# High-Level Document (HLD)

**Movie Recommendation Application Based on Category**

---

**1. Project Title and Overview**

**Title: Movie Recommendation Application Based on Category**

Overview:

The Movie Recommendation Application suggests movies to users based on their preferred categories. Users can select a category, and the application will recommend movies from that category using a recommendation algorithm. This project aims to provide a personalized movie-watching experience by leveraging user preferences and movie data.

**2. Objective**

- To develop a movie recommendation system that suggests movies based on user-selected categories.

- To implement a user-friendly interface where users can select categories and receive movie recommendations.

- To utilize a recommendation algorithm to ensure the suggestions are relevant and personalized.

---

**3. Scope**

\*\*Included:\*\*

- User interface for selecting movie categories.

- Integration with a movie database or API to fetch movie data.

- Implementation of a recommendation algorithm to generate movie suggestions.

- Display of movie recommendations with basic details (title, poster, description).

\*\*Excluded:\*\*

- Advanced recommendation algorithms (e.g., collaborative filtering).

- Real-time data updates or movie streaming functionality.

- Integration with multiple movie databases or APIs.

---

**4. Technical Stack**

- \*\*Frontend:\*\* Streamlit (for creating the web application interface)

- \*\*Backend:\*\* Python (for implementing the recommendation algorithm and data processing)

- \*\*Data Storage:\*\* Local storage or cloud-based movie database/API (e.g., TMDb API)

- \*\*Libraries and Frameworks:\*\* `pickle` (for loading pre-trained models), `requests` (for API calls)

---

**6.Design Description:**

1. \*\*User Interface:\*\*

- Users interact with the Streamlit app to select movie categories.

- The interface allows users to view recommendations based on their selected category.

2. \*\*Backend:\*\*

- Processes user input and interacts with the movie database/API.

- Implements the recommendation algorithm to generate a list of movie suggestions.

3. \*\*Movie Database/API:\*\*

- Provides movie data, including titles, posters, and descriptions.

- Example: TMDb API.

4. \*\*Recommendation Algorithm:\*\*

- Uses movie data to generate recommendations based on the selected category.

---

7. Implementation Details

- \*\*Frontend Implementation:\*\*

- Developed using Streamlit to create an interactive web interface.

- Allows users to select a movie category and view recommendations.

- \*\*Backend Implementation:\*\*

- Python script handles data processing and interaction with the movie database/API.

- Implements a basic recommendation algorithm based on movie categories.

- \*\*Data Integration:\*\*

- Fetch movie data using the TMDb API or a local dataset.

- Process and format data for recommendation.

---

**8. Challenges and Solutions**

- \*\*Challenge:\*\* Limited movie data or API access.

- \*\*Solution:\*\* Use a local dataset as a backup or ensure API access is reliable.

- \*\*Challenge:\*\* Ensuring accurate and relevant recommendations.

- \*\*Solution:\*\* Implement and test different recommendation algorithms to improve accuracy.

---

**9. Results and Outcomes**

- \*\*Functionality:\*\* Successfully implemented a recommendation system that provides movie suggestions based on user-selected categories.

- \*\*User Experience:\*\* Users can easily navigate the interface and receive relevant movie recommendations.

---

**10. Future Work**

- Enhance the recommendation algorithm to incorporate user ratings and preferences.

- Integrate additional movie categories or genres.

- Implement user accounts and save preferences for a more personalized experience.

---

**11. Conclusion**

The Movie Recommendation Application provides a simple and effective way for users to discover movies based on their interests. By using a basic recommendation algorithm and leveraging movie data from an API, the application delivers personalized suggestions and enhances the user experience.

---

**12. References**

- [TMDb API Documentation](https://www.themoviedb.org/documentation/api)

- Streamlit Documentation: [Streamlit](https://docs.streamlit.io/)

- Python `requests` Library: [Requests Documentation](https://requests.readthedocs.io/)

Feel free to adjust the details based on your specific implementation and requirements.