Phase 3

By Alvin Chen, Albert Ma, Chris Espinoza

Project Overview

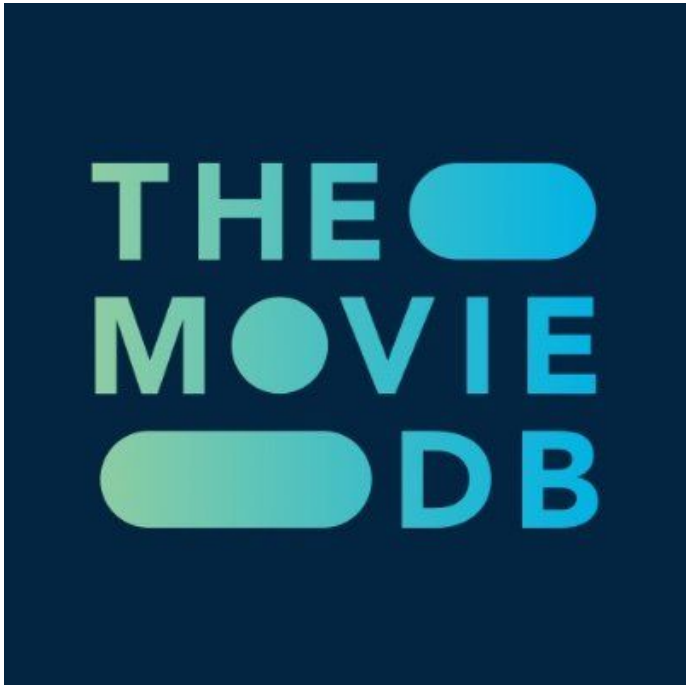
What it is:
A web platform that allows users to browse, search, and view information about movies and TV shows.

It supports user accounts, reviews, discussions, watchlists, favorites, and reactions.

What the system does:

- Displays movies and shows by genre, rating, popularity, or search
- Lets users write reviews, join discussions, and interact with content
- Provides cast, seasons, and genre information
- Stores all user interactions in our relational database

Data source:
Uses the TMDb API to populate the catalog with up-to-date movie and TV show data.



Implementation Details

Frontend (React + TypeScript)

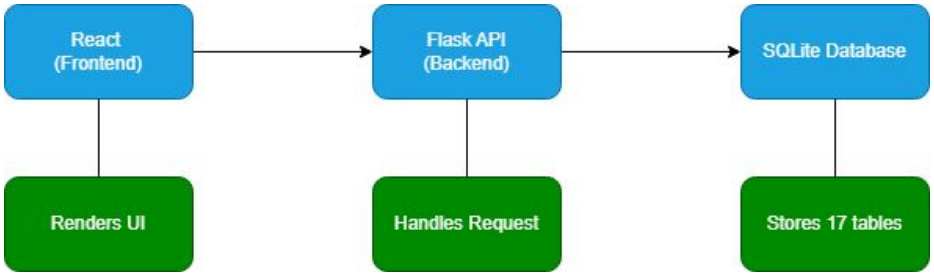
- React **+ TypeScript**
- Built with **Vite**
- Uses **React Router** for client-side routing
- Uses **Axios** for API requests
- Consumes JSON responses from Flask backend

Automated ETL Pipeline (APScheduler)

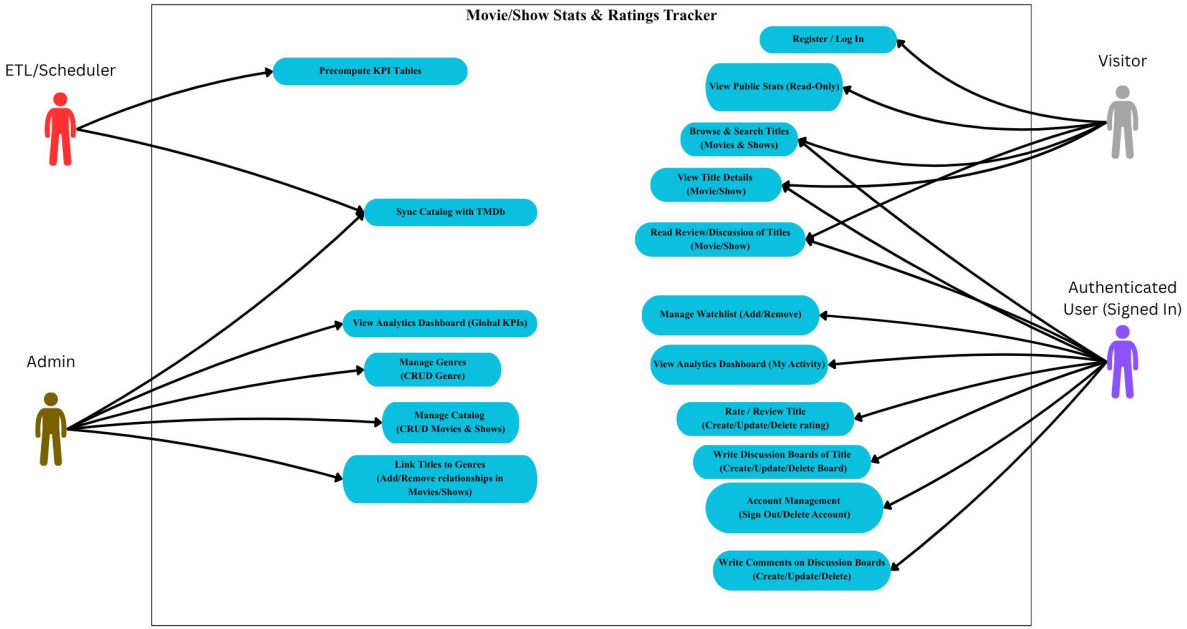
- **APScheduler** triggers scheduled ETL jobs
- TMDb ingestion + media updates
- KPI computation & logging
- Fully automated data refresh (not manual)

Backend API (Flask) and SQLite3 (Database Engine)

- REST endpoints for:
  - Movies, shows, seasons, episodes
  - Watchlists & favorites (M:N junction tables)
  - Reviews, discussions, comments
- Parameterized SQLite3 queries
- JSON output (dict-based)
- Error handling + input validation



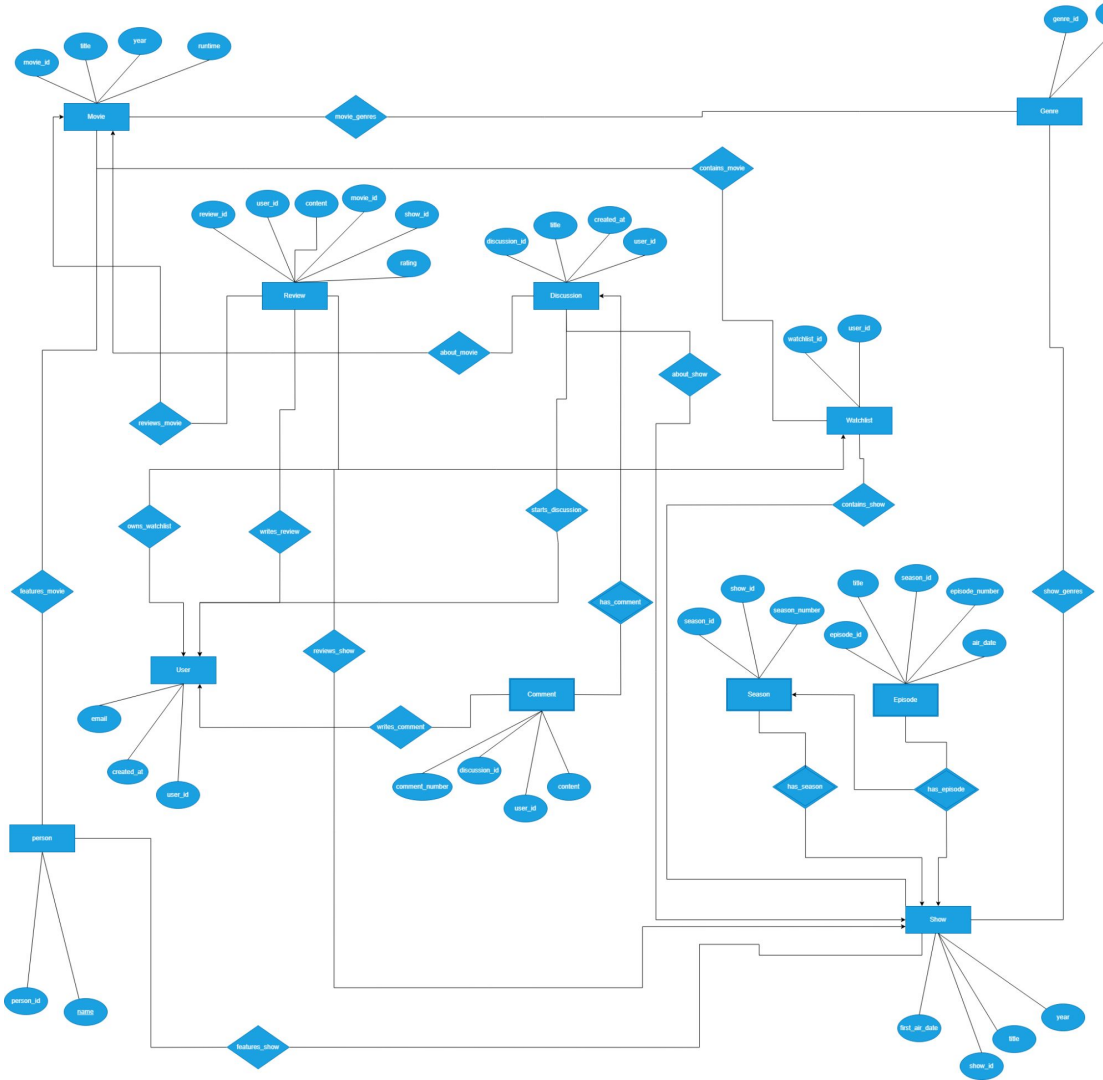
UML Use Case Diagram



Key Use Cases

- Visitor browses titles, filters by genre/rating
- User signs in, adds to watchlist/favorites, writes reviews
- User make discussions & comments
- Admin managing catalog and viewing analytics

ER Diagram


movie_id
title
year
runtime
Movie
movie_genres
genre_id
name
Genre
contains_movie
review_id
user_id
content
movie_id
show_id
rating
Review
discussion_id
title
created_at
user_id
Discussion
about_movie
about_show
watchlist_id
user_id
Watchlist
reviews_movie
owns_watchlist
writes_review
starts_discussion
contains_show
features_movie
has_comment
reviews_show
show_genres
User
email
created_at
user_id
writes_comment
Comment
comment_number
discussion_id
content
user_id
show_id
season_id
season_number
Season
title
season_id
episode_number
episode_id
air_date
Episode
has_season
has_episode
person
person_id
name
features_show
Show
first_air_date
title
year
show_id

Relational Schema - Table Overview

Entities - Total 16

- User (**user_id**, email, …)
- Movie (**movie_id**, title, year, runtime, …)
- Show (**show_id**, tmdb_id, title, year, …)
- Season (**season_id**, **show_id**, season_number, …)
- Episode (**episode_id**, **season_id**, episode_number, …)
- Genre (**genre_id**, name)
- Person (**person_id**, name)
- Review (**review_id**, **user_id**, **movie_id**, **show_id**, content, rating, created_at)
- Discussion (**discussion_id**, **user_id**, **movie_id**, **show_id**, title, created_at)
- Comment (**comment_id**, **discussion_id**, **user_id**, content, created_at)
- Watchlist(**(user_id**, **movie_id**, **show_id)**, added_at)
- Favorite (**(user_id**, **movie_id**, **show_id)**, added_at)
- MovieGenres (**(movie_id, genre_id)**)
- ShowGenres (**(show_id, genre_id)**)
- MovieCast (**(movie_id, person_id)**, character, cast_order)
- ShowCast (**(show_id, person_id)**, character, cast_order)

- Core media tables (movies, shows, seasons, episodes)
- Relationship tables (MovieCast, ShowCast, MovieGenres, ShowGenres, ..)
- User content tables (users, reviews, watchlists, favorites)
- Social feature tables (discussions, comments)

Many-to-Many (M:N) Relationships

Movies ↔ Genres

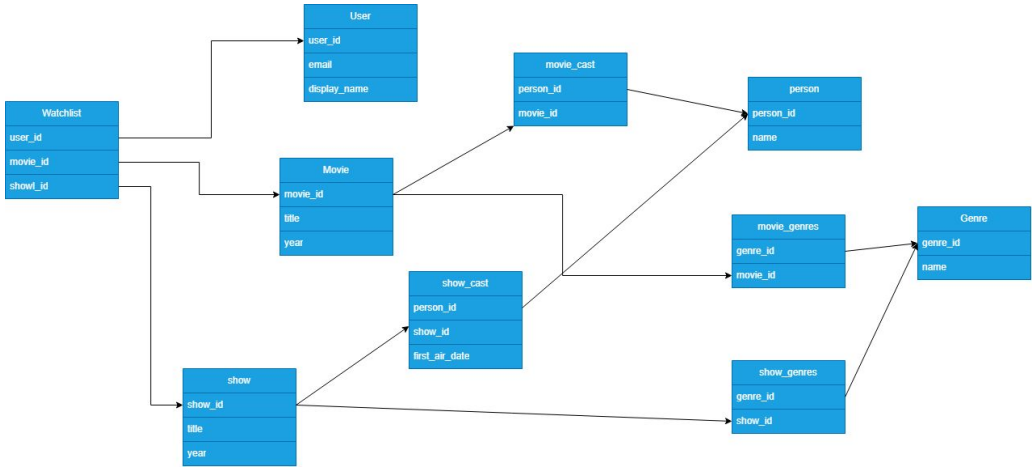
- Junction: **movie_genres(movie_id, genre_id)**
- Composite PK
- (i.e. One movie = many genres; one genre = many movies)

Shows ↔ Genres

- Junction: **show_genres(show_id, genre_id)**
- Composite PK

Movies ↔ People (Cast)

- Junction: **movie_cast(movie_id, person_id)**
- Composite PK



Total 6 M:N relationships

1:M Relationships with Cascading Constraints

Shows → Seasons

- seasons has an independent primary key (season_id)
- show_id is a foreign key (NOT NULL)
- **ON DELETE CASCADE** enforces cleanup to create dependencies

Seasons → Episodes

- episodes has an independent primary key (episode_id)
- season_id is a foreign key (NOT NULL)
- **ON DELETE CASCADE**

Discussions → Comments

- Comments has an independent primary key (comment_id)
- discussion_id is a foreign key (NOT NULL)
- **ON DELETE CASCADE**

Total 5 1:M Relationships

DEMO

https://github.com/Mr-StudyHard/CSE-111-Class-Project/tree/main