**System Analysis & Design for Movie Recommendation Website:**

**1. Introduction**

The Movie Recommendation Website is an online platform designed to help users discover movies based on their preferences, ratings, and viewing history. The system uses React.js for the frontend, an external Movie Database API (e.g., TMDb API) for movie details, and a backend for user data and recommendations.

**2. System Objectives**

* Provide personalized movie recommendations using AI/algorithms.
* Enable users to search and filter movies based on various criteria.
* Allow users to rate movies and write reviews.
* Implement user authentication for a personalized experience.
* Offer a watchlist feature to save favorite movies.

**3. System Requirements**

**3.1 Functional Requirements**

**User Authentication & Profile Management:**

* Register/Login via Email & Password (JWT Authentication or Firebase Auth).
* User profile with preferences and history.
* Movie Search & Filtering
* Search movies by title, genre, year, rating, or actors.
* Apply filters based on popularity, trending, or user interests.

**Movie Recommendation Engine**

* Collaborative Filtering (based on similar users’ preferences).
* Content-Based Filtering (based on movie genres, actors, directors).
* Movie Details Page Show movie description, genre, cast, trailer, and ratings.
* Fetch details from TMDb API or an internal database.
* Watchlist & Favorites Allow users to add movies to their watchlist.
* Save favorite movies for future reference.

**Reviews & Ratings**

* Users can rate movies from 1 to 10 stars.
* Write and read reviews from other users.
* Admin Panel (Optional - for future expansion) Manage movies, users, and recommendations.

**3.2 Non-Functional Requirements**

* Performance: Fast UI rendering with optimized API calls.
* Scalability: Supports increasing user traffic.
* Security: User authentication and secured API calls.
* Usability: Simple, intuitive UI for a better user experience.
* Availability: Ensure high uptime and quick response times.

**4. System Design**

**4.1 System Architecture**

The system follows a 3-tier architecture:

* Built with React.js + Redux / Context API.
* Uses Material UI or Tailwind CSS for styling.
* Fetches data via REST API or GraphQL.
* Backend (Node.js + Express.js)
* Handles user authentication and API communication.
* Stores user preferences and ratings.
* Implements recommendation logic.
* Database (MongoDB / PostgreSQL) Stores user profiles, watchlists, and ratings.
* Uses TMDb API for movie data.

**4.2 Data Flow Diagram (DFD)**

**Level 1: High-Level Overview**

* User interacts with the frontend UI (search, rate movies, etc.).
* The frontend makes an API request to the backend.
* The backend retrieves movie data from the TMDb API.
* The backend processes recommendations based on user behavior.
* The system sends recommended movies back to the frontend.

**5. Database Design**

**5.1 Database Tables**

Table Name Fields

Users user\_id, name, email, password, profile\_picture, preferences

Movies movie\_id, title, genre, director, actors, release\_year, rating, description

Ratings rating\_id, user\_id, movie\_id, rating, review, timestamp

Watchlist watchlist\_id, user\_id, movie\_id, added\_date

**6. UI/UX Design**

**6.1 Pages & Components**

* **Home Page:**

Displays Trending Movies, Top Rated, and User Recommendations.

* **Movie Details Page:**

Shows full movie description, cast, and trailer.

Displays user reviews and average rating.

* **Search & Filter Page:**

Users can search by title, genre, or rating.

* **Watchlist Page:**

Lists movies added to the user's watchlist.

* **User Profile:**

Shows user ratings, reviews, and favorite movies.

**7. Technology Stack**

Layer Technology

Frontend React.js, Redux/Context API, Tailwind CSS / Material UI

Backend Node.js, Express.js

Database MongoDB (NoSQL) / PostgreSQL (SQL)

External API TMDb API for movie data

Authentication Firebase Auth / JWT

Hosting Vercel / Netlify (Frontend), Heroku / DigitalOcean (Backend)

**8. Recommendation Engine**

* **Approach 1: Content-Based Filtering:**

Recommends movies similar to ones the user has liked.

Matches movies based on genre, actors, director, and keywords.

* **Approach 2: Collaborative Filtering:**

Suggests movies based on similar users' ratings.

Uses techniques like KNN (K-Nearest Neighbors) or Matrix Factorization.

* **Approach 3: Hybrid Approach:**

Combines content-based and collaborative filtering for better recommendations.

**9. Future Enhancements**

* AI-Powered Recommendations using Machine Learning (ML).
* User Social Features (Follow friends, see their watchlist).
* Mobile App Version (React Native for cross-platform support).
* Real-time Chat for movie discussions.
* Multi-language support for a global audience.

**10. Conclusion**

This Movie Recommendation Website is designed for a personalized user experience, leveraging React, Node.js, and APIs to deliver high-quality movie suggestions. With scalability and future expansion in mind, the system provides a robust foundation for recommending movies effectively.