Platformă Web pentru Streaming și Recomandare de Filme

Aplicații Multimedia

Zărnescu Dragoș-Ioan
Grupa 331AB

AN: 2025

Cuprins

1 Introducere **1**

2 Suportul Tehnic și Realizări Similare **2**

3 Prezentare tehnică a etapei de realizare/implementare **3**
3.1 Arhitectură generală și organizare modulară 3
3.2 Stack-ul tehnologic și biblioteci folosite 4
3.2.1 Frontend – React.js 4
3.2.2 Backend – Node.js + Express.js 4
3.2.3 Sistem de recomandare – Microserviciu Flask (Python) 5
3.2.4 Baza de date – PostgreSQL 5
3.3 Stocare și transcodare video 5
3.3.1 **Transcodare video cu FFmpeg (2-pass encoding)** 6
3.3.2 **Primul pas – Analiza video (fără output audio/video)** 6
3.3.3 **Al doilea pas – Compresie finală cu audio inclus** 6
3.3.4 **Stocare fișiere pe Google Cloud Storage** 7
3.4 **Sistem de recomandare – Microserviciu AI cu Flask** 7
3.4.1 **Arhitectură** 7
3.4.2 **Explicație matematică a modelului SVD** 8
3.4.3 **Ce învață de fapt modelul** 8

4 Mod de utilizare, interacțiune cu utilizatorul și configurare **8**
4.1 Navigare și interacțiune în interfața web 9
4.2 Funcționalități interactive 9
4.3 Configurare aplicație și rulare locală 10

5 Concluzii **11**
5.1 Obiective atinse 12
5.2 Utilitate și relevanță 12
5.3 Performanță și actualitate 12
5.4 Impact și perspective 12

6 Bibliografie **13**

1 Introducere

Întotdeauna unul din cele mai mari impedimente în a avea o seară minunată de relaxare este găsirea unui **film bun**. Nu știu de alte persoane, dar mie îmi ia mai mult să găsesc un film la care să mă uit decât să mă uit propriu-zis la el. Dar și atunci când îl găsesc fie intru pe un site cu sute de reclame și player-ul nu mai merge sau și mai rău să ajung să găsesc un film pe care îl căutam și să nu aibă niciun fel de subtitrare sau cu subtitrarea descentrată și eventual să vorbească și persană.

Astfel am ales această temă, ce constă în realizarea unei **platforme web pentru streaming de filme**, deoarece consider că este necesar într-o lume în care din raționamente de marketing și consumerism, se pierd ușor, ușor filmele ce au dat idei, direcții, un sens cinematografiei. Deși, în fapt, este cum spune **Tarkovsky**, *"Cinema is an unhappy art as it depends on the money. Not only because a film is very expensive but is then also marketed like cigarettes"*

Scopul principal este acela de a oferi o alternativă serviciilor **Netflix** sau **HBO MAX** unde se află un serviciu pay2watch, fără a avea prea multe recomandări relevante, în principal decât pe cele ale ultimelor filme apărute și fără o varietate de filme prea interesante. Propunându-mi să ofer deschidere către filme precum cele restaurate de **Criterion** cu o largă bază de date oferită de **TMDB** și cu o platformă ce dorește să ofere utilități precum cele de pe **Letterboxd**.

Desigur inspirația inițială a pornit de la site-urile ce se termină în onlinesubtitrat.ro, dar propunându-mi să combin toate aspectele menționate mai sus într-o singură platformă.

Obiectivele principale ale proiectului sunt:

- Realizarea unei platforme web responsive, conectată la o bază de date **PostgreSQL**, care gestionează **6500+** filme populate cu ajutorul API-ului TheMovieDB.
- Implementarea unui **player video** modern cu suport pentru subtitrări, control viteză, Picture-in-Picture și alte funcții avansate.
- Integrarea unei componente de învățare automată pentru recomandări personalizate bazate pe algoritmul **SVD** (Singular Value Decomposition).
- Permisiunea utilizatorilor de a adăuga rating-uri, comentarii și de a crea wishlist-uri personalizate.
- Posibilitatea importului de date sub formă de CSV de pe alte platforme populare pentru antrenarea și utilizarea modelului de recomandare.

2 Suportul Tehnic și Realizări Similare

Proiectul este dezvoltat folosind un stack tehnologic modern: frontend-ul este realizat în **React**, backend-ul în **Express.js**, iar baza de date folosită este **PostgreSQL**. Redarea conținutului video se face printr-un player HTML5 compatibil cu fișiere .mp4 transcodate cu **FFmpeg** și stocate în **Google Cloud Storage**.

Transcodarea este esențială pentru a asigura o redare fluidă și compatibilitate cu majoritatea browserelor. Formatul ales este **H.264 (video)** și AAC (audio), cu bitrate de 1000k pentru video și 64k pentru audio, folosind două treceri (2-pass encoding) pentru un echilibru optim între calitate și dimensiune. Flag-ul **+faststart** permite inițierea mai rapidă a streamingului prin relocarea metadatelor la începutul fișierului.

Pentru recomandări, este folosit algoritmul **SVD** din biblioteca Surprise. Acesta este antrenat pe evaluări ale utilizatorilor pentru filme, oferind predicții personalizate în funcție de istoricul fiecărui user. Modelul este antrenat pe un set complet de date, iar pentru fiecare film nevizionat se prezice scorul estimat, ordonându-se descrescător pentru a oferi cele mai relevante **sugestii**.

În ceea ce privește soluțiile similare, putem menționa următoarele:

- **Netflix** – lider în streamingul global, cu un motor complex de recomandări și suport pentru streaming adaptiv HLS/DASH. Puncte tari: calitate ridicată a serviciilor, scalabilitate, experiență UX. Puncte slabe: lipsa accesului la codul sursă, imposibilitatea personalizării de către utilizatori. [Gomez-Uribe & Hunt, 2015]
- **Plex** – platformă orientată spre self-hosting, cu funcții avansate de organizare și streaming. Puncte tari: control local complet. Puncte slabe: dependența de configurația hardware și lipsa unui sistem avansat de recomandări. [Plex.tv]
- **Popcorn Time** – aplicație open-source care folosește torrente pentru a reda filme. Puncte tari: acces rapid, cod open-source. Puncte slabe: probleme legale și lipsa unui sistem de recomandări real. [Popcorn-time.tw]

Comparativ cu acestea, proiectul propus combină avantajele: este accesibil, extensibil, legal (în condiții necomerciale) și include un sistem propriu de recomandări personalizate.

Aspecte multimedia fundamentale:

- **Protocoale de streaming:** HTTP este folosit pentru redare progresivă, iar aplicația poate fi extinsă pentru suport HLS (HTTP Live Streaming), ce fragmentează video-ul și adaptează calitatea în funcție de conexiune.
- **Compresia video:** Se folosește codec-ul H.264, eficient și larg suportat. Compresia reduce lățimea de bandă necesară, păstrând o calitate acceptabilă.
- **Compresia audio:** AAC este folosit pentru eficiență și compatibilitate largă.
- **Subtitrări:** Sunt suportate fișiere .vtt sau .srt, integrate în player-ul HTML5.

3 Prezentare tehnică a etapei de realizare/implementare

Realizarea acestei platforme de streaming a presupus proiectarea unei arhitecturi robuste, alegerea atentă a tehnologiilor și integrarea unor componente avansate, de la stocare video și transcodare, până la un sistem inteligent de recomandări. Proiectul este complet realizat de mine, de la concept și design până la implementare și testare, fără utilizarea de template-uri externe sau soluții comerciale preexistente.

3.1 Arhitectură generală și organizare modulară

Realizarea acestei platforme de streaming a presupus proiectarea unei arhitecturi robuste, alegerea atentă a tehnologiilor și integrarea unor componente avansate, de la stocare video și transcodare, până la un sistem inteligent de recomandări.

Arhitectura generală

Aplicația este construită pe o arhitectură de tip MVC (Model-View-Controller) în partea de backend, ceea ce permite o separare clară între date (Model), logica de procesare (Controller) și interfața de interacțiune prin API (View, în cazul nostru, rutele REST) ce ulterior vor fi fetch-uite în aplicația realizată în React.

- **Modelul** este responsabil de interacțiunea cu baza de date PostgreSQL, folosind ORM-uri sau interogări SQL directe. Sunt definite tabele pentru movies, users, ratings, comments, wishlist.
- **Controller-ele** implementează logica aplicației, de exemplu: adăugarea unui comentariu, procesarea unui CSV cu ratinguri, returnarea datelor unui film după ID, returnarea datelor unui utilizator.
- **Rutele** (partea de View) sunt definite în Express.js și expun API-uri REST:

```
1 router.get("/movies/:movieId/comments",
     listCommentsbyMovie);
2 router.get("/users/:userId/comments", listCommentsbyUser)
     ;
3 router.get("/users/:userId/ratings", listRatingsByUser);
4 router.post("/movies/:movieId/comments",
     authenticateToken, createComment);
5 router.put('/users/:userId/ratings/:movieId',changeRating
     );
6
```

Partea de recomandare este realizată separat arhitecturii MVC, folosind un server Flask pentru a procesa modelul de recomandare pe care îl voi explica ulterior când voi explica sistemul de recomandare.

3.2 Stack-ul tehnologic și biblioteci folosite

3.2.1 Frontend – React.js

- **React** – aplicația este dezvoltată folosind *functional components*, cu hook-uri precum useState și useEffect pentru gestionarea stării și a ciclului de viață al componentelor.
- **FETCH API** – folosit pentru efectuarea cererilor HTTP către backend-ul REST (e.g., pentru autentificare, obținerea filmelor, trimiterea comentariilor etc.).
- **React Router** – utilizat pentru navigația între paginile aplicației:
  - Rute precum /, /movie/:id, /login, /profile, /wishlist sunt gestionate prin Route și BrowserRouter.
- **Material UI (MUI)** – folosit pentru construirea și stilizarea interfeței grafice:
  - Butoane, carduri, ferestre modale, formulare și layout-uri responsive.
  - Design unitar și accesibil cu ajutorul temelor MUI.
- **Player video HTML5** – integrat direct prin tag-ul <video>, îmbunătățit cu JavaScript pentru funcționalități avansate:
  - Subtitrări .vtt – pot fi încărcate de utilizator și sincronizate cu redarea.
  - Controlul vitezei de redare – permite viteze între 0.5x și 2x.
  - Seeking – derulare rapidă înainte/înapoi prin slider sau shortcut-uri.
  - Fullscreen, volum, mute – controale standard personalizate.
  - Picture-in-Picture (PiP) – suport pentru vizionare în fereastră flotantă.

3.2.2 Backend – Node.js + Express.js

- **Express.js** – framework pentru rutare RESTful și construirea API-ului server-side.
- **pg** – client oficial pentru PostgreSQL, folosit pentru executarea de interogări SQL din Node.js.
- **cors** – middleware care activează CORS (Cross-Origin Resource Sharing), permițând comunicația între frontend-ul React și serverul Express chiar dacă sunt pe porturi diferite.
- **express.json** – middleware care permite parsarea automată a payload-ului de tip JSON din request-urile HTTP.
- **jsonwebtoken (JWT)** – pentru autentificare bazată pe token:
  - La login, utilizatorului i se generează un token JWT semnat.

  - Tokenul este trimis în headerul cererilor ulterioare (Authorization: Bearer <token>) și verificat în middleware înainte de accesarea rutelor protejate.

Stocare video – Google Cloud Storage

- Filmele în format .mp4 sunt stocate într-un bucket privat GCS.
- Accesarea fișierelor video se face prin linkuri semnate.
- Interacțiunea cu GCS se face inițial prin tool CLI (gsutil) și poate fi extinsă cu biblioteci oficiale.

3.2.3 Sistem de recomandare – Microserviciu Flask (Python)

Un server Flask rulează separat pentru gestionarea algoritmului de recomandare personalizată, antrenat pe baza ratingurilor utilizatorilor.

Librării Python utilizate:

- **Flask** – framework web minimalist pentru API REST.
- **flask_cors** – pentru activarea CORS, permițând comunicația între serverul Flask și aplicația React.
- **Surprise** – biblioteca principală folosită pentru sistemul de recomandare:
  - SVD (Singular Value Decomposition)
  - Dataset, Reader – pentru procesarea datelor de tip user–item–rating
- **pandas** – pentru manipularea datelor în format DataFrame.

3.2.4 Baza de date – PostgreSQL

- Bază relațională ce conține tabelele:
  - movies (id, titlu, video_url, descriere, etc.)
  - users (id, email, parola, etc.)
  - ratings (user_id, movie_id, rating)
  - comments (user_id, movie_id, comentariu)
  - wishlist (user_id, movie_id)
- Populare automată cu date din TMDb API folosind un script Node.js care trimite cereri asincrone și salvează răspunsurile în baza de date.

3.3 Stocare și transcodare video

Pentru a permite streaming fluid, compatibil cu majoritatea browserelor moderne, fișierele video originale (de mari dimensiuni, în unele cazuri depășind 2GB) sunt transcodate și optimizate înainte de a fi publicate în platformă. Procesul urmat include:

3.3.1 Transcodare video cu FFmpeg (2-pass encoding)

Am utilizat FFmpeg, o unealtă open-source robustă pentru procesare video/audio, pentru a realiza conversia fișierelor brute în format .mp4 (H.264 + AAC), formatul cel mai bine suportat de browsere și dispozitive moderne.

Procesul este compus din două faze, pentru a obține o compresie eficientă, menținând în același timp o calitate bună a imaginii:

3.3.2 Primul pas – Analiza video (fără output audio/video)

```
ffmpeg -y -i "$INPUT" \
-c:v libx264 -b:v $TARGET_VIDEO_BITRATE \
-pass 1 -an -f mp4 /dev/null
```

- `-c:v libx264` — folosește codec-ul H.264 pentru compresia video.
- `-b:v $TARGET_VIDEO_BITRATE` — setează bitrate-ul dorit (1000K folosit pentru rezoluție de 480p).
- `-pass 1` — marchează faptul că este prima trecere din cele două.
- `-an` — elimină audio pentru această fază.
- Outputul este aruncat în /dev/null – doar se analizează video-ul.

3.3.3 Al doilea pas – Compresie finală cu audio inclus

```
ffmpeg -i "$INPUT" \
-c:v libx264 -b:v $TARGET_VIDEO_BITRATE \
-c:a aac -b:a $TARGET_AUDIO_BITRATE \
-movflags +faststart \
-vf "scale='min(854,iw)':'min(480,ih)':
   force_original_aspect_ratio=decrease" \
-pass 2 "$OUTPUT"
```

- `-c:a aac -b:a 64k` — encode audio folosind AAC la 64kbps (eficient pentru streaming).
- `-movflags +faststart` — mută metadatele MP4-ului la începutul fișierului pentru streaming progresiv, permițând playerului să înceapă redarea imediat.
- `-vf "scale=..."` — redimensionează rezoluția la maximum 854x480, păstrând proporțiile originale.
- `-pass 2` — finalizează procesul pe baza analizei din primul pas.

Această metodă asigură o dimensiune redusă, compatibilitate ridicată și o experiență optimă de streaming pe toate platformele.

3.3.4 Stocare fișiere pe Google Cloud Storage

După procesul de transcodare, fișierele .mp4 rezultate sunt încărcate manual pe **Google Cloud Storage**, într-un bucket configurat public pentru streaming:

1. Mă conectez în consola GCP și accesez bucket-ul creat special pentru video-uri.
2. Încarc fișierul video din local.
3. Obțin URL-ul public de acces și îl actualizez în baza de date pentru filmul aferent.

3.4 Sistem de recomandare – Microserviciu AI cu Flask

Sistemul de recomandare nu este integrat direct în backendul Node.js, ci rulează ca un **microserviciu separat**, construit în Python cu **Flask**. Această decuplare permite o scalabilitate mai bună și evită blocarea serverului principal cu procese grele de ML.

3.4.1 Arhitectură

- Flask expune un endpoint /recommend/<user_id> pe un alt port față de portul serverului în Express care returnează filmele recomandate în format JSON.
- Flask rulează independent și are propriul script care antrenează modelul și răspunde la cereri.

```
@app.route("/recommend/<int:user_id>", methods=["GET"])
def recommend(user_id):
    if not model_ready:
        return jsonify({"error": "Model is not available"}),
   503

    try:
        top_movies = get_recommendations(user_id)
        return jsonify(top_movies)
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

- Antrenarea modelului se face într-un **thread separat**, la pornirea aplicației Flask:

```
def start_recommender():
    run_recommender() #   ncarc   datele  i  antreneaz
   modelul SVD
    model_ready = True
```

- Codul din model.py include:

- extragerea ratingurilor din tabela ratings cu aproximativ 6 milioane de rating-uri salvate dintr-un dataset public.
- construirea unui DataFrame
- antrenarea unui model **SVD (Singular Value Decomposition)** folosind SURPRISE

3.4.2 Explicație matematică a modelului SVD

Modelul SVD folosește o aproximare latentă a matricei de ratinguri R, astfel încât:

$$R \approx U \cdot \Sigma \cdot V^T \tag{1}$$

În contextul sistemului de recomandare, fiecare utilizator u și fiecare film i sunt reprezentați în spațiul latent de dimensiune k (număr de factori):

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T \cdot q_i \tag{2}$$

- μ este media globală a ratingurilor
- b_u, b_i sunt deviațiile pentru utilizator și film
- p_u este vectorul latent al utilizatorului
- q_i este vectorul latent al filmului

Toți parametrii (p_u, q_i, b_u, b_i) sunt antrenați astfel încât să **minimizeze eroarea pătratică medie** (RMSE). Aceasta se face prin algoritmi de **gradient descent** deja implementați.

3.4.3 Ce învață de fapt modelul

- Preferințele latente ale utilizatorilor (de ex. preferă SF, drame, filme vechi etc.)
- Atribute latente ale filmelor (ex. este popular printre fanii SF, are ton dramatic etc.)
- Relația între aceste preferințe → similaritatea este dată de produsul scalar.

4 Mod de utilizare, interacțiune cu utilizatorul și configurare

Această secțiune descrie modul în care utilizatorul interacționează cu platforma de streaming și recomandare de filme, precum și pașii necesari pentru configurarea aplicației.

4.1 Navigare și interacțiune în interfața web

Platforma oferă o interfață prietenoasă, dezvoltată cu React.js și stilizată cu Material UI (MUI). Utilizatorul poate naviga ușor între pagini precum:

- **Homepage-ul**, unde sunt afișate toate filmele disponibile, cu opțiuni de sortare în funcție de genul filmului, filtrare dacă filmul e disponibil pe platformă și căutare după numele filmului sau al regizorului;
- **Pagina detaliată a unui film**, cu player video, subtitrări, comentarii și ratinguri;
- **Pagina de autentificare/înregistrare**;
- **Pagina Utilizatorului**
  - Istoricul său de comentarii, de recenzii
  - **Secțiunea de recomandări**, unde sunt afișate sugestii personalizate pentru fiecare utilizator înregistrat;
  - **Formularul de încărcare CSV** cu ratinguri proprii, pentru recalibrarea recomandărilor.

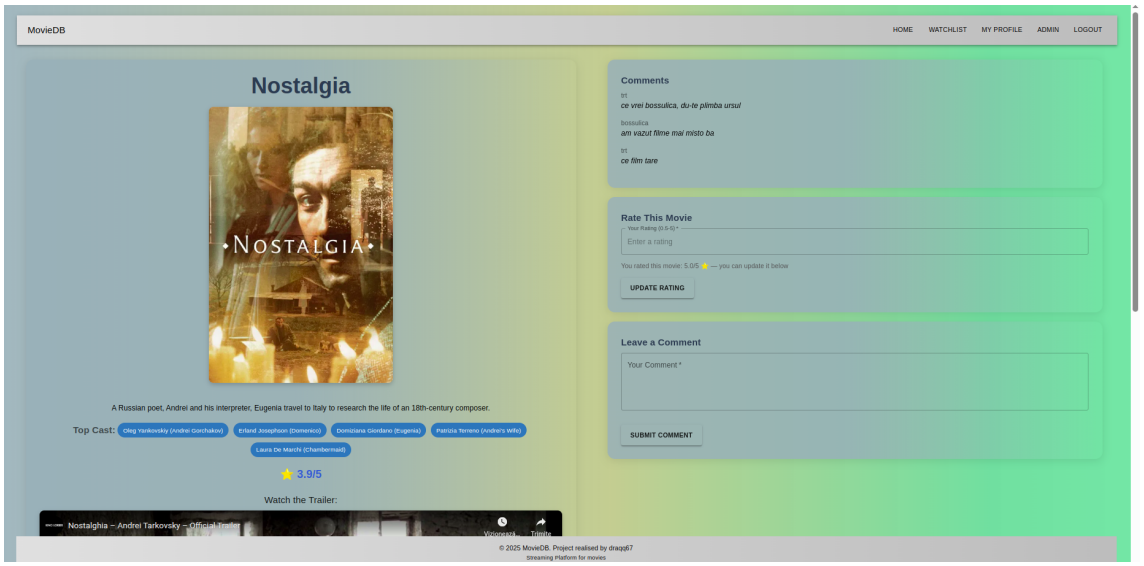

Figura 1: Pagina Filmului

4.2 Funcționalități interactive

După autentificare, utilizatorul beneficiază de:

- **Player video personalizat**, cu suport pentru:
  - Subtitrări .vtt încărcate manual;

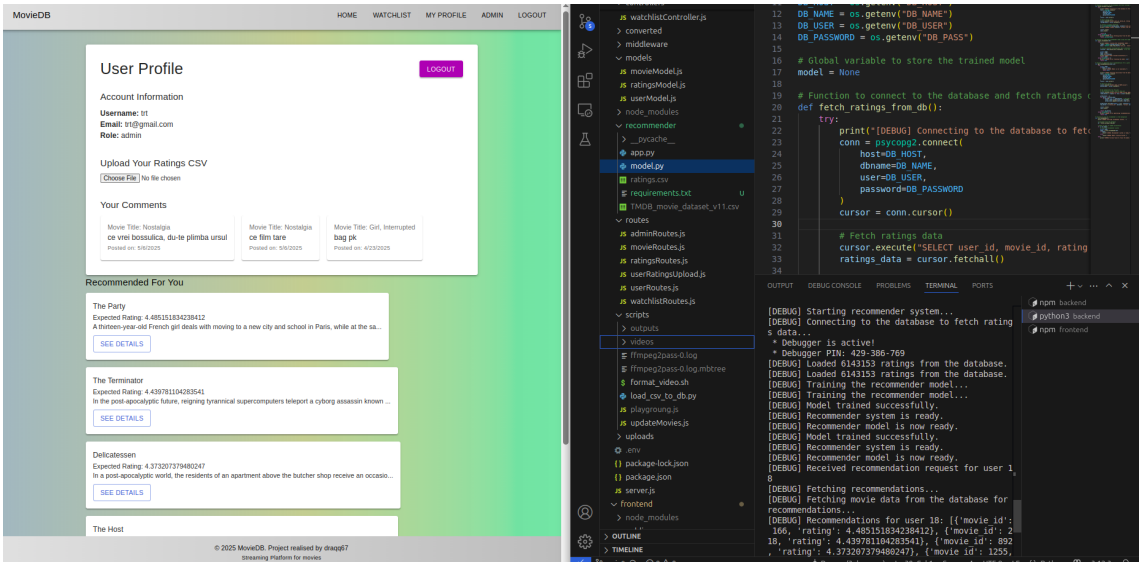

Figura 2: Recomandarile oferire pe pagina utilizatorului alaturi de log-urile procesului de recomandare

  - Control al vitezei de redare (0.5x – 2x);
  - Picture-in-Picture, fullscreen, control volum, mute;
  - Seeking și derulare rapidă.

- **Adăugare comentarii și ratinguri** vizibile instant în pagina filmului.
- **Creare wishlist**, pentru salvarea filmelor preferate.
- **Vizualizare recomandări** personalizate bazate pe preferințele utilizatorului.

4.3 Configurare aplicație și rulare locală

Aplicația conține 3 componente:

- **Frontend - React** pentru a porni aplicația pe portul 3000:

```
cd frontend
npm install
npm start

```

- **Backend Node.js (Express)** pe portul 5000:

```
cd backend
npm install
npm start

```

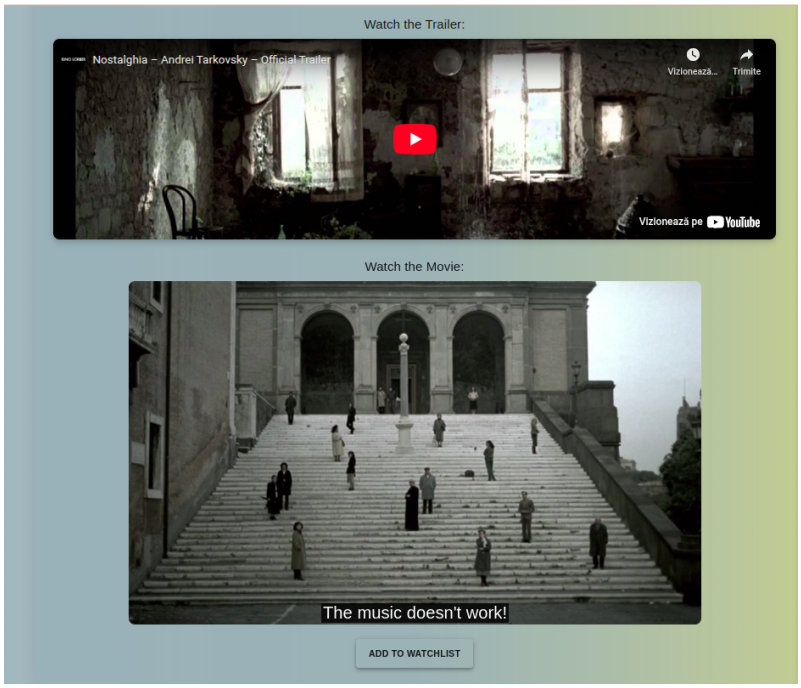

Figura 3: Player video alaturi de subtitrari

- **Server Flask**, responsabil de rularea modelului SVD, pe portul 5001. La pornire, antrenează modelul pe toate ratingurile din baza de date. Comunicarea între componente se face prin HTTP folosind fetch:

```
cd backend/recommender
pip install -r requirements.txt
python3 app.py

```

Datele sensibile au fost adăugate într-un fișier .env ce ulterior au fost transmise acolo unde trebuia să se facă conexiunile cu baza de date, cât și cu token-ul de autentificare, cât și cheia API folosită pentru a popula baza de date cu filmele de pe TMDB.

Aplicația lucrează momentan doar pe serverul local, în cadrul acestui proiect nu am realizat procesul de deploy, dar în viitor este de luat în considerare.

5 Concluzii

Proiectul prezentat și-a atins în mod complet obiectivele stabilite inițial, vizând dezvoltarea unei aplicații web moderne de tip platformă de streaming video, cu funcționalități avansate de redare multimedia și un sistem inteligent de recomandare personalizată. Componentele au fost dezvoltate modular, utilizând un stack tehnologic actual și scalabil, iar integrarea acestora a fost realizată într-un mod coerent și funcțional.

5.1 Obiective atinse

- Crearea unei aplicații de tip platformă video cu funcționalități moderne: player HTML5 cu suport pentru subtitrări, *Picture-in-Picture*, control al vitezei și streaming progresiv optimizat.
- Implementarea unui sistem de recomandare bazat pe un algoritm SVD, cu model antrenat pe date reale dintr-o bază de date PostgreSQL și expus printr-un API dedicat scris în Flask.
- Posibilitatea ca utilizatorii să interacționeze cu platforma: adăugare în wish-list, rating, comentarii și primirea de recomandări personalizate.
- Optimizarea video pentru web prin transcodare `FFmpeg` și livrare din Google Cloud.

5.2 Utilitate și relevanță

Această aplicație se înscrie perfect în tendințele actuale de personalizare a experienței utilizatorului. Utilitatea practică este evidentă: o astfel de platformă oferă atât conținut de calitate, cât și o experiență interactivă și personalizată.

5.3 Performanță și actualitate

- Video-urile sunt optimizate pentru încărcare rapidă și compatibilitate cu toate browserele moderne.
- SVD este un algoritm consacrat și eficient, utilizat în sisteme de recomandare comerciale (Netflix, Amazon).
- Separarea arhitecturii în microservicii (Flask pentru recomandări, Express pentru backend general) permite scalare ușoară și întreținere facilă.

5.4 Impact și perspective

Pe termen lung, această aplicație poate fi extinsă cu ușurință pentru a susține:

- mai multe formate video,
- modele de recomandare avansate (ex: deep learning, content-based),
- funcționalități sociale (ex: grupuri, recenzii).

Proiectul demonstrează aplicarea inteligenței artificiale în contexte reale, managementul eficient al fișierelor multimedia și dezvoltarea de aplicații web moderne, scalabile și modulare.

6 Bibliografie

1. TMDB Developers, "The Movie Database (TMDB) API," [Online]. Available: `https://developers.themoviedb.org/`.

2. FFmpeg Developers, "FFmpeg Documentation," [Online]. Available: `https://ffmpeg.org/documentation.html`.

3. ReactJS, "React – A JavaScript library for building user interfaces," [Online]. Available: `https://reactjs.org/`.

4. ExpressJS, "Express - Node.js web application framework," [Online]. Available: `https://expressjs.com/`.

5. PostgreSQL Global Development Group, "PostgreSQL: The World's Most Advanced Open Source Relational Database," [Online]. Available: `https://www.postgresql.org/`.

6. Google Cloud, "Cloud Storage Documentation," [Online]. Available: `https://cloud.google.com/storage/docs`.

7. S. Funk, "Netflix Update: Try This at Home," [Online]. Available: `https://sifter.org/~simon/journal/20061211.html`.

8. C. A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, Dec. 2015.

9. Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.