεγχος των Components υλοποιείται συνήθως με τη χρήση των εργαλείων **Jest** και **React Testing Library**.

Σε αντίθεση με παλαιότερες τεχνικές που έλεγχαν την εσωτερική κατάσταση (state) των συστατικών, η React Testing Library προσεγγίζει τη δοκιμή από την οπτική γωνία του τελικού χρήστη. Ελέγχεται δηλαδή το τι ακριβώς αποδίδεται (renders) στο Εικονικό DOM (Virtual DOM).

Ενδεικτικό σενάριο δοκιμής στο CINEHUB: Στο συστατικό BookingInterface (Διαδραστικό Πλάνο Θέσεων), οι δοκιμές μπορούν να προσομοιώσουν το κλικ του χρήστη σε μια διαθέσιμη θέση. Η δοκιμή επαληθεύει ότι η συγκεκριμένη θέση αποκτά άμεσα την κλάση CSS «selected» (αλλάζοντας χρώμα) και ότι το ποσό στο πεδίο Total Price ενημερώνεται δυναμικά. Ομοίως, προσομοιώνεται η είσοδος εσφαλμένων δεδομένων στη φόρμα πληρωμής της πιστωτικής κάρτας, επαληθεύοντας ότι τα μηνύματα σφάλματος (validation errors) εμφανίζονται εγκαίρως στην οθόνη του χρήστη, βελτιώνοντας την εμπειρία χρήσης (UX) και αποτρέποντας την αποστολή άκυρων δεδομένων στον διακομιστή.

4.11 Προτεινόμενες Στρατηγικές Ανάπτυξης, Containerization (Docker) και Συνεχής Ενσωμάτωση (CI/CD)

Η ανάπτυξη ενός πληροφοριακού συστήματος δεν ολοκληρώνεται με τη συγγραφή του πηγαίου κώδικα. Η μετάβαση από το περιβάλλον ανάπτυξης (development environment) στο περιβάλλον παραγωγής (production environment) αποτελεί μια από τις μεγαλύτερες προκλήσεις στη Μηχανική Λογισμικού. Παραδοσιακά, το φαινόμενο "δουλεύει στον υπολογιστή μου" (it works on my machine) προκαλούσε αμέτρητα προβλήματα, καθώς οι διαφορές στις εκδόσεις των λειτουργικών συστημάτων, της Java ή της Node.js οδηγούσαν σε αστοχίες κατά την ανάπτυξη στον διακομιστή (deployment).

Στην τρέχουσα έκδοση, η πλατφόρμα CINEHUB εκτελείται σε τοπικό περιβάλλον ανάπτυξης (embedded Tomcat για το Spring Boot και Vite dev server για τη React). Για τη μετάβαση σε περιβάλλον παραγωγής προτείνεται η υιοθέτηση των σύγχρονων πρακτικών του DevOps, με αξιοποίηση τεχνολογιών Containerization και αυτοματοποιημένων ροών εργασίας (CI/CD), οι οποίες αναλύονται στη συνέχεια.

4.11.1 Προτεινόμενη Αρχιτεκτονική Containerization με χρήση Docker

Η τεχνολογία Docker φέρνει επανάσταση στον τρόπο με τον οποίο το λογισμικό πακετάρεται και διανέμεται. Σε αντίθεση με τις παραδοσιακές Εικονικές Μηχανές (Virtual Machines - VMs) που απαιτούν την εγκατάσταση ενός πλήρους λειτουργικού συστήματος (Guest OS) δεσμεύοντας τεράστιους πόρους, τα Containers (υποδοχείς) μοιράζονται τον πυρήνα (kernel) του συστήματος φιλοξενίας. Αυτό τα καθιστά εξαιρετικά ελαφριά, γρήγορα στην εκκίνηση και απόλυτα απομονωμένα.

Για την πλατφόρμα CINEHUB, η προτεινόμενη προσέγγιση του Dockerization διαχωρίζεται στα εξής επίπεδα:

Επίπεδο Backend (Spring Boot): Η εφαρμογή Java πακετάρεται μέσω ενός αρχείου ρυθμίσεων (Dockerfile). Αρχικά, χρησιμοποιείται μια ελαφριά βασική εικόνα (π.χ. eclipse-temurin:17-jdk-alpine) που περιέχει μόνο τα απολύτως απαραίτητα στοιχεία του Java Development Kit. Στη συνέχεια, το παραγόμενο εκτελέσιμο αρχείο (.jar) αντιγράφεται μέσα στο container. Η εντολή εκκίνησης αναλαμβάνει την εκτέλεση της εφαρμογής, εκθέτοντας αποκλειστικά τη θύρα λειτουργίας του API (π.χ. θύρα 8080). Αυτό διασφαλίζει ότι το Backend θα εκτελεστεί με την ίδια ακριβώς συμπεριφορά σε οποιονδήποτε cloud provider.

Επίπεδο Frontend (React.js) και Multi-stage Builds: Για το επίπεδο παρουσίασης, ακολουθείται η προηγμένη στρατηγική των Πολλαπλών Σταδίων κατασκευής (Multi-stage builds). Στο πρώτο στάδιο, χρησιμοποιείται μια εικόνα Node.js για να κατεβάσει τις εξαρτήσεις (npm install) και να μεταγλωττίσει (build) τον κώδικα της React σε στατικά αρχεία HTML, CSS και παραλλαγμένη (minified) JavaScript. Στο δεύτερο στάδιο, τα παραγόμενα στατικά αρχεία μεταφέρονται σε μια νέα, πολύ ελαφριά εικόνα που τρέχει τον εξυπηρετητή Nginx. Ο Nginx δεν λειτουργεί μόνο ως web server για την εξυπηρέτηση των αρχείων στον πελάτη, αλλά ρυθμίζεται και ως Reverse Proxy (αντίστροφος διαμεσολαβητής), δρομολογώντας με ασφάλεια τα αιτήματα του API προς το Backend container.

4.11.2 Προτεινόμενη Ενορχήστρωση με Docker Compose

Εφόσον το σύστημα αποτελείται από πολλαπλά αυτόνομα υποσυστήματα (Frontend, Backend, Σχεσιακή Βάση Δεδομένων), η διαχείρισή τους ως μεμονωμένα containers καθίσταται πολύπλοκη. Για τον συντονισμό τους, χρησιμοποιείται το εργαλείο Docker Compose.

Μέσω ενός ενιαίου αρχείου (docker-compose.yml), ορίζονται όλες οι υπηρεσίες (services), τα δίκτυα επικοινωνίας (bridge networks) και οι μόνιμοι χώροι αποθήκευσης δεδομένων (volumes). Με αυτόν τον τρόπο, το CINEHUB μπορεί να εκκινηθεί πλήρως σε οποιονδήποτε διακομιστή με μία μόνο εντολή (docker-compose up -d). Το Docker Compose αναλαμβάνει να εκκινήσει πρώτα τη βάση δεδομένων, να περιμένει να τεθεί σε λειτουργία, και έπειτα να εκκινήσει το Backend και το Frontend, διασφαλίζοντας την ορθή αλληλουχία (dependency resolution).

4.11.3 Συνεχής Ενσωμάτωση και Συνεχής Παράδοση (CI/CD)

Στα σύγχρονα περιβάλλοντα ανάπτυξης λογισμικού, η χειροκίνητη μεταφορά κώδικα στον διακομιστή θεωρείται επισφαλής και παρωχημένη. Για τον λόγο αυτό, η θεωρητική αρχιτεκτονική του CINEHUB περιλαμβάνει τον σχεδιασμό μιας αυτοματοποιημένης ροής εργασιών Συνεχούς Ενσωμάτωσης και Συνεχούς Παράδοσης (Continuous Integration / Continuous Deployment - CI/CD), αξιοποιώντας πλατφόρμες όπως το GitHub Actions ή το GitLab CI.

Η ροή εργασιών (Pipeline) δομείται στα εξής αυτοματοποιημένα στάδια:

1. Στάδιο Ενσωμάτωσης (Continuous Integration): Κάθε φορά που ο προγραμματιστής "ανεβάζει" (push) νέο κώδικα στο κεντρικό αποθετήριο (repository), ενεργοποιείται αυτόματα η ροή ελέγχου. Το σύστημα CI κατεβάζει τον κώδικα, τον μεταγλωττίζει και εκτελεί το σύνολο των Μοναδιαίων Δοκιμών (Unit Tests) και των Δοκιμών Ενοποίησης (Integration Tests) που αναλύθηκαν στο προηγούμενο κεφάλαιο. Εάν έστω και μία δοκιμή αποτύχει, η διαδικασία διακόπτεται (build failure), αποτρέποντας την προώθηση ελαττωματικού κώδικα στην παραγωγή.
2. Στάδιο Κατασκευής (Build and Register): Εφόσον όλοι οι έλεγχοι περάσουν επιτυχώς, ο μηχανισμός CI/CD προχωρά στην αυτόματη κατασκευή των νέων Docker Images για το Frontend και το Backend. Στη συνέχεια, τα images αυτά "μαρκάρονται" με τον αριθμό της τρέχουσας έκδοσης (version tagging) και ανεβαίνουν (push) σε ένα ασφαλές μητρώο καταγραφής (Container Registry), όπως το Docker Hub.
3. Στάδιο Παράδοσης (Continuous Deployment): Στο τελικό βήμα, το σύστημα επικοινωνεί μέσω κρυπτογραφημένων κλειδιών (SSH) με τον διακομιστή παραγωγής (Cloud Server - π.χ. AWS, DigitalOcean). Δίνει την εντολή να κατεβούν οι νέες εκδόσεις των υποδοχέων και να αντικαταστήσουν τις παλιές, με ελάχιστο έως μηδενικό χρόνο διακοπής των υπηρεσιών (Zero-Downtime Deployment).

Μέσω της παραπάνω προτεινόμενης αρχιτεκτονικής DevOps, η πλατφόρμα CINEHUB θα μπορούσε να επιτύχει υψηλή αξιοπιστία, ταχύτητα στην κυκλοφορία νέων χαρακτηριστικών και σταθερότητα, καθιστάμενη μια εφαρμογή πλήρως προετοιμασμένη για απαιτητικά περιβάλλοντα παραγωγής.

4.12 Εφαρμογή Σχεδιαστικών Προτύπων Λογισμικού (Software Design Patterns)

Η ανάπτυξη πολύπλοκων συστημάτων λογισμικού, όπως η πλατφόρμα CINEHUB, απαιτεί την υιοθέτηση δοκιμασμένων λύσεων σε επαναλαμβανόμενα αρχιτεκτονικά προβλήματα. Οι λύσεις αυτές ονομάζονται Σχεδιαστικά Πρότυπα (Design Patterns) και η ορθή εφαρμογή τους διασφαλίζει την ευελιξία, τη συντηρησιμότητα και την επεκτασιμότητα του πηγαίου κώδικα. Κατά την υλοποίηση του Backend στο οικοσύστημα του Spring Boot, αξιοποιήθηκαν ευρέως τα ακόλουθα σχεδιαστικά πρότυπα.

4.12.1 Πρότυπο Αντιστροφής Ελέγχου και Έγχυσης Εξαρτήσεων (IoC & Dependency Injection)

Το θεμελιώδες πρότυπο πάνω στο οποίο εδράζεται ολόκληρο το Spring Framework είναι η Έγχυση Εξαρτήσεων (Dependency Injection - DI), μια εξειδικευμένη μορφή της Αντιστροφής Ελέγχου (Inversion of Control - IoC). Στον παραδοσιακό προγραμματισμό, κάθε κλάση είναι υπεύθυνη για τη δημιουργία (instantiation) των αντικειμένων από τα οποία εξαρτάται, χρησιμοποιώντας τη λέξη-κλειδί "new". Αυτό οδηγεί σε στενή σύζευξη (tight coupling), καθιστώντας τον κώδικα δυσνόητο και δύσκολο στη δοκιμή (testing).

Στο CINEHUB, ο έλεγχος αντιστρέφεται: το ίδιο το Spring Container (IoC Container) αναλαμβάνει να δημιουργήσει και να διαχειριστεί τον κύκλο ζωής των αντικειμένων (Beans). Στους ελεγκτές της εφαρμογής (όπως στον AuthController και στον CinemaController), η έγχυση των εξαρτήσεων υλοποιείται μέσω του κατασκευαστή (Constructor Injection). Με τη χρήση του annotation "@RequiredArgsConstructor" της βιβλιοθήκης Lombok, η Java παράγει αυτόματα τον κατασκευαστή για όλα τα τελικά (final) πεδία, εξασφαλίζοντας ότι ο ελεγκτής δεν μπορεί να δημιουργηθεί εάν δεν του παρασχεθούν πρώτα οι απαραίτητες υπηρεσίες (Repositories και Services). Η προσέγγιση αυτή προάγει τη χαλαρή σύζευξη (loose coupling) και διευκολύνει την αντικατάσταση των εξαρτήσεων με εικονικά αντικείμενα (mocks) κατά τις μοναδιαίες δοκιμές (unit tests).

4.12.2 Πρότυπο Αντικειμένου Πρόσβασης Δεδομένων (DAO / Repository Pattern)

Το επίπεδο επικοινωνίας με τη σχεσιακή βάση δεδομένων υλοποιήθηκε μέσω του προτύπου Repository, το οποίο αποτελεί μια σύγχρονη εξέλιξη του παραδοσιακού Data Access Object (DAO) Pattern. Ο σκοπός αυτού του προτύπου είναι να απομονώσει πλήρως την επιχειρησιακή λογική (Service Layer) από τις ιδιαιτερότητες της βάσης δεδομένων (SQL ερωτήματα, συνδέσεις, transactions).

Μέσω του Spring Data JPA, δημιουργήθηκαν διεπαφές (Interfaces) όπως το UserRepository και το BookingRepository. Οι διεπαφές αυτές κληρονομούν (extend) έτοιμες λειτουργίες από το JpaRepository. Με αυτόν τον τρόπο, το σύστημα εκτελεί πολύπλοκες αναζητήσεις (όπως το "findByHall_Cinema_Id") αποκλειστικά μέσω της ονοματολογίας των μεθόδων. Το μοτίβο Repository αποκρύπτει την πολυπλοκότητα του Object-Relational Mapping (ORM) και παρέχει μια καθαρή, αντικειμενοστρεφή διεπαφή στο Service Layer (Data Hiding).

4.12.3 Πρότυπο Κατασκευαστή (Builder Pattern)

Κατά τη δημιουργία πολύπλοκων οντοτήτων (Entities), όπως η κράτηση (Booking) ή η προβολή (Screening), η χρήση πολλαπλών κατασκευαστών (constructors) ή αλυσιδωτών κλήσεων "set" μπορεί να οδηγήσει σε δυσνόητο και επιρρεπή σε σφάλματα κώδικα (Telescoping Constructor Anti-Pattern).

Για την επίλυση αυτού του ζητήματος, ενσωματώθηκε το Creational Design Pattern "Builder". Χρησιμοποιώντας το annotation "@Builder" της βιβλιοθήκης Lombok σε επίπεδο κλάσης, παράγεται αυτόματα ένας εσωτερικός μηχανισμός δόμησης αντικειμένων. Η τεχνική αυτή, όπως εφαρμόζεται κατά την εγγραφή ενός νέου χρήστη ή τη δημιουργία μιας προβολής, επιτρέπει τη συνένωση μεθόδων (method chaining) για τη ρύθμιση των πεδίων με τρόπο ευανάγνωστο (fluent API style), χωρίς να απαιτείται η δημιουργία ενδιάμεσων μεταβλητών, διασφαλίζοντας παράλληλα την αμεταβλητότητα (immutability) των δεδομένων κατά τη στιγμή της δημιουργίας του αντικειμένου.

4.13 Προτεινόμενη Τεκμηρίωση Διεπαφών και Προδιαγραφές REST API (OpenAPI / Swagger)

Στις σύγχρονες κατανεμημένες αρχιτεκτονικές, όπως το μοντέλο Πελάτη-Εξυπηρετητή (Client-Server) που υιοθετεί η πλατφόρμα CINEHUB, τα υποσυστήματα του Frontend και του Backend αναπτύσσονται και λειτουργούν απολύτως ανεξάρτητα. Αυτός ο αυστηρός διαχωρισμός απαιτεί την ύπαρξη ενός ξεκάθαρου "συμβολαίου επικοινωνίας" (communication contract) μεταξύ τους. Οι προγραμματιστές που αναπτύσσουν τη διεπαφή στη React πρέπει να γνωρίζουν με απόλυτη ακρίβεια τα διαθέσιμα τελικά σημεία (endpoints) του διακομιστή, τις αποδεκτές μεθόδους HTTP (GET, POST, PUT, DELETE), τη δομή των Αντικειμένων Μεταφοράς Δεδομένων (DTOs) και τους πιθανούς κωδικούς σφαλμάτων.

Στο παρελθόν, η τεκμηρίωση των APIs γινόταν χειροκίνητα σε στατικά έγγραφα, μια πρακτική χρονοβόρα και επιρρεπής σε σφάλματα, καθώς η τεκμηρίωση έχανε γρήγορα τον συγχρονισμό της με τον πραγματικό κώδικα. Για την επίλυση αυτού του προβλήματος προτείνεται η υιοθέτηση του βιομηχανικού προτύπου OpenAPI Specification (OAS), το οποίο υλοποιείται οπτικά μέσω της σουίτας εργαλείων Swagger.

4.13.1 Αυτοματοποιημένη Παραγωγή Τεκμηρίωσης με Springdoc OpenAPI

Για την ενσωμάτωση του προτύπου στο επίπεδο του διακομιστή (Spring Boot), προτείνεται η αξιοποίηση της βιβλιοθήκης springdoc-openapi. Η συγκεκριμένη βιβλιοθήκη λειτουργεί δυναμικά κατά τον χρόνο εκτέλεσης (runtime). Αναλύει (scans) μέσω αντανάκλασης (reflection) όλες τις κλάσεις που φέρουν τη σήμανση @RestController. Στη συνέχεια, διαβάζει τα annotations των μεθόδων (όπως @GetMapping και @PostMapping), τις παραμέτρους (όπως @RequestBody και @PathVariable) και τη δομή των επιστρεφόμενων αντικειμένων (Response Entities).

Το αποτέλεσμα αυτής της αυτοματοποιημένης ανάλυσης είναι η παραγωγή ενός δομημένου αρχείου JSON ή YAML, το οποίο περιγράφει με αυστηρή μαθηματική τυποποίηση ολόκληρο το API της πλατφόρμας. Κάθε φορά που ένας προγραμματιστής προσθέτει ένα νέο πεδίο σε μια οντότητα (π.χ. στην κράτηση) ή ένα νέο endpoint στον κώδικα της Java, η τεκμηρίωση ενημερώνεται αυτόματα. Αυτό εξασφαλίζει την απόλυτη ταύτιση κώδικα και προδιαγραφών (Single Source of Truth), εξαλείφοντας τον κίνδυνο παρωχημένης τεκμηρίωσης.

4.13.2 Διαδραστική Διεπαφή Δοκιμών (Swagger UI)

Η πραγματική αξία του προτύπου OpenAPI αναδεικνύεται μέσω του Swagger UI. Πρόκειται για μια ενσωματωμένη, γραφική διεπαφή χρήστη η οποία παράγεται αυτόματα και προσπελάζεται μέσω του φυλλομετρητή (συνήθως στο μονοπάτι /swagger-ui.html του εξυπηρετητή).

Μέσω αυτής της διεπαφής, οι προγραμματιστές του Frontend μπορούν όχι μόνο να διαβάσουν τις προδιαγραφές, αλλά και να αλληλεπιδράσουν ζωντανά με το Backend. Το Swagger UI παρέχει οπτικές φόρμες για την εισαγωγή δεδομένων και την εκτέλεση πραγματικών HTTP κλήσεων. Για παράδειγμα, ο προγραμματιστής μπορεί να εκτελέσει το endpoint /api/auth/login, να εισάγει δοκιμαστικά διαπιστευτήρια, να λάβει το JSON Web Token (JWT) και να το εισάγει στον ειδικό μηχανισμό εξουσιοδότησης (Authorize button) της σελίδας.

Από εκείνη τη στιγμή, το Swagger ενσωματώνει το Token στο HTTP Header όλων των επόμενων αιτημάτων, επιτρέποντας τη ζωντανή δοκιμή προστατευμένων endpoints (όπως η δημιουργία μιας αίθουσας κινηματογράφου ή η ανάκτηση του ιστορικού κρατήσεων) χωρίς την ανάγκη συγγραφής ούτε μίας γραμμής κώδικα στο Frontend ή τη χρήση εξωτερικών εργαλείων όπως το Postman. Η προσέγγιση αυτή επιταχύνει κατακόρυφα τον κύκλο ανάπτυξης (development cycle), μειώνει τις ασυνεννοησίες μεταξύ των ομάδων και εγγυάται ότι οι κλήσεις από τη React θα ταιριάζουν απόλυτα με τις απαιτήσεις του Spring Boot.

4.14 Κανονικοποίηση Σχεσιακής Βάσης Δεδομένων (Database Normalization)

Η αποδοτικότητα και η ακεραιότητα ενός πληροφοριακού συστήματος εξαρτώνται άμεσα από τον σχεδιασμό της βάσης δεδομένων του. Όταν τα δεδομένα αποθηκεύονται χωρίς συγκεκριμένη δομή, ελλοχεύει ο κίνδυνος εμφάνισης πλεονασμών (data redundancy) και ανωμαλιών κατά την εισαγωγή, διαγραφή ή ενημέρωση των εγγραφών (modification anomalies). Για την αποτροπή αυτών των φαινομένων, ο σχεδιασμός του σχεσιακού σχήματος της πλατφόρμας CINEHUB βασίστηκε στους αυστηρούς κανόνες της Κανονικοποίησης (Database Normalization), φτάνοντας έως την Τρίτη Κανονική Μορφή (3NF).

4.14.1 Πρώτη Κανονική Μορφή (1NF - First Normal Form)

> Ο θεμελιώδης κανόνας της Πρώτης Κανονικής Μορφής επιβάλλει την ατομικότητα των δεδομένων (atomicity). Κάθε πεδίο (στήλη) ενός πίνακα πρέπει να περιέχει μία και μόνο μία τιμή, και δεν επιτρέπονται επαναλαμβανόμενες ομάδες ή πίνακες μέσα σε ένα κελί.

Στο CINEHUB, η αρχή αυτή τηρήθηκε με αυστηρότητα, ιδιαίτερα στον χειρισμό των συλλογών (Collections) του Spring Boot. Για παράδειγμα, στην οντότητα "Booking", ένας χρήστης μπορεί να κλείσει πολλαπλές θέσεις (π.χ. A1, A2, A3). Αντί οι θέσεις αυτές να αποθηκευτούν ως μια ενιαία συμβολοσειρά διαχωρισμένη με κόμματα μέσα στον πίνακα των κρατήσεων (γεγονός που θα παραβίαζε την 1NF και θα καθιστούσε αδύνατη την αναζήτηση με SQL), το σύστημα αξιοποιεί το annotation @ElementCollection του JPA. Η προσέγγιση αυτή δημιουργεί αυτόματα έναν ξεχωριστό, πλήρως κανονικοποιημένο πίνακα "booking_seats". Ο πίνακας αυτός συνδέεται με την κράτηση μέσω ενός ξένου κλειδιού (booking_id) και περιέχει μία εγγραφή για κάθε μεμονωμένη θέση, εξασφαλίζοντας την ατομικότητα της πληροφορίας.

4.14.2 Δεύτερη Κανονική Μορφή (2NF - Second Normal Form)

Για να βρίσκεται μια βάση στη Δεύτερη Κανονική Μορφή, πρέπει αρχικά να ικανοποιεί την 1NF και, επιπλέον, να μην υπάρχουν μερικές εξαρτήσεις (partial dependencies). Αυτό σημαίνει ότι κάθε μη-κλειδί χαρακτηριστικό (non-key attribute) πρέπει να εξαρτάται από ολόκληρο το πρωτεύον κλειδί (Primary Key) και όχι μόνο από ένα τμήμα του, κάτι που αφορά κυρίως πίνακες με σύνθετα κλειδιά.

Στην αρχιτεκτονική του CINEHUB, η συμμόρφωση με την 2NF επιτεύχθηκε καθολικά μέσω της χρήσης Τεχνητών Πρωτευόντων Κλειδιών (Surrogate Keys). Σε κάθε οντότητα (User, Movie, Cinema, Screening), έχει οριστεί ένα μοναδικό ακέραιο αναγνωριστικό (id τύπου Long) το οποίο παράγεται αυτόματα από τη βάση δεδομένων (GenerationType.IDENTITY). Εφόσον τα πρωτεύοντα κλειδιά αποτελούνται πάντα από μία μόνο στήλη, καθίσταται μαθηματικά αδύνατη η ύπαρξη μερικών εξαρτήσεων, ικανοποιώντας πλήρως τον κανόνα της 2NF.

4.14.3 Τρίτη Κανονική Μορφή (3NF - Third Normal Form)

Η Τρίτη Κανονική Μορφή αποτελεί το χρυσό πρότυπο για τις περισσότερες συναλλακτικές βάσεις δεδομένων (OLTP). Για την ικανοποίησή της, ο πίνακας πρέπει να βρίσκεται σε 2NF και να μην περιέχει μεταβατικές εξαρτήσεις (transitive dependencies). Μια μεταβατική εξάρτηση προκύπτει όταν ένα μη-κλειδί χαρακτηριστικό εξαρτάται από ένα άλλο μη-κλειδί χαρακτηριστικό, αντί να εξαρτάται αποκλειστικά από το πρωτεύον κλειδί.

Το σχεσιακό μοντέλο του CINEHUB έχει σχεδιαστεί προσεκτικά για την εξάλειψη αυτών των εξαρτήσεων. Ένα χαρακτηριστικό παράδειγμα είναι η σχέση μεταξύ των Κινηματογράφων (Cinema) και των Αιθουσών (Hall). Εάν τα στοιχεία της αίθουσας και τα στοιχεία του κινηματογράφου (όπως η πόλη και η διεύθυνση) συνυπήρχαν στον ίδιο πίνακα, η αλλαγή της διεύθυνσης ενός κινηματογράφου θα απαιτούσε την ενημέρωση εκατοντάδων εγγραφών (μία για κάθε προβολή ή αίθουσα).

Στη θέση αυτής της πρακτικής, το σύστημα διαχωρίζει τις οντότητες. Ο πίνακας "Screening" (Προβολή) δεν αποθηκεύει τον τίτλο της ταινίας ή το όνομα του κινηματογράφου. Αντιθέτως, αποθηκεύει μόνο τα Ξένα Κλειδιά (Foreign Keys: movie_id, hall_id). Εάν ο διαχειριστής αλλάξει τον τίτλο της ταινίας ή τη διάρκεια της, η αλλαγή γίνεται αποκλειστικά σε ένα σημείο (στον πίνακα "Movie"). Οι αλλαγές αυτές αντανακλώνται αυτόματα σε όλες τις προβολές και τις κρατήσεις μέσω των σχεσιακών συνδέσεων (JOINs), διασφαλίζοντας την απόλυτη ακεραιότητα των δεδομένων και αποφεύγοντας τις ανωμαλίες ενημέρωσης (update anomalies).

4.15 Δυσκολίες και Προβλήματα κατά την Ανάπτυξη

Η ανάπτυξη ενός πλήρους συστήματος Full-Stack δεν αποτελεί μια γραμμική διαδικασία. Κατά τη διάρκεια της υλοποίησης του CINEHUB ανέκυψαν συγκεκριμένα τεχνικά εμπόδια, η αντιμετώπιση των οποίων συνέβαλε καθοριστικά στη διαμόρφωση της τελικής αρχιτεκτονικής. Στην ενότητα αυτή καταγράφονται τα σημαντικότερα από αυτά, μαζί με τις λύσεις που υιοθετήθηκαν και τους περιορισμούς που παραμένουν.

4.15.1 Πολιτική Ίδιας Προέλευσης και Ρυθμίσεις CORS

Κατά την πρώτη διασύνδεση του Frontend με το Backend, όλα τα αιτήματα απορρίπτονταν από τον φυλλομετρητή. Η αιτία εντοπίστηκε στην Πολιτική Ίδιας Προέλευσης (Same-Origin Policy): το επίπεδο παρουσίασης εκτελείται στη θύρα ανάπτυξης του Vite, ενώ ο εξυπηρετητής στη θύρα 8080, με αποτέλεσμα κάθε κλήση να θεωρείται διασταυρούμενης προέλευσης (cross-origin) και να προηγείται ενός αιτήματος ελέγχου (preflight OPTIONS). Η επίλυση απαίτησε τη ρητή δήλωση ενός CorsConfigurationSource στην κλάση SecurityConfig, με καθορισμό των επιτρεπόμενων προελεύσεων, μεθόδων και κεφαλίδων, καθώς και την ενεργοποίηση του αντίστοιχου φίλτρου στην αλυσίδα ασφαλείας του Spring Security.

4.15.2 Ταυτοχρονισμός κατά την Κράτηση Θέσεων

Το δυσκολότερο πρόβλημα του συστήματος υπήρξε ο ταυτοχρονισμός. Ο έλεγχος διαθεσιμότητας μιας θέσης και η αποθήκευση της κράτησης αποτελούν δύο διακριτά βήματα (check-then-act), και επομένως δύο νήματα εκτέλεσης μπορούν θεωρητικά να περάσουν και τα δύο τον έλεγχο πριν προλάβει κάποιο να γράψει στη βάση. Η χρήση της σήμανσης @Transactional στο επίπεδο του BookingService εγγυάται την ατομικότητα της συναλλαγής και την αυτόματη αναίρεση (rollback) σε περίπτωση σφάλματος, δεν αποτρέπει όμως από μόνη της το φαινόμενο της συνθήκης ανταγωνισμού (race condition) υπό υψηλό φόρτο. Για την πλήρη θωράκιση του συστήματος σε περιβάλλον παραγωγής απαιτείται είτε αισιόδοξο κλείδωμα (optimistic locking) με πεδίο έκδοσης στην οντότητα Screening, είτε απαισιόδοξο κλείδωμα (pessimistic locking) της εγγραφής της προβολής, είτε — ως τελευταία γραμμή άμυνας — ένας περιορισμός μοναδικότητας στο επίπεδο της βάσης για τον συνδυασμό προβολής και αριθμού θέσης.

4.15.3 Αποθήκευση του Διακριτικού Πρόσβασης (Token Storage)

Η επιλογή του τόπου αποθήκευσης του JWT στο επίπεδο του πελάτη αποτέλεσε συνειδητό συμβιβασμό. Το LocalStorage προσφέρει απλότητα και άμεση πρόσβαση από τη βιβλιοθήκη Axios, εκθέτει όμως το διακριτικό σε ενδεχόμενη επίθεση Cross-Site Scripting (XSS). Η εναλλακτική λύση ενός cookie με τη σήμανση HttpOnly θωρακίζει το διακριτικό απέναντι σε XSS, εισάγει ωστόσο την ανάγκη προστασίας απέναντι σε επιθέσεις Cross-Site Request Forgery (CSRF). Στο πλαίσιο της παρούσας εργασίας επιλέχθηκε η πρώτη προσέγγιση, με ρητή αναγνώριση του περιορισμού και πρόταση μετάβασης στη δεύτερη κατά την εμπορική αξιοποίηση του συστήματος.

4.15.4 Συμβατότητα Φυλλομετρητών στη Φωνητική Αναζήτηση

Η λειτουργία της φωνητικής αναζήτησης υλοποιήθηκε πάνω στο Web Speech API. Κατά τις δοκιμές διαπιστώθηκε ότι η διεπαφή SpeechRecognition δεν υποστηρίζεται ομοιόμορφα από όλους τους φυλλομετρητές και είναι διαθέσιμη κυρίως στις μηχανές τύπου Chromium μέσω του προθέματος webkitSpeechRecognition. Για να μην αστοχεί η εφαρμογή σε μη υποστηριζόμενα περιβάλλοντα, υλοποιήθηκε έλεγχος ύπαρξης της διεπαφής πριν την ενεργοποίηση του μικροφώνου και ομαλή υποβάθμιση (graceful degradation) στην κλασική αναζήτηση μέσω πληκτρολογίου, συνοδευόμενη από ενημερωτικό μήνυμα προς τον χρήστη.

4.15.5 Παραγωγή Ψηφιακού Εισιτηρίου στο Επίπεδο του Πελάτη

Η απόφαση να παράγεται το ψηφιακό εισιτήριο τοπικά στον φυλλομετρητή, μέσω των βιβλιοθηκών jsPDF και react-qr-code, απαλλάσσει τον εξυπηρετητή από υπολογιστικό φόρτο, δημιούργησε όμως δύο πρακτικές δυσκολίες. Πρώτον, ο κώδικας QR αποδίδεται ως διανυσματικό στοιχείο SVG μέσα στο DOM και απαιτεί μετατροπή σε εικόνα ράστερ πριν ενσωματωθεί στο έγγραφο PDF. Δεύτερον, οι προεπιλεγμένες γραμματοσειρές της jsPDF δεν καλύπτουν πλήρως το ελληνικό αλφάβητο, γεγονός που επιβάλλει είτε την ενσωμάτωση προσαρμοσμένης γραμματοσειράς είτε τη διατήρηση των πεδίων του εισιτηρίου στη λατινική γραφή.

4.15.6 Διαχείριση του Σχήματος της Βάσης Δεδομένων

Κατά τη φάση της ανάπτυξης, το σχήμα της βάσης παράγεται αυτόματα από το Hibernate μέσω της ρύθμισης ddl-auto=update. Η επιλογή αυτή επιταχύνει σημαντικά τους κύκλους ανάπτυξης, καθώς κάθε μεταβολή σε μια οντότητα αντικατοπτρίζεται άμεσα στους πίνακες. Είναι όμως ακατάλληλη για περιβάλλον παραγωγής, διότι δεν διαγράφει παρωχημένες στήλες, δεν υποστηρίζει ελεγχόμενη επαναφορά και δεν παρέχει ιστορικό μεταβολών. Ως ώριμη λύση προτείνεται η υιοθέτηση ενός εργαλείου διαχείρισης μεταναστεύσεων σχήματος (schema migrations), όπως το Flyway ή το Liquibase.

4.15.7 Αποδοτικότητα Ερωτημάτων και το Πρόβλημα N+1

Η οντότητα Booking συνδέεται μεταβατικά με τις οντότητες Screening, Movie, Hall και Cinema. Η αφελής ανάκτηση του ιστορικού κρατήσεων ενός χρήστη οδηγούσε στην εκτέλεση ενός ερωτήματος για τη λίστα των κρατήσεων και ενός επιπλέον ερωτήματος για κάθε συσχέτιση κάθε εγγραφής — του γνωστού προβλήματος N+1. Η απενεργοποίηση του μηχανισμού open-in-view, σε συνδυασμό με τη ρητή χαρτογράφηση σε επίπεδα αντικείμενα μεταφοράς δεδομένων (mapToDto), περιόρισε το φαινόμενο και κατέστησε προβλέψιμο τον αριθμό των ερωτημάτων προς τη βάση.

4.15.8 Καθολικός Χειρισμός Εξαιρέσεων

Η ύπαρξη ενός καθολικού χειριστή εξαιρέσεων (GlobalExceptionHandler) με τη σήμανση @ControllerAdvice εξασφαλίζει ότι καμία μη αναμενόμενη εξαίρεση δεν διαρρέει προς τον πελάτη ως ίχνος στοίβας (stack trace). Ένας χειριστής όμως που δεσμεύει τον γενικό τύπο Exception υπερκαλύπτει και τις σκόπιμες εξαιρέσεις τύπου ResponseStatusException, μετατρέποντας τους σημασιολογικά ορθούς κωδικούς 403, 404 και 409 σε έναν γενικό κωδικό 400. Η ορθή πρακτική, η οποία και προτείνεται, είναι ο καθολικός χειριστής να εξαιρεί ρητά τις εξαιρέσεις που φέρουν ήδη κωδικό κατάστασης, ώστε να διατηρείται η σημασιολογία των αποκρίσεων του REST API που περιγράφηκε στην ενότητα 4.8.

5. ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

5.1 Τεχνική Παρουσίαση Λειτουργίας (Developer Perspective)

Από την πλευρά της ανάπτυξης λογισμικού, η εφαρμογή CINEHUB λειτουργεί ως ένα κατανεμημένο σύστημα όπου το Frontend (React) λειτουργεί ως ο «εγκέφαλος» της διεπαφής και το Backend (Spring Boot) ως ο «εγγυητής» της επιχειρησιακής λογικής και της ασφάλειας. Η αρχιτεκτονική επιτρέπει την πλήρη διαχείριση της κατάστασης (state management) σε πραγματικό χρόνο, κάτι που είναι εμφανές στη διαδικασία κράτησης.

Κάθε ενέργεια του χρήστη, από την επιλογή μιας θέσης έως την ολοκλήρωση της πληρωμής, επικυρώνεται (validated) στο Backend. Αυτό διασφαλίζει ότι τα δεδομένα παραμένουν συνεπή, αποτρέποντας κρίσιμα σφάλματα όπως οι διπλοκρατήσεις (double bookings) ή η πρόσβαση σε δεδομένα άλλων χρηστών. Η επικοινωνία βασίζεται σε ασύγχρονα αιτήματα, επιτρέποντας στην εφαρμογή να παραμένει «ζωντανή» και αποκρινόμενη χωρίς να απαιτείται η πλήρης επαναφόρτωση της σελίδας.

5.2 Ανάλυση και Λειτουργία Αλγορίθμων

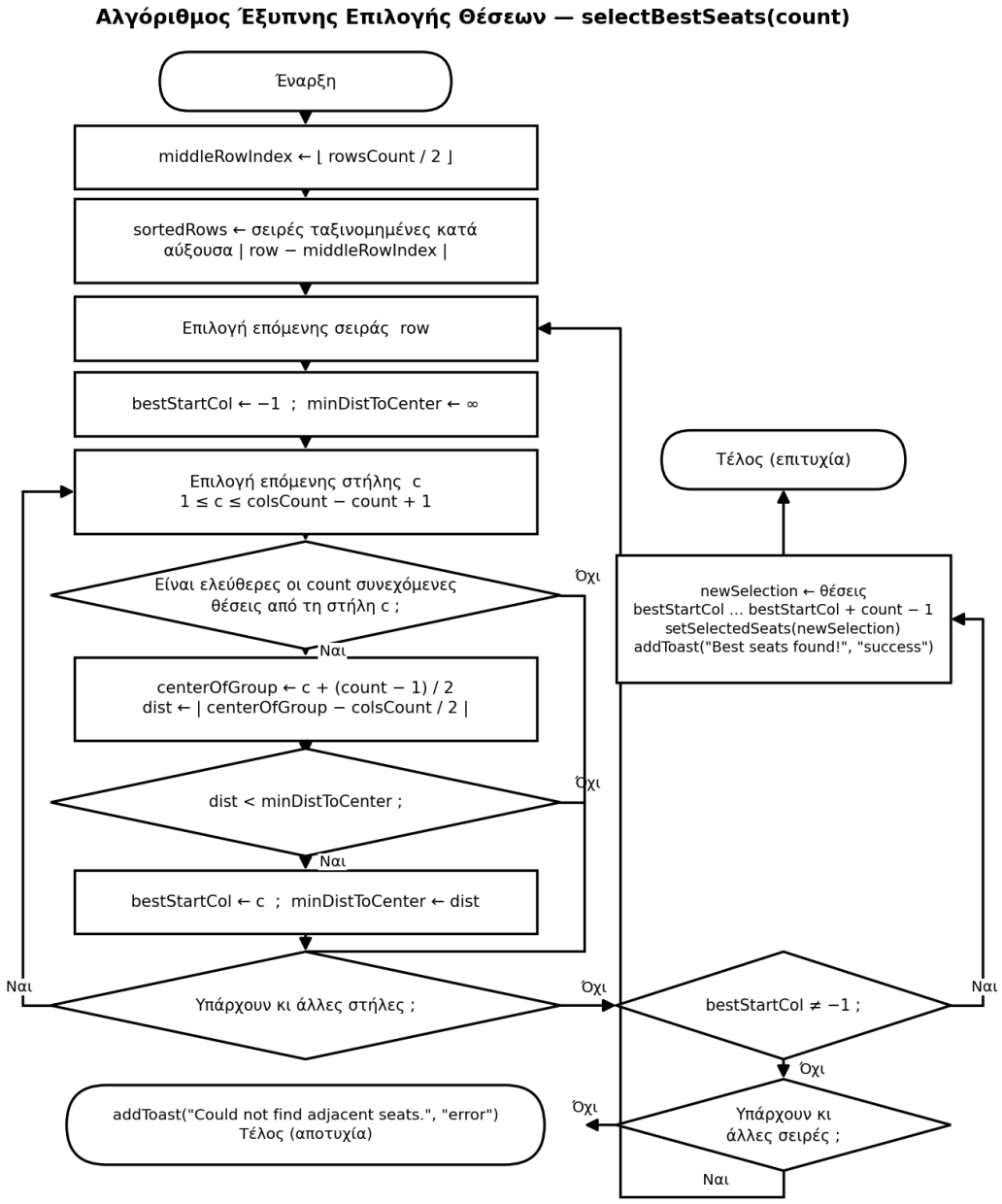
Η «νοημοσύνη» του CINEHUB πηγάζει από εξειδικευμένους αλγορίθμους που αυτοματοποιούν διαδικασίες οι οποίες σε παλαιότερα συστήματα απαιτούσαν χειροκίνητη παρέμβαση.

5.2.1 Αλγόριθμος Έξυπνης Επιλογής Θέσεων (Smart Seat Selection)

Ο αλγόριθμος selectBestSeats αναπτύχθηκε στο επίπεδο του Frontend για να διευκολύνει τον χρήστη στην εύρεση των ιδανικών διαθέσιμων θέσεων βάσει γεωμετρικών και στατιστικών κριτηρίων. Η λογική του αλγορίθμου ακολουθεί τα εξής βήματα:

1. **Προσδιορισμός Κέντρου (Vertical Priority):** Ο αλγόριθμος εντοπίζει τον δείκτη της μεσαίας σειράς της αίθουσας, η οποία θεωρείται η βέλτιστη απόσταση από την οθόνη.
2. **Ταξινόμηση Σειρών:** Οι σειρές ταξινομούνται με βάση την απόλυτη απόστασή τους από τη μεσαία σειρά (πρώτα οι κεντρικές, μετά οι πίσω και τελευταίες οι μπροστινές).
3. **Αναζήτηση Συνεχόμενων Μπλοκ:** Σε κάθε σειρά, ο αλγόριθμος αναζητά ένα συνεχόμενο μπλοκ διαθέσιμων θέσεων που αντιστοιχεί στον αριθμό των εισιτηρίων που επιθυμεί ο χρήστης.

4. **Οριζόντιο Κεντράρισμα (Horizontal Priority):** Εάν βρεθούν πολλαπλά διαθέσιμα μπλοκ στην ίδια σειρά, επιλέγεται αυτό που βρίσκεται πλησιέστερα στο κέντρο της σειράς (οριζόντια), υπολογίζοντας την απόσταση από τη μεσαία στήλη της αίθουσας.



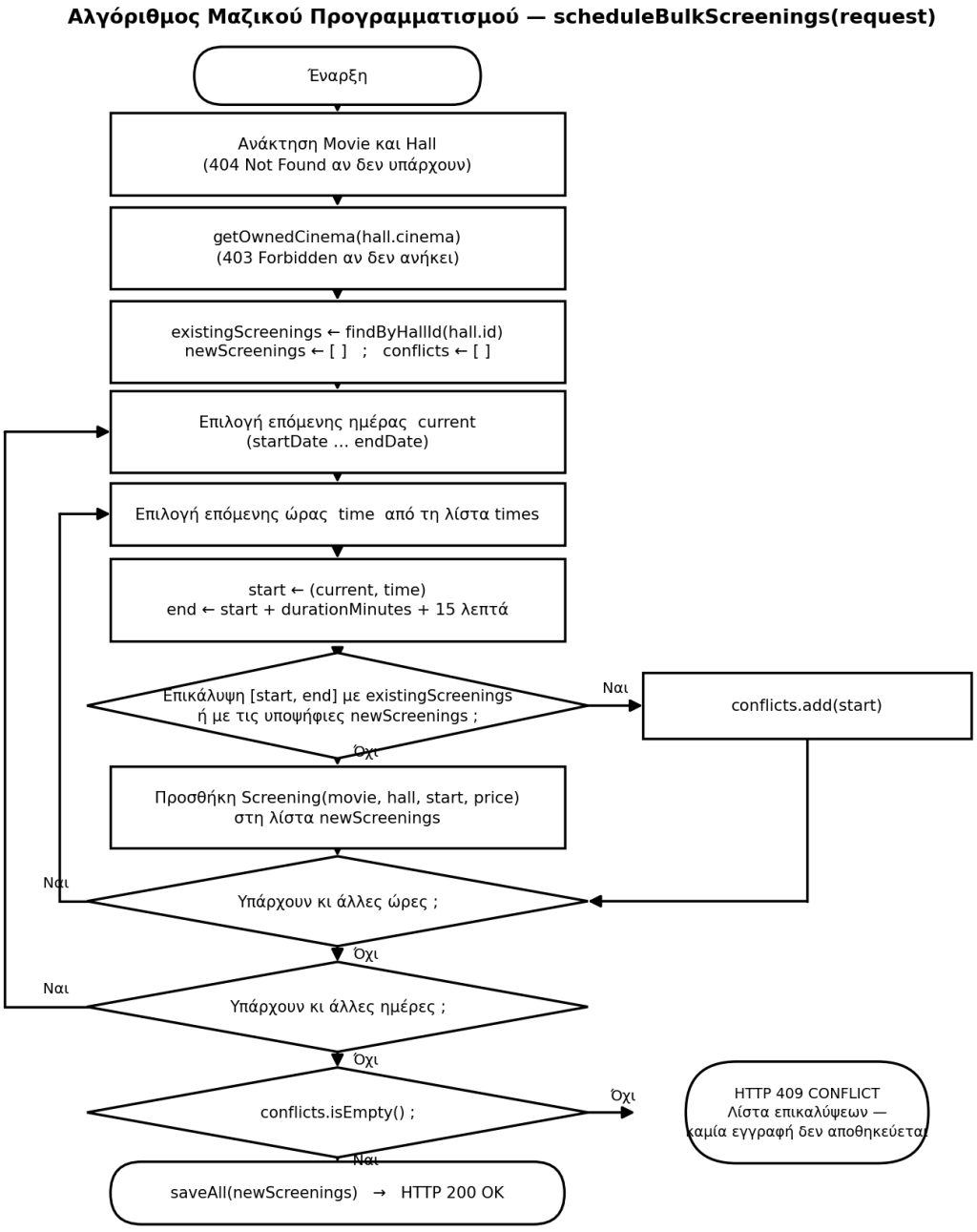
Εικόνα 5.1: Διάγραμμα ροής του αλγορίθμου Έξυπνης Επιλογής Θέσεων (selectBestSeats).

5.2.2 Αλγόριθμος Μαζικού Προγραμματισμού (Bulk Scheduling Logic)

Στο Backend, ο μηχανισμός scheduleBulkScreenings επιτρέπει στους διαχειριστές να προγραμματίζουν ολόκληρες εβδομάδες προβολών με μία κίνηση. Ο αλγόριθμος λειτουργεί ως εξής:

- **Επαναληπτική Επεξεργασία (Looping):** Εκτελεί έναν βρόχο από την ημερομηνία έναρξης έως την ημερομηνία λήξης.
- **Χρονικός Συνδυασμός:** Για κάθε ημέρα, δημιουργεί προβολές για όλες τις επιλεγμένες ώρες.
- **Δυναμική Σύνδεση Οντοτήτων:** Για κάθε συνδυασμό, δημιουργείται μια νέα εγγραφή στη βάση δεδομένων, συνδέοντας την ταινία με την αίθουσα και την τιμή.
- **Conflict Validation:** Πριν την αποθήκευση, το σύστημα εκτελεί έλεγχο επικαλύψεων για κάθε υποψήφια προβολή, τόσο απέναντι στις ήδη αποθηκευμένες προβολές της αίθουσας όσο και απέναντι στις υπόλοιπες προβολές της ίδιας παρτίδας, διασφαλίζοντας ότι η αίθουσα είναι ελεύθερη κατά το διάστημα "Ώρα Έναρξης + Διάρκεια Ταινίας + Χρόνος Καθαρισμού". Εφόσον εντοπιστεί έστω και μία σύγκρουση, η διαδικασία ανακόπτεται και δεν αποθηκεύεται καμία εγγραφή (λογική all-or-nothing), ενώ επιστρέφεται κωδικός 409 CONFLICT με τη λίστα των προβληματικών ωρών.

Η συνολική αρχιτεκτονική και η ροή επικύρωσης (Conflict Validation) του αλγορίθμου αποτυπώνονται σχηματικά στο παρακάτω Διάγραμμα Ροής (Εικόνα 5.2):



Εικόνα 5.2: Διάγραμμα ροής του αλγορίθμου Μαζικού Προγραμματισμού και του ελέγχου επικαλύψεων (Conflict Validation).

5.2.3 Ανάλυση Υπολογιστικής Πολυπλοκότητας (Big-O Notation)

Η αποτελεσματικότητα των αλγορίθμων που ενσωματώνονται σε ένα πληροφοριακό σύστημα δεν κρίνεται μόνο από την ορθότητα των αποτελεσμάτων τους, αλλά και από τον τρόπο με τον οποίο κλιμακώνονται (scalability) καθώς αυξάνεται ο όγκος των δεδομένων. Στην Επιστήμη των Υπολογιστών, η αξιολόγηση της αποδοτικότητας πραγματοποιείται μέσω της Ασυμπτωτικής Ανάλυσης και του συμβολισμού Big-O, ο οποίος περιγράφει το άνω φράγμα της Χρονικής (Time Complexity) και Χωρικής Πολυπλοκότητας (Space Complexity) στη χειρότερη δυνατή περίπτωση (worst-case scenario).

Για την πλατφόρμα CINEHUB, πραγματοποιείται εκτενής θεωρητική ανάλυση στους δύο βασικότερους αλγόριθμους του συστήματος: την Έξυπνη Επιλογή Θέσεων (Frontend) και τον Μαζικό Προγραμματισμό με έλεγχο επικαλύψεων (Backend).

Ανάλυση Αλγορίθμου: Έξυπνη Επιλογή Θέσεων (Smart Seat Selection)

Ο αλγόριθμος "selectBestSeats" λειτουργεί στο επίπεδο του πελάτη (React) και αναλαμβάνει να βρει τις καλύτερες συνεχόμενες θέσεις. Για την ανάλυση, ορίζουμε τις εξής μεταβλητές:

- R: Το συνολικό πλήθος των σειρών της αίθουσας (π.χ. 10).
- C: Το συνολικό πλήθος των στηλών / θέσεων ανά σειρά (π.χ. 15).
- K: Το πλήθος των συνεχόμενων εισιτηρίων που ζητά ο χρήστης (συνήθως K << C).

Η εκτέλεση του αλγορίθμου διαχωρίζεται σε δύο φάσεις:

Φάση 1η - Ταξινόμηση Σειρών: Ο αλγόριθμος υπολογίζει τη μεσαία σειρά, μια πράξη σταθερού χρόνου O(1). Έπειτα, ταξινομεί τον πίνακα των σειρών (R) με κριτήριο την απόλυτη απόσταση από το κέντρο. Η ταξινόμηση ενός πίνακα μήκους R απαιτεί στη χειρότερη περίπτωση χρόνο O(R log R).

Φάση 2η - Σάρωση και Αναζήτηση: Ο αλγόριθμος διατρέχει τις R ταξινομημένες σειρές. Για κάθε σειρά, εξετάζει όλες τις πιθανές αρχικές στήλες. Ο μέγιστος αριθμός εκκινήσεων σε μια σειρά είναι (C - K + 1). Για κάθε πιθανή εκκίνηση, πραγματοποιεί έναν εσωτερικό βρόχο μεγέθους K για να ελέγξει αν οι θέσεις είναι ελεύθερες στον πίνακα των δεσμευμένων θέσεων. Συνεπώς, η πολυπλοκότητα αυτού του ένθετου βρόχου είναι O(R * (C - K + 1) * K).

Συνολική Χρονική Πολυπλοκότητα (Time Complexity): Η συνολική χρονική πολυπλοκότητα διαμορφώνεται ως O(R log R + R * C * K). Δεδομένου ότι σε πραγματικές συνθήκες ο αριθμός των ζητούμενων θέσεων K είναι μια πολύ μικρή σταθερά (π.χ. 2 έως 4), ο αλγόριθμος προσεγγίζει γραμμικό χρόνο O(R * C) ως προς το σύνολο των θέσεων της αίθουσας. Αυτό σημαίνει ότι ο αλγόριθμος εκτελείται ακαριαία στον φυλλομετρητή του πελάτη, χωρίς να προκαλεί «πάγωμα» (blocking) στο κύριο νήμα εκτέλεσης (main thread) της JavaScript.

Χωρική Πολυπλοκότητα (Space Complexity): Απαιτείται μνήμη O(R) για τη δημιουργία του ταξινομημένου αντιγράφου των σειρών και επιπλέον O(K) για την αποθήκευση της τελικής επιλογής των θέσεων. Επομένως, η συνολική χωρική πολυπλοκότητα ανέρχεται σε O(R + K), καθιστώντας τον αλγόριθμο εξαιρετικά αποδοτικό σε κατανάλωση μνήμης.

Ανάλυση Αλγορίθμου: Μαζικός Προγραμματισμός (Bulk Scheduling Validation)

Ο αλγόριθμος "scheduleBulkScreenings" εκτελείται στο επίπεδο του διακομιστή (Spring Boot) και έχει διπλή αποστολή: τη δημιουργία των προβολών και την αποτροπή επικαλύψεων (Conflict Validation). Για την ανάλυση, ορίζουμε τις μεταβλητές:

- D: Ο αριθμός των ημερών του εύρους προγραμματισμού (Start Date έως End Date).
- T: Ο αριθμός των διακριτών ωρών προβολής ανά ημέρα (π.χ. 18:00, 20:30, 23:00).
- E: Ο αριθμός των ήδη υπαρχουσών προβολών (Existing Screenings) στη συγκεκριμένη αίθουσα, οι οποίες ανασύρονται από τη βάση δεδομένων.

Κατά την εκτέλεση, το σύστημα χρησιμοποιεί έναν εξωτερικό βρόχο για τη διάσχιση των ημερών (D) και έναν εσωτερικό για τη διάσχιση των ωρών (T), προκύπτοντας συνολικά D * T επαναλήψεις. Σε κάθε επανάληψη, το σύστημα πρέπει να ελέγξει αν η νέα προβολή επικαλύπτεται με κάποια από τις ήδη υπάρχουσες.

Στην τρέχουσα υλοποίηση (Linear Conflict Check), η σύγκριση γίνεται ελέγχοντας τα διαστήματα [Έναρξη, Λήξη] απέναντι σε κάθε στοιχείο της λίστας των υπαρχουσών προβολών (E), αλλά και απέναντι στις υποψήφιες προβολές της ίδιας παρτίδας που έχουν ήδη γίνει αποδεκτές. Επομένως, για κάθε μία από τις D * T νέες προβολές, εκτελούνται έως E + D * T συγκρίσεις.

Συνολική Χρονική Πολυπλοκότητα (Time Complexity): Ο χρόνος εκτέλεσης στη χειρότερη περίπτωση είναι O(D * T * (E + D * T)), με κυρίαρχο όρο τον O(D * T * E) όταν το πλήθος των ήδη υπαρχουσών προβολών υπερβαίνει το μέγεθος της παρτίδας. Παρόλο που η πολυπλοκότητα είναι υπερ-γραμμική ως προς τον συνδυασμό των μεταβλητών, ο αλγόριθμος παραμένει αποδοτικός στην πράξη επειδή το Backend εκτελεί τις συγκρίσεις απευθείας στη μνήμη (In-Memory Processing μέσω του Java Stream API) και όχι πραγματοποιώντας ξεχωριστά ερωτήματα (queries) στη βάση δεδομένων για κάθε προβολή ξεχωριστά, αποφεύγοντας το διαβόητο πρόβλημα N+1.

Χωρική Πολυπλοκότητα (Space Complexity): Η χωρική πολυπλοκότητα είναι O(E) για τη φόρτωση των υπαρχουσών προβολών στην κύρια μνήμη RAM του διακομιστή, συν O(D * T) για τη δημιουργία της λίστας (newScreenings) που περιέχει τις νέες προβολές πριν αυτές αποθηκευτούν μαζικά στη βάση με την εντολή saveAll(). Συνολικά, η μνήμη που απαιτείται είναι O(E + D * T), γεγονός που διασφαλίζει ότι η διαδικασία δεν θα προκαλέσει εξάντληση μνήμης (OutOfMemoryError), ακόμη και για προγραμματισμό ολόκληρων μηνών.

5.3 Μηχανισμοί Ασφάλειας και Ταυτοποίησης (JWT Flow)

Η ασφάλεια της εφαρμογής βασίζεται στο πρωτόκολλο JWT (JSON Web Token), το οποίο επιτρέπει μια αρχιτεκτονική "stateless" (χωρίς κατάσταση στον διακομιστή).

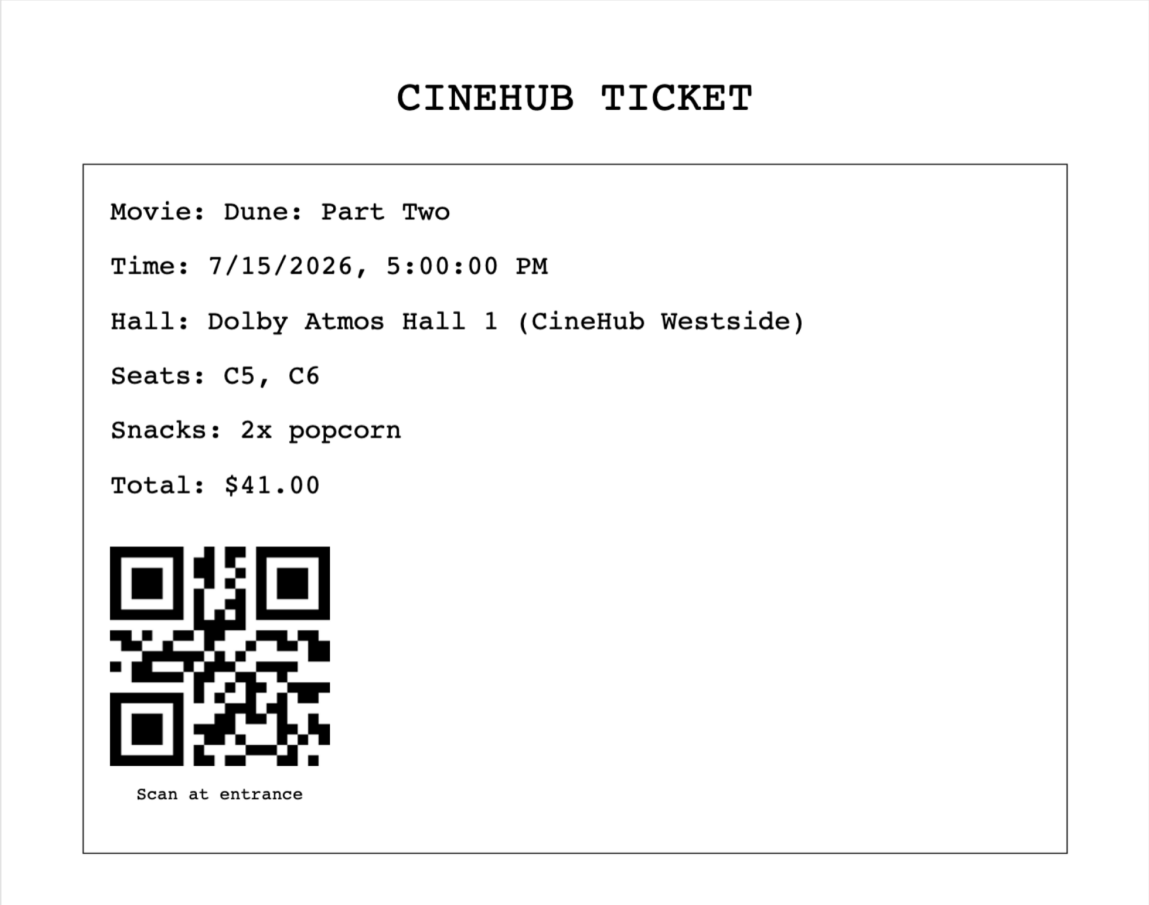
Η ροή εργασίας ακολουθεί τα εξής στάδια:

1. **Authentication:** Ο χρήστης υποβάλλει τα διαπιστευτήριά του, τα οποία επικυρώνονται από το Spring Security.
2. **Token Generation:** Παράγεται ένα μοναδικό, ψηφιακά υπογεγραμμένο token που περιέχει την ταυτότητα και τους ρόλους του χρήστη.
3. **Authorization:** Το token αποθηκεύεται στο Frontend και αποστέλλεται σε κάθε αίτημα. Το Backend το αποκωδικοποιεί και επιτρέπει την πρόσβαση σε προστατευμένες λειτουργίες, όπως η αγορά εισιτηρίων, μόνο αν ο χρήστης διαθέτει τον απαιτούμενο ρόλο (π.χ. ROLE_CUSTOMER).

5.4 Αυτοματοποιημένη Παραγωγή Εισιτηρίων και QR Codes

Μετά την επιτυχή ολοκλήρωση μιας κράτησης, το σύστημα ενεργοποιεί αυτοματοποιημένες διαδικασίες παραγωγής παραστατικών:

- **Παραγωγή PDF:** Χρησιμοποιώντας τη βιβλιοθήκη jsPDF, το σύστημα δημιουργεί δυναμικά ένα έγγραφο που περιέχει όλα τα στοιχεία της κράτησης (ταινία, αίθουσα, θέσεις, snacks).
- **Ενσωμάτωση QR Code:** Παράγεται ένας μοναδικός κώδικας QR που περιέχει κρυπτογραφημένο το ID της κράτησης. Ο μηχανισμός αυτός επιτρέπει την άμεση επικύρωση της εισόδου στο φυσικό χώρο του κινηματογράφου μέσω μιας συνοδευτικής εφαρμογής σάρωσης (scanner app).



Εικόνα 5.3: Το παραγόμενο ψηφιακό εισιτήριο (PDF) με τον ενσωματωμένο κώδικα QR.

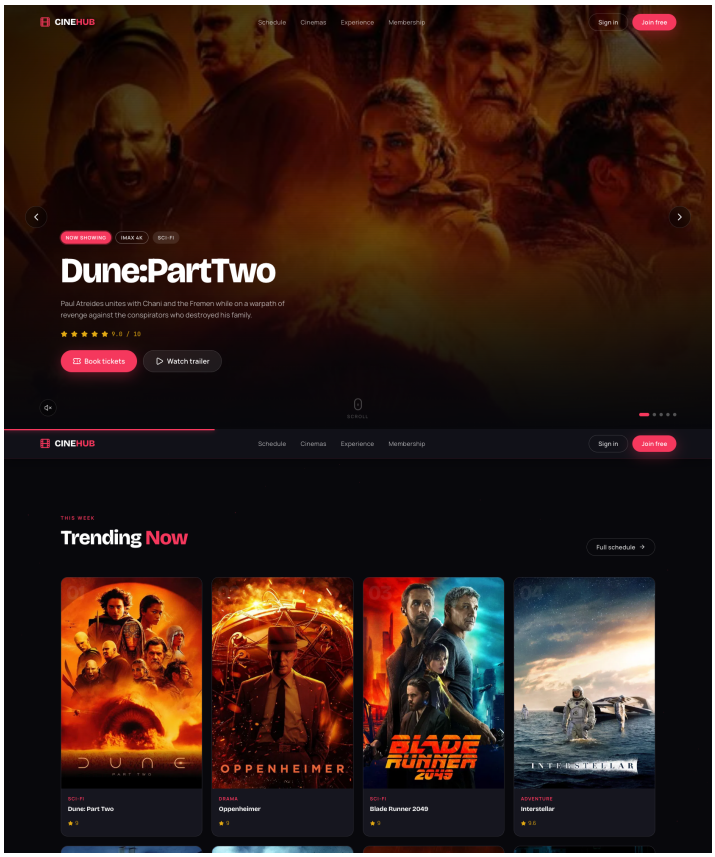
6. ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΤΗ

6.1 Εισαγωγή στη Διεπαφή Χρήστη

Η σχεδίαση της διεπαφής χρήστη (User Interface - UI) της πλατφόρμας CINEHUB βασίστηκε σε σύγχρονες αρχές της Αλληλεπίδρασης Ανθρώπου-Υπολογιστή (HCI), με στόχο την ελαχιστοποίηση του γνωστικού φόρτου (cognitive load) και τη μεγιστοποίηση της ευχρηστίας (usability). Το σύστημα προσφέρει διαφορετικές, προσαρμοσμένες όψεις (views) ανάλογα με τον ρόλο του χρήστη (Πελάτης, Διαχειριστής Κινηματογράφου, Διαχειριστής Συστήματος), εξασφαλίζοντας ότι ο κάθε χρήστης έχει άμεση πρόσβαση μόνο στα εργαλεία που τον αφορούν, προστατεύοντας παράλληλα την ακεραιότητα του συστήματος.

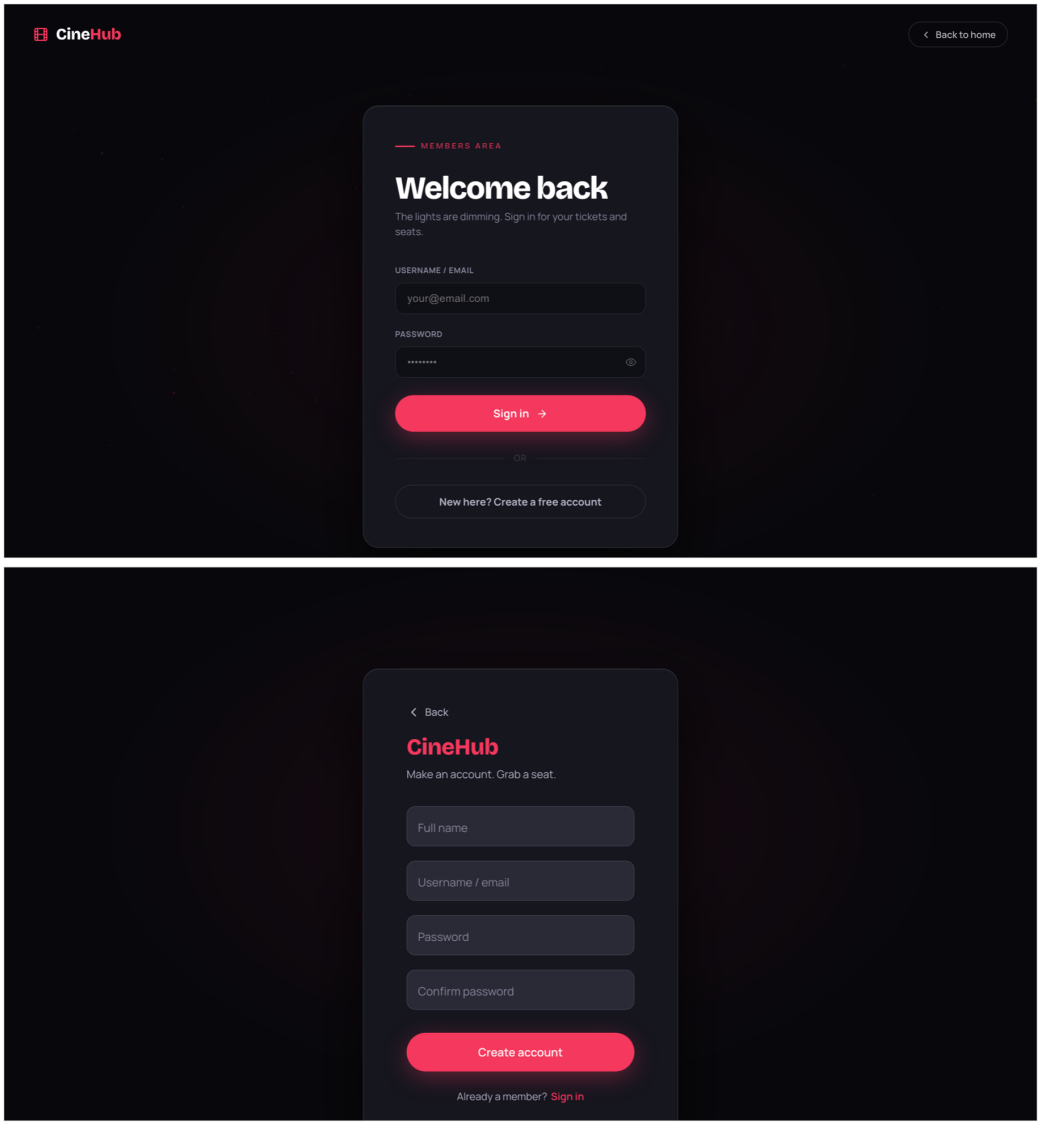
6.2 Εγγραφή και Πρόσβαση στο Σύστημα

Η αρχική επαφή του χρήστη με την εφαρμογή πραγματοποιείται μέσω της Αρχικής Σελίδας (Landing Page), η οποία λειτουργεί ως η «βιτρίνα» της πλατφόρμας, προβάλλοντας τις δημοφιλέστερες ταινίες (Trending Now) και τα τεχνικά χαρακτηριστικά των συνεργαζόμενων κινηματογράφων.



Εικόνα 6.1: Η αρχική σελίδα (Landing Page) της πλατφόρμας με την ενότητα «Trending Now».

- **Δημιουργία Λογαριασμού (Signup):** Ο νέος χρήστης, επιλέγοντας το αντίστοιχο κουμπί, μεταφέρεται στη φόρμα εγγραφής. Εκεί καλείται να συμπληρώσει βασικά στοιχεία, όπως το Ονοματεπώνυμο, το Username (ή Email) και τον προσωπικό του κωδικό πρόσβασης. Κατά την πληκτρολόγηση, το σύστημα ελέγχει δυναμικά την εγκυρότητα των πεδίων.
- **Σύνδεση (Login):** Οι ήδη εγγεγραμμένοι χρήστες εισάγουν τα διαπιστευτήριά τους στη φόρμα εισόδου. Το σύστημα ταυτοποιεί τον χρήστη και τον ανακατευθύνει αυτόματα στο κεντρικό ταμπλό (Dashboard). Η ανακατεύθυνση είναι έξυπνη (role-based routing): οι απλοί πελάτες οδηγούνται στον κατάλογο ταινιών, ενώ οι διαχειριστές μεταφέρονται απευθείας στα κέντρα ελέγχου τους.



Εικόνα 6.2: Η φόρμα σύνδεσης (Login) και εγγραφής (Signup) της εφαρμογής, σχεδιασμένη με κινηματογραφική αισθητική.

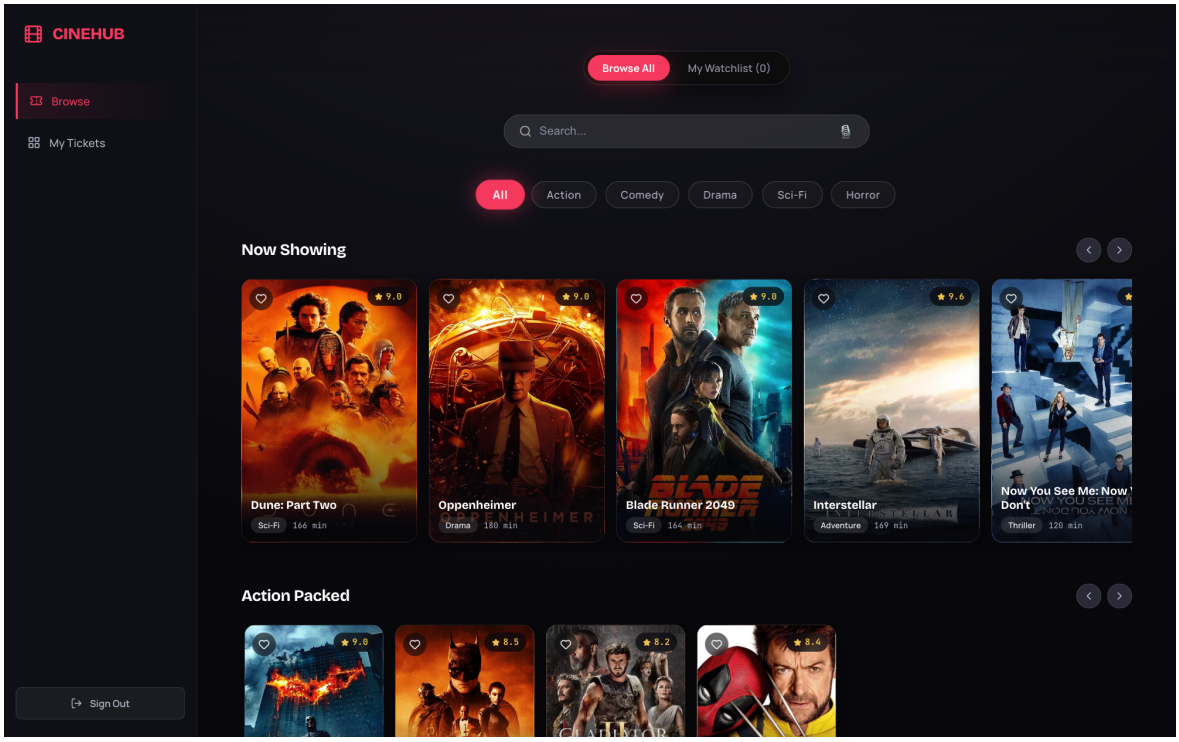
6.3 Εγχειρίδιο Απλού Χρήστη (Customer)

Ο απλός χρήστης του συστήματος, φέροντας τον ρόλο ROLE_CUSTOMER, έχει στη διάθεσή του εργαλεία για την αναζήτηση περιεχομένου, την αγορά εισιτηρίων και τη διαχείριση του προσωπικού του ιστορικού.

6.3.1 Αναζήτηση και Επιλογή Ταινίας

Στην κεντρική οθόνη ("Browse"), ο χρήστης αντικρίζει τον πλήρη κατάλογο των διαθέσιμων ταινιών οργανωμένο σε θεματικές κατηγορίες (π.χ. "Now Showing", "Top Rated").

- **Έξυπνη Αναζήτηση και Φίλτρα:** Στο επάνω μέρος της οθόνης, παρέχεται μια μπάρα αναζήτησης. Ο χρήστης μπορεί να αναζητήσει ταινίες βάσει τίτλου ή να χρησιμοποιήσει τα «φίλτρα ειδών» (chips) για να απομονώσει ταινίες συγκεκριμένου είδους (π.χ. Action, Comedy).
- **Φωνητική Αναζήτηση (Voice Search):** Ενσωματώθηκε εικονίδιο μικροφώνου, το οποίο επιτρέπει την υπαγόρευση του τίτλου της ταινίας. Το σύστημα αναγνωρίζει τη φωνή και εκτελεί την αναζήτηση, ενισχύοντας την προσβασιμότητα της εφαρμογής.
- **Λίστα Αγαπημένων (Watchlist):** Πατώντας το εικονίδιο σε σχήμα «καρδιάς» πάνω στην αφίσα (poster) μιας ταινίας, η ταινία προστίθεται στην προσωπική λίστα του χρήστη, την οποία μπορεί να προβάλει επιλέγοντας την καρτέλα "My Watchlist".

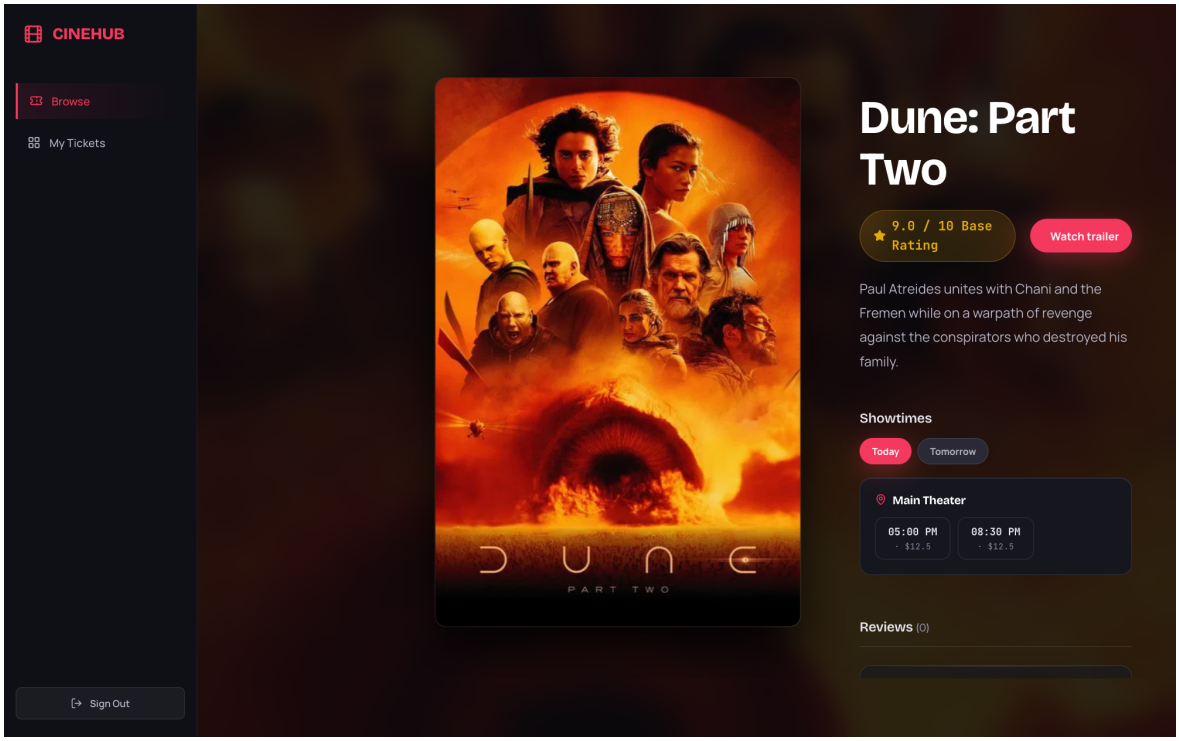


Εικόνα 6.3: Ο κατάλογος ταινιών (Browse) με τα φίλτρα ειδών και τη φωνητική αναζήτηση.

6.3.2 Διαδικασία Κράτησης (E-Ticketing Flow)

Επιλέγοντας μια ταινία, ο χρήστης οδηγείται στη σελίδα λεπτομερειών (Movie Details), όπου μπορεί να δει την περιγραφή, να αναπαραγάγει το trailer και να δει την τρέχουσα βαθμολογία. Η ροή κράτησης ακολουθεί τρία διακριτά βήματα:

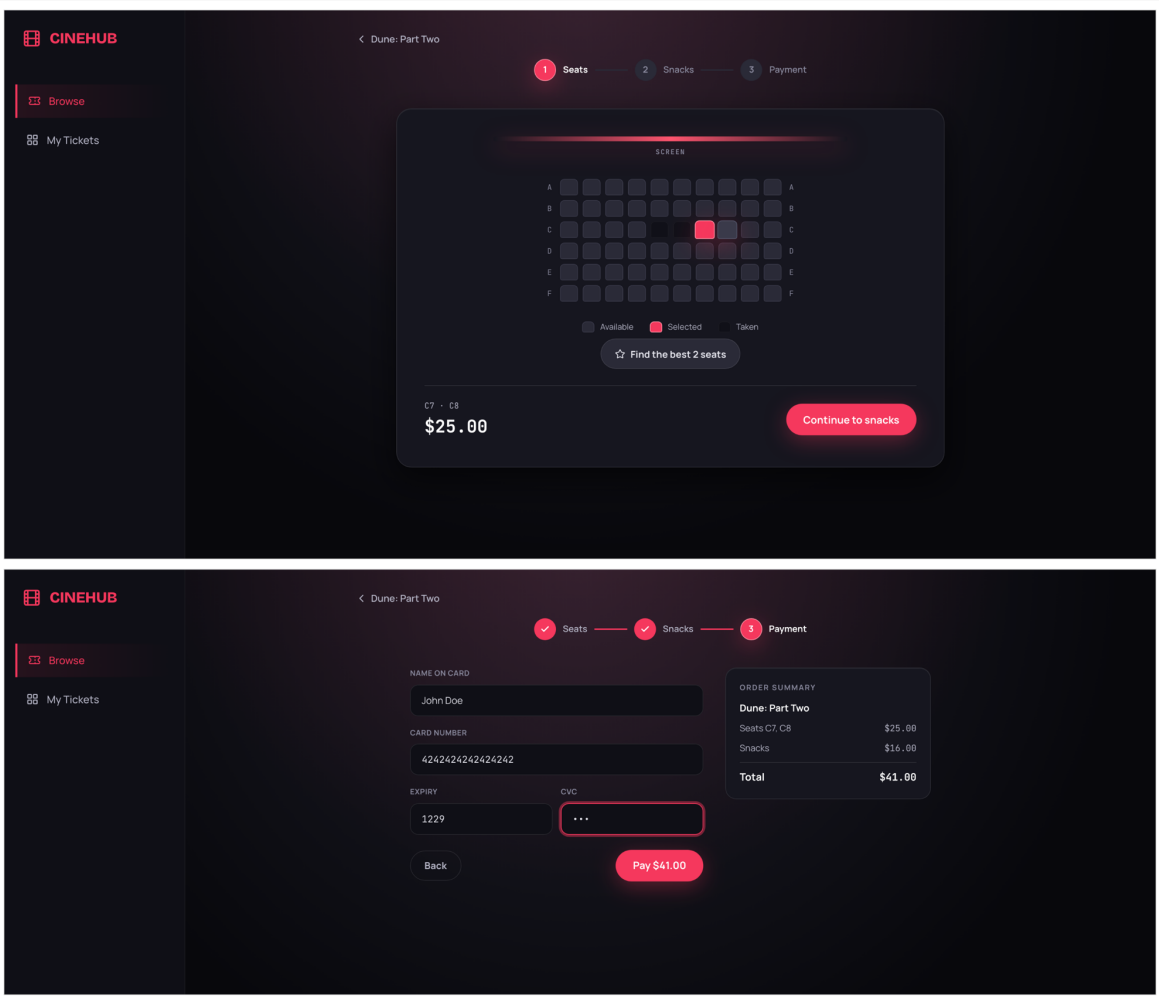
1. **Επιλογή Ημερομηνίας και Ώρας:** Μέσω ενός κυλιόμενου μενού (carousel) ημερομηνιών, ο χρήστης βλέπει τις διαθέσιμες προβολές ταξινομημένες ανά κινηματογράφο.



Εικόνα 6.4: Η σελίδα λεπτομερειών ταινίας (Movie Details) με το trailer και το καρουζέλ ημερομηνιών.

2. **Διαδραστικό Πλάνο Θέσεων (Seat Map):** Ανοίγει ένα ψηφιακό αντίγραφο της αίθουσας. Οι ελεύθερες θέσεις είναι προσβάσιμες, ενώ οι κατειλημμένες εμφανίζονται απενεργοποιημένες (γκρίζες). Ο χρήστης μπορεί να κάνει κλικ στις θέσεις της αρεσκείας του. Για μέγιστη ευκολία, παρέχεται το κουμπί **"Auto-Select Best 2"**, το οποίο (βάσει αλγορίθμου) προεπιλέγει τις δύο καλύτερες κεντρικές θέσεις.

3. **Κυλικείο (Snacks) και Πληρωμή (Checkout):** Πριν την πληρωμή, δίνεται η δυνατότητα προσθήκης προϊόντων κυλικείου (ποπ-κορν, αναψυκτικά). Τέλος, αναδύεται το ασφαλές παράθυρο πληρωμής (Payment Modal) όπου ο χρήστης εισάγει τα στοιχεία της κάρτας του για να ολοκληρώσει τη συναλλαγή.

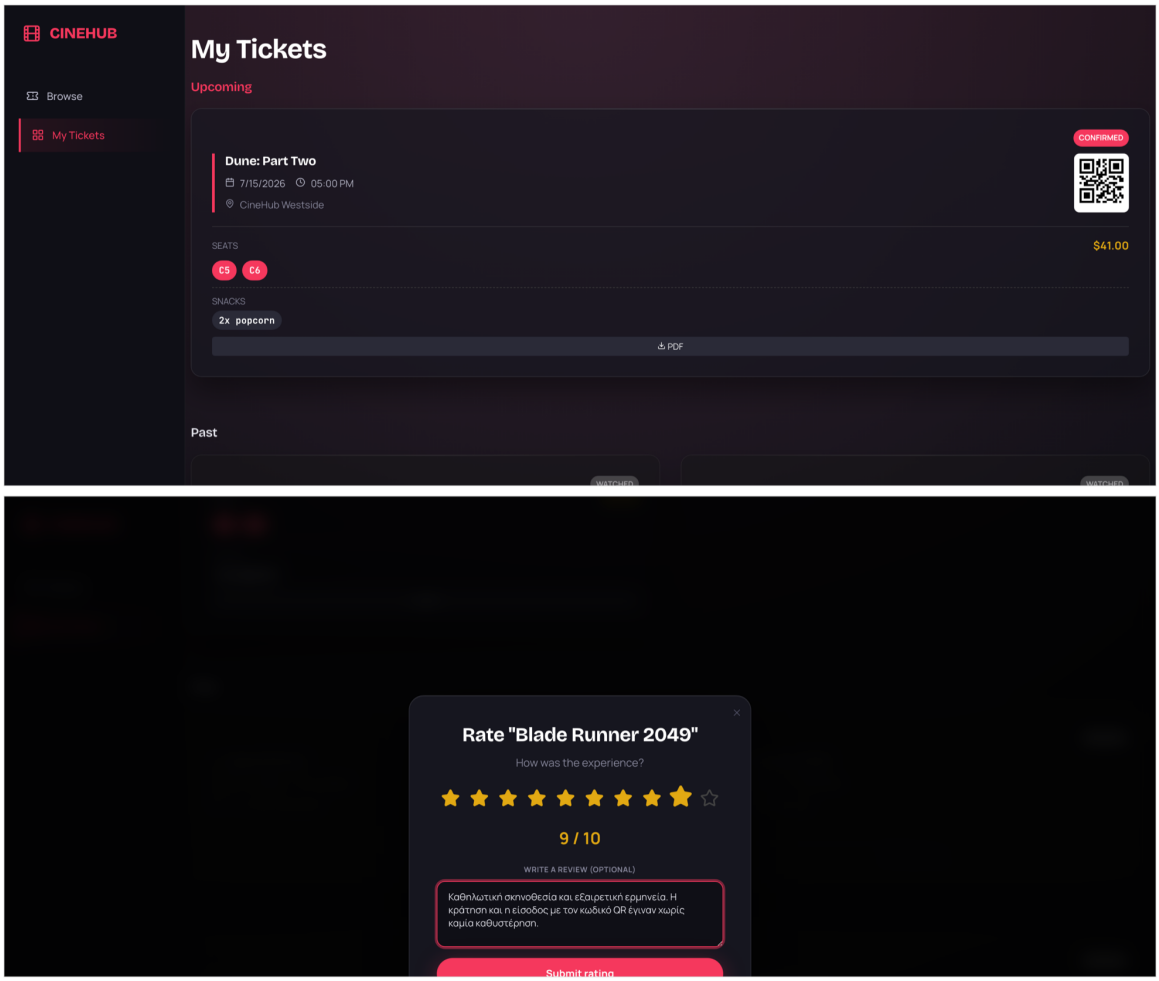


Εικόνα 6.5: Η διαδραστική επιλογή θέσεων (Seat Grid) και η προσομοίωση του συστήματος ηλεκτρονικής πληρωμής.

6.3.3 Το Πορτοφόλι μου και οι Αξιολογήσεις (My Tickets & Ratings)

Στην καρτέλα "My Tickets", συγκεντρώνεται όλο το ιστορικό δραστηριότητας:

- **Μελλοντικές Προβολές:** Τα επερχόμενα εισιτήρια εμφανίζουν εμφανώς το **QR Code**. Ο πελάτης μπορεί να κατεβάσει το εισιτήριο τοπικά πατώντας το κουμπί "Download PDF".
- **Παρελθοντικές Προβολές & Αξιολόγηση:** Για τις ταινίες που ο χρήστης έχει ήδη παρακολουθήσει, το σύστημα ξεκλειδώνει τη λειτουργία αξιολόγησης ("Rate Movie"). Αυτή η λειτουργία εμφανίζει ένα αναδυόμενο παράθυρο όπου ο χρήστης μπορεί να καταχωρήσει τη βαθμολογία του (κλίμακα 1-10) και μια γραπτή κριτική, τροφοδοτώντας το σύστημα με πολύτιμα δεδομένα ανατροφοδότησης.

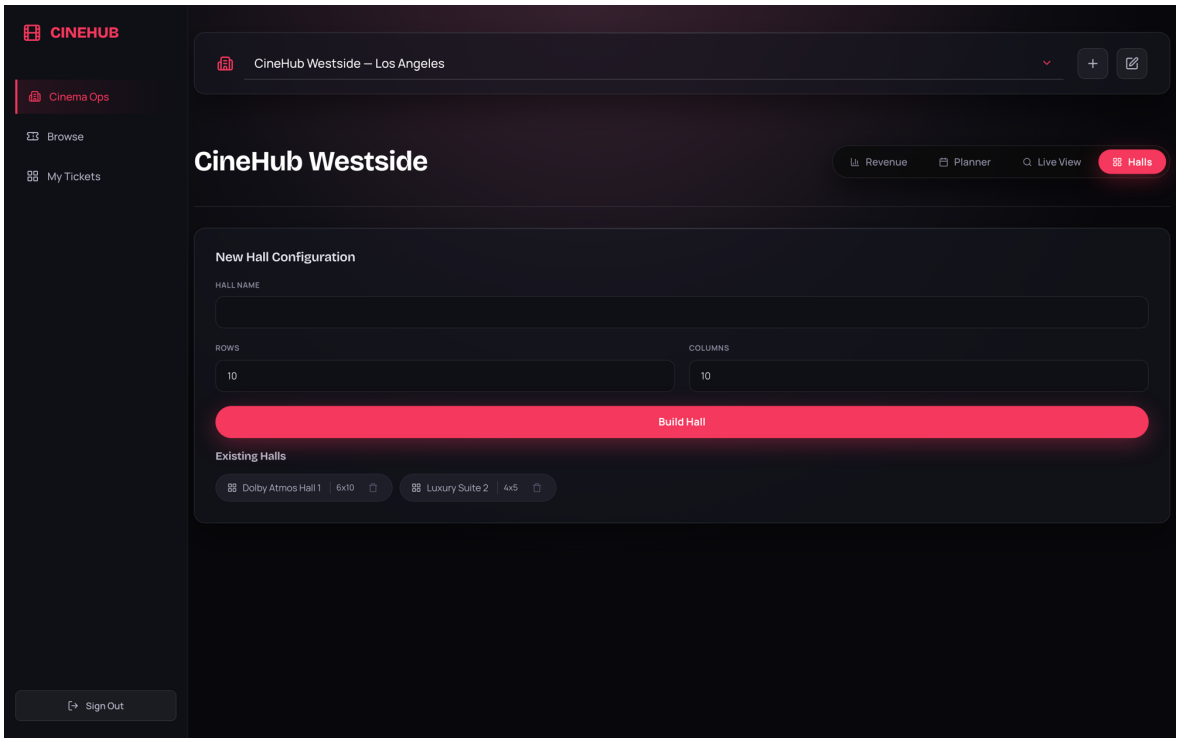


Εικόνα 6.6: Το ψηφιακό πορτοφόλι (My Tickets) με τον κώδικα QR και το παράθυρο αξιολόγησης ταινίας.

6.4 Εγχειρίδιο Διαχειριστή Κινηματογράφου (Cinema Manager)

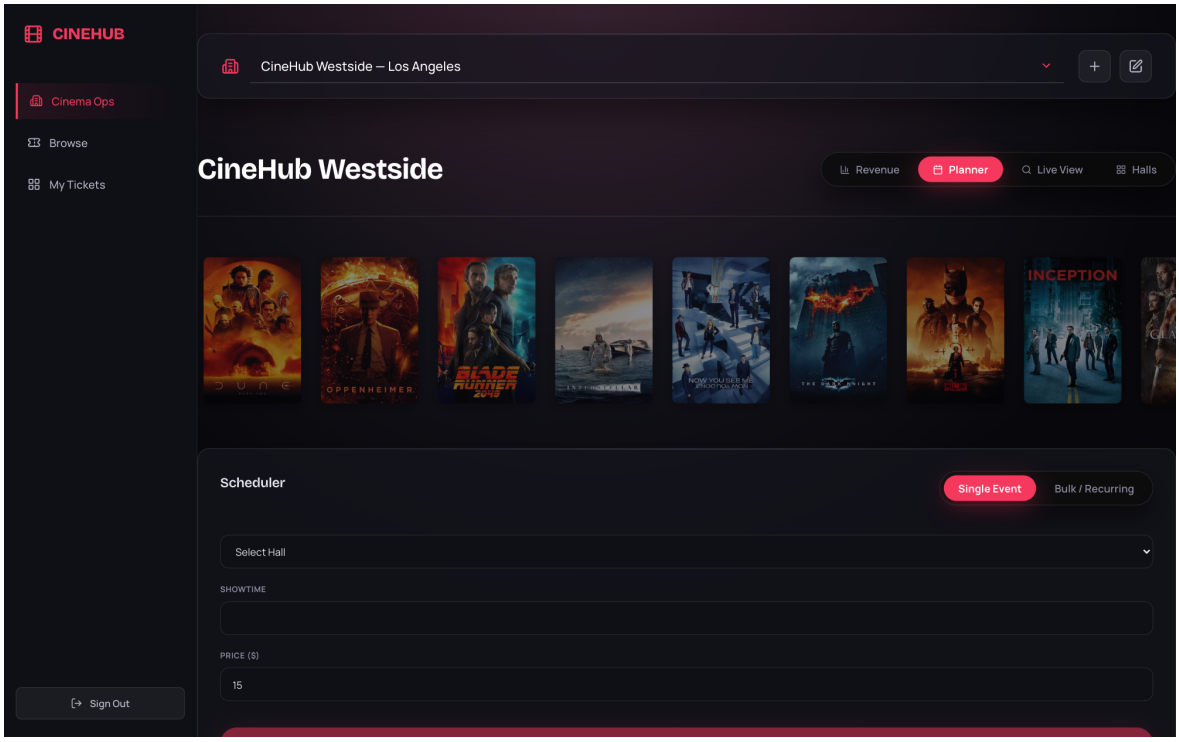
Ο Διαχειριστής Κινηματογράφου (ROLE_CINEMA_MANAGER) δραστηριοποιείται στην καρτέλα **"Cinema Ops"**.

- **Χαρτογράφηση Αιθουσών (Halls):** Ο διαχειριστής εισάγει τα στοιχεία του κινηματογράφου του και δημιουργεί νέες αίθουσες, ορίζοντας απλώς τον αριθμό των γραμμών (Rows) και στηλών (Cols).



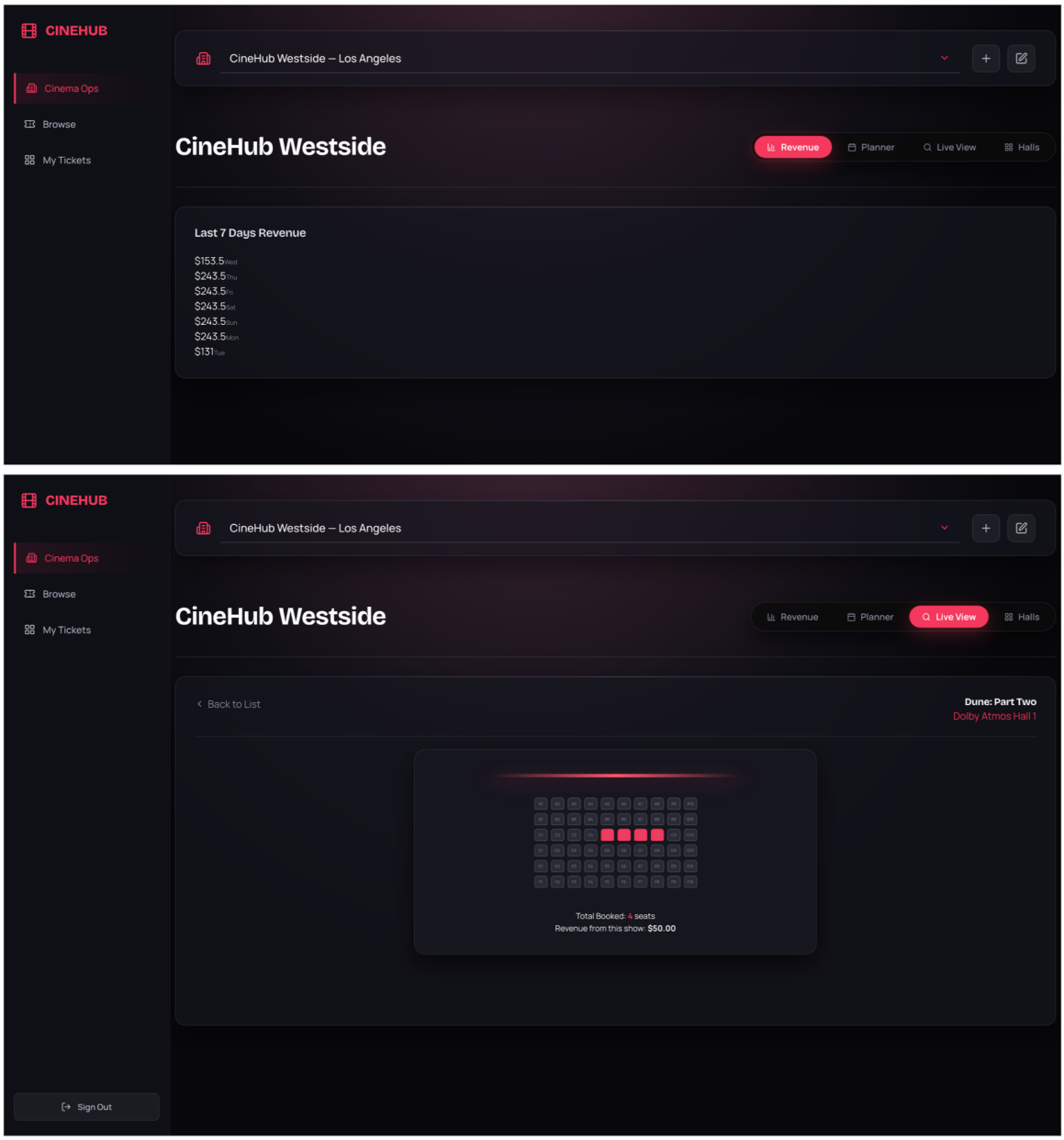
Εικόνα 6.7: Η χαρτογράφηση αιθουσών (Halls) στην καρτέλα Cinema Ops.

- **Προγραμματισμός (Scheduler):** Παρέχεται η δυνατότητα μιας μονής προβολής (Single Event) ή η χρήση του Μαζικού Προγραμματισμού (Bulk / Recurring), όπου με τον ορισμό ημερομηνίας έναρξης, λήξης και ώρας, παράγεται αυτόματα το πρόγραμμα εβδομάδων.



Εικόνα 6.8: Η λειτουργία Μαζικού Προγραμματισμού (Bulk / Recurring Scheduling) προβολών.

- **Στατιστικά και Ζωντανή Εποπτεία:** Μέσω της ενότητας "Revenue", ο διαχειριστής βλέπει ραβδογράμματα με τα έσοδα των τελευταίων 7 ημερών. Στην ενότητα "Live View", μπορεί να επιλέξει οποιαδήποτε μελλοντική προβολή και να δει το ακριβές πλάνο της αίθουσας με τις δεσμευμένες θέσεις σε πραγματικό χρόνο, γνωρίζοντας ακριβώς πόσα άτομα αναμένονται.

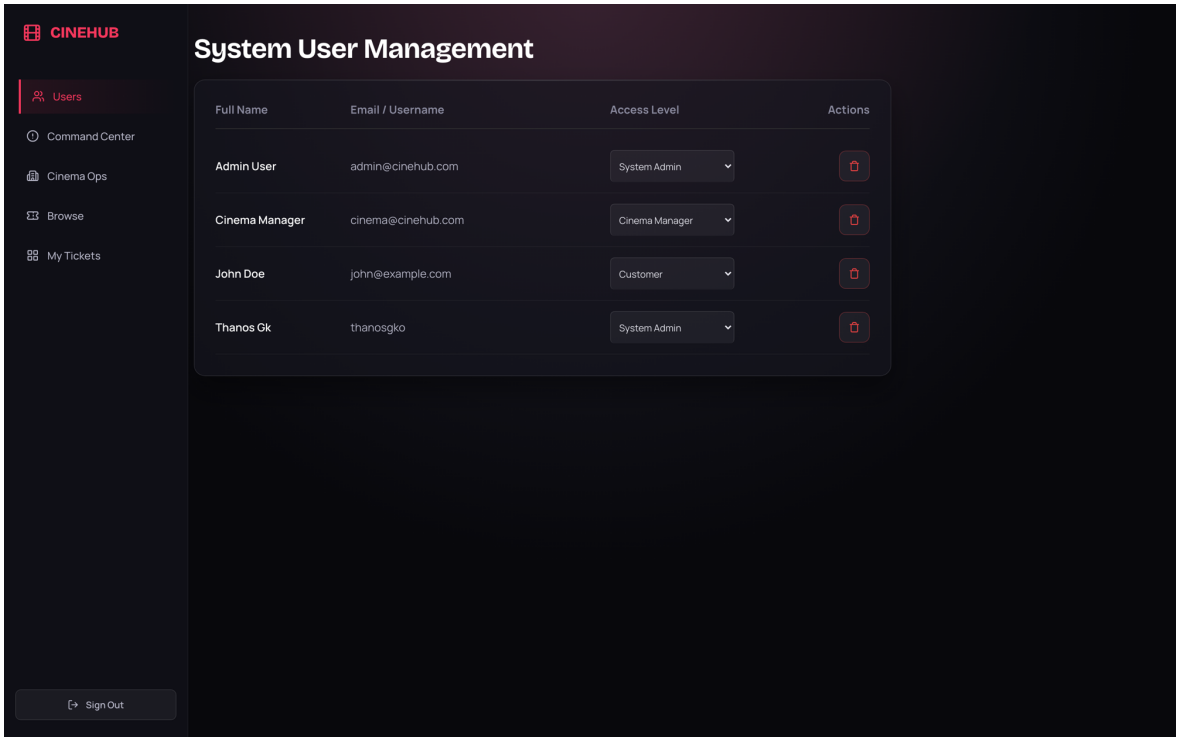


Εικόνα 6.9: Τα στατιστικά εσόδων (Revenue) και η ζωντανή εποπτεία κρατήσεων (Live View).

6.5 Εγχειρίδιο Κεντρικού Διαχειριστή (System Admin)

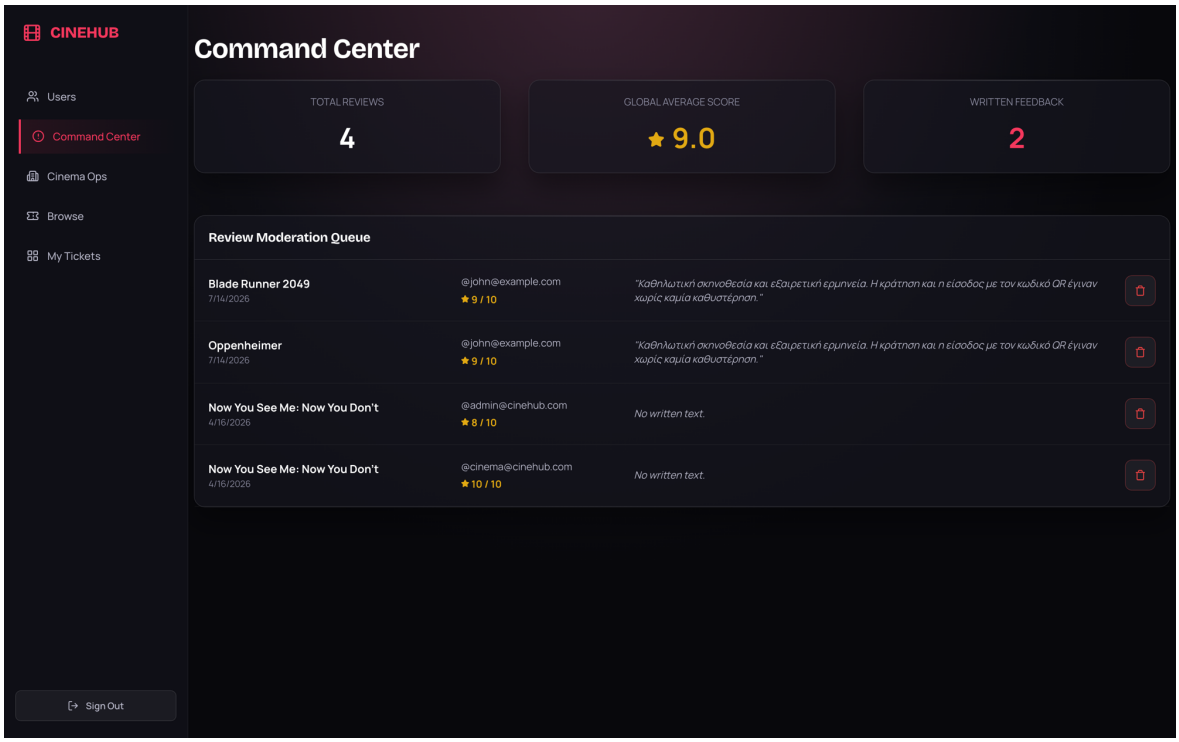
Ο Διαχειριστής Συστήματος (ROLE_ADMIN) κατέχει τα υψηλότερα δικαιώματα πρόσβασης και συντηρεί τη συνολική υγεία της πλατφόρμας.

- **Διαχείριση Χρηστών (Users):** Στην αντίστοιχη καρτέλα εμφανίζεται μια δομημένη λίστα με όλα τα εγγεγραμμένα μέλη. Ο Admin έχει την εξουσία να τροποποιήσει το επίπεδο πρόσβασης (Access Level) ενός χρήστη, αναβαθμίζοντάς τον, για παράδειγμα, από απλό Πελάτη σε Διαχειριστή Κινηματογράφου (Cinema Manager) ή να προχωρήσει σε οριστική διαγραφή λογαριασμών.



Εικόνα 6.10: Η διαχείριση χρηστών και ρόλων (User Management) από τον Διαχειριστή Συστήματος.

- **Κέντρο Ελέγχου (Command Center):** Παρέχει μια πανοραμική εικόνα (System Health Overview), παρουσιάζοντας το συνολικό πλήθος των κριτικών, τον παγκόσμιο μέσο όρο βαθμολογιών και τον αριθμό των γραπτών σχολίων. Στο κάτω μέρος της οθόνης, παρουσιάζεται η ουρά ελέγχου περιεχομένου (Moderation Queue), όπου ο Admin μπορεί να διαβάσει κάθε κριτική και να την αφαιρέσει άμεσα (Delete) εάν περιέχει ακατάλληλο περιεχόμενο, διατηρώντας έτσι την ποιότητα των πληροφοριών.



Εικόνα 6.11: Το Κέντρο Ελέγχου (Command Center) του Διαχειριστή Συστήματος με την ουρά ελέγχου κριτικών.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

7.1 Αποτίμηση και Συμπεράσματα της Πτυχιακής Εργασίας

Ο πρωταρχικός στόχος της παρούσας πτυχιακής εργασίας ήταν η εννοιολογική σύλληψη, ο αρχιτεκτονικός σχεδιασμός και η πλήρης προγραμματιστική υλοποίηση μιας σύγχρονης, κατανεμημένης διαδικτυακής πλατφόρμας με την ονομασία CINEHUB. Η πλατφόρμα σχεδιάστηκε με όραμα να αναδιαμορφώσει τον τρόπο με τον οποίο οι χρήστες και οι διαχειριστές αλληλεπιδρούν με το οικοσύστημα των κινηματογραφικών προβολών, ενοποιώντας λειτουργίες που παραδοσιακά βρίσκονταν κατακερματισμένες σε διαφορετικά συστήματα (πληροφόρηση, ηλεκτρονική κράτηση, συστήματα αξιολόγησης).

Μέσα από την αυστηρή τήρηση του Κύκλου Ζωής Ανάπτυξης Λογισμικού (SDLC) και την υιοθέτηση σύγχρονων προτύπων της Τεχνολογίας Λογισμικού (Software Engineering), εξήχθησαν τα ακόλουθα κρίσιμα συμπεράσματα και επιτεύγματα:

1. **Επιτυχής Υλοποίηση Αρχιτεκτονικής N-Tier:** Αποδείχθηκε στην πράξη η υπεροχή της αποσυνδεδεμένης αρχιτεκτονικής (decoupled architecture). Η χρήση της React.js στο επίπεδο παρουσίασης (Frontend) επέτρεψε τη δημιουργία μιας ταχύτατης Single Page Application (SPA), εξαλείφοντας τους περιττούς κύκλους επαναφόρτωσης (page reloads) και προσφέροντας μια ρευστή εμπειρία χρήστη, όμοια με αυτή των εγγενών (native) εφαρμογών. Αντίστοιχα, το Spring Boot στο επίπεδο επιχειρησιακής λογικής (Backend) παρείχε την απαραίτητη σταθερότητα, ασφάλεια και επεκτασιμότητα για τη διαχείριση ταυτόχρονων αιτημάτων.
2. **Ενοποίηση Ηλεκτρονικών Κρατήσεων και Κοινωνικής Αλληλεπίδρασης:** Το σύστημα CINEHUB κατάφερε να γεφυρώσει το χάσμα μεταξύ της άμεσης συναλλαγής (e-ticketing) και της συλλογής ανατροφοδότησης (feedback). Η ενσωμάτωση του συστήματος αξιολογήσεων (ratings/reviews) απευθείας στο ψηφιακό πορτοφόλι (My Tickets) του χρήστη, εξασφαλίζει ότι οι κριτικές προέρχονται από επαληθευμένους θεατές, δημιουργώντας μια αξιόπιστη βάση δεδομένων που μπορεί να αξιοποιηθεί ερευνητικά στο μέλλον.

3. **Αυτοματοποίηση Διαχειριστικών Διαδικασιών:** Για τους διαχειριστές του συστήματος, η εργασία απέδωσε ισχυρά εργαλεία τα οποία ελαχιστοποιούν το διοικητικό (administrative) κόστος. Η ανάπτυξη του αλγορίθμου Μαζικού Προγραμματισμού (Bulk Scheduling) επέλυσε το πρόβλημα της χρονοβόρας, χειροκίνητης καταχώρησης προβολών, ενώ η διαδραστική εποπτεία των αιθουσών (Live View) προσφέρει πολύτιμα δεδομένα επιχειρηματικής ευφυΐας (Business Intelligence) σε πραγματικό χρόνο.
4. **Θωράκιση και Ασφάλεια Πληροφοριών:** Η εφαρμογή του ελέγχου πρόσβασης βάσει ρόλων (Role-Based Access Control) σε συνδυασμό με το πρωτόκολλο JSON Web Tokens (JWT) εξασφάλισε ότι η εφαρμογή είναι ανθεκτική σε μη εξουσιοδοτημένες προσβάσεις. Η "stateless" φύση των JWT εγγυάται ότι η αυθεντικοποίηση είναι κλιμακώσιμη (scalable), ενώ ο σχεδιασμός του Backend αποτρέπει ενεργά τις επικαλύψεις (overlaps) κατά την κράτηση θέσεων.

Συνοψίζοντας, η πτυχιακή εργασία απέδειξε επιτυχώς ότι ο αρμονικός συνδυασμός θεωρητικών αρχών της Πληροφορικής (Αλγόριθμοι, Δομές Δεδομένων, Αλληλεπίδραση Ανθρώπου-Υπολογιστή) με σύγχρονα βιομηχανικά εργαλεία ανάπτυξης μπορεί να αποδώσει ένα λογισμικό υψηλών προδιαγραφών, ικανό να σταθεί σε πραγματικά παραγωγικά περιβάλλοντα.

7.2 Μελλοντικές Επεκτάσεις

Παρά το γεγονός ότι το σύστημα CINEHUB αποτελεί μια πλήρως λειτουργική και ολοκληρωμένη λύση, η ραγδαία εξέλιξη του πεδίου της Πληροφορικής ανοίγει συνεχώς νέους ορίζοντες. Το λογισμικό έχει σχεδιαστεί με γνώμονα την επεκτασιμότητα (extensibility), επιτρέποντας την απρόσκοπτη προσθήκη νέων υποσυστημάτων στο μέλλον. Οι κυριότερες προτεινόμενες μελλοντικές επεκτάσεις αναλύονται παρακάτω:

7.2.1 Ενσωμάτωση Τεχνητής Νοημοσύνης (AI) και Συστημάτων Συστάσεων

Όπως αναλύθηκε στο θεωρητικό πλαίσιο της εργασίας, η Τεχνητή Νοημοσύνη παίζει καθοριστικό ρόλο στην εξατομίκευση της εμπειρίας του χρήστη. Με βάση τον τεράστιο όγκο δεδομένων που θα συλλέγει η πλατφόρμα από τις αξιολογήσεις των χρηστών, προτείνεται η ανάπτυξη ενός έξυπνου Συστήματος Συστάσεων (Recommender System). Χρησιμοποιώντας αλγόριθμους Μηχανικής Μάθησης, όπως το Συνεργατικό Φιλτράρισμα (Collaborative Filtering) ή το Φιλτράρισμα Βάσει Περιεχομένου (Content-Based Filtering), το CINEHUB θα μπορεί να μαθαίνει από τις προτιμήσεις του χρήστη. Το σύστημα θα αναγνωρίζει μοτίβα (π.χ. προτίμηση σε ταινίες επιστημονικής φαντασίας) και θα προτείνει στοχευμένα νέες κυκλοφορίες, αυξάνοντας κατακόρυφα την ικανοποίηση του πελάτη (customer retention) και τα έσοδα του κινηματογράφου.

Επιπλέον, η πρόσφατη πρόοδος στα Μεγάλα Γλωσσικά Μοντέλα (Large Language Models) ανοίγει τον δρόμο για την ενσωμάτωση ενός συνομιλιακού βοηθού, ικανού να απαντά σε ερωτήσεις των χρηστών και να προτείνει ταινίες σε φυσική γλώσσα. Η αξιοποίησή τους, ωστόσο, εγείρει σημαντικά ζητήματα αξιοπιστίας, διαφάνειας και ηθικής, τα οποία απαιτούν προσεκτική μελέτη πριν από κάθε παραγωγική εφαρμογή [44, 45, 46].

7.2.2 Επέκταση στο Οικοσύστημα Κινητών Συσκευών (Mobile Computing)

Με δεδομένη τη στροφή των χρηστών προς τις κινητές συσκευές (Mobile-First approach), μια λογική μελλοντική επέκταση είναι η ανάπτυξη μιας εγγενούς εφαρμογής (Native Mobile App) για λειτουργικά συστήματα Android και iOS, αξιοποιώντας τεχνολογίες όπως το React Native ή το Flutter [15, 16]. Μια κινητή εφαρμογή θα επέτρεπε λειτουργίες προστιθέμενης αξίας (value-added features), όπως:

- **Ειδοποιήσεις (Push Notifications):** Ενημέρωση των χρηστών σε πραγματικό χρόνο για επερχόμενες πρεμιέρες, εκπτώσεις ή την έναρξη της προβολής τους [39].
- **Ενσωμάτωση σε Ψηφιακά Πορτοφόλια (Digital Wallets):** Δυνατότητα αποθήκευσης του παραγόμενου ψηφιακού εισιτηρίου απευθείας στο Google Wallet ή στο Apple Wallet για άμεση, ανέπαφη πρόσβαση.

7.2.3 Ανάπτυξη Συστήματος Ελέγχου Πρόσβασης (Scanner App)

Συμπληρωματικά με την πλατφόρμα, θα μπορούσε να αναπτυχθεί ένα ανεξάρτητο υποσύστημα (Scanner Application) σχεδιασμένο αποκλειστικά για το προσωπικό υποδοχής των κινηματογράφων. Η εφαρμογή αυτή θα χρησιμοποιεί την κάμερα των φορητών συσκευών για να σαρώνει (scan) τους κωδικούς QR που ήδη παράγει το CINEHUB. Η εφαρμογή θα επικοινωνεί μέσω API με το Backend για να επιβεβαιώνει την εγκυρότητα του εισιτηρίου σε πραγματικό χρόνο, αποτρέποντας απόπειρες παραποίησης (fraud prevention) και επιταχύνοντας τη ροή ελέγχου στην είσοδο της αίθουσας.

7.2.4 Διασύνδεση με Πραγματικές Πύλες Πληρωμών (Payment Gateways)

Ενώ στο πλαίσιο της παρούσας πτυχιακής εργασίας το στάδιο της πληρωμής λειτουργεί σε περιβάλλον προσομοίωσης για λόγους επίδειξης (mocking), το επόμενο βήμα για την εμπορική αξιοποίηση του συστήματος είναι η διασύνδεση (integration) με πραγματικές πύλες πληρωμών. Η ενσωμάτωση σύγχρονων APIs (όπως της Stripe ή του PayPal) θα διασφαλίσει την απόλυτη ασφάλεια των συναλλαγών (PCI DSS compliance) και θα αυτοματοποιήσει την εκκαθάριση των χρηματικών ποσών μεταξύ της πλατφόρμας και των συνεργαζόμενων κινηματογράφων.

7.2.5 Πολυγλωσσικότητα και Προσβασιμότητα (i18n & Web Accessibility)

Τέλος, μια σημαντική προσθήκη θα ήταν η διεθνοποίηση (Internationalization - i18n) της εφαρμογής για την υποστήριξη πολλαπλών γλωσσών, διευρύνοντας το κοινό-στόχο της πλατφόρμας. Παράλληλα, περαιτέρω βελτιώσεις στον κώδικα του Frontend βάσει των προτύπων Προσβασιμότητας Περιεχομένου Ιστού (WCAG - Web Content Accessibility Guidelines), όπως η καλύτερη υποστήριξη πλοήγησης μέσω πληκτρολογίου και προγραμμάτων ανάγνωσης οθόνης (screen readers), θα καταστήσουν το CINEHUB απόλυτα φιλικό και προσβάσιμο σε άτομα με αναπηρίες (ΑμεΑ) [37].

8. ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] G. A. Tsihrintzis, M. Virvou, and I. Hatzilygeroudis, “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*, vol. 32, no. 05, p. 2302003, 2023.

[2] M. Virvou, G. A. Tsihrintzis, N. G. Bourbakis, and L. C. Jain, *Handbook on Artificial Intelligence-Empowered Applied Software Engineering: VOL. 2: Smart Software Applications in Cyber-Physical Systems*, vol. 3. Springer Nature, 2022.

[3] G. A. Tsihrintzis, M. Virvou, and T. Saruwatari, “Special Collection of Extended Selected Papers on ‘Novel Research Results Presented at the 14th International Joint Conference on Knowledge-based Software Engineering (JCKBSE2022)’,” *Intelligent Decision Technologies*, vol. 16, no. 4, pp. 715-716, 2022.

[4] M. Virvou, G. A. Tsihrintzis, and L. C. Jain, “Introduction to advances in selected artificial intelligence areas,” in *Advances in Selected Artificial Intelligence Areas: World Outstanding Women in Artificial Intelligence*, Cham: Springer, 2022, pp. 1-7.

[5] G. A. Tsihrintzis, M. Virvou, and G. Phillips-Wren, “Surveys in artificial intelligence-based technologies,” *Intelligent Decision Technologies*, vol. 13, no. 4, pp. 393-394, 2019.

[6] S. Weng, C. S. Shieh, and G. A. Tsihrintzis, Eds., *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the 18th IIH-MSP 2022 Kitakyushu, Japan*. Springer, 2023.

[7] M. Virvou and T. Nakamura, Eds., *Knowledge-based Software Engineering: Proceedings of the Eighth Joint Conference on Knowledge-Based Software Engineering*, vol. 180. IOS Press, 2008.

[8] G. A. Tsihrintzis and M. Virvou, Eds., *Multimedia Services in Intelligent Environments: Software Development Challenges and Solutions*, vol. 2. Springer, 2010.

[9] G. A. Tsihrintzis, C. Toro, S. A. Rios, R. J. Howlett, and L. C. Jain, “27th KES International Conference on Knowledge-Based and Intelligent Information & Engineering Systems KES2023,” *Procedia Computer Science*, vol. 225, pp. 1-11, 2023.

[10] M. Virvou, G. A. Tsihrintzis, N. G. Bourbakis, and L. C. Jain, “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*, Cham: Springer, 2022, pp. 1-8.

[11] I. K. Hatzilygeroudis, G. A. Tsihrintzis, and L. C. Jain, Eds., *Fusion of Machine Learning Paradigms: Theory and Applications*, vol. 236. Springer Nature, 2023.

[12] M. Virvou, E. Alepis, G. A. Tsihrintzis, and L. C. Jain, *Machine learning paradigms: advances in learning analytics*. Springer, 2020, pp. 1-5.

[13] M. Virvou, “The emerging era of human-AI interaction: Keynote address,” in *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, IEEE, 2022, pp. 1-10.

[14] M. Virvou, “Artificial Intelligence and User Experience in reciprocity: Contributions and state of the art,” *Intelligent Decision Technologies*, vol. 17, pp. 73-125, 2023.

[15] E. Alepis and M. Virvou, *Object-oriented user interfaces for personalized mobile learning*. Springer, 2014.

[16] M. Virvou and E. Alepis, “Mobile educational features in authoring tools for personalised tutoring,” *Computers & Education*, vol. 44, no. 1, pp. 53-68, 2005.

[17] G. A. Tsihrintzis, M. Virvou, E. Alepis, and I. O. Stathopoulou, “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)*, IEEE, 2008, pp. 32-37.

[18] I. O. Stathopoulou, E. Alepis, G. A. Tsihrintzis, and M. Virvou, “On assisting a visual-facial affect recognition system with keyboard-stroke pattern information,” *Knowledge-Based Systems*, vol. 23, no. 4, pp. 350-356, 2010.

[19] G. Katsionis and M. Virvou, “A cognitive theory for affective user modelling in a virtual reality educational game,” in *2004 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, IEEE, 2004, pp. 1209-1213.

[20] E. Alepis, I. O. Stathopoulou, M. Virvou, G. A. Tsihrintzis, and K. Kabassi, “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, IEEE, 2010, pp. 274-281.

[21] E. Alepis and M. Virvou, “Emotional Intelligence: Constructing user stereotypes for affective bi-modal interaction,” in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, Springer, 2006, pp. 435-442.

[22] E. Alepis, M. Virvou, and K. Kabassi, “Affective student modeling based on microphone and keyboard user actions,” in *Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, IEEE, 2006, pp. 139-141.

[23] M. Virvou, “Automatic reasoning and help about human errors in using an operating system,” *Interacting with Computers*, vol. 11, no. 5, pp. 545-573, 1999.

[24] M. Virvou and B. Du Boulay, “Human plausible reasoning for intelligent help,” *User modeling and user-adapted interaction*, vol. 9, pp. 321-375, 1999.

[25] M. Virvou, “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*, 2018, pp. 2-3.

[26] K. Chrysafiadi and M. Virvou, “Dynamically personalized e-training in computer programming and the language C,” *IEEE Transactions on Education*, vol. 56, no. 4, pp. 385-392, 2013.

[27] K. Chrysafiadi, M. Virvou, and G. A. Tsihrintzis, “A fuzzy-based mechanism for automatic personalized assessment in an e-learning system for computer programming,” *Intelligent Decision Technologies*, vol. 16, no. 4, pp. 699-714, 2022.

[28] S. Papadimitriou, K. Chrysafiadi, and M. Virvou, “Adaptive quizzes using fuzzy genetic algorithm,” in *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, IEEE, 2023, pp. 1-8.

[29] K. Chrysafiadi, M. Virvou, G. A. Tsihrintzis, and I. Hatzilygeroudis, “An Adaptive Learning Environment for Programming Based on Fuzzy Logic and Machine Learning,” *International Journal on Artificial Intelligence Tools*, vol. 32, no. 05, p. 2360011, 2023.

[30] V. Tsiriga and M. Virvou, “Modelling the student to individualise tutoring in a web-based ICALL,” *International Journal of Continuing Engineering Education and Life Long Learning*, vol. 13, no. 3-4, pp. 350-365, 2003.

[31] M. Virvou, “On the evaluation of the combined role of virtual reality games and animated agents in edutainment,” *Intelligent Computer Graphics 2011*, pp. 79-95, 2012.

[32] M. Virvou and K. Kabassi, “Evaluating an intelligent graphical user interface by comparison with human experts,” *Knowledge-based systems*, vol. 17, no. 1, pp. 31-37, 2004.

[33] M. Virvou and K. Kabassi, “Adapting the human plausible reasoning theory to a graphical user interface,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 4, pp. 546-563, 2004.

[34] M. Virvou and K. Kabassi, “Reasoning about users' actions in a graphical user interface,” *Human-Computer Interaction*, vol. 17, no. 4, pp. 369-398, 2002.

[35] M. Virvou and M. Moundridou, “Student and instructor models: two kinds of user model and their interaction in an ITS authoring tool,” in *User Modeling 2001*, Springer, 2001, pp. 158-167.

[36] M. Moundridou and M. Virvou, “Authoring and delivering adaptive Web-based textbooks using WEAR,” in *Proceedings IEEE International Conference on Advanced Learning Technologies*, IEEE, 2001, pp. 185-188.

[37] P. Pavlakis, E. Alepis, and M. Virvou, “Intelligent mobile multimedia application for the support of the elderly,” in *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, IEEE, 2012, pp. 297-300.

[38] A. Kontogianni, E. Alepis, M. Virvou, and C. Patsakis, *“Smart Tourism: The Impact of Artificial Intelligence and Blockchain”*, Intelligent Systems Reference Library, Springer, 2024.

[39] N. Alonistioti, E. A. Tsichrintzi, K. Chrysafiadi, and E. Alepis, “Requirements for Fuzzy Logic in Personalisation of Fire Emergency Alerts,” in *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, IEEE, 2023, pp. 1-8.

[40] E. Mougiakou and M. Virvou, “Based on GDPR privacy in UML: Case of e-learning program,” in *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, IEEE, 2017, pp. 1-8.

[41] E. A. Politou, E. Alepis, M. Virvou, and C. Patsakis, *Privacy and Data Protection Challenges in the Distributed Era*. Springer, 2022, pp. 1-185.

[42] D. P. Panagoulias, M. Virvou, and G. A. Tsihrintzis, “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*, vol. 248, p. 123375, 2024.

[43] M. Virvou and G. A. Tsihrintzis, “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)*, IEEE, 2023, pp. 1-6.

[44] M. Virvou and G. A. Tsihrintzis, “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)*, IEEE, 2023, pp. 1-8.

[45] M. Virvou and G. A. Tsihrintzis, “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)*, IEEE, 2023, pp. 1-8.

[46] M. Virvou, G. A. Tsihrintzis, D. N. Sotiropoulos, K. Chrysafiadi, E. Sakkopoulos, and E. A. Tsichrintzi, “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)*, IEEE, 2023, pp. 1-9.