**System Analysis & Design**

**1. Problem Statement & Objectives**

**Problem Statement: In today’s digital age, users struggle to find relevant movies based on their interests. With vast amounts of movie data available, an intelligent recommendation system is needed to enhance user experience.**

**Objectives:**

* **Develop an interactive movie recommendation system.**
* **Fetch real-time movie data using a third-party API.**
* **Implement user-friendly search, filter, and recommendation features.**
* **Ensure scalability, performance, and security of the system.**

**2. Use Case Diagram & Descriptions**

**Use Case Diagram: The system will include the following actors and interactions:**

* **User: Searches for movies, applies filters, views recommendations.**
* **System: Fetches and displays movie data, processes recommendations.**
* **API Provider (TMDb API): Supplies movie information.**

**(Insert Use Case Diagram Here)**

**Use Case Descriptions:**

1. **Search Movies: Users input a movie title, and the system fetches relevant results.**
2. **Apply Filters: Users filter movies based on genre, rating, and year.**
3. **Get Recommendations: The system suggests movies based on trending data and user preferences.**

**3. Functional & Non-Functional Requirements**

**Functional Requirements:**

* **Users can browse, search, and filter movies.**
* **The system fetches data from the API and displays it.**
* **Users receive personalized movie recommendations.**
* **Pagination and sorting functionalities are implemented.**

**Non-Functional Requirements:**

* **The system must load data within 2 seconds.**
* **API requests should be secure and optimized.**
* **The UI must be responsive and accessible.**
* **The architecture should support scalability and maintainability.**

**4. Software Architecture**

**The system follows a React-Redux frontend architecture with API integration.**

**Architecture Style:**

* **Frontend: React with Redux for state management.**
* **Data Source: The Movie Database (TMDb) API.**

**High-Level Design:**

* **UI Layer: Handles user interactions and displays movie data.**
* **State Management Layer: Manages global state using Redux.**
* **API Integration Layer: Fetches movie data from TMDb.**
* **Recommendation Engine: Filters and sorts movie data based on user preferences.**

**This document outlines the foundational structure of the movie recommendation system, ensuring clarity in development and implementation.**