# Αρχιτεκτονική συστήματος σύστασης ταινιών με χρήση μηχανικής μάθησης

# Περιεχόμενα

# 1. Εισαγωγή

Το σύστημα σύστασης ταινιών (movie recommendation system) που παρουσιάζουμε, στοχεύει σε γρήγορες και ακριβής συστάσεις ταινιών με επίκεντρο την εύκολη πλοήγηση του χρήστη στην πλατφόρμα. Μέσα από την διαδικτυακή πλατφόρμα, ο θεατής θα έχει στη διάθεση του μία ευρύ ποικιλία ταινιών, με προτάσεις που έχουν παραχθεί από την συνεργασία 2 αλγορίθμων μηχανικής μάθησης τους LightGCN και FIRE, αλγόριθμοι που η ακρίβεια τους έχει αποδειχθεί ερευνητικά. Στόχος του παρόν εγγράφου είναι η περιγραφή της αρχιτεκτονικής του συστήματος με σκοπό να τεθούν οι βάσεις για την υλοποίηση του συνόλου των προτάσεων του έργου.

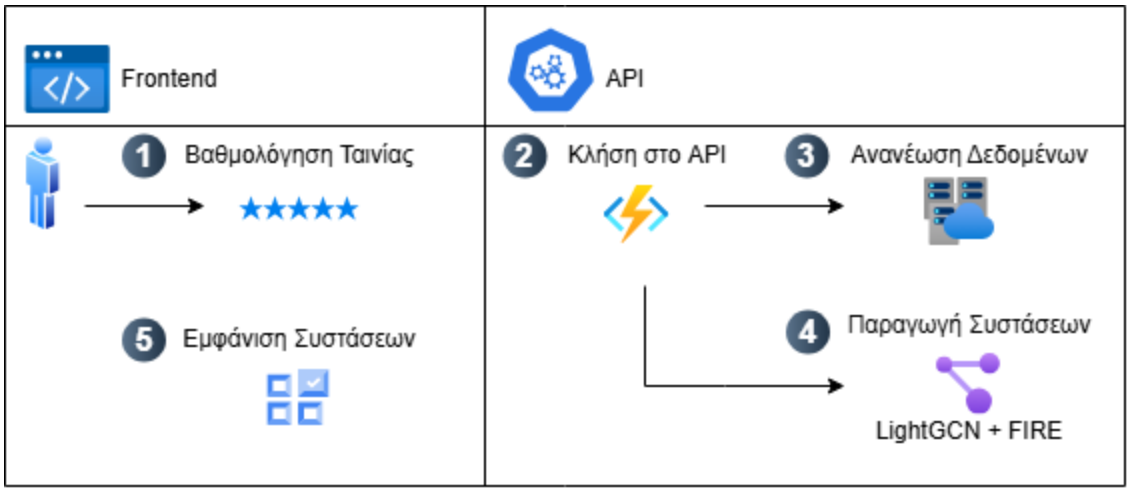
# 2. Αρχιτεκτονική Συστήματος

Το αρχιτεκτονικό σχέδιο του συστήματος διαχωρίζεται σε 2 βασικά επίπεδα:

- Το **επίπεδο frontend,** που εξασφαλίζει την διεπαφή χρήστη.
- Το **επίπεδο API**, που είναι χωρισμένο σε 2 τμήματα: α) Ανανέωση Συστάσεων και β) Αποθήκευση Δεδομένων.

Όταν ο χρήστης πραγματοποιεί μια ενέργεια που απαιτεί ενημέρωση των προτάσεων, το frontend θα καλέσει το API ώστε να ζητήσει νέες συστάσεις από το backend. Αυτό συμβαίνει, όταν ο χρήστης αλληλεπιδράσει με μία ταινία βαθμολογώντας την. Η διαδικασία έχει ως εξής:

1. **Αξιολόγηση θεατή:** Ο θεατής αξιολογεί την ταινία βαθμολογώντας την από το 1 έως το 5.
2. **Κλήση στο API:** Το frontend, σε απόκριση αυτής της ενέργειας, στέλνει ένα αίτημα HTTP προς ένα καθορισμένο endpoint (local host).
3. **Αποθήκευση δεδομένων:** Τα δεδομένα αναβαθμίζονται μέσα στα CSV αρχεία.
4. **Ανανέωση συστάσεων:** Με βάση τα νέα δεδομένα παράγει τη λίστα των προτεινόμενων ταινιών.
5. **Εμφάνιση αποτελεσμάτων:** Η νέα λίστα συστάσεων στέλνεται πίσω στο frontend μέσω του API, και εμφανίζεται στην διεπαφή χρήστη.




Εικόνα 1: Λειτουργικό διάγραμμα συστήματος.

Με αυτόν τον τρόπο, κάθε φορά που ο χρήστης χρειάζεται ενημερωμένες συστάσεις, η εφαρμογή frontend επικοινωνεί με το backend μέσω του ορισμένου API αντί να προσπαθεί να παράξει προτάσεις τοπικά. Αυτό διατηρεί το frontend απλό και μεταφέρει την απαιτητική επεξεργασία στο backend.

## 2.1 Επίπεδο Frontend

Το frontend της εφαρμογής υλοποιείται με React και παρέχεται ως στατικό web περιεχόμενο. Η εφαμογη είναι μονοσέλιδη (Single Page Application, SPA), έχει χτιστεί σε αρχεία HTML, CSS και JavaScript, τα οποία δίνονται απευθείας στον χρήστη χωρίς να απαιτείται δυναμική δημιουργία σε server. Δηλαδή, όλες οι διαδρομές (routes) φορτώνονται από ένα μόνο αρχείο HTML.

Ο ρόλος του frontend είναι να προσφέρει το περιβάλλον διεπαφής χρήστη (UI) όπου ο χρήστης μπορεί να αλληλεπιδρά με το σύστημα. Όλη η λογική των συστάσεων υλοποιείται μέσω κλήσεων API σε ένα backend, γι’ αυτό το frontend δεν χρειάζεται να γνωρίζει λεπτομέρειες του μοντέλου μηχανικής μάθησης, απλώς εμφανίζει τα αποτελέσματα και στέλνει αιτήματα.

Το frontend όπως και το backend παρέχονται μέσω local host.

## 2.2 Επίπεδο API

Όσο αφόρα το επίπεδο API η κλήση από το frontend φτάνει στο API μέσω του PHP server, που περιέχει τη διασύνδεση με τη λογική του μοντέλου LightGCN και FIRE, και ενεργοποιείται από εισερχόμενα HTTP αιτήματα. Όταν φτάσει το αίτημα με τα στοιχεία του χρήστη (user ID), ο

κώδικας του server εκτελείται, φορτώνει το εκπαιδευμένο μοντέλο LightGCN και το χρησιμοποιεί για να παράξει τις νέες συστάσεις.

Αφότου υπολογίσει τις προτεινόμενες επιλογές για τον συγκεκριμένο θεατή με βάση τα δεδομένα εισόδου, διαμορφώνει την απόκριση σε μορφή αρχείου JSON, που περιλαμβάνει τη λίστα των συστάσεων. Στη συνέχεια επιστρέφει αυτή την απόκριση πίσω στο frontend ως αποτέλεσμα του HTTP αιτήματος.

Επιπλέον αποκλειστικά για σύσταση των ταινιών χρησιμοποιείται ένα Flask server για να είναι πιο εύκολη η διασύνδεση με του αλγορίθμους σε python.

Με αυτόν τον τρόπο, η υπολογιστική εργασία του συστήματος συστάσεων παραμένει στο backend, και το frontend απλώς λαμβάνει έτοιμα τα αποτελέσματα. Επιπλέον, κατά τη διάρκεια αυτής της διαδικασίας να ενημερώνονται τα αποθηκευμένα δεδομένα για παράδειγμα, να καταγράψει σε ένα αρχείο CSV τη νέα αλληλεπίδραση του χρήστη ώστε η πληροφορία αυτή να χρησιμοποιηθεί αργότερα για ανανέωση του μοντέλου.

Για την ανάπτυξη του μοντέλου στο περιβάλλον του υπολογιστή του χρήστη, το μοντέλο προεκπαιδεύεται και στη συνέχεια οι παράμετροί του διατίθενται στη λειτουργία. Συμπεριλάμβάνεται το αρχείο των βαρών του μοντέλου στο πακέτο ανάπτυξης, ώστε να φορτωθεί δυναμικά στον κώδικα.

Όταν ο θεατής προσθέσει μία νέα βαθμολογία ανανεώνεται η συγκεκρίμενη στήλη που αντιστοιχεί στο user ID του. Το μοντέλο επανεκπαιδεύεται σε αυτά τα νέα δεδομένα μέσω του μοντέλου Fire που συνεργάζεται με το LightGCN για τον γρήγορο υπολογισμό των νέων συστάσεων, κατά την αίτηση του θεατή.

Αν ο χρήστης δεν είναι ήδη εγγεγραμμένος ή δεν έχει βαθμολογήσει κάποια ταινία, ενδεικτικά ταυτίζουμε τις πιθανές του προτιμήσεις με έναν τυχαίο εγγεγραμμένο θεατή που έχει ιστορικό παρακολούθησης ταινιών. Αυτό γίνεται διότι, η πρώτη κλήση για παραγωγή συστάσεων (cold start problem, ισχύει για όλα τα είδη συστάσεων) είναι συνήθως πιο αργή και ανακριβής γιατί δεν υπάρχει προϊστορία για τις προτιμήσεις του χρήστη, όμως οι επόμενες κλήσεις θα είναι πολύ ταχύτερες καθώς το μοντέλο είναι ήδη φορτωμένο.

## 2.3 Ανανεώση Συστάσεων

Μόλις το frontend λάβει την απόκριση από το API με τις νέες συστάσεις, προχωρά στην ανανέωση του περιεχομένου που εμφανίζεται στον χρήστη. Συγκεκριμένα, ο κώδικας στο frontend ενημερώνει τη συγκεκριμένη κλάση με τη νέα λίστα συστάσεων.

Πιο συγκεκριμένα, ο χρήστης πατά το κουμπί σύστασης ταινιών και μεταβαίνει σε μια νέα σελίδα που επεξεργάζεται τις ταινίες που έχει δει με βάση το id τους. Ενεργοποιείται ο Flask server

και στέλνει στον PHP server, σε μορφή JSON, τα id των ταινιών που είναι τα πορίσματα των 2 αλγορίθμων. Ο PHP server αντιστοιχεί τα id στους τίτλους και τα στέλνει στο UI.

Έτσι, χωρίς να απαιτείται πλήρης ανανέωση της σελίδας, ο χρήστης βλέπει δυναμικά ανανεωμένες προτάσεις. Η εμπειρία του θεατή είναι ομαλή, καθώς οι αλλαγές εμφανίζονται άμεσα στη διεπαφή και του επιτρέπουν να συνεχίσει την περιήγησή του αξιοποιώντας τις νέες προτάσεις.

## 2.4 Αποθήκευση Δεδομένων

Όλες οι πληροφορίες που χρειάζεται το σύστημα συστάσεων (τα ιστορικά αλληλεπίδρασης χρηστών) διατηρούνται σε αρχεία CSV. Συγκεκριμένα αποθηκεύουμε στο ακόλουθα αρχεία:

- processed_movies.csv: περιέχει τις στήλες, movieId (ακέραιος δείκτης που ξεκινά από 0 και αναπαριστά μοναδικά κάθε ταινία), title (τίτλος ταινίας με το έτος κυκλοφορίας, π.χ. “Toy Story (1995)”) και genres (λίστα κωδικών ειδών χωρισμένων με ‘|’, π.χ. 12|16|35|14 αντιστοιχεί σε Adventure, Animation, Comedy, Fantasy) και χρησιμοποιείται για να συνδέσει τα metadata των ταινιών με τις αξιολογήσεις και για content based φιλτράρισμα στο σύστημα συστάσεων.
- processed_ratings.csv: περιέχει τις στήλες userId (εσωτερικός κωδικός χρήστη, ξεκινά από 0), movieId (εσωτερικός κωδικός ταινίας, ξεκινά από 0), rating (κανονικοποιημένη βαθμολογία 0–1), timestamp (χρονική σήμανση UNIX) και m (side info, εξήγηση ακολουθεί παρακάτω) και χρησιμοποιείται για εκπαίδευση και αξιολόγηση των προτιμήσεων στο μοντέλο συστάσεων.
- movie_genres.csv: έχει δύο στήλες: genre_id (αριθμητικός κωδικός της κατηγορίας ταινίας) και genre_name (το όνομα της κατηγορίας), και χρησιμεύει για να μετατρέπουμε τους αριθμητικούς κωδικούς σε αναγνώσιμα ονόματα .
- movie_metadata.csv: περιέχει επτά στήλες, movieId (εσωτερικός ακέραιος δείκτης που συνδέει κάθε εγγραφή με την ταινία στο σύστημα), title (ο τίτλος της ταινίας), actors (λίστα κορυφαίων ηθοποιών χωρισμένη με κόμμα), director (όνομα σκηνοθέτη), release_date (ημερομηνία κυκλοφορίας σε μορφή YYYY-MM-DD), summary (περίληψη/συστολή πλοκής ως κείμενο) και duration (διάρκεια προβολής σε λεπτά). Χρησιμοποιείται για να εμπλουτίσει το σύστημα συστάσεων ή τη βάση δεδομένων με λεπτομέρειες κάθε ταινίας.
- users.csv: περιέχει τρεις στήλες, το userId (ακέραιος δείκτης που αναγνωρίζει μοναδικά κάθε χρήστη), το username (το όνομα με το οποίο συνδέεται ο χρήστης) και το password (η κρυπτογραφημένη τιμή του κωδικού πρόσβασης), και χρησιμοποιείται για την αυθεντικοποίηση των χρηστών στην εφαρμογή.
- watchlist.csv: περιέχει δύο στήλες, το userId (εσωτερικός ακέραιος κωδικός που αναγνωρίζει κάθε χρήστη) και το movieId (εσωτερικός ακέραιος κωδικός που αναγνωρίζει κάθε ταινία) και χρησιμοποιείται για να συσχετίζει κάθε χρήστη με τις ταινίες που έχει δει, βαθμολογήσει ή προσθέσει στη λίστα του.

## 2.5 Προεπεξεργασία Δεδομένων

Η προεπεξεργασία των δεδομένων που λάβαμε από το MovieLens 100K dataset γίνεται σε 2 στάδια, το καθένα στο δικό του αρχείο python.

Αρχικά στο data_preprocessing.py:

- Διορθώνουμε ένα λάθος στην σειρά της χρονολογίας ορισμένων τίτλων.
- Αλλάζουμε το scale των βαθμολογιών όλων των ταινιών, από 1-5 σε 0-1 για πιο έγκυρα και γρήγορα αποτελέσματα των αλγορίθμων μηχανικής μάθησης.
- Κωδικοποιούμε όλα τα ids να ξεκινούν από το 0.

Έπειτα στο preprocess.py, χρησιμοποιούμε τα αποτελέσματα του παραπάνω αρχείου:

- Αντικαθιστούμε στο processed_movies.csv, το string του ονόματος της κατηγορίας με id της, όπως φαίνεται από το movie_genres.csv.
- Παράγουμε τα train_matrix.npz, test_matrix.npz και train_matrix.npy, το train_matrix.npy είναι ένα απλό αρχείο numpy που αποθηκεύει έναν δισδιάστατο πίνακα με τα δεδομένα εκπαίδευσης. Το test_matrix.npz είναι ένα συμπιεσμένο αρχείο numpy που περιέχει τους πινάκες για το σετ ελέγχου. Το train_matrix.npz είναι επίσης συμπιεσμένο αρχείο numpy που περιέχει τους πινάκες για το σετ εκπαίδευσης.

Επιπλέον για την συλλογή των λεπτομερειών των ταινιών που βρίσκονται στο movie_metadata.csv, χρειάστηκε η δημιουργία ενός ακόμη αρχείου του get_metadata.py. Με βάση το tmdbId που υπάρχει για κάθε movieId στο links.csv, το οποίο παρέχεται από το MovieLens 100K dataset, για κάθε ταινία παίρνουμε τις λεπτομέρειες της μέσω του API της TMDB, μία ιστοσελίδα που έχει δεδομένα για πολλές ταινίες. Έτσι αποθηκεύουμε πληροφορίες για τους α) 5 βασικούς πρωταγωνιστές, β) σκηνοθέτη, γ) ημερομηνία έκδοσης, δ) περίληψη και ε) διάρκεια σε λεπτά. Να σημειωθεί ότι για μερικές ταινίες δεν υπήρχαν επαρκής δεδομένα.

# 3. Εργαλεία και Τεχνολογίες Ανάπτυξης

## 3.1 Γλώσσες Προγραμματισμού

Για την ανάπτυξη του συστήματος, χρησιμοποιήθηκαν γλώσσες προγραμματισμού τόσο για την υλοποίηση του αλγορίθμου μηχανικής μάθησης, όσο και για την δημιουργία της διαδικτυακής πλατφόρμας. Χρησιμοποιήθηκε Python για την ανάπτυξη του αλγορίθμου, ενώ για την δημιουργία ιστοσελίδων αξιοποιήθηκε η βιβλιοθήκη React της Javascript.

## Python

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε η Python, μια δυναμική, ερμηνευόμενη γλώσσα υψηλού επιπέδου με απλή, ευανάγνωστη σύνταξη που επιτρέπει ταχύτερη γραφή και συντήρηση κώδικα. Η μεγάλη κοινότητά της και το πλήθος βιβλιοθηκών που

υποστηρίζει καλύπτουν όλες τις ανάγκες από την προεπεξεργασία δεδομένων μέχρι την απεικόνιση τους.

Η Python αποτελεί πλέον την κλασσική γλώσσα υλοποίησης αλγορίθμων μηχανικής μάθησης, πράγμα που γίνεται εφικτό μέσω βιβλιοθηκών όπως η Pytorch, που είναι εξειδικευμένες στη βαθιά μάθηση.

### PHP

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε η PHP, μια δυναμική, ερμηνεύσιμη γλώσσα, που διευκολύνει τη γρήγορη γραφή και συντήρηση κώδικα. Η τεράστια κοινότητα της PHP και ο πλούτος πακέτων, καλύπτουν κάθε ανάγκη, από την απλή διαχείριση φορμών και συνδέσεων σε βάσεις δεδομένων έως τη δημιουργία RESTful API και CLI εργαλείων.

Η PHP παραμένει η κλασική επιλογή για ανάπτυξη web εφαρμογών, και την χρησιμοποιούμε ως API για την επικοινωνία με του frontend με το backend.

### React

Η React είναι μία open source βιβλιοθήκη της διαδεδομένης γλώσσας προγραμματισμού Javascript, που αναπτύσσει η εταιρία Meta, και αλλάζει την παραδοσιακή παραγωγή ιστοσελίδων με HTML. Αντί ο προγραμματιστής να γράφει μία μεγάλη σελίδα HTML, με scripts που αλλάζουν το DOM, η React επιτρέπει την δημιούργια διεπαφών χρήστη (UI) ως επαναχρησιμοποιούμενα κομμάτια (components).

Το React λειτουργεί δημιουργώντας ένα εικονικό DOM στη μνήμη αντί να χειρίζεται απευθείας το DOM του προγράμματος περιήγησης. Πραγματοποιεί τις απαραίτητες μετατροπές σε αυτήν την εικονική αναπαράσταση πριν εφαρμόσει τις αλλαγές στο πραγματικό DOM . Έτσι απλοποιεί τη δημιουργία διαδικτυακών εφαρμογών δίνοντας ιδιαίτερη έμφαση στην απόδοση και τη συντηρησιμότητα του λογισμικού.

Δηλαδή η React δεν αντικαθιστά το HTML, απλά προτείνει έναν πιο βολικό και προγραμματιστικό τρόπο υλοποίησης του HTML κώδικα, χωρίς ο προγραμματιστής να χρειαστεί να γράψει άμεσα HTML.

Το DOM (Document Object Model) είναι το δέντρο που αναπαριστά τη σελίδα HTML (μαζί με το CSS). Κάθε στοιχείο HTML αντιστοιχεί σε έναν κόμβο στο DOM.

## 3.2 Βασικές Βιβλιοθήκες

Εκτός από τις γλώσσες προγραμματισμού, χρησιμοποιήθηκαν και βιβλιοθήκες που αυξάνουν τις δυνατότητες τους.

### NumPy

Στον πυρήνα των αριθμητικών υπολογισμών βρίσκεται η βιβλιοθήκη NumPy, η οποία παρέχει το πολυδιάστατους πίνακες αποδοτικά υλοποιημένους σε C και ένα ευρύ σύνολο συναρτήσεων για γραμμική άλγεβρα, στατιστική και άλλα.

## Pandas

Η βιβλιοθήκη Pandas δίνει τα εργαλεία για την προετοιμασία των datasets που χρειάζονται για να εκπαιδευτούν οι αλγόριθμοι μηχανικής μάθησης. Με μεθόδους ομαδοποίησης, καθαρισμού και γενικότερα επεξεργασίας των δεδομένων θεωρείται απαραίτητη.

## Pytorch

Η Pytorch, με γνώμονα το Tensor (πολυδιάστατους πίνακες), μπορεί να επιταχυνθεί σε GPU, ενσωματώνει έναν αυτόματο υπολογισμό παραγώγων μέσω υπολογιστικών γραφημάτων, προσφέροντας ευελιξία στο σχεδιασμό μοντέλων. Είναι σχεδιασμένη για τις εφαρμογές της βαθιάς μάθησης παρέχοντας έτοιμα επίπεδα, συναρτήσεις βελτιστοποίησης, κόστους, εκπαίδευσης και πολλά άλλα. Επιπλέον, η οικογένεια βιβλιοθηκών TorchVision, TorchText και TorchAudio διευκολύνει την πρόσβαση σε datasets και μετασχηματισμούς για εικόνες, κείμενο και ήχο, ενώ παρέχει και εργαλεία που επιτρέπουν την εξαγωγή μοντέλων σε περιβάλλοντα παραγωγής.

## Flask

Το Flask είναι ένα ελαφρύ framework για web εφαρμογές σε Python που βασίζεται στην ιδέα του routing (δημιουργία διαφρομών) και της δημιουργίας εφαρμογών γύρω από την κλάση Flask. Με τις μεθόδους app.route γίνεται ο χειρισμός των εισερχόμενων HTTP αιτημάτων, ενώ με το jsonify επιστρέφονται εύκολα JSON απαντήσεις σε clients.

Αν και χρησιμοποιούμε PHP για το API το Flask app αποτελεί μια διασύνδεση μεταξύ του PHP και των αλγορίθμων στην Python, και το χρησιμοποιούμε μόνο για την διαδικασία σύστασης

## Scipy

Το Scipy είναι μια βιβλιοθήκη επιστημονικών υπολογισμών σε Python, που βασίζεται στο Numpy και παρέχει εξειδικευμένα εργαλεία για βελτιστοποίηση, στατιστική, επεξεργασία σήματος και άλλα. Το πακέτο της, scipy.sparse χειρίζεται αραιούς πίνακες, επιτρέποντας αποδοτικούς αλγορίθμους γραμμικής άλγεβρας σε δεδομένα όπου τα περισσότερα στοιχεία των πινάκων είναι μηδενικά (sparce matrices).

## 3.3 Εισαγωγή στο LightGCN

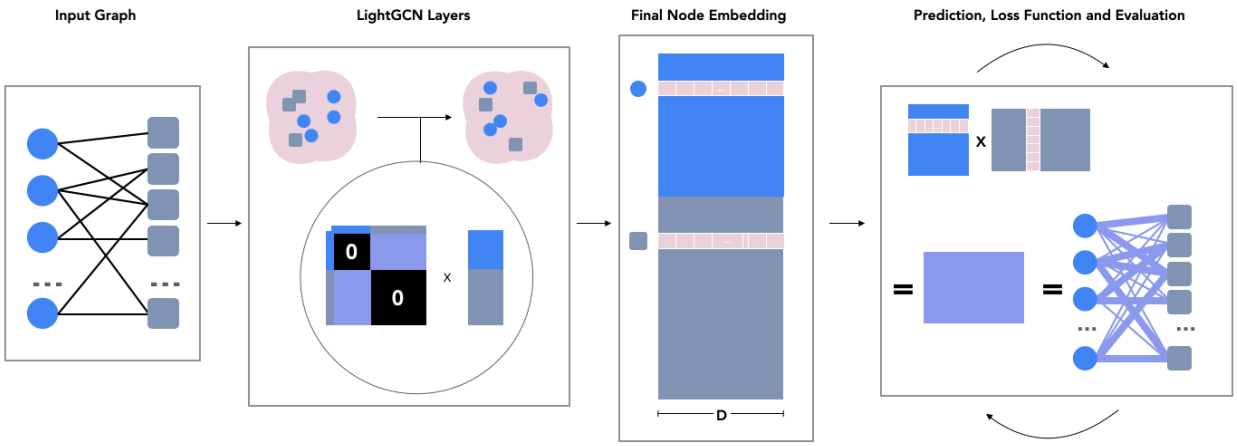
Ο LightGCN αποτελεί την λογική με την οποία παράγονται οι προτάσεις των ταινιών, δηλαδή οι προβλέψεις του συστήματος για το ποιά ταινία θα άρεσε περισσότερο στον χρήστη με βάση τις προηγούμενες ταινίες που έχει δει. Όπως οι περισσότεροι αλγόριθμοι σύστασης, βασίζεται σε Graph Neural Network (GNN). Σε αντίθεση όμως με τα συνηθισμένα πιο περίπλοκα

δίκτυα γραφημάτων, το LightGCN κρατά μόνο τα απολύτως απαραίτητα στοιχεία, τις σχέσεις που συνδέουν τον χρήστη με τα αντικείμενα, που στην προκειμένη περίπτωση είναι η βαθμολογία που δίνει ο θεατής στις ταινίες που έχει δει.

Πρακτικά ο αλγόριθμος προσπαθεί να αναθέσει σε κάθε θεατή και κάθε ταινία ένα διανυσματικό "προφίλ" (embedding), έτσι ώστε παρόμοιοι θεατές ή ταινίες να έχουν κοντινά διανύσματα. Τελικός σκοπός του αλγορίθμου είναι για κάθε θεατή να βαθμολογήσει νοητά όλες τις ταινίες που δεν έχει δει, και να τις κατατάξει με αυτήν την βαθμολογία ξεκινώντας από αυτές που δεν πιστεύει ότι αυτός ο χρήστης θα έβλεπε και καταλήγοντας σε αυτές που πιστεύει ότι θα ήθελε να δει πολύ.

Το πρόβλημα της σύστασης έτσι μετατρέπεται σε ένα πρόβλημα πρόβλεψης συνδέσμων σε έναν γράφο. Το μοντέλο προσπαθεί να προβλέψει ποιες ακμές (συνδέσεις θεατή-ταινίας) που δεν υπάρχουν ακόμη θα έπρεπε να υπάρχουν, δηλαδή ποιες ταινίες θα άρεσαν σε ποιον χρήστη.

Εδώ συνδέεται και η συνέλιξη (convolution) που προσθέθηκε στην κλασσική αρχιτεκτονική των GNN ώστε να μετατραπεί σε GCN (Graph Convolutional Network). Η μαθηματική πράξη της συνέλιξης, σε πρακτικό επίπεδο εντοπίζει συγκεκριμένα χαρακτηριστικά ανάλογα με τους αριθμούς που υπάρχουν στον πίνακα της συνέλιξης. Για το LightGCN τα χαρακτηριστικά είναι οι σχέσεις μεταξύ θεατών και ταινιών. Αυτή η μινιμαλιστική σχεδίαση μειώνει την πολυπλοκότητα και το κόστος υπολογισμού, ενώ παράλληλα συλλαμβάνει αποτελεσματικά τη δομή του γράφου.




Εικόνα 2: Λειτουργικό διάγραμμα του LightGCN

## 3.4 Πώς διαφέρει από άλλες μεθόδους σύστασης

Το LightGCN προέκυψε από την παρατήρηση ότι οι κλασικές αρχιτεκτονικές GNN ήταν υπερβολικά πολύπλοκες και περιττές για το πρόβλημα του φιλτραρίσματος αντικειμένων. Στις συστάσεις, οι κόμβοι (χρήστες και αντικείμενα) δεν έχουν πλούσια αρχικά χαρακτηριστικά και ουσιαστικά αναπαρίστανται μόνο από το μοναδικό τους ID. Με άλλα λόγια, κάθε χρήστης/αντικείμενο ξεκινά με ένα αναγνωριστικό χωρίς νόημα πέρα από την ταυτότητά του. Σε

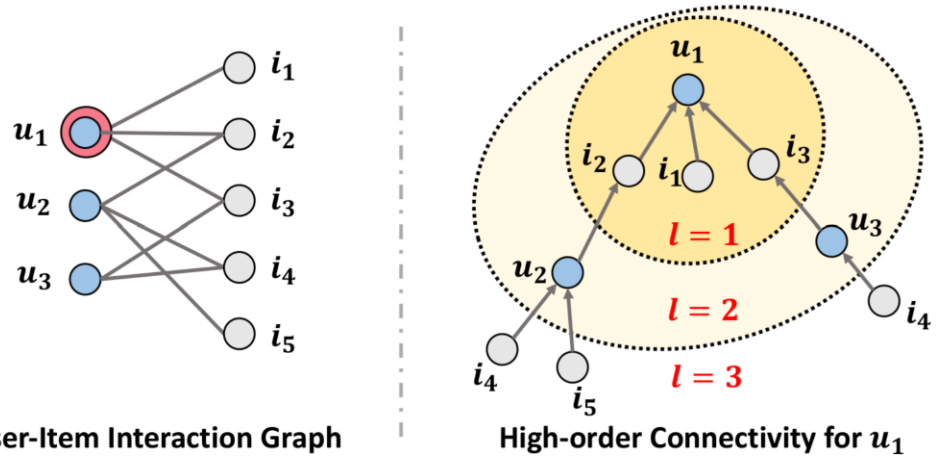
μια τέτοια περίπτωση, το να εφαρμόσουμε πολλαπλά στρώματα με μη γραμμικούς μετασχηματισμούς κάνει το μοντέλο πιο δύσκολο να εκπαιδευτεί. Ερευνητές διαπίστωσαν πειραματικά ότι αφαιρώντας αυτά τα στοιχεία η απόδοση όχι μόνο δεν πέφτει, αλλά βελτιώνεται σημαντικά.

## 3.5 Αρχιτεκτονική του LightGCN

Ο γράφος του LightGCN ονομάζεται διμερές γράφημα (bipertite graph), και το βασικό του χαρακτηριστικό είναι ότι οι κόμβοι του γράφου ανοίκουν αναγκαστικά σε 2 κατηγορίες χρήστες και αντικείμενα (στη προκειμένη περίπτωση θεατές και ταινίες), σχηματίζοντας ένα γράφημα όπου ακμές υπάρχουν μόνο μεταξύ χρήστη και αντικειμένου και όχι μεταξύ 2 χρηστών ή 2 αντικειμένων. Κάθε χρήστης και κάθε αντικείμενο ξεκινά με ένα αρχικό διάνυσμα που είναι ουσιαστικά τα μόνα παραμετροποιήσιμα μεγέθη του μοντέλου. Δηλαδή, δεν χρειάζεται κανένα επιπλέον γνώρισμα για τους κόμβους πέρα από αυτά τα ενσωματωμένα διανύσματα τα οποία μαθαίνονται κατά τη διάρκεια της εκπαίδευσης του μοντέλου, όπως σε όλους τους αλγορίθμους μηχανικής μάθησης.

Σε επίπεδο αρχιτεκτονικής, το LightGCN πραγματοποιεί "K" επίπεδα συνελικτικών βημάτων διάδοσης (propagation layers) πάνω στο γράφημα. Σε κάθε τέτοιο επίπεδο, μεταδίδεται η πληροφορία από τους γείτονες ενός κόμβου προς τον ίδιο. Αντίστοιχα, το embedding ενός αντικειμένου ενημερώνεται από τα embeddings των χρηστών που το έχουν ως γείτονές τους.

Μετά την ολοκλήρωση όλων των επιπέδων διάδοσης, υπολογίζει το τελικό embedding κάθε κόμβου (χρήστη ή αντικειμένου) ως έναν συνδυασμό όλων των επιπέδων. Με αυτό τον συνδυασμό, το τελικό διάνυσμα του χρήστη περιλαμβάνει τόσο το δικό του αρχικό embedding, όσο και τις πληροφορίες από τους γείτονες κάθε βαθμίδας. Διατηρούμε έτσι την πληροφορία τόσο από κοντινές όσο και από πιο μακρινές σχέσεις, αποφεύγοντας την πλήρη αλλοίωση της μοναδικότητας κάθε κόμβου




Εικόνα 3: Αριστερά, διμερές γράφημα, Δεξιά, τα διαφορετικά επίπεδα με τους κόμβους και γείτονες τους

## 3.6 Εισαγωγή στο FIRE

Στα συστήματα σύστασης, οι νέες αλληλεπιδράσεις χρηστών και αντικειμένων καταγράφονται συνεχώς, απαιτώντας γρήγορη ενημέρωση του μοντέλου για διατήρηση της απόδοσης. Οι παραδοσιακές προσεγγίσεις με Graph Neural Networks (όπως το LightGCN) χρειάζονται χρονοβόρα επανεκπαίδευσή, γεγονός που περιορίζει τη συχνότητα ανανέωσης των συστάσεων.

Η μέθοδος FIRE σχεδιάστηκε για να αντιμετωπίσει αυτές τις προκλήσεις, προσφέροντας μία μη παραμετρική λύση που επιτρέπει ενημερώσεις σε O(nnz) χρόνο (όπου nnz ο αριθμός μη μηδενικών στοιχείων στον πίνακα αλληλεπιδράσεων) χωρίς την ανάγκη για χρονοβόρες επανεκπαιδεύσεις. Σε αυτόν τον χρόνο συμπεριλαμβάνεται και η εκπαίδευση του.

Βασίζεται στην επεξεργασία σημάτων σε γράφους (GSP), εφαρμόζοντας ειδικά φίλτρα για να ενσωματώσει τόσο τη χρονική δυναμική των αλληλεπιδράσεων όσο και πλευρικές πληροφορίες (side info), διατηρώντας παράλληλα έναν πίνακα αλληλεπιδράσεων και έναν πίνακα χρονικών στιγμών (για αυτό στο processed_ratings.csv διατηρούμε τις στήλες timestamp και m).

## 3.7 Αρχιτεκτονική του FIRE

Το FIRE μεταχειρίζεται το διμερές γράφημα θεατών ταινίων ως σήμα, εφαρμόζοντας φίλτρα που εξομαλύνουν (με low pass και high pass) ή διατηρούν συστατικά με βάση τη δομή του γράφου. Αποφεύγεται έτσι το κόστος του gradient descent, αφού όλες οι πράξεις γίνονται γραμμικές.

Αποθηκεύονται δύο αραιά matrices, ένα βαθμολογιών και ένα χρονοσφραγίδων για τις παλαιές αλληλεπιδράσεις.

Τα Φίλτρα Πλευρικής Πληροφορίας (Side Info Filters) χωρίζονται σε user based filter ($P_1$) και σε item based filter ($P_2$).

Το user based filter κατασκευάζεται από user-user similarity matrix (cosine similarity), κανονικοποιείται και εφαρμόζεται για την βελτίωση της εξατομίκευσης. Το σήμα του τρέχοντος χρήστη ενισχύεται από τις αλληλεπιδράσεις άλλων χρηστών με παρόμοιο προφίλ, ώστε να λαμβάνονται υπόψη συλλογικά μοτίβα συμπεριφοράς.

Το item based filter κατασκευάζεται από το αντίστοιχο item-item φίλτρο που αξιοποιεί similarity matrix αντικειμένων για πιο έγκυρες συστάσεις.

Το νεοσχηματιζόμενο adjacency matrix, δηλάδη με τον συνδιασμό του ιστορικού και της τωρινής κατάστασης όλων των θεατών, κανονικοποιείται συμμετρικά ώστε να εξασφαλίζεται σταθερότητα στη διάδοση σήματος.

Εφαρμόζεται sparsesvd ($P_3$) στον κανονικοποιημένο πίνακα για την εξαγωγή των βασικών χαρακτηριστικών. Η SVD αντιστοιχεί σε ένα ιδανικό low pass graph filter, που εντοπίζει τα ολικά (από όλο το dataset) συνεργατικά μοτίβα. Στην υλοποίηση μας, δεν χρησιμοποιούμε την sparsesvd που παρέχεται από την βιβλιοθήκη με αντίστοιχο όνομα, αλλά το svds από την βιβλιοθήκη scipy.sparce με αντίστοιχο αποτέλεσμα.

Singular Value Decomposition, SVD (Παράγοντοποίηση Μονών Τιμών) είναι μια μέθοδος γραμμικής άλγεβρας που διασπά έναν πίνακα σε τρία απλούστερα συστατικά, επιτρέποντας αποδοτική ανάλυση και μείωση της διάστασης των δεδομένων.

Τέλος, τα $P_1$, $P_2$ και $P_3$ αθροίζονται για το τελικό score matrix. Έτσι, επιτυγχάνεται ένα mixed frequency graph filter που συνδυάζει ιδανικές και γραμμικές συνιστώσες.

Το FIRE επιστρέφει για κάθε χρήστη τις κορυφαίες θέσεις βάσει του τελικού score, έτοιμες για reranking με το LightGCN.

## 3.8 Αλληλεπίδραση και Συνεργασία LightGCN και FIRE

Αρχικά, πριν την δημοσίευση του συστήματος έχουμε εκπαιδεύσει το LightGCN μοντέλο με βάση όλα τα δεδομένα των πινάκων. Αποθηκεύουμε το αποτέλεσμα στο αρχείο lightgcn_final.pth και το χρησιμοποιούμε τις σχέσεις που αντιπροσωπεύει, ως βάση για όλες τις μετέπειτα επανεκπαιδεύσεις.

Κάθε φορά που χρήστης ζητά συστάσεις, πρώτα, το FIRE φιλτράρει το σύνολο των ταινιών από 10.000 (που βρίσκονται στο dataset) σε μερικές εκατοντάδες υποψήφιους για όλους τους χρήστες. Στη συνέχεια, το LightGCN επανεκπαιδεύεται μόνο πάνω σε αυτό το μικρό υποσύνολο, μειώνοντας το κόστος κατά πολύ.

Το LightGCN έπειτα, μαθαίνει τα embeddings των θεατών και των ταινίων στο διμερές γράφημα, παρέχοντας συνεργατικές αναπαραστάσεις. Πιο συγκεκριμένα το pipeline, για κάθε φορά που χρήστης ζητά τις συστάσεις, είναι το εξής:

1. **Candidate Generation**
    a. Φορτώνεται το μοντέλο FIRE για να παραχθεί ο πλήρης πίνακας score θεατή-αντικειμένου.
    b. Για κάθε χρήστη επιλέγονται οι κορυφαίες 100 υποψήφιες θέσεις. Αυτή η φάση είναι πολύ γρήγορη, καθώς βασίζεται σε sparse φίλτρα και low-rank SVD.
2. **Επανεκπαίδευση με LightGCN**
    a. Φορτώνεται το LightGCN με τα τελικά βάρη από την αρχική εκπαίδευση.
    b. Παράγονται τα embeddings θεατών και ταινιών.
    c. Για τον ζητούμενο θεατή υπολογίζεται το εσωτερικό του γινόμενο μεταξύ του embedding τού και των embeddings των 100 υποψήφιων, και επιλέγονται οι τελικές 20.

**<u>Να σημειωθεί επειδή δεν υπάρχει τυχαιότητα στους 2 αλγορίθμους, αν δεν γίνει κάποια ιδιαίτερη αλλαγή στις βαθμολογίες που έχει δώσει ο θεατής οι συστάσεις που θα παράγονται θα είναι πάντα ίδιες.</u>**

# 4. Αναμενόμενη είσοδος και έξοδος

Για τη δημιουργία προτάσεων μέσω του LightGCN και του FIRE, η είσοδος είναι το user ID του θεατή για τον οποίο θέλουμε συστάσεις. Με αυτό το δεδομένο, το μοντέλο ανακτά το διάνυσμα που αντιστοιχεί σε αυτόν τον θεατή και παράχθηκε κατά την εκπαίδευση.

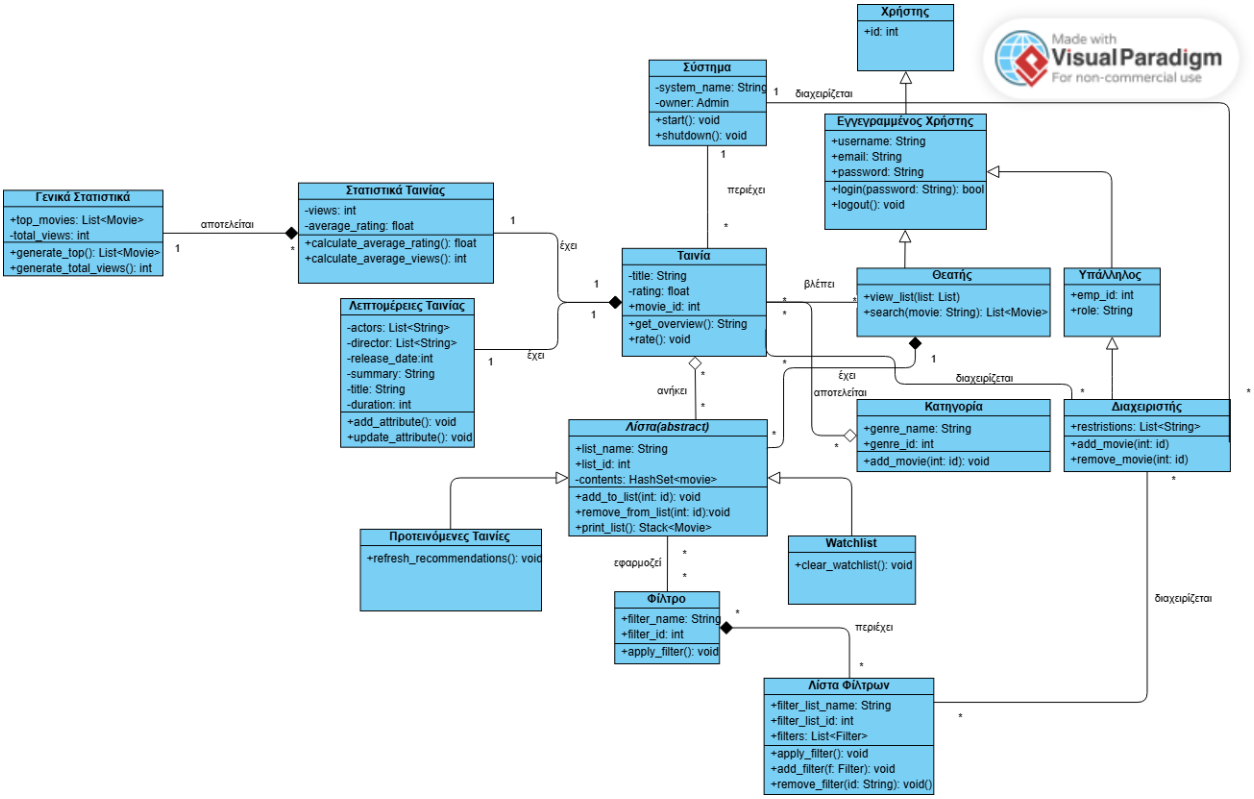
Πρακτικά το σύστημα λαμβάνει αυτόματα το user ID του συνδεδεμένου χρήστη στη συσκευή και κάθε φορά που θεατής βαθμολογεί μια νέα ταινία, το API αναβαθμίζει τα αντίστοιχα στοιχεία του πίνακα csv, προσθέτοντας μια νέα γραμμή με user ID την νεά βαθμολογία και το movie ID της βαθμολογημένης ταινίας

Η έξοδος του αλγορίθμου είναι μια βαθμολογία συνάφειας για κάθε ταινία, που υποδεικνύει πόσο πιθανό είναι ο χρήστης να προτιμήσει αυτή τη ταινία. Το LightGCN χρησιμοποιεί το εσωτερικό γινόμενο του διανύσματος του θεατή και του διανύσματος της ταινίας, που δίνει ως αποτέλεσμα τη βαθμολογία προτίμησης. Αυτές οι βαθμολογίες στη συνέχεια να ταξινομούνται για να δημιουργήσουν την τελική λίστα με τις κορυφαίες 20 ταινίες. Να σημειωθεί ότι το FIRE δεν παράγει ταξινομημένη λίστα, παράγει ένα σύνολο ταινιών μικρότερο από το ολικό.

Οι προτεινόμενες ταινίες είναι αυτές με τις υψηλότερες βαθμολογίες. Έπειτα μέσω του API στέλνονται οι συστάσεις στο frontend για να τις λάβει ο θεατής.

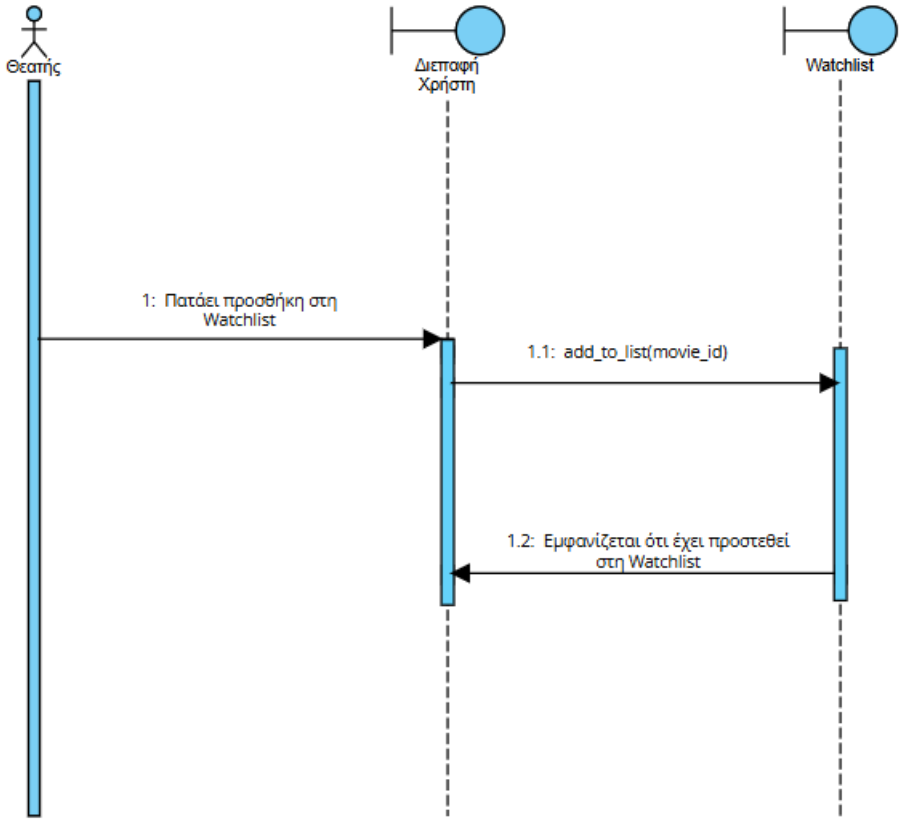
# 5. Σχεδίαση Πλατφόρμας

## 5.1 Διάγραμμα Κλάσεων



## 5.2 Διαγράμματα Ακολουθίας

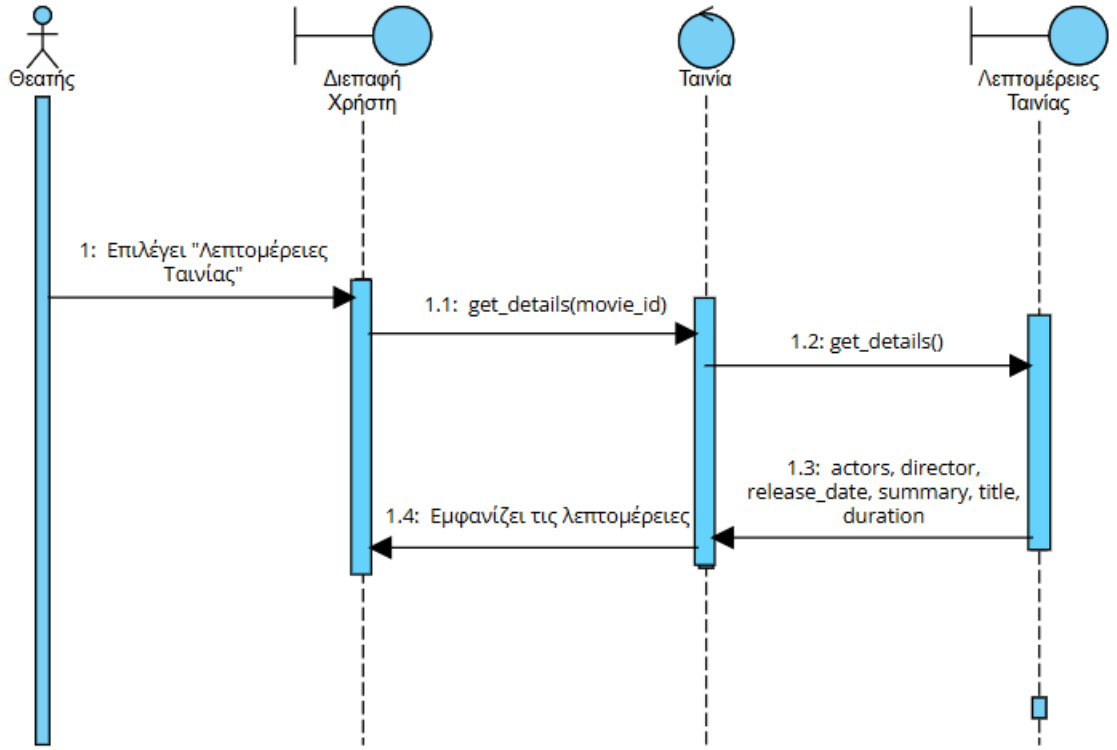
Διάγραμμα ακολουθίας για την περίπτωση «Προσθήκη στη Watchlist»




Το παραπάνω διάγραμμα ακολουθίας παρουσιάζει τη διαδικασία με την οποία ένας θεατής προσθέτει μια ταινία που επιθυμεί στη λίστα Watchlist. Αρχικά, ο θεατής πατάει την επιλογή "Προσθήκη στη Watchlist" (βήμα 1). Η Διεπαφή Χρήστη καλεί τη συνάρτηση add_to_list(movie_id) (βήμα 1.1), περνώντας το αναγνωριστικό της ταινίας. Η ενέργεια αυτή αποστέλλεται στην κλάση Watchlist, η οποία χειρίζεται την αποθήκευση της ταινίας στη λίστα

του χρήστη. Μόλις ολοκληρωθεί η προσθήκη, η Διεπαφή Χρήστη λαμβάνει επιβεβαίωση και εμφανίζει μήνυμα που ενημερώνει ότι η ταινία έχει προστεθεί επιτυχώς στη Watchlist (βήμα

Διάγραμμα ακολουθίας για την περίπτωση «Προβολή Λεπτομερειών Ταινίας»




Το παραπάνω διάγραμμα ακολουθίας απεικονίζει τη διαδικασία προβολής των λεπτομερειών μιας ταινίας. Ο θεατής ουσιαστικά, πατάει στο κουμπί "Λεπτομέρειες Ταινίας" (βήμα 1), και η Διεπαφή Χρήστη καλεί τη συνάρτηση get_details(movie_id) (βήμα 1.1), περνώντας το αναγνωριστικό της επιλεγμένης ταινίας. Η κλάση Ταινία, λειτουργώντας ως ενδιάμεσο controller, ζητά τις λεπτομέρειες από την κλάση Λεπτομέρειες Ταινίας μέσω της get_details() (βήμα 1.2). Αυτή επιστρέφει όλα τα απαραίτητα δεδομένα της ταινίας, όπως ηθοποιούς, σκηνοθέτη, ημερομηνία κυκλοφορίας, περίληψη, τίτλο και διάρκεια (βήμα 1.3). Τέλος, τα δεδομένα αποστέλλονται πίσω στη Διεπαφή Χρήστη, όπου και προβάλλονται στον θεατή (βήμα 1.4).

## Βιβλιογραφία

Xiangnan He, Kuan Deng, Xiang Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation"

Jiafeng Xia, Dongsheng Li, Hansu Gu, "FIRE: Fast Incremental Recommendation with Graph Signal Processing"