•Software Design Document

1. Introduction

This document outlines the design of a movie booking application, which facilitates users to browse, select, and book movies. It also provides administrative functionalities to manage movies and bookings. The application aims to provide a user-friendly interface for users to interact with movies and bookings.

2. System Overview

The system consists of two main components:

- **User Interface: Provides an interface for users to interact with the application.**
- **Backend Server: Handles requests from the user interface, performs necessary operations, and communicates with the database.**

3. Functional Requirements

3.1 Admin Functionality

1. **Save Movie Details (POST): Admin can add new movies to the database by providing movie details.**
2. **Update Movie Details (PUT): Admin can update existing movie details such as title, description, genre, etc.**
3. **Delete Movie (DELETE): Admin can delete movies from the database.**
4. **Save Booking Details (POST): Admin can record booking details including user information, movie, date, and time.**
5. **List Movies (GET): Admin can view the list of available movies.**
6. **List Booking Details (GET): Admin can view the list of all booking details.**

7. **List Booked Movies (GET): Admin can view the list of movies that have been booked.**

3.2 User Functionality

1. **User Registration (POST): Users can register by providing necessary details.**
2. **User Login (POST): Registered users can log in to the application.**
3. **User Logout (POST): Users can log out of the application.**
4. **List Movies (GET): Users can view the list of available movies.**
5. **View Movie Details (GET): Users can see detailed information about a particular movie.**
6. **Add Movie to Favorites (POST): Users can add movies to their favorites list for easy access.**
7. **Remove Movie from Favorites (DELETE): Users can remove movies from their favorites list.**
8. **Choose Booking Details: Users can select movie, date, and time for booking.**
9. **Book Movies (POST): Users can book movies by providing necessary booking details.**

4. Architecture Overview

The application follows a client-server architecture:

- **Client Side: Implemented using a web or mobile application.**
- **Server Side: Implemented using a RESTful API server.**

4.1 Frontend Architecture

The frontend architecture includes components for user registration, login, movie browsing, movie details, favorites management, and booking functionalities.

4.2 Backend Architecture

The backend architecture comprises the following components:

- **Controller Layer: Handles incoming requests and delegates them to appropriate service classes.**
- **Service Layer: Contains business logic for processing requests and interacts with the data access layer.**
- **Data Access Layer: Communicates with the database to perform CRUD operations on movie and booking data.**

5. Database Schema

The database schema consists of tables to store information about users, movies, favorites, booking details, and booked movies.

5.1 User Table

- **Table Name: user**
- **Columns:**
  - **id (Primary Key, Auto-generated): Unique identifier for each user.**
  - **email: Email address of the user.**
  - **password: Password of the user.**
  - **role: Role of the user (e.g., admin, user).**
  - **fullname: Full name of the user.**

5.2 MyMovies Table

- **Table Name: mymovies**
- **Columns:**
  - **id (Primary Key): Unique identifier for each movie.**
  - **imageUrl: URL of the movie image.**
  - **name: Name of the movie.**
  - **director: Director of the movie.**
  - **writer: Writer of the movie.**

5.3 Movies Table

- **Table Name: movies**
- **Columns:**
  - **id (Primary Key, Auto-generated): Unique identifier for each movie.**
  - **imageUrl: URL of the movie image.**
  - **name: Name of the movie.**
  - **director: Director of the movie.**
  - **writer: Writer of the movie.**
  - **cast: List of cast members (element collection).**
  - **description: Description of the movie.**
  - **genre: Genre of the movie.**
  - **country: Country of origin of the movie.**
  - **releaseDate: Release date of the movie.**
  - **rating: Rating of the movie.**

5.4 BookingDetails Table

- **Table Name: bookingdetails**
- **Columns:**
  - **id (Primary Key, Auto-generated): Unique identifier for each booking detail.**
  - **capacity: Capacity of the cinema.**
  - **location: Location of the cinema.**
  - **cinema: Name of the cinema.**
  - **date: Date of the booking.**
  - **showTime: Time of the movie show.**
  - **kindOfTicket: Type of ticket booked.**

5.5 BookedMovies Table

- **Table Name: bookedmovies**
- **Columns:**
  - **id (Primary Key, Auto-generated): Unique identifier for each booked movie.**
  - **cinema: Name of the cinema where the movie is booked.**
  - **date: Date of the booking.**
  - **showTime: Time of the movie show.**
  - **ticketType: Type of ticket booked.**
  - **num: Number of tickets booked.**
  - **title: Title of the booked movie.**
  - **movieID: ID of the booked movie.**

6. Technologies Used

- **Backend Framework: Springboot**
- **Database: MySQL**

Authentication:

1- User Authentication Process:

- **Describe the process of how users are authenticated within the system.**
- **Mention the use of form-based authentication where users provide their credentials (username and password) through a login form.**
- **Explain how the authentication process is initiated when a user attempts to access a secured resource or endpoint.**

2- Custom Success Handler:

- **Explain the purpose of the custom success handler (CustomSuccessHandler) used in the authentication configuration.**

- **Mention that the success handler is responsible for redirecting authenticated users to the appropriate page based on their role (e.g., admin-page or user-page).**

3- User Details Service:

- **Highlight the usage of a custom user details service (CustomUserDetailsService) for loading user-specific data during authentication.**
- **Mention that the user details service retrieves user details (e.g., username, password, authorities) from the database.**

4- Password Encoding:

- **Explain the importance of password encoding for securely storing user passwords.**
- **Mention the use of the BCryptPasswordEncoder for hashing and verifying passwords.**

5- Authorization Rules:

- **Provide an overview of the authorization rules configured in the system.**
- **Explain how access to different resources and endpoints is restricted based on user roles (e.g., ADMIN or USER).**
- **Mention that certain endpoints are accessible only to users with the ADMIN role, while others are open to all users or authenticated users.**

5- Logout Configuration:

- **Describe the configuration for handling user logout.**
- **Mention the invalidation of the HTTP session, clearing of authentication, and redirection to the login page after logout.**

7. Conclusion

This software design document provides a comprehensive overview of the movie booking application's design, including its functionality, architecture, database schema, and technologies used. It serves as a guideline for development and maintenance activities throughout the project lifecycle.