**Movie Recommendation App - Requirements & System Design**

**1. Requirements Gathering**

**1.1 Stakeholder Analysis**

**Key Stakeholders:**

* **End Users**: People who will use the app to search and discover movies.
* **Developers**: Responsible for building and maintaining the app.
* **Business Owners**: The person or company funding the project and interested in its success.
* **API Providers**: Third-party movie database services.

**Stakeholder Needs:**

* Users: Easy-to-use interface, accurate movie recommendations, and smooth browsing experience.
* Developers :require clear documentation and modular architecture.
* Business owners : A functional app that meets user expectations, potential for monetization, and high user engagement.
* API providers demand proper usage of their services and compliance.

**1.2 User Stories & Use Cases**

**User Stories:**

* As a user, I want to search for movies by name so that I can find specific films.
* As a user, I want to filter movies by genre, rating, and release year to find relevant movies.
* As a user, I want to view trending movies so that I can see what's popular.
* As an admin, I want to track user interactions to improve recommendations.

**Use Case Example:**

* **Use Case Name**: Searching for a movie
  + **Actors**: User
  + **Description**: User enters a movie title in the search bar and gets relevant results.
  + **Preconditions**: User has accessed the search function.
  + **Postconditions**: Relevant movies are displayed.

**1.3 Functional Requirements**

* User authentication (Sign up, Login, Logout)
* Movie search and filtering by genre, rating, and release year
* Trending movie recommendations
* Movie Filtering Users can filter movies by:
  + **Genre** (e.g., Action, Comedy).
  + **Release Year**.
  + **Rating** (e.g., IMDb score)
* API integration with a third-party movie database

**1.4 Non-functional Requirements**

* **Performance**: Fast response times for API calls
* **Security**: Secure API calls and user authentication
* **Usability**: User-friendly UI/UX with smooth navigation
* **Reliability**: Robust error handling and fallback mechanisms

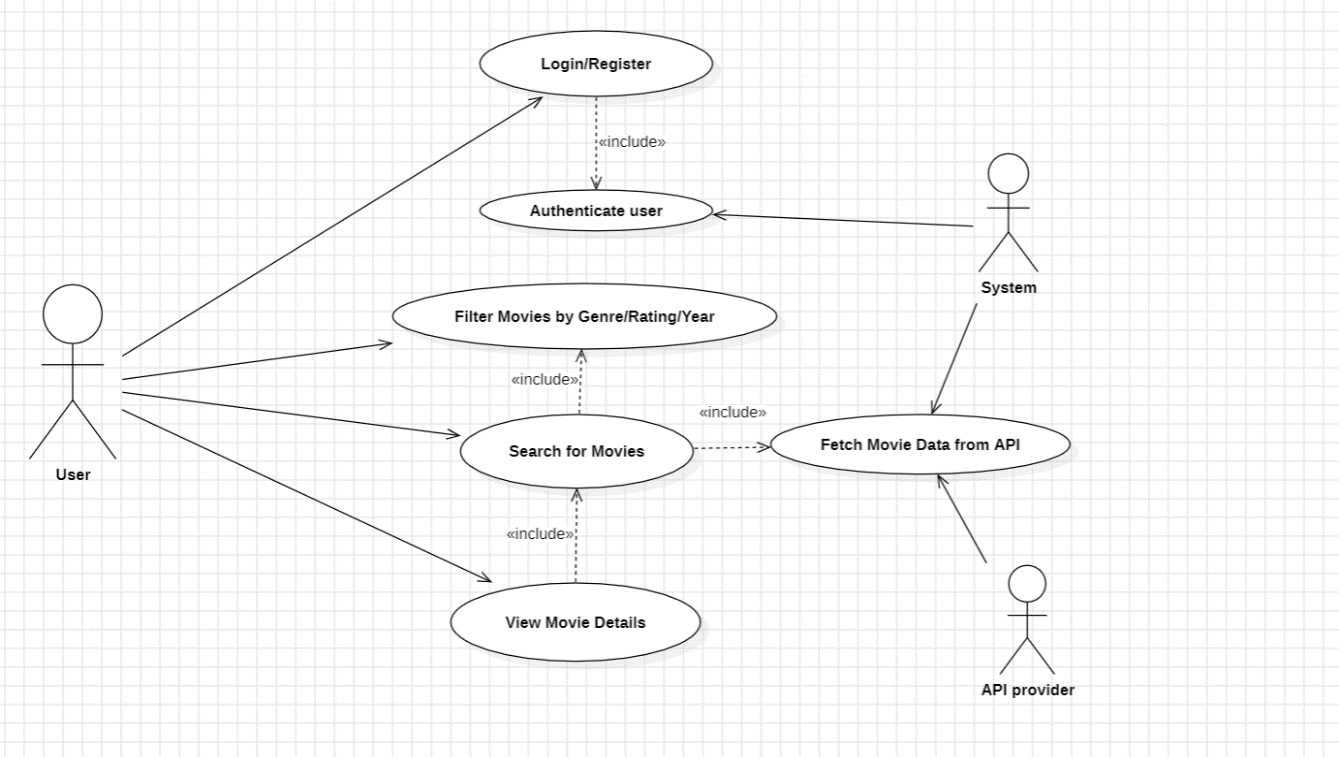
**2. System Analysis & Design**

**2.1 Problem Statement & Objectives**

**Problem Statement:** Users struggle to find relevant movies based on their interests due to overwhelming options.

**Objectives:**

* Build an easy-to-use app for discovering movies based on user preferences.
* Implement a scalable solution with an intuitive interface.
* Ensure real-time fetching of movies using a reliable API.

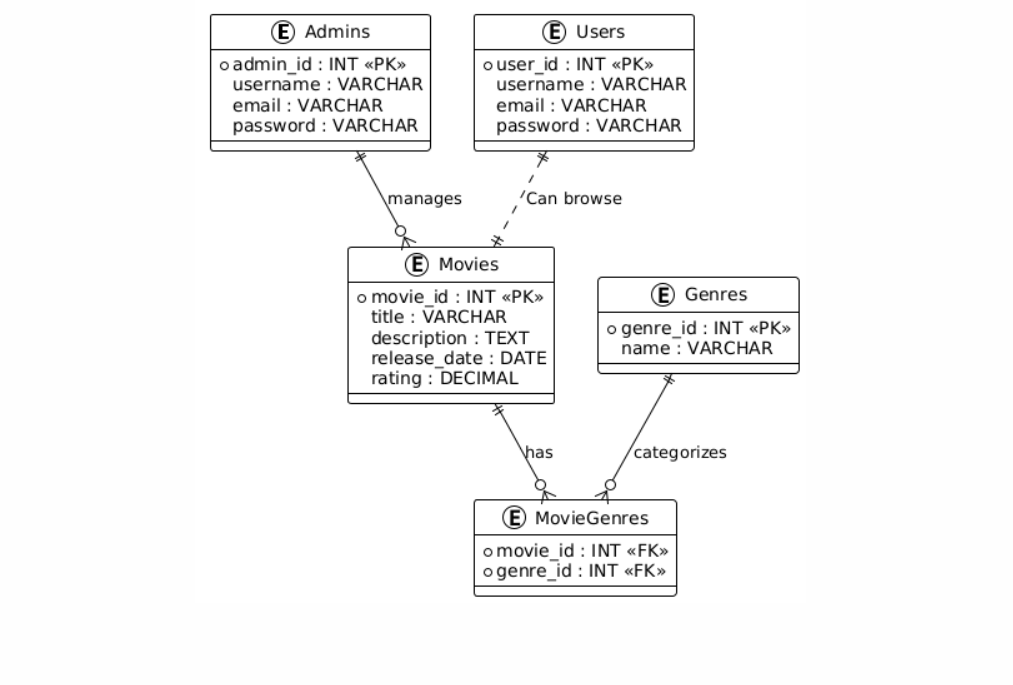
**2.2 Use Case Diagram**

**2.3 Software Architecture**

* **Architecture Style:** Component-based architecture using **React**
* **State Management:** Redux for global state handling
* **Backend API:** Node.js for server-side interactions (if needed)
* **Frontend Framework:** React with modular components

**3. Database Design & Data Modeling**

**3.1 ER Diagram**



**3.2 Logical & Physical Schema**

| **Table** | **Attributes** | **Primary Key (PK)** | **Foreign Key (FK)** |
| --- | --- | --- | --- |
| **Users** | user\_id, username, email, password | user\_id | - |
| **Admins** | admin\_id, username, email, password | admin\_id | - |
| **Movies** | movie\_id, title, description, release\_date, rating | movie\_id | admin\_id (FK) |
| **Genres** | genre\_id, name | genre\_id | - |
| **MovieGenres** | movie\_id, genre\_id | movie\_id + genre\_id | movie\_id, genre\_id |