INTERNSHIP PROJECT DOCUMENT ON

“MOVIE RECOMMENDATION SYSTEM”

Submitted by

(Intern)

Sai Kiran Annoju

Submitted to

Founder - Kanduri Abhinay

CEO & CTO - Rithin Varma

INTRODUCTION:

The Movie Recommendation System is a web-based application designed to provide personalized movie recommendations based on user preferences. This system allows users to search for movies, view detailed information (such as ratings, cast, and synopsis), and create/manage a watchlist of their favorite movies. The project demonstrates the implementation of a full-stack web application using modern technologies and best practices.

The system is built using Vue.js for the frontend and Node.js (Express) for the backend, with The Movie Database (TMDB) API for fetching movie data. The application allows users to search for movies, view recommendations, and manage their watchlist. The project showcases the use of modern web development practices, including API integration, dynamic rendering, and personalized recommendations.

The Movie Recommendation System is a practical example of how web technologies can be used to build interactive and scalable applications. It also highlights the importance of user experience, real-time updates, and responsive design in modern web applications.

SOFTWARE DEVELOPMENT LIFE CYCLE(SDLC)

The development of the Movie Recommendation System followed the **SDLC framework** to ensure a structured and efficient process:

1. Planning

- **Objective**: Develop a web-based movie recommendation system for searching, viewing, and managing movies.
- **Scope**: Build a full-stack application with a Vue.js frontend, Node.js backend, and TMDB API integration.
- **Feasibility**: Utilize modern web technologies for scalability and responsiveness.

2. Defining Requirements

- **Software Requirements**:
  - **Operating System**: Windows, macOS, or Linux.
  - **Frontend**: Vue.js, HTML5, CSS3, JavaScript (ES6+), Axios.
  - **Backend**: Node.js, Express.js, TMDB API.
  - **Development Tools**: Visual Studio Code, Git, Node.js, npm.
- **Hardware Requirements**: Modern web browser with JavaScript support.

3. Designing

- **Frontend Architecture**: Component-based design using Vue.js.
- **Backend Architecture**: RESTful API design using Express.js.
- **Database Design**: Local storage for managing the watchlist.
- **UI Design**: Responsive and user-friendly interface for seamless navigation.

4. Building (Implementation)

- **Frontend**: Developed using Vue.js with Vue Router for navigation and Axios for API requests.
- **Backend**: Built using Node.js and Express.js for handling API requests to TMDB.
- **Features Implemented**:
  - Search movies and view details (ratings, cast, synopsis).
  - Generate personalized recommendations.
  - Create and manage a watchlist.
  - Delete movies from the watchlist.

5. Testing

- **Unit Testing**: Validated API endpoints and frontend components.
- **Edge Cases**: Tested invalid inputs, empty search results, and watchlist functionality.
- **Performance**: Ensured fast and responsive performance for real-time updates.

6. Deployment

- Deployed as a standalone web application.
- **Future Deployment Plans**:
  - Integration with cloud services for scalability.
  - Mobile app development for on-the-go recommendations.

7. Conclusion

Following the SDLC methodology ensured a systematic approach to developing the Movie Recommendation System, resulting in a robust and user-friendly application with scalability for future enhancements.

PROCEDURE AND METHODS USED:

1. **Frontend Development:**
   - Built using Vue.js with a component-based architecture.
   - Used Vue Router for navigation and Axios for making HTTP requests to the backend.
   - Implemented dynamic rendering for movie cards and watchlist items.
2. **Backend Development:**
   - Developed using Node.js and Express.js for handling API requests.
   - Integrated TMDB API for fetching movie data.
   - Implemented CORS for handling cross-origin requests.
3. **Watchlist Management:**
   - Used localStorage to store and manage the user's watchlist.
   - Implemented functionality to add and delete movies from the watchlist.
4. **Validation and Error Handling:**
   - Input validation ensures data integrity.
   - Error handling manages invalid searches and unexpected issues.

ALGORITHM:

1. **Movie Search:**
   - User inputs a search query.
   - Data is sent to the backend via a GET request to the TMDB API.
   - Backend fetches movie data and returns it to the frontend.
   - Frontend displays the search results dynamically.
2. **Watchlist Management:**
   - User clicks the "Add to Watchlist" button on a movie.
   - Movie data is stored in localStorage.
   - User can view and delete movies from the watchlist.
3. **Movie Details:**
   - User clicks on a movie card to view details.
   - Frontend sends a GET request to fetch detailed movie data from the TMDB API.
   - Backend returns the data, and the frontend displays it.

FUTURE SCOPE:

1. **User Authentication:** Add user authentication for personalized experiences.
2. **Advanced Recommendations:** Integrate machine learning for personalized movie recommendations.
3. **Social Sharing:** Allow users to share movies on social media.
4. **Multi-Platform Support:** Develop mobile apps for iOS and Android**.**

FUTURE ENHANCEMENTS:

1. **User Profiles:** Allow users to create profiles and save preferences.
2. **Movie Categories**: Introduce categories (e.g., genre, year) for better organization**.**
3. **Advanced Analytics:** Provide insights into user preferences and watchlist trends**.**
4. **Offline Mode:** Enable offline access to the watchlist.
5. **Multi-Language Support:** Add support for multiple languages.

ADVANTAGES:

1. **Personalized Recommendations:** Provides tailored movie suggestions based on user preferences.
2. **User-Friendly Interface:** Easy to use and navigate.
3. **Responsive Design:** Works seamlessly on desktop and mobile devices**.**
4. **Scalable Architecture**: Designed to handle a large number of users and movies**.**
5. **Real-Time Updates:** Users can see search results and watchlist updates in real-time

CONCLUSION :

The **Movie Recommendation System** successfully demonstrates the implementation of a modern web application using Vue.js and Node.js. The system provides a solid foundation for future enhancements and highlights the importance of user experience, real-time updates, and responsive design in modern web applications.

REFERENCES:

1. **Vue.js Documentation: https://vuejs.org/**
2. **Node.js Documentation: https://nodejs.org/**
3. **Express.js Documentation: https://expressjs.com/**
4. **TMDB API Documentation: https://developers.themoviedb.org/**
5. **MDN Web Docs: https://developer.mozilla.org/**