

CSE3086– NoSQL Databases

J Component - Project Report

Review III

Movie Reviews

By

21MIA1067

Yazhini R

21MIA1153

Saahith M S

M.Tech CSE with Specialization

Submitted to

Dr.A.Bhuvaneswari,

Assistant Professor Senior,
SCOPE, VIT, Chennai

School of Computer Science and Engineering



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

November 2024



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

WINTER SEM 22-23

Worklet details

Programme	M.Tech with Specialization	
Course Name / Code	NoSQL Databases (CSE3086)	
Slot	D1	
Faculty Name	Dr.A.Bhuvaneshwari	
Digital Assignment	Review III	
Team Members Name Reg. No	Yazhini R	21MIA1067
	Saahith MS	21MIA1153

Team Members(s) Contributions – Tentatively planned for implementation:

<i>Worklet Tasks</i>	<i>Contributor's Names</i>
Dataset Collection	Yazhini R
Preprocessing	Saahith MS
Architecture/ Model/ Flow diagram	Yazhini R
Model building (suitable algorithm)	Saahith MS
Results – Tables, Graphs	Saahith MS and Yazhini R
Technical Report writing	Yazhini R and Saahith M S
Presentation preparation	Saahith MS

ABSTRACT

The Movie Review Web Application is an interactive dynamic framework through which the navigation, assessment, and review of movies occur in a most efficient manner. Built with the solid MERN stack comprising MongoDB, Express.js, React.js, and Node.js, the application delivers a scalable, efficient, and user-centric interface.

The core of the site is user registration and personalization. Users can create secure accounts, log in, and then enjoy features adapted to them - like their own collections of movies, personalized recommendations, and their history of reviews. From that point, all logged-in users are given an exhaustive list of films with each having an average rating, a brief summary of the movie, and available reviews that enable them to make informed choices as to which will be their next viewing.

The app will have an extended search and filter system that is meant to make it more usable. Movies can be found based on title, genre, actors, or the language involved and further filtered by rating, release year, and popularity. This ensures that users can locate movies that exactly match their preference. The impressive feature of the app is its real-time review and rating system. Users can rate and comment on movies, and the app updates it in real time to ensure that the data is current and relevant. Moreover, they may make a watchlist to easily track their preferred films. The watchlist enables them to quickly add or remove movies and hence provides an easy means for the user to maintain his entertainment preferences in order.

The social features make an app possible in engaging the community. A user will be able to view and write reviews and rate others post know prevailing trends. This interaction element adds to the user experience together with fostering a sense of community among film enthusiasts.

Technically, this application is built with React.js for the frontend part; in this way, it offers an extremely responsive and fluid navigation and dynamic content rendering user interface. The system has been constructed with very efficient APIs and database triggers to update the system in real time throughout the platform.

The application allows coherent and engaging user experiences by offering smooth navigation and updated information. Additionally, the application has provisions for real-time updates to user reviews and ratings for providing users with the most accurate information possible. Also, the scalability of the system ensures it to accommodate an increasingly growing user base as well as an expanding film database without degrading performance.

Introduction

The movie review web application is to meet the need of users which include giving them a single location where they can find information on movies, read reviews and recommend which movies to watch. Movies fans generally seek information about them with the help of several resources, which results in a dispersed experience in the present digital era. Currently, all these functionalities are lacking or scattered in various forms in the different sites and our project aims at developing a single web application that is easy to use and is developed using the MERN stack technology which includes MongoDB, Express. js, React. js and Node. js.

Problem Statement: The problem exists in the form of the lack of easy and feasible ways to find accurate recommendations of movies and genuine reviews from different sources based on the users' preferences. Current networks do not contain specified filters which makes it difficult for the users to search movies according to their preferences for instance by genre, actors or language. Also, handling and streaming large data of movie details and user generated content in real-time may be challenging, thus need for a good backend solution.

Objectives: The primary objective of this project is to create a scalable, efficient, and interactive web application that allows users to:

- Sign up and sign in safely.
- Watch a list of movies consolidated with their respective ratings.
- Movies can be searched using different parameters such as the title, the genre, the actors and the language.
- Rate movies and post your review.
- To implement the filtering and getting the recommendations according to the users own parameters.

Challenges: Major issues involve handling the big and unformatted data that is in most times involved with movie information and user generated information. Other essential issues are to guarantee the application's scalability when it comes to many users simultaneously and the real-time delivery of data. Furthermore, the creation of the user interface with regard to being adaptive and easy to navigate also involves UX design.

Dataset and Database Tool used

Dataset link: [Dataset Link](#)

TMDb has an extensive API that gives full access to one of the largest movie databases containing movie titles, genres, actors, languages, release dates, ratings, and reviews.

- Advantages: Currently, it is up-to-date, has a wide range of topics and offers high-quality information.
- Disadvantages: API access to data may be provided with rate limits which control how often data can be requested.

IMDb offers data sets of basic information about the films, their ratings and other related information. These are normally available for download in the IMDb's official website.

- Advantages: Extensive historical data and summary of the complete information of the movies.
- Disadvantages: Few drawbacks in terms of depth of data if you do not pay for the service.

MongoDB is a scalable NoSQL database for handling large quantities of various datasets. The information about the users is stored in it, containing account information, watchlists, and user preferences, besides all movie information such as title, genre, ratings, and descriptions. MongoDB also handles the structure of the nested user reviews and ratings and will update and retrieve much more efficiently.

This enables adding movies in real-time to the watched queues or upload reviews without anxiety about duplication using operators like \$addToSet. The database enables rapid searches and filters using indexes on title, genre, actors, or language. The very schema-less architecture makes it scalable with regards to managing growing data and changing attributes. In a nutshell, MongoDB is the base framework for a responsive and user-centric application.

FireBase Database is quite flexible and efficient at managing user and movie information using the backend. The application creates very efficient real-time synchronization that eliminates any interruption to ensure every update-like user reviews, ratings, or changes in the watchlists-appear simultaneously on various devices. Data storage is hierarchical and based on JSON structure that's really essential with regards to the effective categorization of movie information, user details, and their activities. This app also supports offline capabilities wherein one can retrieve information even if the internet connection is unavailable, and these automatic synchronizations are done once a connection comes back. With the features of real-time from Firebase, this application enables a dynamic interactive experience, making it suitable for modern applications on mobile devices.

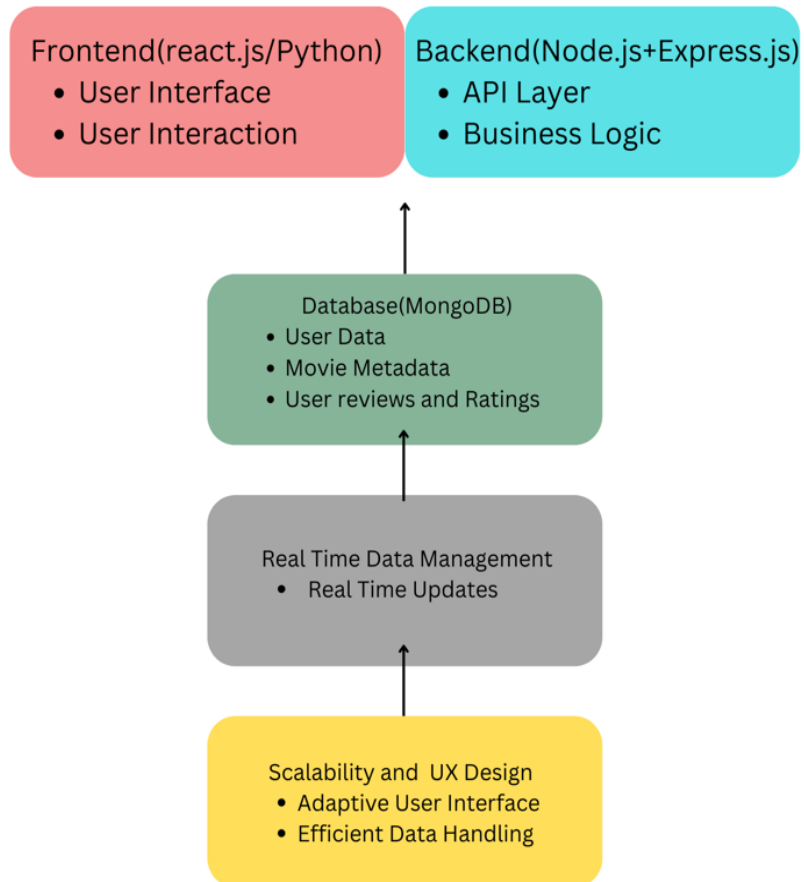
Algorithms / Techniques description

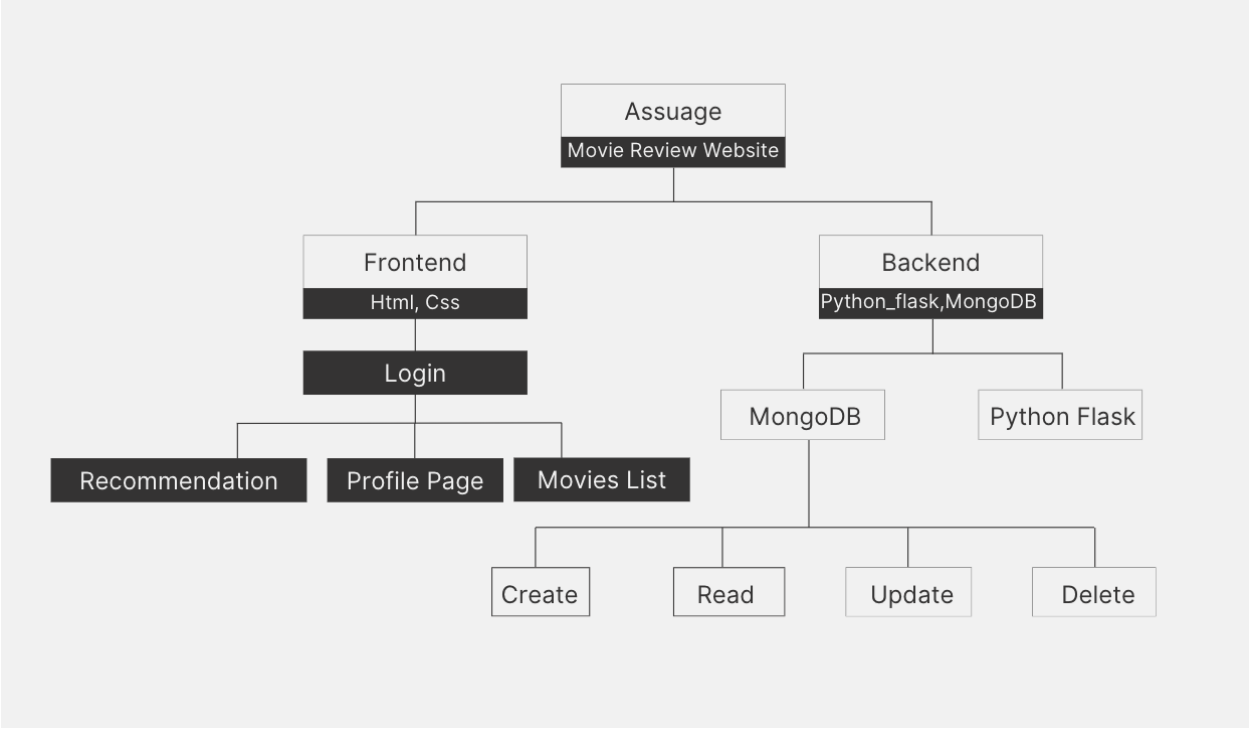
Recommendation algorithm based on **collaborative filtering** is used to personalize listings of films using user-specific data. The application captures information from user-engagements, including ratings, reviews, watchlists, and search histories. It checks the similarity between users or movies using methodologies such as cosine similarity or Pearson correlation. User-based filtering works in that it suggests movies those users who have liked similar preferences, and item-based filtering suggests movies which are similarities to the ones the user has rated favorably previously. The suggested movies are ranked based on the expected relevance, using attributes like genre or year, etc. Also, the recommendations of the system change in real time as users continue submitting more ratings and reviews.

Pseudocode:

1. Input Data:
 - Get user preferences (ratings) from the database.
 - Get the movie database with metadata.
2. Identify Movies Rated by User:
 - Retrieve movies rated by the target user.
3. Calculate Similarity Between Movies:
 - For each rated movie, compare it with other movies in the database.
 - Calculate similarity score
4. Sort Movies by Similarity Score:
 - Rank movies based on their similarity scores to the user's rated movies.
5. Recommend Top Movies:
 - Select top 5 movies based on similarity score.
6. Display Recommendations:
 - Display the recommended movies on the website.

Block Diagram of the proposed work / system design





Implementation Details

1. User Authentication:

User authentication serves as the foundation of our platform, ensuring secure and personalized interactions for every user.

- **User Registration:** During registration, users provide essential details such as:
 - Email: A unique identifier for each user.
 - Name: Used for personalized greetings and reviews.
 - Password: Stored securely in an encrypted format using hashing techniques like bcrypt.
 - Profile Picture: An optional feature allowing users to upload images, enhancing their profile's appearance.

MongoDB's users collection stores this information, ensuring efficient retrieval and scalability. The inclusion of a profile picture field provides a personal touch and allows users to express their identity visually.

- **Login and Session Management:** Upon successful login, a Flask session is created, storing the user's session ID. This session ensures that users remain authenticated as they navigate the platform. The session-based approach improves security by avoiding persistent client-side credentials.
- **Data Security:**
 - Passwords are never stored in plain text. Instead, a hashed version is saved, making it virtually impossible for attackers to retrieve the original password.
 - MongoDB's unique indexing ensures that duplicate registrations with the same email address are prevented.

These authentication features provide the foundation for personalized user experiences, including watchlists, recommendations, and reviews.

2. Movie Management:

The platform's core functionality revolves around managing a rich catalog of movies, each equipped with detailed information to enhance user engagement.

- **Movie Catalog:** The movies collection in MongoDB stores information such as:

- Movie name
- Genre(s)
- Director and cast
- Rating
- Poster image (URL or file path)

This data is fetched dynamically to populate the front-end pages, ensuring scalability and easy updates.

- **Dynamic Search:** Users can search for movies based on titles, genres, or ratings. MongoDB's text indexes enable efficient querying, allowing users to quickly find relevant content.
- **Movie Recommendations:**
 - The system employs content-based filtering to recommend movies. When a user selects a genre and a minimum rating threshold, the platform queries MongoDB to fetch matching movies.
 - For example, if a user selects "Drama" with a rating of 8.0 or higher, MongoDB's query filters movies in this genre and sorts them by rating.

This approach ensures users receive tailored recommendations, enhancing their experience and encouraging continued engagement.

3. Watchlist Management:

The watchlist feature empowers users to curate a personalized list of movies they wish to watch.

- **Adding Movies:**
 - Users can add movies to their watchlist with a single click. MongoDB's \$addToSet operator prevents duplicates by ensuring each movie appears only once.
 - A real-time confirmation message is displayed using AJAX, ensuring seamless user interaction without page reloads.
- **Removing Movies:**
 - The profile page allows users to manage their watchlist. A simple button click removes a movie from the watchlist, executed using MongoDB's \$pull operator.
- **Database Structure:**

- The users collection contains a watchlist array, which stores the names of movies added by the user. This structure allows efficient updates and retrievals.

By integrating these functionalities, the watchlist becomes a powerful tool for users to organize their movie-watching plans.

4. Movie Reviews:

Reviews are a vital feature, fostering community engagement and user interaction.

- **Adding Reviews:**

- Users can post reviews for movies. Each review includes:
 - Reviewer name (fetched automatically from the user's session)
 - Review text
 - Date of submission
 - Count of likes and dislikes (defaulted to 0)
- MongoDB stores reviews as an array of objects within each movie document. This nested structure simplifies data management and retrieval.

- **Editing and Deleting Reviews:**

Users can update or delete their reviews. The system ensures that only the reviewer's entry is modified, leveraging MongoDB's \$elemMatch operator for precise targeting.

- **Displaying Reviews:**

Reviews are displayed in chronological order on each movie's page. MongoDB's aggregation framework ensures efficient sorting and retrieval.

These features create a vibrant review ecosystem, encouraging users to share their opinions and engage with the community.

5. Profile Management:

The profile page acts as a centralized hub for user-specific data, including activity and preferences.

- **Liked/Disliked Movies:**

- MongoDB maintains separate arrays for movies a user has liked or disliked. This data is dynamically fetched and displayed on the profile page.
- Users can toggle their preferences, with MongoDB updating the corresponding arrays in real-time.

- **Watchlist:**

The watchlist is prominently displayed on the profile page, offering users an overview of their saved movies.

- **Profile Updates:**

Users can edit their name or upload a new profile picture. MongoDB efficiently updates these fields while retaining other user data.

This feature personalizes the platform, making users feel connected to their accounts and activities.

6. Content-Based Recommendations:

The recommendation engine is a standout feature, enhancing user engagement through personalized suggestions.

- **Recommendation Logic:**

- Based on user input (e.g., genre and rating), MongoDB queries the movies collection to fetch relevant results.
- The use of indexes ensures rapid execution, even with large datasets.

- **User Experience:**

Recommendations are displayed on a dedicated page with detailed movie cards, including posters, genres, and ratings.

By tailoring suggestions to user preferences, this feature transforms the platform into a discovery tool for movie enthusiasts.

Integration with MongoDB: A Detailed Overview

MongoDB serves as the backbone of our Movie Review and Recommendation System, enabling efficient data management, scalability, and seamless integration for all features. Each feature leverages MongoDB's NoSQL architecture to ensure robust functionality and performance. Below is a detailed explanation of how MongoDB supports and integrates with each feature of the platform.

1. User Authentication and Profile Management:

MongoDB is instrumental in managing user data, ensuring security and scalability.

Database Design for User Authentication: The users collection is used to store user information, with each document representing a user. A sample schema:

```
{
  "_id": "unique_user_id",
  "email": "user@example.com",
  "name": "John Doe",
  "password": "hashed_password",
  "profile_picture": "path_or_url_to_image",
  "watchlist": ["Movie1", "Movie2"],
  "liked_movies": ["Movie3"],
  "disliked_movies": ["Movie4"]
}
```

- **Key Features:**

- Unique Index: An index is created on the email field to ensure each user registers with a unique email address.
- Password Hashing: Only encrypted passwords are stored, enhancing security.

CRUD Operations in User Authentication:

- **Create:** During registration, a new document is inserted into the users collection.
- **Read:** On login, the user's email and hashed password are validated against the database.
- **Update:** Profile updates (e.g., name, profile picture) are handled using MongoDB's \$set operator.
- **Delete:** Although not implemented directly, a user account deletion feature could remove the corresponding document from the users collection.

2. Movie Management:

The movies collection acts as a central repository for all movie-related data, providing detailed information about each movie.

Database Design for Movies:

```
{
  "_id": "unique_movie_id",
  "movie_name": "Inception",
  "genre": ["Action", "Sci-Fi"],
  "director": "Christopher Nolan",
  "cast": ["Leonardo DiCaprio", "Joseph Gordon-Levitt"],
  "rating": 8.8,
  "poster": "path_or_url_to_poster",
  "reviews": []
}
```

- **Key Features:**

- **Nested Data:** The reviews field is an array of objects, enabling efficient storage and retrieval of all reviews for a movie.
- **Indexes:** Indexes are created on fields like genre and rating to optimize recommendation queries.

CRUD Operations in Movie Management:

- **Create:** New movies can be added to the collection as part of system updates or database initialization.
- **Read:** Users can browse all movies or search for specific ones using filters. MongoDB's \$regex operator supports flexible text searches.
- **Update:** Movie details (e.g., ratings) can be updated, such as when new ratings are aggregated.
- **Delete:** Movies can be removed from the collection if required, though this feature is typically restricted to administrators.

3. Watchlist Management:

The watchlist feature relies on MongoDB's array operations for efficient storage and manipulation of user-specific movie lists.

Database Design for Watchlists: Watchlists are stored as arrays within the user's document in the users collection:

```
{
  "watchlist": ["Inception", "The Dark Knight"]
}
```

CRUD Operations in Watchlists:

- **Create:** When a user adds a movie to their watchlist, MongoDB's \$addToSet operator ensures no duplicates are added:

```
{ $addToSet: { "watchlist": "Inception" } }
```

- **Read:** The watchlist field is retrieved during profile loading to display the user's saved movies.
- **Update:** Movies can be reordered or modified in the watchlist (if implemented in the future).
- **Delete:** MongoDB's \$pull operator removes a movie from the watchlist:

```
{ $pull: { "watchlist": "Inception" } }
```

4. Movie Reviews:

Reviews are a core feature that facilitates user engagement. MongoDB's flexible schema makes it easy to store and manage reviews within the movies collection.

Database Design for Reviews: Reviews are stored as an array of objects within each movie document:

```
{
  "reviews": [
    {
      "reviewer": "John Doe",
      "review_text": "Amazing movie!",
      "date": "2024-11-19",
      "likes": 10,
    }
  ]
}
```

```

    "dislikes": 2
  }
]
}

```

CRUD Operations in Reviews:

- **Create:** A new review is added to the reviews array using the \$push operator:

```
{ $push: { "reviews": { "reviewer": "John Doe", "review_text": "Great!", ... } } }
```

- **Read:** All reviews for a movie are retrieved and displayed on the movie's page.
- **Update:** MongoDB's \$set operator updates specific fields (e.g., review text or likes) within a review object:

```
{ $set: { "reviews.$[elem].review_text": "Updated review" } }
```

- **Delete:** MongoDB's \$pull operator removes a specific review:
- ```
{ $pull: { "reviews": { "reviewer": "John Doe" } } }
```

## 5. Profile Management:

The profile section consolidates all user-specific data, leveraging MongoDB's ability to store related information in a single document.

### Database Design for Profiles:

The users collection stores:

- Liked\_movies: Array of liked movie names.
- Disliked\_movies: Array of disliked movie names.
- Watchlist: Array of saved movie names.

### CRUD Operations in Profiles:

- **Create:** Movies are added to Liked\_movies or Disliked\_movies using the \$addToSet operator.
- **Read:** The profile page queries the user's document to fetch and display liked/disliked movies and the watchlist.
- **Update:** Profile picture or name updates are handled using \$set.
- **Delete:** Removing a movie from Liked\_movies or Disliked\_movies is achieved using \$pull.



## 6. Content-Based Recommendations:

The recommendation engine utilizes MongoDB's querying and aggregation capabilities to fetch personalized results.

**Database Design for Recommendations:** Movies are stored with attributes like genre, rating, and cast, enabling flexible filtering.

### Recommendation Logic:

Query Example: To fetch action movies with a rating above 8:

```
{ "genre": "Action", "rating": { $gte: 8.0 } }
```

**Sorting:** Recommendations are sorted by rating using MongoDB's sort method:

```
{ $sort: { "rating": -1 } }
```

MongoDB's aggregation framework allows complex queries, combining filters, sorting, and projection for efficient results.

## **Mobile App Implementation**

The project presents the implementation of a movie review and recommendation system as a mobile app using Android Studio, integrated with Firebase as the backend database. The app provides users with an interactive platform to explore movies, manage their watchlists, and post reviews. The app's key features include user authentication, movie catalogue management, personalized movie recommendations, and CRUD operations on reviews and watchlists. Firebase is used to handle real-time data storage and synchronization.

### **Features and Implementation:**

1. **User Authentication:** User authentication is central to ensuring that users can have personalized experiences in the app.
  - **User Registration:** Users can register by providing an email address, name, password, and an optional profile picture. Firebase Authentication is used to securely handle user sign-ups and login requests. The profile picture is uploaded to Firebase Storage and linked to the user's profile.
  - **Login and Session Management:** Firebase Authentication is used for login and session management, ensuring that users can securely access the app's features like movie reviews and watchlists. User sessions are maintained using Firebase Authentication tokens, ensuring a seamless experience.
  - **Data Security:** Passwords are securely managed by Firebase Authentication, which uses encrypted tokens for login sessions, ensuring that sensitive information is never exposed.
2. **Movie Management:** The movie catalogue is an essential part of the app, providing users with a wide range of movies to explore, rate, and review.
  - **Movie catalogue:** Movie details (name, genre, director, cast, rating, and poster) are stored in Firebase Firestore. The app fetches this data in real-time, allowing users to browse movies or search for specific ones based on different criteria.
  - **Movie Recommendations:** The app offers personalized movie recommendations based on user preferences such as genre and minimum rating. Firebase Cloud Firestore queries are used to fetch movies that match the user's preferred genres and ratings.
3. **Watchlist Management :** The watchlist feature allows users to save movies they wish to watch in the future.

- **Adding Movies:** Users can add movies to their watchlist by clicking an "Add to Watchlist" button. Firebase Firestore stores these movies under the user's profile, ensuring that the movies are added only once.
  - **Removing Movies:** Users can remove movies from their watchlist by accessing the profile page and selecting the "Remove" button next to the respective movie.
  - **Real-Time Feedback:** Firebase Realtime Database is used to ensure that any action, like adding or removing a movie from the watchlist, is immediately reflected in the app without requiring a page reload.
4. **Movie Reviews:** The app allows users to write and manage reviews for movies they have watched.
- **Adding Reviews:** Logged-in users can post reviews for movies, including text and a rating. Reviews are stored in Firebase Firestore under the corresponding movie document as a sub-collection.
  - **Editing and Deleting Reviews:** Users can edit or delete their reviews at any time. Firebase's real-time data synchronization ensures that the latest version of a review is reflected immediately on the movie page.
  - **Displaying Reviews:** All reviews for a movie are displayed on the movie's page, encouraging community interaction and providing valuable insights to other users.
5. **Profile Management:** The profile section consolidates all user-specific information and personalized data.
- **Personalized Data:** The user profile displays information like the watchlist, liked/disliked movies, and review history. Data is dynamically fetched from Firebase Firestore, ensuring that the user's profile is always up-to-date.
  - **Profile Updates:** Users can update their profile information, such as their name and profile picture, from within the profile page. The app updates this data in Firebase in real-time, ensuring the profile remains current.
6. **Content-Based Recommendations:** This feature provides personalized movie recommendations based on user preferences.
- **Recommendation Logic:** Users can specify the genre and minimum rating they prefer. The app queries Firebase Firestore to retrieve movies that match these criteria. For example, if a user selects the "Action" genre and a minimum rating of 8.0, the app fetches top-rated action movies.

- **Query Optimization:** Firebase Firestore indexes and sorts the data to ensure that recommendation queries are efficient, even with large datasets.

## **Integration with Firebase**

Firebase serves as the backbone for the mobile app, handling user authentication, real-time data storage, and synchronization.

- **Firebase Authentication:** Manages user sign-ups, logins, and session management securely.
- **Firebase Firestore:** Stores and retrieves movie data, user profiles, reviews, and watchlist information in real-time. Firestore's flexible NoSQL structure allows for easy storage of nested data, such as reviews as sub-collections within movie documents.
- **Firebase Realtime Database:** Handles real-time updates for actions like adding/removing movies from the watchlist, ensuring that changes are reflected instantly across all users.
- **Firebase Storage:** Stores user profile pictures, movie posters, and other media, ensuring that files are uploaded and retrieved securely.

### **Workflow:**

#### **1. User Interaction:**

- Users register and log in using Firebase Authentication.
- Upon logging in, users are taken to the home page where they can browse movies, post reviews, and manage their watchlist.
- Personalized recommendations are displayed based on user preferences.
- Users can navigate to their profile to update personal details and view their activity, including liked movies and watchlists.

#### **2. Dynamic Content Updates:**

Firebase's real-time synchronization ensures that any changes made by the user, such as adding/removing movies from the watchlist or posting/editing a review, are immediately reflected across devices.

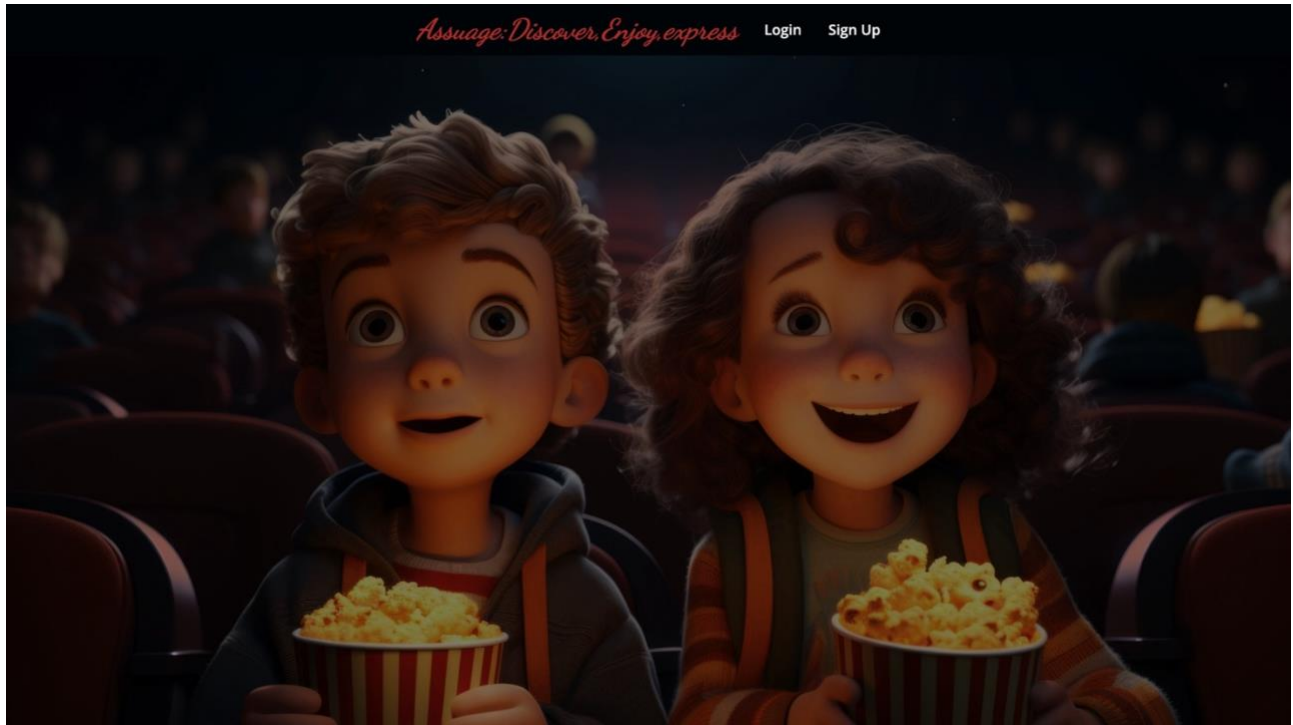
### **3. Database Operations:**

Every user interaction triggers an appropriate Firebase query, ensuring that movie data, reviews, and user-specific data are updated and fetched in real-time.

### **4. Recommendation Engine:**

The recommendation engine queries Firebase Firestore based on user preferences and generates a list of movies that match the criteria, ensuring users are always presented with relevant content.

## Screenshots



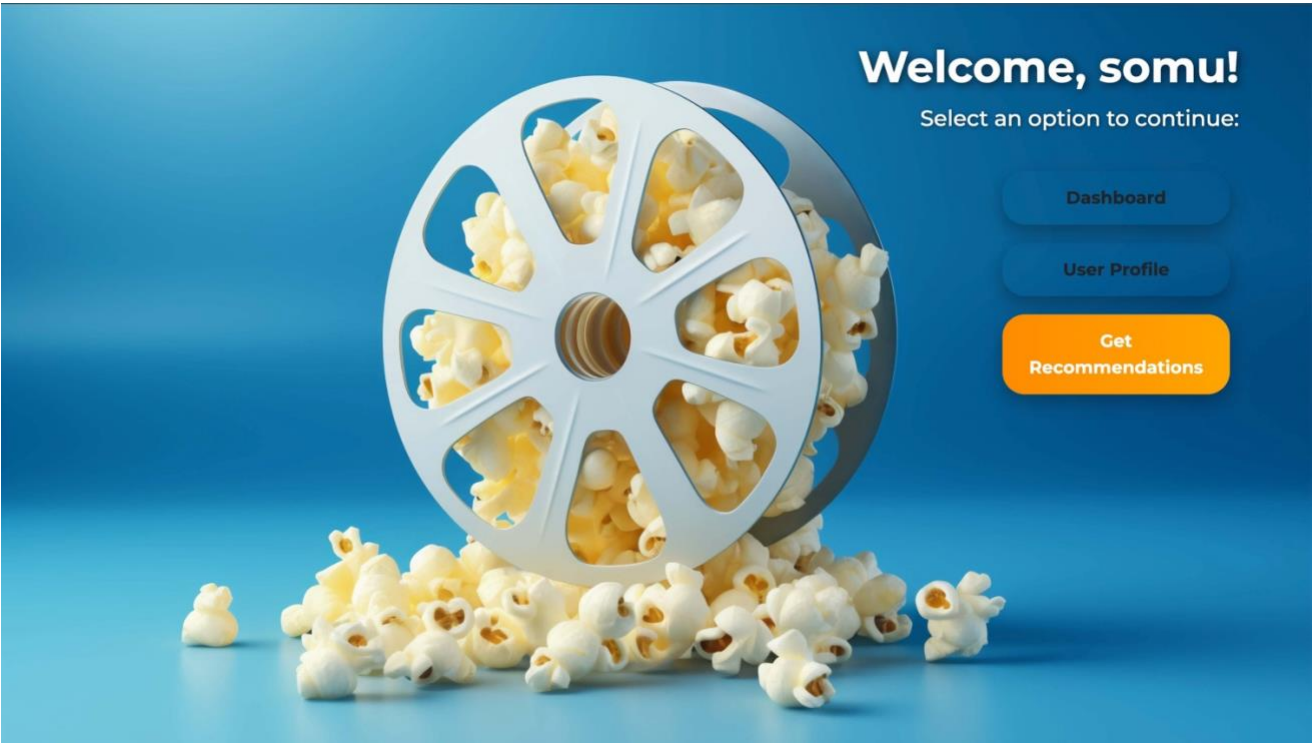
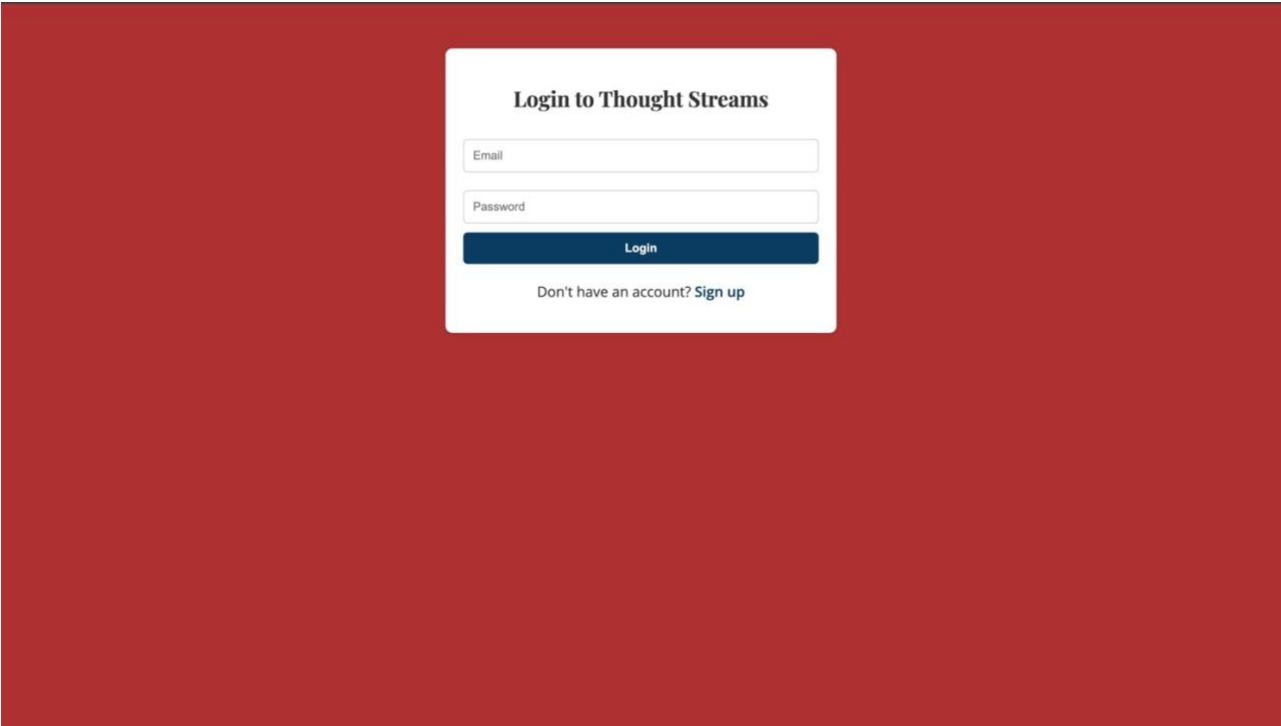
### Create Your Account

Choose file

No file chosen

Sign Up

Already have an account? [Login](#)



## My Profile



Welcome, blessy

Here is a summary of your movie preferences and watchlist.

Update Profile Picture

### My Watchlist 1

• The Shawshank Redemption

### Liked Movies 1

### Disliked Movies 1

© 2024 ASSUAGE Movie Review. All rights reserved.

## ASSUAGE Movie Review

Go to My Profile

Search for a Movie

Select a movie...

Search

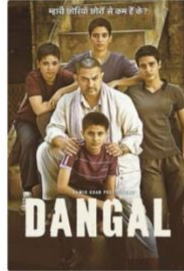


[Go to My Profile](#)

Q Search for a Movie

Select a movie...

[Q Search](#)



## Dangal

Genre: [Biography], [Drama], [Sport]

Release Date: 2016

Director: Nitesh Tiwari

Cast: Aamir Khan, Sakshi Tanwar, Fatima Sana Shaikh

Ratings: 8.4

[Add to Watchlist](#) [Like](#) [Dislike](#)

### User Reviews

Raj

A remarkable sports drama with Aamir Khan's stellar performance and inspiring story.

Sita

Emotionally powerful and motivational with exceptional performances and direction.

rahul

remarkable

### Add Your Review

Write your review here...

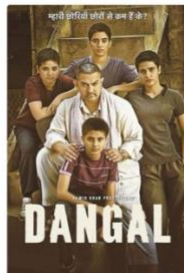
[Submit](#)

[Go to My Profile](#)

Q Search for a Movie

Select a movie...

[Q Search](#)



## Dangal

Genre: [Biography], [Drama], [Sport]

Release Date: 2016

Director: Nitesh Tiwari

Cast: Aamir Khan, Sakshi Tanwar, Fatima Sana Shaikh

Ratings: 8.4

[Add to Watchlist](#) [Like](#) [Dislike](#)

### User Reviews

Raj

A remarkable sports drama with Aamir Khan's stellar performance and inspiring story.

Sita

Emotionally powerful and motivational with exceptional performances and direction.

rohini

remarkable

#### Add Your Review

Write your review here...

Submit

### Find Recommended Movies

Select Genre:

Comedy

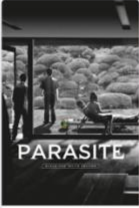


Minimum Rating:


7

Get Recommendations

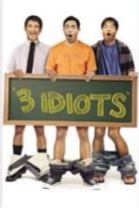
Top 5 Recommended Movies in Comedy with Rating Above 7.0




**Parasite**  
Rating: 8.5  
Director: Bong Joon Ho  
Cast: [Kang-ho Song, Sun-yeon Lee, Woo-sung Jo]  
[Add to Wishlist](#)




**Super Deluxe**  
Rating: 8.4  
Director: Thiagarajan Kumararaja  
Cast: [Vijay Sethupathi, Fathath Pasha, Samantha Ruth Prabhu]  
[Add to Wishlist](#)



**3 Idiots**  
Rating: 8.4  
Director: Rajkumar Hirani  
Cast: [Aamir Khan, Ir. Madhavan, Sharman Joshi]  
[Add to Wishlist](#)



**Jigarthanda**  
Rating: 8.3  
Director: Karthi Subasing  
Cast: [Dhanraj, Nishu Senthil, Lakshmi Menon]  
[Add to Wishlist](#)



**Amélie**  
Rating: 8.3  
Director: Jean-Pierre Jeunet  
Cast: [Audrey Tautou, Mathieu Kassovitz, Rufus]  
[Add to Wishlist](#)

[Go Back](#)

## **Results and Discussion**

### **Results:**

The Movie Review and Recommendation System provides the key functionality on both web and mobile interfaces, including secure authentication of a user, movie database maintenance, watchlists management, movie reviews, and content-based recommendations. On the web interface, Flask and MongoDB offer flexible and customizable backend solutions; however, on mobile, Firebase Authentication and Firestore are exploited to yield a richer, more real-time experience.

Both the platforms support secured methods for user authentication-the mobile app's Firebase Authentication simplifies session management, whereas for Flask, it is through the cookie-based approach. The movie management system is quite efficient on both platforms. MongoDB's aggregation framework allows MongoDB to perform the queries efficiently on web, and Firestore offers real-time synchronization that provides immediate updates on the mobile. The watchlist management works great on both, but the mobile application is superior as a change display immediately, while the web uses AJAX for partial page updates. Movie reviews function well on both with the use of full CRUD, where Firebase real time synchronization makes it an interactive experience on mobile.

In terms of content-based recommendations, both applications provide personalized suggestions to the user, dependent upon his preferences, such as genre or rating. MongoDB's strong querying ability makes it efficient in filtering on the web, whereas Firebase Firestore's real-time updates give a quicker response on a mobile device.

### **Conclusion:**

The Movie Review and Recommendation System is built using the web and mobile applications, safe as well as efficient to work on the pros and cons of each. For requisites, the web application being Flask with MongoDB is way more flexible with efficient querying; hence, it is pretty suitable for systems that may require operations or larger data storage. Using Firebase, however, ensures that this is a mobile application offering real-time synchronization and user interaction-a platform more dynamic and responsive.

Future enhancement can include OAuth so that logging in is much easier for the user, and using more complex algorithms for recommendation systems, possibly using machine learning. The data of the users can even be synced between both platforms for a much more seamless experience in case users log in and out between web and mobile.

## **Future Potential Works**

### **1. Collaborative Filtering Integration:**

- The current recommendation system is based on **content-based filtering** (genre and rating). Future versions could integrate **collaborative filtering** techniques to recommend movies based on **user behaviour** (e.g., movies liked by similar users).
- Combining **content-based** and **collaborative filtering** could improve the recommendation quality and offer more personalized suggestions.

### **2. Machine Learning Enhancements:**

- **User Reviews Analysis:** The current system doesn't analyse user reviews beyond text input. Future implementations could use **Natural Language Processing (NLP)** to analyse sentiment and keywords in reviews, which could further personalize movie recommendations based on user sentiments.
- **Deep Learning Models** could be employed to better predict ratings and preferences, integrating data from multiple sources (e.g., viewing history, reviews, etc.).

### **3. Cross-Platform Sync:**

- Users should be able to **sync** their movie watchlist, ratings, and reviews across both the mobile and web platforms. This cross-platform functionality will ensure users have a consistent experience on any device.

### **4. Enhanced Social Features:**

- **Follow Friends/Users:** Allow users to follow their friends and view their movie ratings and reviews. This would introduce a social aspect to the platform and improve engagement.
- **User-generated Lists:** Let users create personalized movie lists (e.g., "Top 10 Movies," "Movies to Watch," etc.) and share them with others.

### **5. Video Streaming Integration:**

- For added value, future versions could integrate with **streaming services** (e.g., Netflix, Amazon Prime, etc.) to allow users to watch movies directly from the app.

## **REFERENCES**

- [1] Akshaya, M. G., & Mahalakshmi, T. (2023). Movie review application. International Journal of Advanced Research in Science, Communication and Technology. <https://doi.org/10.48175/ijarsct-12938>
- [2] Kumar, P. (2024). Movie review sentiment analysis and AI-story generation web application. International Journal for Science Technology and Engineering. <https://doi.org/10.22214/ijraset.2024.58826>
- [3] Manjunath, D. R., & Hadimani, B. S. (2019). Hierarchical clustering and regression classification-based review analysis on movie-based applications. IEEE International Conference on Advanced Techniques in Computing and Engineering (ICATIECE), 2019. <https://doi.org/10.1109/ICATIECE45860.2019.9063861>
- [4] Topal, K., & Ozsoyoglu, G. (2016). Movie review analysis: Emotion analysis of IMDb movie reviews. Proceedings of the ACM Conference on Information and Knowledge Management. <https://doi.org/10.5555/3192424.3192641>
- [5] Aishwarya, W., Parth, W., & Singh, P. (2020). A new sentiment analysis-based application for analyzing reviews of web series and movies of different genres. IEEE International Conference on Confluence. <https://doi.org/10.1109/CONFLUENCE47617.2020.9058137>
- [6] Li, H., Zhang, H., Zhang, S., Shi, J., Dai, H., Yang, Y., & Cai, H. (2018). Movie review information retrieval system and method based on emotion analysis. Proceedings of the International Conference on Smart Computing (ICSCA).
- [7] Constantinov, C., Mocanu, M., Barbulescu, N., Popescu, E., & Mocanu, A. (2017). MovieRate: Considerations on applying a custom social reputation engine for movie reviews. Proceedings of the IEEE Carpathian Conference. <https://doi.org/10.1109/CARPATHIANCC.2017.7970394>
- [8] Chaovalit, P., & Zhou, L. (2005). Movie review mining: A comparison between supervised and unsupervised classification approaches. Proceedings of the Hawaii International Conference on System Sciences (HICSS). <https://doi.org/10.1109/HICSS.2005.445>
- [9] Golbeck, J. (2006). Filmtrust: Movie recommendations from semantic web-based social networks. Proceedings of the IEEE Consumer Communications and Networking Conference. <https://doi.org/10.1109/CCNC.2006.1593265>
- [10] Mishra, R., Hemant, & Bajaj, P. (2024). Movie review system using machine learning. Research Square. <https://doi.org/10.21203/rs.3.rs-4282512/v1>

[GitHub Repository](#)