MovieHub

Software Requirements Specification

Capstone | COS 301

Team name: Girls Gone Code

Client: EPI-USE

“We declare variables not wars.”

Table of Contents

Introduction

The purpose of MovieHub is to revolutionise the way users discover, review, and engage with films by providing a comprehensive, user-friendly platform that combines advanced recommendation algorithms, community features, and a sleek interface.

There is a growing demand for a centralised platform that caters to movie enthusiasts' needs, from discovering new films to engaging with a community of like-minded individuals. Existing platforms lack a combination of comprehensive movie metadata, personalised recommendations, and strong social networking features. MovieHub aims to fill this gap by providing an all-in-one solution that enhances the movie-watching experience through data-driven recommendations and social interactions.

The scope of the MovieHub project encompasses the development of an innovative movie review platform that integrates a comprehensive movie database with detailed metadata, allows users to generate and rate reviews, and offers personalised movie recommendations through advanced algorithms. It includes social networking features such as following, liking, and commenting, along with robust search and filtering options for easy movie discovery. The project emphasises a responsive and intuitive user interface compatible with various mobile devices.

User Stories/ User Characteristics

Movie Enthusiasts

Movie enthusiasts are the primary users who deeply enjoy watching and discussing films. They will use MovieHub to discover new movies through advanced search and filtering options, read and write detailed reviews, and engage with the community by following other users, liking, and commenting on reviews. The personalised recommendation system will help them find movies that match their tastes based on their viewing history and preferences. Enthusiasts will also appreciate curated lists and the ability to categorise movies using tags.

Critics

Professional critics will use MovieHub as a platform to publish their reviews and reach a broader audience. They can build a following on the platform, interact with their readers through comments, and gain visibility by having their reviews liked and shared. The integration with external platforms like IMDb and Rotten Tomatoes will enhance their credibility and allow cross-platform connectivity.

Aspiring Filmmakers

Aspiring filmmakers will use the app to study popular and critically acclaimed movies to understand industry trends and audience preferences. They will use the movie metadata and reviews to learn what works in the industry, engage with the community to gather feedback and insights and follow other filmmakers and critics to stay inspired and informed.

Account Management

Story 1: As a new user, I want to create an account using my social media account/email so that I can quickly sign up without entering my details manually.

Story 2: As a registered user, I want to update my personal information so that my profile reflects my current details.

Story 3: As a registered user, I want to change my profile picture so that I can personalise my account.

Story 4: As a registered user, I want to manage my privacy settings so that I can control who sees my information and activities.

Story 5: As a registered user, I want to be able to permanently delete my account so that my data is removed from the system.

Authentication

Story 6: As a registered user, I want to securely log into my account so that my information is protected.

Story 7: As a registered user, I want to securely log out of my account so that my session is closed and my data is safe.

Story 8: As a registered user, I want to reset my password if I forget it so that I can regain access to my account.

Story 9: As a user, I want to receive an email to reset my forgotten password so that I can restore my access.

Movie Database Integration

Story 10: As a registered user, I want the platform to integrate with a movie database so that I can access accurate and detailed movie information.

Story 11: As a registered user, I want to view detailed information about each movie so that I can make informed decisions about what to watch.

User Interaction and Engagement

Story 13: As a registered user, I want to rate and review movies so that I can share my opinions and see what others think.

Story 14: As a registered user, I want to receive notifications for interactions and updates so that I can stay informed about activity on my account.

Personalised Movie Recommendations

Story 15: As a registered user, I want to receive personalised movie recommendations so that I can discover films that match my tastes.

Story 16: As a registered user, I want to adjust my preferences so that the movie recommendations I receive are more relevant to my interests.

Advanced Search and Filtering

Story 17: As a registered user, I want advanced search and filtering options so that I can easily discover new films that match specific criteria.

Story 18: As a registered user, I want to filter movies based on release date, rating, and genre so that I can find movies that fit my preferences.

Story 19: As a registered user, I want a search bar so that I can quickly find movies, reviews, and other users.

Social Networking Features

Story 20: As a registered user, I want to create and share curated lists of my favourite movies so that I can showcase my tastes to others.

Story 21: As a registered user, I want to like and comment on other users' reviews so that I can engage with the community and share my thoughts.

Story 23: As a registered user, I want to be able to follow and unfollow other users to manage my social connections.

Story 24: As a registered user, I want to like and comment on reviews so that I can engage with the community and share my thoughts.

User-Generated Movie Reviews and Ratings

Story 25: As a registered user, I want to write and submit movie reviews so that I can share my opinions with other users.

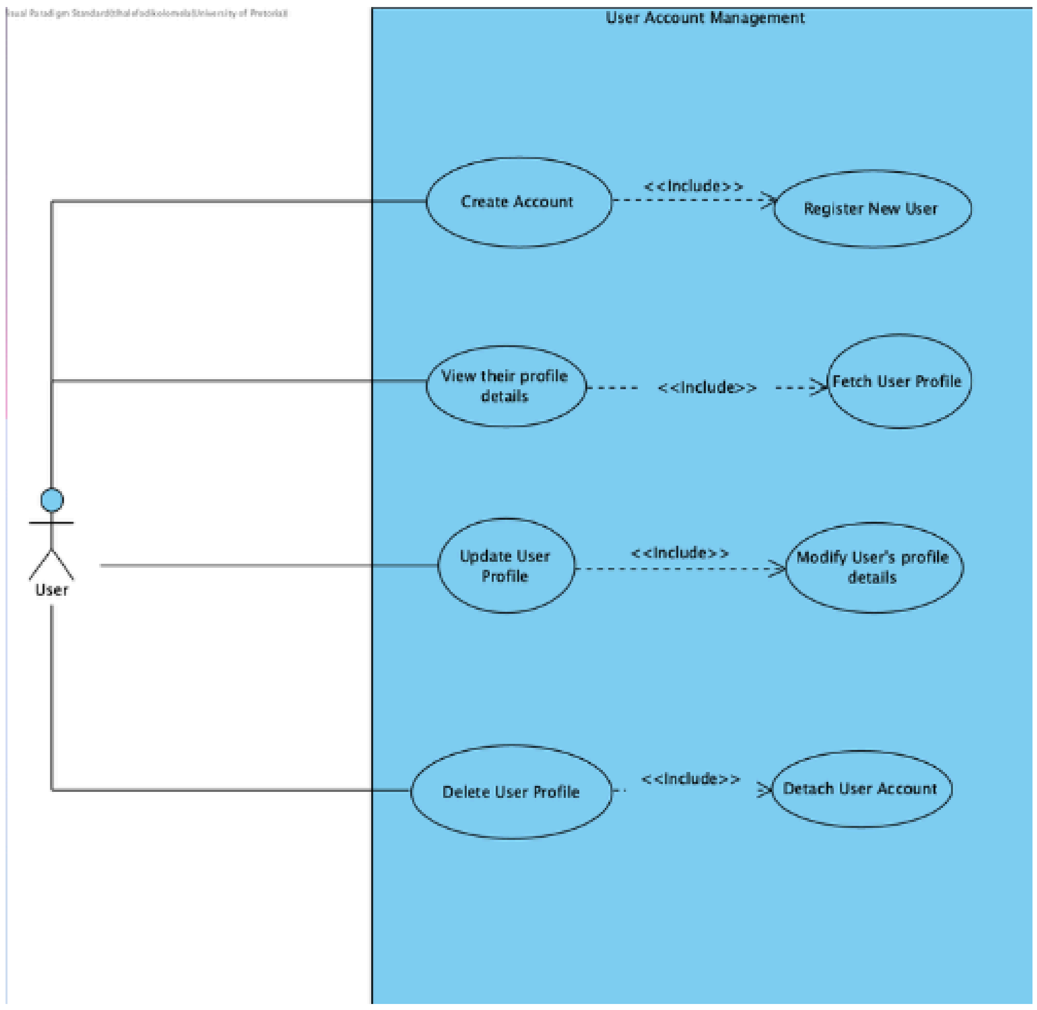
Logging Movies

Story 27: As a registered user, I want to view the movies, reviews, and watchlists I have liked so that I can easily find and revisit my favourite content.

Story 28: As a registered user, I want to create watchlists for movies I have watched or plan to watch so that I can keep track of my viewing history and future plans.

Functional Requirements

Use Case Diagram



Requirements

Account Management

R1.1: The system shall allow users to create accounts using email addresses or social media accounts.

R1.2: The system shall allow users to edit their profiles, including updating personal information and profile pictures.

R1.3: Users can manage privacy settings.

R1.4: Users can permanently delete their accounts.

Authentication

R2.1: The system shall provide secure login functionality.

R2.2: Users can securely log out of their account.

R2.3: The system allows users to reset their passwords.

R2.4: Users can reset a forgotten password via email.

R2.5: Users can change their existing password while still logged in.

Responsive User Interface

R3.1: The platform shall have a responsive and intuitive interface for seamless navigation.

R3.2: Support for multiple device types and screen resolutions shall be provided.

Movie Database Integration

R4.1: The platform shall integrate with a comprehensive movie database to retrieve metadata.

R4.2: Users shall be able to search for movies by title, genre, director, or actor.

R4.3: Detailed information about each movie (synopsis, release date, runtime, cast) shall be accessible.

R4.4: Trailers and multimedia content related to movies shall be available.

User Interaction and Engagement

R5.1: Users shall be able to create accounts securely.

R5.2: Users shall have the ability to rate and review movies.

R5.4: Users shall receive notifications for interactions and updates.

Personalised Recommendations

R6.1: The platform shall provide personalised movie recommendations based on user preferences.

R6.2: Users shall be able to adjust their preferences to refine recommendations.

Advanced Search and Filtering

R7.1: Advanced search and filtering options for discovering new films shall be available.

R7.2: Users shall be able to filter movies based on various criteria (e.g., release date, rating, genre).

R7.3: There will be a search bar to search for movies, reviews, and users.

Social Networking Features

R8.1: Users shall have the ability to create and share curated lists of favourite movies.

R8.2: Tagging movies for categorization and organisation shall be supported.

R8.3: Trending and popular movies within the community shall be visible.

R8.4: Users shall be able to follow other users to keep up with their reviews and activities.

R8.5: Users can follow and unfollow other users.

R8.6: Users can like/comment on reviews.

Content Generation

R9.1: Users can write and submit reviews for movies.

R9.2: Users shall be able to edit or delete their reviews.

R9.3: Users shall be able to comment on movie reviews.

R9.4: Users can create posts to share their thoughts and engage with other users.

Logging Movies

R10.1: Users can view the movies, reviews, and watchlists they have liked.

R10.2: Users can make watchlists for movies they have watched or plan on watching.

Service Contracts

Create User Account

Endpoint: POST /users

Headers:

Content Type: application/json

Request Body:

```
{
  "username": "Jane",
  "email": "jane.doe@gmail.com",
  "password": "janeDaDoe"
}
```

Response:

```
{
  "userId": "pTjrHHYS2qWczf4mKExik40KgLH3",
  "username": "Jane",
  "email": "jane.doe@gmail.com",
  "createdAt": "2 Jun 2024"
}
```

Update User Profile

Endpoint: PUT /users/{userId}

Headers:

Content Type: application/json

Request Body:

```
{
  "username": "Jane",
  "email": "jane.doe@gmail.com",
  "profilePicture": "string",
  "privacySettings": {
    "showEmail": "boolean",
    "showActivity": "boolean"
  }
}
```

Response:

```
{
  "userId": "pTjrHHYS2qWczf4mKExik40KgLH3",
  "username": "Jane",
  "email": "jane.doe@gmail.com",
  "profilePicture": "string",
  "privacySettings": {
    "showEmail": "boolean",
    "showActivity": "boolean"
  },
  "updatedAt": "2 Jun 2024"
}
```

Delete User Account

Endpoint: DELETE /users/{userId}
Headers:
    Content Type: application/json
Response:

```
{
  "message": "Account successfully deleted."
}
```

Authenticate User

Endpoint: POST /auth/login
Headers:
    Content Type: application/json
Request Body:

```
{
  "email": "jane.doe@gmail.com",
  "password": "janeDaDoe"
}
```

Response:

```
{
  "userId": "pTjrHHYS2qWczf4mKExik40KgLH3",
  "username": "Jane",
  "token": "string" // JWT or session token
}
```

Logout User

Endpoint: POST /auth/logout
Headers:
    Content Type: application/json
    Authorization: Bearer <token>
Response:

```
{
  "message": "Successfully logged out."
}
```

Reset Password

Endpoint: POST /auth/reset-password
Headers:
    Content Type: application/json
Request Body:

```
{
  "email": "jane.doe@gmail.com"
}
```

Response:

```
{
  "message": "Password reset email sent."
}
```

Update Password

Endpoint: PUT /auth/update-password

Headers:

    Content Type: application/json
    Authorization: Bearer <token>

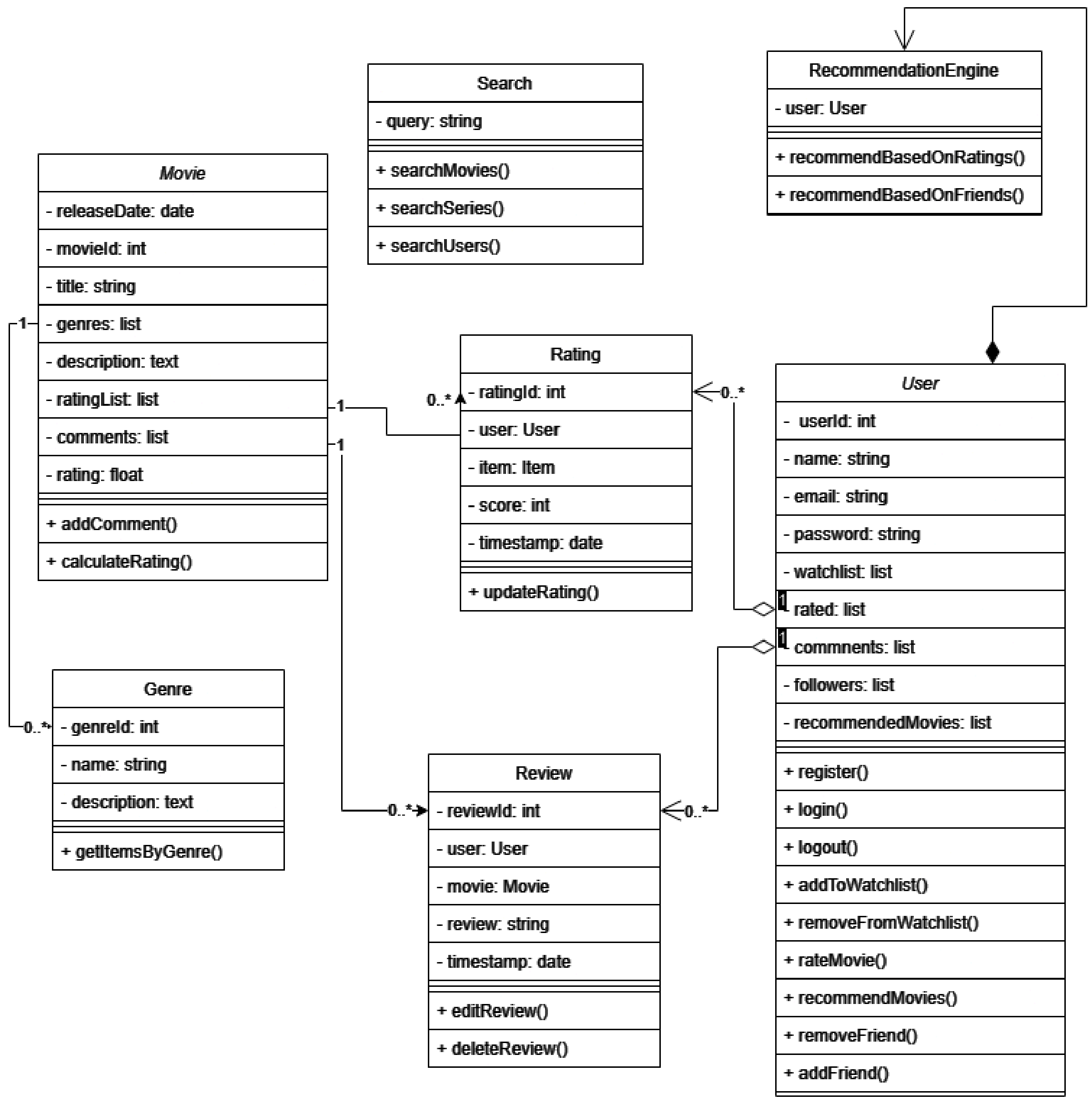
Request Body:

```
{
  "currentPassword": "string",
  "newPassword": "string"
}
```

Response:

```
{
  "message": "Password successfully updated."
}
```

Class Diagram



Architectural Requirements

Quality Requirements

1. Performance. The system should:
    - Responds promptly to user actions and requests - deliver real time results
    - Maintain consistent performance under varying user loads.
2. Scalability. The system should:
    - Be able to handle increasing user traffic/demand data volume.
    - Ensure the database can scale to handle growing amounts of data and transactions.
3. Security. The system should:
    - Implement robust authentication and authorization mechanisms to protect user data.
    - Encrypt sensitive information
4. Portability.The system should:
    - Ensure compatibility across different mobile devices and screen sizes
5. Reliability. The system should:
    - Ensure the system is consistently available for users.
    - Implement measures to recover gracefully from failures and prevent data loss.
    - Ensure the integrity and accuracy of user data.

Architectural Patterns

Model-View-Controller (MVC) Pattern

MVC separates the application into three interconnected components: Model (data), View (UI), and Controller (business logic).

Separation of Concerns: Keeps the codebase organised and easier to maintain.

Scalability: Different components can be developed and scaled independently.

Testability: Each component can be tested independently, facilitating easier debugging and testing.

Service-Oriented Architecture (SOA)

SOA is similar to microservices but services are often larger and communicate through an enterprise service bus (ESB).

Modular Development: Encourages the development of modular services that can be reused across the application.

Interoperability: Facilitates integration with external systems and services, which is useful for integrating with third-party movie APIs.

Client-Server Architecture

A distributed structure where service providers (servers) handle core functions and clients (e.g., mobile apps, web browsers) request services.

Implementation: Use Node.js for backend server logic, Firebase for real-time data synchronisation and authentication, and Neo4j for complex data relationships.

Benefits:

Separation of Concerns: Clients handle the user interface, while servers manage data and business logic.

Scalability: Servers and clients can be scaled independently to handle increased load.

Security: Centralised control over authentication, authorization, and data protection.

Flexibility: Supports multiple types of clients (web, mobile) using the same APIs.

Maintainability: Easier updates and debugging with a modular codebase.

Design Patterns

1. **Repository Pattern**
    1.1. Abstracts the data layer, providing a way to interact with the database through a set of defined methods.
    1.2. Implementation: Use repositories in the backend to handle CRUD operations with Firebase and Neo4j.
    1.3. Benefits: Simplifies data access, improves testability, and enforces separation of concerns.
2. **Factory Pattern**
    2.1. Provides an interface for creating objects without specifying the exact class of object that will be created.
    2.2. Implementation: Use factories to create instances of services or objects in the application.
    2.3. Benefits: Promotes loose coupling and enhances scalability.
3. **Observer Pattern**
    3.1. Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified.
    3.2. Implementation: Use for implementing real-time features like notifications and updates.
    3.3. Benefits: Enables real-time communication and event handling.

Constraints

Technology Requirements

1. **Frontend Development:**

   1.1. Use React Native for mobile.

   1.2. Use component libraries like NativeBase or React Native Paper for UI consistency and rapid development.

2. **Backend Development:**

   2.1. Alternatively, use Node.js with Express for a unified JavaScript stack.

   2.2. Leverage Firebase for real-time data synchronisation, authentication, and hosting.

3. **API Development:**

   3.1. Use REST or GraphQL for API development, depending on the specific requirements.

   3.2. Utilise Firebase Functions to handle server-side logic and API endpoints.

4. **Database Management:**

   4.1. Use Neo4j for handling complex relationship queries and graph data.

   4.2. Use Firebase Firestore for real-time data storage and synchronisation.

5. **CI/CD and Deployment:**

   5.1. Implement CI/CD pipelines using GitHub Actions.

5.2. Deploy applications using Expo for unified mobile and web deployment.

5.3. Use Firebase Hosting for web application deployment.

6. **Search and Filtering:**

6.1. Use Elasticsearch for advanced search and filtering capabilities, ensuring high performance and scalability.

7. **Testing:**

7.1. Use Jest for unit testing, Supertest for API testing, and Cypress for end-to-end testing.

8. **Documentation and Team Collaboration:**

8.1. Use Google Docs and Markdown for documentation.

8.2. Use GitHub for project management.

8.3. Use Discord and Blackboard Collaborate for real-time communication and collaboration.