Movie Ticket Booking System

1. Introduction

Team Details

The following table includes the team member details.

Name	Roll No	Email
Sathvik	IMT2023001	SVN.Sathvik@iiitb.ac.in
Sudhir	IMT2023546	Kh.Sudhir@iiitb.ac.in
Jitin	IMT2023057	Jitin.Karumudi@iiitb.ac.in
Likith	IMT2023573	Likith.Kasam@iiitb.ac.in
Sai Ganesh	IMT2023525	Ganesh.upadrasta@iiitb.ac.in
Kapil Aditya	IMT2023052	Karrikapil.aditya@iiitb.ac.in

Project Overview

The Movie Ticket Booking System is designed to provide a convenient platform for users to browse available movies, select showtime, and book tickets. The system allows users to view available seats and choose according to their preference. It will be accessible through web interface and will include user authentication, real-time seat availability updates, and ticket confirmation through email.

Scope

The scope of this project includes the following functionalities:
1. User registration and login.
2. Browsing available movies and showtimes.
3. Real-time seat selection and availability check.
4. Ticket booking .
5. Ticket confirmation and email notification.
6.Ticket Cancellation.
7.Adding theater and managing it options available for admins

The project does not include:
1. Refunds.
2. Integration with third-party payment gateways beyond basic implementation.

2. Objectives

The main objectives of this project are:
1. To provide a user-friendly interface for booking movie tickets.
2. To ensure real-time seat availability.
3. To implement secure user authentication.
4. To generate and send booking confirmations through email.
5. To support multiple movie locations and showtimes.

3. System Overview

Technical Specifications

Technology Stack:
1. Frontend: React
2. Backend: C++ and Node.js
3. Database: MySQL or MongoDB
4. Middleware: Java (to handle communication between the frontend and backend)

Input/Output Requirements

Input:
- User inputs include city selection, Théâtre selection/movie selection and date selection.

We will also implement the admin actions for adding, deleting movies.
Output:
- System outputs include ticket confirmation and booking details.

4. Functional Requirements

Detailed Features

The key features include:
1. Movie browsing: Users can view available movies and showtimes.
2. Seat selection: Users can choose their preferred seats from real-time availability data.
3. User authentication: Users must register and log in before booking tickets.
4. Email notification: After booking, users will receive a ticket confirmation via email.

Use Cases

Use Case 1: Movie Selection
The user browses the available movies and selects one to view showtimes.
Use Case 2: Seat Booking
The user selects seats based on availability and proceeds to payment.

Use case 3: Admin features
1. Add new theaters and cinema locations.
2. Manage movie listings by adding, updating, or removing movies.

3. Schedule showtimes for each movie.
4. Monitor booking activity and generate reports.

5. Non-functional Requirements

Performance Requirements

The system is expected to handle high traffic during peak hours, ensuring a response time of less than 2 seconds for seat selection and booking confirmation.

Usability

The system will have an intuitive and responsive interface to provide a seamless booking experience for users.

Security Requirements

Access to the system will require user authentication.

6. Development Setup

To set up the development environment, follow these steps:
1. Install the required versions of C++, Node JS, Java, React and MySQL/Mongo DB.
2. Clone the repository using the command:
```
git clone https://github.com/example/movie-ticket-booking.git
```
3. Install dependencies and check the branches using:
```
git checkout main
```
4. Set up the backend server using C++ and connect it to the MySQL/MongoDB database.
5. Set up the frontend server using React and ensure communication with the backend.

7. Workflow

Backend

The backend will handle the core logic of the system, including seat selection and booking confirmation. Classes like Movie, Show, Seat, and Booking will be used to implement this functionality.
Example class structure:
```
class Movie {
  string name;
  vector<Show> shows;
};

class Show {
  string time;
  vector<Seat> availableSeats;
};

class Seat {
  int seatNumber;
  bool isAvailable;
};

class Booking {
  string user;
  Movie movie;
  Show show;
  vector<Seat> seatsBooked;
};
```

The admin interface will have additional functionality for theater and show management. Classes such as Admin, Theater, and ShowSchedule will be added to support these features. Example admin-related class structure:

```
class Admin {
  string name;
  void addTheater(Theater theater);
  void scheduleShow(Theater theater, Movie movie, string showTime);
};

class Theater {
  string name;
  string location;
  vector<ShowSchedule> schedules;
};

class ShowSchedule {
  Movie movie;
  string showTime;
  Theater theater;
};
```

Frontend

The frontend will be designed using Reactjs to allow users to interact with the backend system for browsing movies, selecting seats, and making payments.

Middleware

The middleware (Java) will handle communication between the frontend and backend systems, ensuring secure data transfer and real-time updates.

8. Important Files & Folders

Key files in the project include:
1. movie.cpp: Contains the logic for movie management.
2. seat.cpp: Handles seat availability and booking logic.
3. booking.cpp: Processes user bookings and payments.
4. frontend/: Contains React code for the user interface.
5. middleware/: Java code handling communication between frontend and backend.

9. Testing & Logging

We will implement unit tests for the core classes (Movie, Show, Seat, Booking) and use logging mechanisms to record booking failures and payment errors.

10. Conclusion

This project aims to develop a user-friendly movie ticket booking system with real-time seat availability and efficient booking confirmation. We expect to deliver a robust system that simplifies the movie ticket booking process for users.