MangoDB: Advanced IMDB Clone

Database, Backend, and Frontend Architecture

Atharva Bendale
22B0901
atharvaab@cse.iitb.ac.in

Nivesh Aggarwal
22B0912
22B0912@iitb.ac.in

Dhvanil Gheewala
22B0923
dhvanilg@cse.iitb.ac.in

Vishal Bysani
22B1061
vishalbysani@cse.iitb.ac.in

May 2, 2025

Goals

MangoDB is a one-stop platform for discovering, tracking, and reviewing movies, shows, and books. Our goal is to create a comprehensive and user-friendly database system where users can share ratings and reviews and follow their cinephile/readaholic friends or their favourite movie/book reviewers. Users can browse their collections to discover entertainment that matches their interests. We also recommend users movies corresponding to the movie page they are looking. The git repository for the project :

```
https://github.com/AtharvaBendale/IMDB
```

Note: The AI model used for review summary generation, the PostgreSQL schema (`DDL`) and the relations can be found in the following google drive link : CS349 Project

User's Perspective

MangoDB's homepage features a curated selection of popular titles across movies, TV shows, and books. Users can search for specific titles or enter partial keywords to discover relevant results. Advanced filtering options make it easy to refine searches by genre, media category, cast, release year, minimum rating, and sort results by popularity or ratings. Both critic and audience ratings are displayed separately to help users choose.

Each movie, show, or book detail page includes an AI-generated review summary that saves time. MangoDB also provides personalized recommendations based on factors such as genre, decade, director or author, cast, rating, and popularity—intelligently weighted to deliver relevant suggestions. For TV shows, dedicated seasons pages are available, each featuring its own trailer

to help users explore episodes in depth. Trailers are also provided for all movies and TV shows (at the series level), while book pages include a preview link that redirects to the corresponding Google Books page. Visual elements like posters for books, movies, shows, and seasons, along with descriptive summaries for each title, further enrich the user experience.

By creating an account (with an optional profile picture), users can easily track their media engagement. They can add movies and shows to their watchlist and books to their to-read list. When users rate or review a title, it is automatically added to their watched or read list for easy tracking.

MangoDB also fosters community interaction by allowing users to follow others and browse their collections—even without logging in.

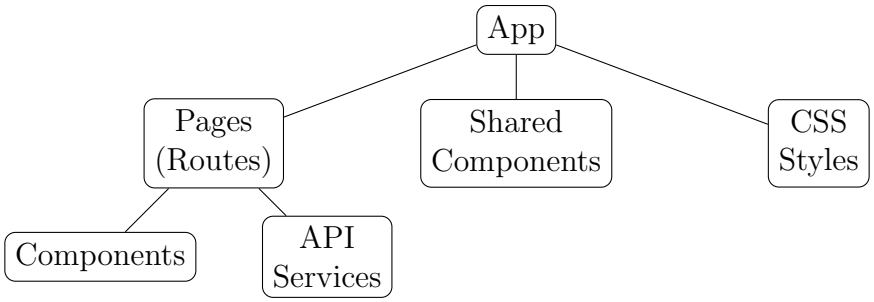
With all these features we can claim:

MangoDB — Your ultimate hub for movies, shows, and books.

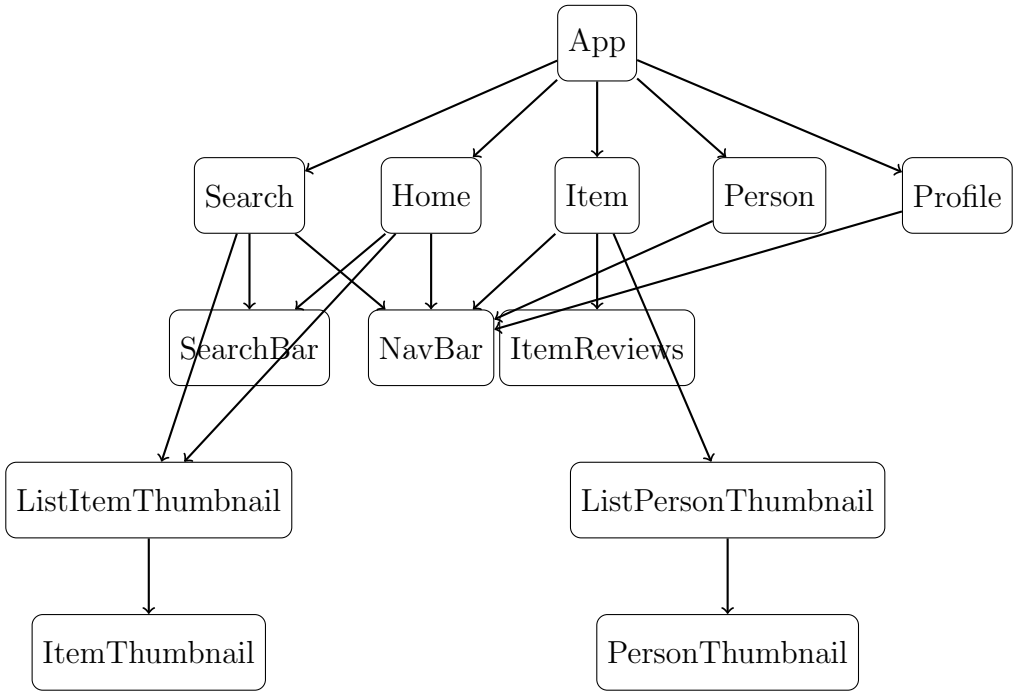
Project Architecture

Frontend Architecture

The frontend project follows a hierarchical structure:



The following diagram illustrates the component hierarchy and data flow in the application:



The application uses several reusable components to maintain consistency and reduce code duplication:

Component Name	Description
NavBar	Provides navigation across the application with logo, search functionality with dropdown suggestions, authentication-based rendering, and responsive design. Essential for usability and appears on all pages.
ListItemThumbnail	Displays horizontal scrollable list of movie or TV show thumbnails with pagination controls, dynamic row limit calculation, and consistent item display. Efficiently presents collections of content in a visually appealing manner.
ItemThumbnail	Shows a single movie or TV show card with image (with fallback handling), title, rating visualization, year information, and cast preview. Standardizes content item presentation throughout the application.
ListPersonThumbnail	Presents horizontal scrollable list of people with images, fallbacks, name and role display, and responsive design. Specialized for displaying people rather than content items.
PersonThumbnail	Displays a single person card with image (with fallback), name, character name when applicable, and navigation to person detail page. Standardizes person presentation throughout the application.
ListItemOverview	Shows vertical list of movie or TV show items with more detailed information than thumbnails, used primarily for search results where users need more information to make decisions.
ItemOverview	Presents detailed overview of a single movie or TV show in larger format including image, title, rating, year, cast information, and description preview. Middle ground between compact thumbnail and full item page.
SearchBar	Provides consistent search input interface with text input, placeholder, search button, enter key support, and optional parent state synchronization. Standardizes search experience across the application.
SearchDropdown	Offers search input with dropdown selection functionality including filtered options, selection handling, value display, and support for selection or direct entry. Used for filters and advanced search options.
Popup Component	Provides modal dialog functionality with overlay, close button, and background interaction prevention. Enhances user experience for focused tasks like rating or reviewing.

Table 1: Components Overview

Backend Architecture

Overview

MangoDB's backend is built on a relational database system that stores comprehensive information about movies, TV shows, books, cast, crew, authors and user interactions. The schema for all the relations are available in `DDL.sql` file

1.2.1 Entity-Relationship Diagrams for our Database

The diagrams below uses the following conventions:

- `Primary Keys` are indicated using a key symbol (🔑).
- `Foreign Keys` are indicated using a connector symbol, referencing the primary key of the related entity.
- `Not Null` constraints are shown using the label `NN` next to the corresponding attribute.
- `One-to-One, One-to-Many`, and `Many-to-One Relationships` are represented using a line with a cut on the "one" side.

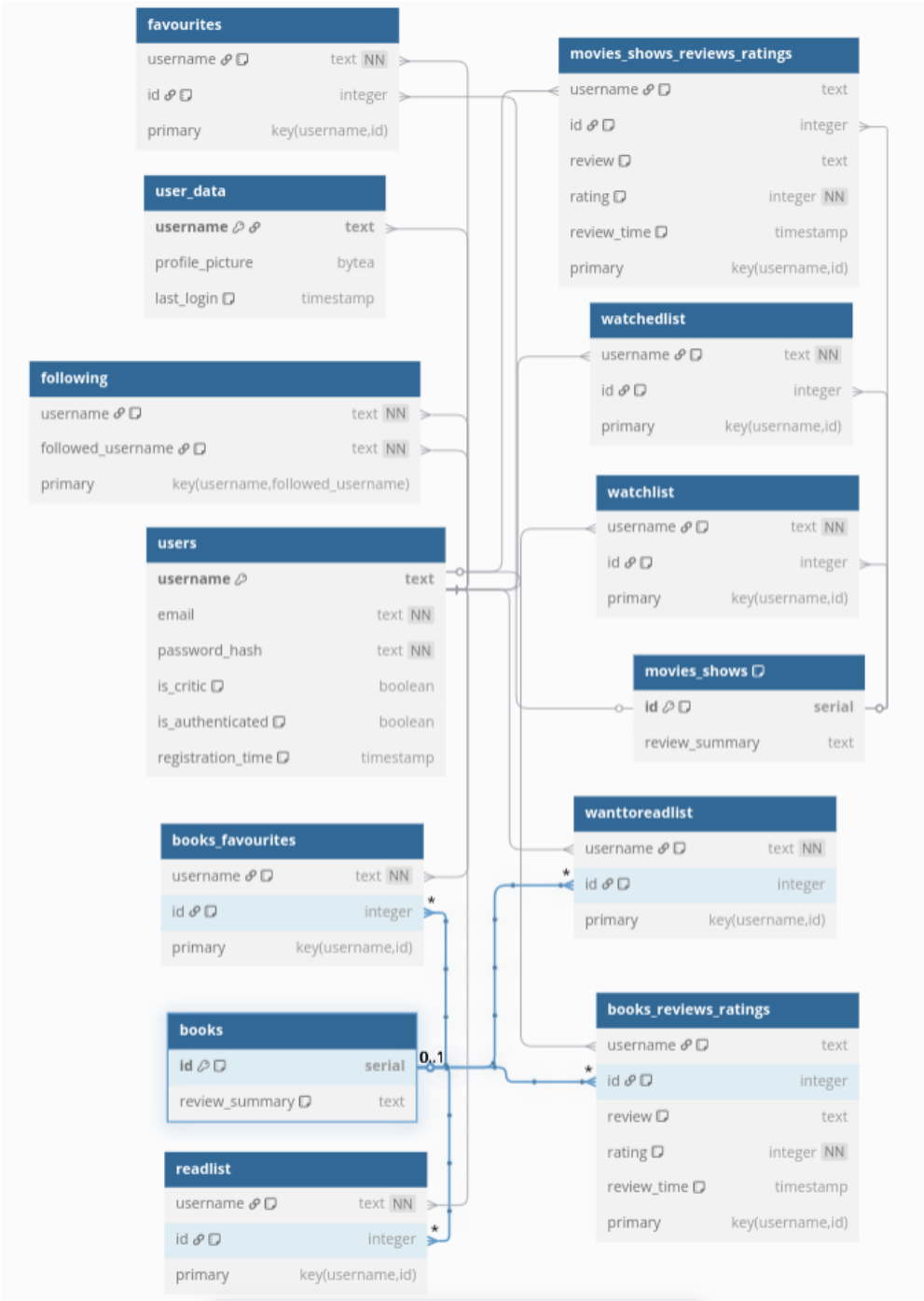


Figure 1: User Relations

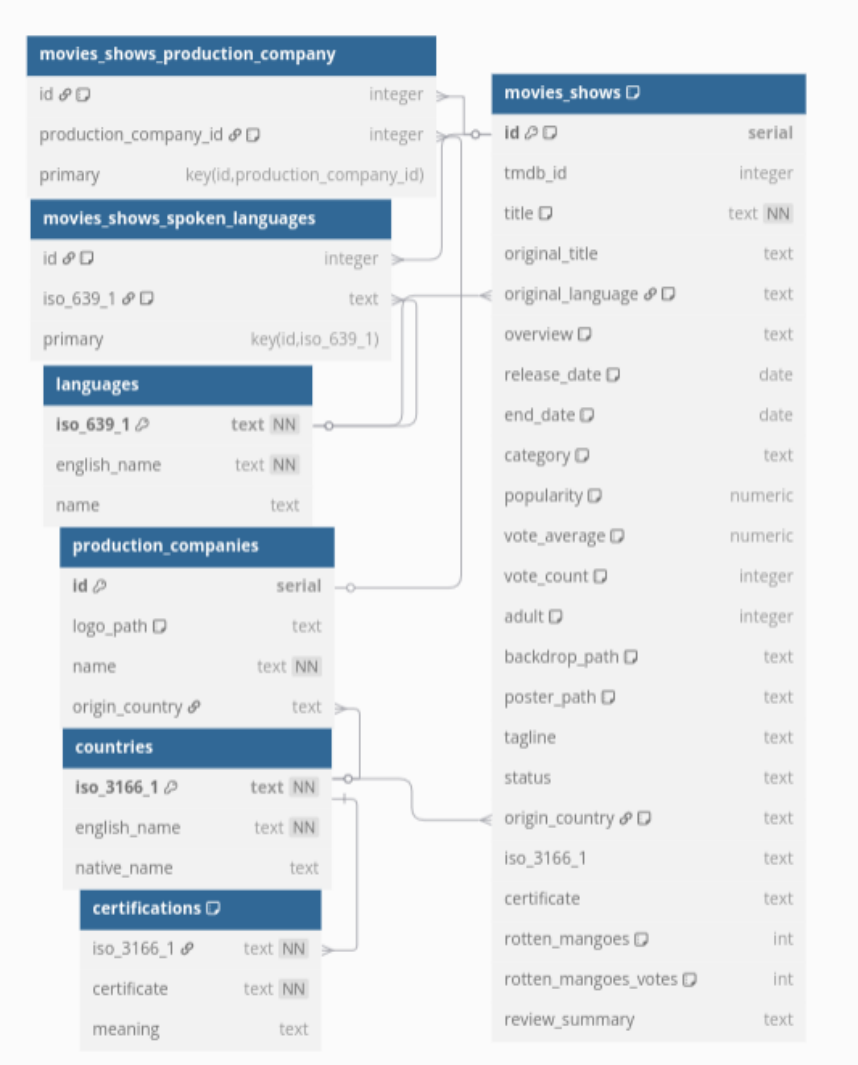


Figure 2: Movies and Shows Details

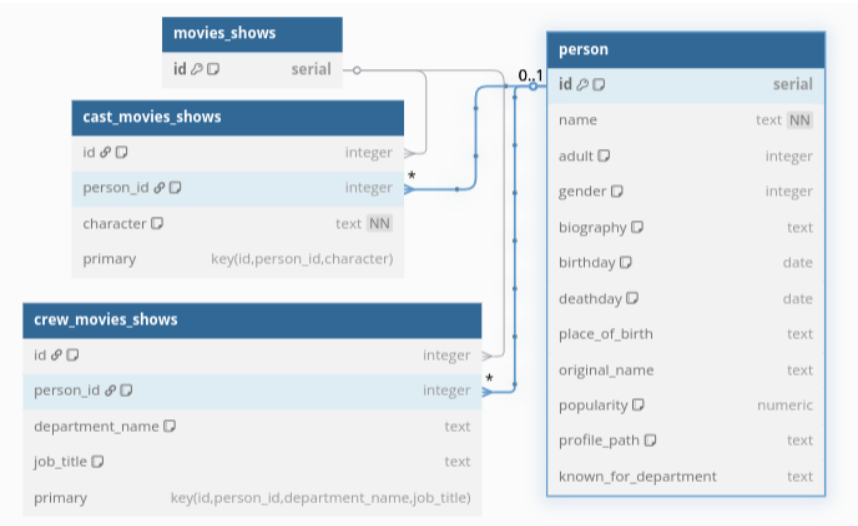


Figure 3: Crew and Cast details for the movies and shows (Note: Some attributes of `movies_shows` from the previous diagrams have not been included for clarity)

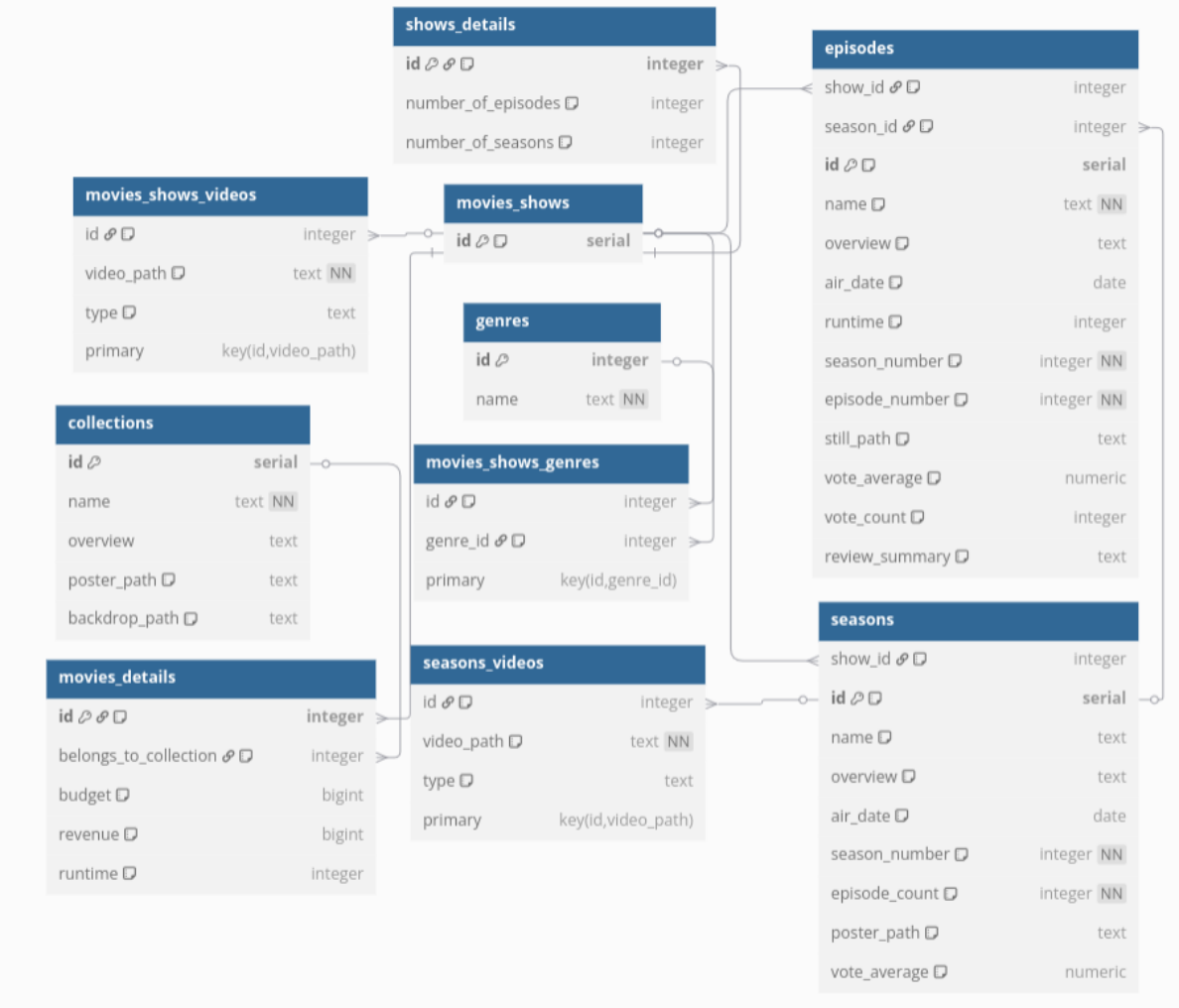


Figure 4: Collection details for movies and shows

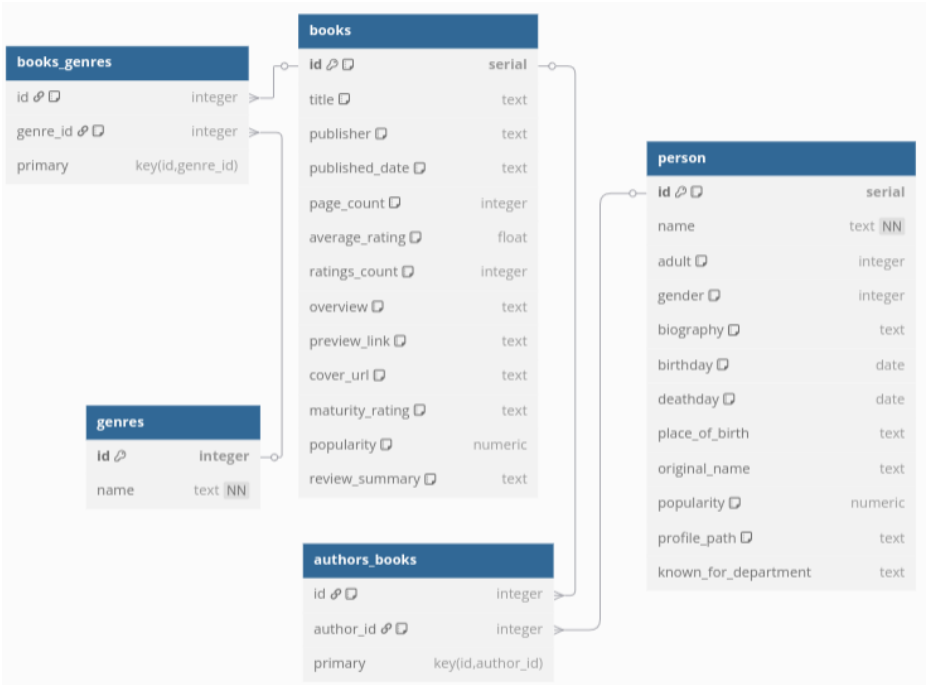


Figure 5: Book details

Project Functionalities

Frontend Functionalities

The application consists of the following main pages, each serving a specific purpose in the user journey:

Page Name	Description
Home Page	Landing page displaying trending movies and TV shows with a hero section containing welcoming message and search functionality. Detects authentication state to personalize the experience.
Item Page	Displays comprehensive details about a specific movie or TV show including title, year, rating, duration, cast and crew information, user rating functionality, review submission, video trailer integration, favorite toggling, and user reviews section with pagination.
Season Page	Shows information about TV show seasons including season name, poster, all episodes of the season, and a season navigator to browse various seasons.
Collections Page	Displays information about collections (e.g., Marvel Collection) including collection name, description, and all items (movies/shows) belonging to the collection.
Person Page	Shows biographical information about actors, directors, or other film industry professionals including name, birth/death dates, professional roles, popular works, awards, and popularity metrics.
Search Page	Allows users to search for movies, TV shows, and people with advanced filtering options including genre, person, year, minimum rating, content type selection, sorting options, and paginated results.
Genre Page	Displays movies and TV shows belonging to a specific genre, showing genre title, information, and popular movies and TV shows within that genre.
Authentication Pages	Handles user login with username/email and password, new user registration with validation, authentication state management, and error handling with user feedback.
Profile Page	Shows user information, activity statistics, favorite movies and shows, and watchlist items.
List Persons Page	Displays comprehensive lists of people associated with a movie or show in specific roles (actors, directors, writers) with navigation to individual person pages.
Not Found Page	Displays a Page Not Found message when website path does not map to any page or requested item is not available.
Network Error Page	Shows Network Error message with option to retry.

Table 2: Pages Overview

Backend Functionalities

On the backend, we implemented a set of API endpoints and server-side functions to handle key operations like handling user management, media retrieval, search and filtering, user interactions, and AI-powered enhancements such as review summarization and recommendations. A summary of the key backend calls and their respective roles is provided below.

Backend Call / Function	**Explanation**
`isAuthenticated()`	Checks if the user is authenticated
`rateLimit()`	middleware limits each IP to 10,000 requests every 15 minutes
`createTransport()`	sets up a Nodemailer transporter to send emails using Gmail
`lastSeen()`	returns how long ago a given user was active
`heartBeats()`	Updates the last_login timestamp
`/signUp`	Handles user registration and sends a verification link
`cron job`	Deletes unverified users from the database
`/verify-email`	Handles email verification by verifying the token
`/login`	Authenticates users and creates session tokens
`/isLoggedIn`	Checks if the user is logged in by verifying the session
`/logout`	Logs out the user by destroying the session
`/getMatchingItem`	Handles a search request to find media matching the query
`/getMatchingItemPages`	Handles a paginated search request to find media matching the query
`/getMovieShowDetails`	Handles a request to fetch detailed information of movies/shows
`/getSeasonDetails`	Handles a request to fetch detailed information of seasons of shows
`/getPersonDetails`	Handles a request to fetch detailed information about a celebrity
`/getMovieShowByCollectionId`	fetch movies or shows of a collection
`/getMoviesByPopularity`	retrieves movies sorted by popularity with pagination and watchlist status
`/getShowsByPopularity`	retrieves shows sorted by popularity with pagination and watchlist status
`/filterItems`	Enables filtering using genres,year, minimum rating, cast, category and order by popularity or rating
`/listGenres`	fetches and returns a list of all genres, sorted by name.
`/matchingPersons`	Fetch people whose names match the given search text, returning a list sorted by popularity
`/submitEpisodeRatingReview`	handle the submission of ratings and reviews for books, movies/shows, and episodes, updating relevant average ratings, review counts, and generating a review summary using an AI model.
`/getFavourites`	retrieves a user's favorite movies, shows, and books
`/addToFavourites`	adds a movie/show or book to the user's favorites

/removeFromFavourites	removes a movie/show or book from the user's favorites
/getWatchList	retrieves the user's watchlist
/getWantToReadList	retrieves the user's To-Readlist
/addToWatchList	adds a movie/show to the user's watchlist
/addWantToReadList	adds a book to the user's To-Readlist
/removeFromWatchList	removes a movie/show from the user's watchlist
/getWatchedList	fetches the user's watchedlist
/removeFromWantToReadList	removes a book from the user's To-Readlist
/getReadList	fetches the user's Readlist
/addToWatchedList	adds a movie/show to user's watchedlist
/addToReadList	adds a book to user's Readlist
/removeFromWatchedList	removes movies/shows from Watched List.
/removeFromReadList	removes books from Read List.
/getFollowers	fetches the list of users who follow the current user, along with their profile pictures
/getFollowing	fetches the list of users who the current user follows, along with their profile pictures
/followUser	allows the current user to follow another user
/getUserDetails	fetches a user's profile info, favourites, lists, and social connections
/getBooksDetails	fetches detailed information, authors, genres, reviews, similar books, and recommendations
/getBooksByAuthorId	fetches a paginated list of books written by a given author
/uploadProfilePicture	uploads and updates the authenticated user's profile picture and its MIME type

Table 3: Backend functions and their roles in MangoDB

Implementation Approach and Tooling

For the backend, PostgreSQL was used as the relational database management system to design a normalized and scalable schema capable of handling complex relationships between media, users, reviews, and interactions. The server-side logic and RESTful APIs were developed using Node.js. These APIs handled data retrieval, filtering, user authentication, and interaction management. AI-based review summarization , were integrated using pre-trained language models and python script. The recommendation system was carefully tuned by assigning weights to factors like genre, decade, cast, director/author, popularity, and ratings to ensure relevant suggestions. The majority of the backend was developed from scratch. However, for certain advanced functionalities—such as sending verification emails, automatically deleting unverified users after a specified period using cron jobs, storing profile pictures in blob format and implementing rate limiting to mitigate potential DoS attacks—we used external resources like GeeksforGeeks ([1]), PostgreSQL documentation ([3]) and AI tools such as ChatGPT to guide our implementation. These tools also assisted in identifying suitable models for generating review summaries and in performing data scraping tasks using Selenium. Additionally, GitHub Copilot was utilized to expedite the development of routine or repetitive code segments through intelligent autocompletion.

Data Sources and Collection

Our data scraping module utilises the TMDb (The Movie Database) API[5], along with selected Rotten Tomatoes[4] web pages accessed via Selenium and BeautifulSoup and Google Books API([2]), as primary sources for collecting detailed information on movies ,TV shows and books. We follow a structured methodology to extract metadata related to media content, including cast, crew, authors and other relevant attributes, which are then used to populate our relational database.

The project was version-controlled using Git and hosted on GitHub to facilitate organized and collaborative development.

Future Work

With additional time, we intended to extend MangoDB to incorporate games as a new category within the media database. Several advanced features were also envisioned to further enrich the platform's capabilities. These included functionalities such as allowing users to delete their accounts, configure account visibility as public or private, participate in genre-based community forums, and engage in threaded discussions through replies on reviews. We also planned to implement review moderation and censorship tools, introduce age-based content access controls, and host a dedicated server to support an Android application built with React Native. Additionally, we aimed to offer users detailed media consumption statistics and develop an enhanced recommendation system that uses both a user's own collections (favorites, watchlists, highest rated movies, etc.) and those of the users they follow to generate personalized suggestions. We also planned to integrate a form for users to apply as critics, with fields to submit information such as previous works, a resume, and other relevant details and sent to the admins for personal verification.

References

[1] GeeksforGeeks. *GeeksforGeeks: A Computer Science Portal for Geeks*. 2025. URL: https://www.geeksforgeeks.org/.

[2] Google. *Google Books APIs*. Web API. Google LLC, 2025. URL: https://developers.google.com/books.

[3] *PostgreSQL: The World's Most Advanced Open Source Relational Database*. URL: https://www.postgresql.org.

[4] Rotten Tomatoes. *Rotten Tomatoes*. 2025. URL: https://www.rottentomatoes.com/.

[5] TMDb. *The Movie Database (TMDb) API*. 2024. URL: https://developer.themoviedb.org/.