Software Documentation for Movie Recommendation Website

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the development of the Movie Recommendation Website. This platform allows users to register, search for movies, receive personalized recommendations, and filter movies by genre, rating, and release year.

1.2 Scope

The website will provide users with a dynamic movie browsing experience, utilizing data from the TMDB API. The backend is powered by Node.js and Express.js, while the frontend is built using React.js. The platform will include user authentication, movie search, filtering, and recommendation features.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirements Specification
- **TMDB:** The Movie Database (API for movie data)
- **API:** Application Programming Interface
- **React.js:** A JavaScript library for building user interfaces
- **Node.js:** A JavaScript runtime for backend development
- **Express.js:** A web application framework for Node.js

2. Overall Description

2.1 Product Perspective

The Movie Recommendation Website is a new product designed to cater to users interested in exploring movie content and receiving recommendations based on their search queries. It will be accessible through a browser and requires internet access.

2.2 Product Functions

- User registration and login with authentication.
- Home page displaying a list of popular movies fetched from the TMDB API.
- Search functionality to find movies by name.
- Movie recommendations based on search queries.
- Filtering options for movies by genre, rating, and release year.
- Detailed view for individual movies, displaying information such as title, rating, description, and release date.

2.3 User Classes and Characteristics

- **General Users:** Can browse, search, and filter movies. Users need to register and log in to receive personalized recommendations.
- **Admin (optional):** Can manage content and moderate user activity (if implemented).

2.4 Operating Environment

- **Frontend:** React.js (compatible with major browsers like Chrome, Firefox, Safari)
- **Backend:** Node.js with Express.js, running on a server (Linux/Windows)
- **API:** TMDB API for movie data and custom filtering API

2.5 Design and Implementation Constraints

- The website requires integration with the TMDB API, thus API rate limits must be considered.
- Movie data updates rely on TMDB's API availability and accuracy.
- Authentication security should comply with best practices (e.g., password hashing, HTTPS).

2.6 Assumptions and Dependencies

- Users have access to a stable internet connection.
- Users will need to create an account to save preferences or receive recommendations.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Registration and Authentication

- The system shall provide users with the ability to register using an email address, username, and password.
- The system shall allow registered users to log in with valid credentials.
- The system shall maintain session management (using JWT or sessions) for authenticated users.
- The system shall enforce validation rules (e.g., password length, email format).

3.1.2 Movie Search

- The system shall allow users to search for movies using a search bar.
- The system shall display movie search results fetched from the TMDB API based on user queries.
- The system shall allow users to click on a movie result to view detailed information about the selected movie.

3.1.3 Movie Recommendations

- The system shall suggest a list of recommended movies based on the user's search query.
- Recommendations shall be fetched from the TMDB API based on movie similarity, genre, and popularity.

3.1.4 Movie Filtering

- The system shall provide filters for users to refine movie searches by genre, rating, and release year.
- The system shall update the movie list dynamically as filters are applied.

3.1.5 Home Page

- The system shall display a list of popular movies on the home page, fetched from the TMDB API.
- The home page shall include a navigation bar with a search bar, login, and registration links.

3.1.6 Movie Details Page

- The system shall display detailed movie information when a user selects a movie.
- The movie details page shall include the title, rating, release date, description, genre, and similar movies.

3.2 Non-Functional Requirements

3.2.1 Performance Requirements

- The system shall respond to user actions (e.g., search, filter) within 2 seconds.
- The system shall handle up to 100 concurrent users without degradation in performance.

3.2.2 Security Requirements

- The system shall store passwords securely using hashing algorithms.
- The system shall protect against common web vulnerabilities (e.g., SQL injection, XSS).
- All API calls should be protected using HTTPS to ensure secure communication between the frontend and backend.

3.2.3 Usability Requirements

- The system shall have a responsive design to ensure usability on different devices (mobile, tablet, desktop).
- The system shall be intuitive and easy to navigate for users with basic internet knowledge.

3.2.4 Scalability Requirements

- The system should be scalable to support future enhancements, including features like user profiles, favorites, and advanced recommendation algorithms.

3.3 External Interface Requirements

3.3.1 User Interfaces

- **Login Page:** A form with fields for email and password, and a link to the registration page.
- **Registration Page:** A form with fields for username, email, password, and a confirm password field.
- **Home Page:** A page displaying popular movies with a search bar and navigation links.
- **Movie Details Page:** A page that displays detailed information about the selected movie.
- **Search Results Page:** A page that shows movie results based on user search, along with recommendations.

3.3.2 API Interfaces

- TMDB API: Used for retrieving movie data.
- Custom Filtering API: Endpoints for movie filtering.

3.3.3 Database Interfaces

- User Database: Stores user information including username, email, hashed passwords, and session tokens.
- Movie Database: (Optional) If storing additional movie information locally for caching.

3.3.4 Hardware Interfaces

- The website shall be compatible with standard hardware configurations (PCs, mobile devices, etc.).

4. System Features

- **Feature 1:** User Authentication
- **Feature 2:** Movie Search and Display
- **Feature 3:** Filtering and Recommendation System
- **Feature 4:** Detailed Movie Information

5. Other Non-Functional Requirements

5.1 Reliability

- The system shall have an uptime of 99.9% to ensure constant availability of movie data and user features.

5.2 Maintainability

- The system shall be built with modular code architecture, ensuring ease of future maintenance and updates.