# Abstract

This project is aimed at developing an online movie ticket booking application that is essential for modern cinemas and moviegoers. The Movie Ticket Booking Application is a web-based platform that allows users to browse, select, and book movie tickets conveniently. This system can be accessed by users to view movie listings, choose showtimes, and reserve seats. The application ensures secure user authentication using email-based login, with features of JWT-based token security, movie search, seat selection, booking confirmation, and payment integration. The system is designed to streamline the ticket booking process, making it efficient for both users and cinema administrators.

# Introduction

In the digital era, booking movie tickets online has become a standard expectation for cinemagoers, offering convenience and efficiency. Traditional methods of purchasing tickets, such as standing in long queues or relying on physical box officers, are being replaced by web-based systems that allow users to browse movie listings, select seats, and book tickets from the comfort of their homes.

This movie ticket booking application provides several key advantages:

* Online Accessibility: Users can browse available movies, select showtimes, and book tickets anytime, from any device.
* Secure Authentication: Email-based login and JWT token security ensure that user data is protected.
* Seat Selection: Users can view available seats and make their selection in real-time.
* Booking Confirmation: Instant booking confirmations are provided, with the option to receive email notifications.
* User-Friendly Interface: The application is designed to be easy to use, making the booking process smooth and straightforward.

Cinema administrators can easily:

* Manage Movie Listings: add, update, or remove movies and showtimes.
* Monitor Bookings: View and manage all bookings, ensuring that the system runs efficiently.
* Ensure Security: Utilize built-in security features to protect user data and prevent unauthorized access.

# Scope

## System:

* System to be able to display current selection of movies, allowing users (customers) to book ticket or tickets for a movie.
* System to be able to continue updating the movie selection, seat selection. Making information available.

## User:

* User to be able to see current movie schedules, available seats and cinema plan of seats in relations to each other and the screen.
* User will be able to perform add/delete/update operation on their reservation.
* User to be able to retrieve their booking ticket and view ticket information. (Seat number, movie time etc..)

# Technologies Used

* JSX (JavaScript XML): Used for structuring the content and components of pages on the frontend.
* CSS (Cascading Style Sheets): For styling the application.
* Bootstrap: For basic stylings ( form inputs and etc.. )
* ReactJS: A JavaScript library used to build the frontend, offering a component-based architecture for developing user interfaces.
* Spring Boot: A Java-based framework used for building the backend, simplifying the development of RESTful web services and integrating with various tools.
* MySQL**:** A relational database management system used for storing and managing data related to users, movies, bookings.

# Tools Used

* IntelliJ IDEA: An integrated development environment (IDE) for developing.
* Postman**:** A tool for testing and interacting with RESTful APIs during development.
* Maven: A build automation tool for managing dependencies and the build lifecycle in the Spring Boot project.

# Use Case Diagram

A diagram of a movie ticketing system

Description automatically generated

Both the User and the Admin have specific roles and access to different functionalities:

User:

* Register: Users can create a new account by registering with their email and their desired password.
* Login: Registered users can log in to access the system using their credentials.
* Book Movie Ticket: Users can browse available movies and book tickets for their desired showtimes.
* Search Movie: Users can search for movies based on movie title.
* Making Payment: After selecting a movie and booking tickets, users can proceed to make payments through the payment system.
* View Movie List: Users can view the list of all available movies and their showtimes.
* Modify Booking: Users have the option to modify their existing bookings, such as changing the number of tickets or the selected seats, based on availability.
* Cancel Booking: Users can cancel their bookings.

Admin:

* Adding Movie: Admins can add new movies details to the system.
* Delete Movie: Admins have the authority to remove movies from the list if they are no longer available for booking.
* Update Movie: Admins can update movie details.
* View Booking List: Admins can view the list of all bookings made by users.

# Functional Requirements

It defines what the system should do, specific behaviors, functionalities, or tasks that the system must perform to meet the needs of the users and to the business.

* **User Registration and Login:**
  + The system must allow new users to register using their email and other personal details.
  + The system must provide a secure login feature for registered users using their email and password.
* **Movie Browsing and Search:**
  + Users must be able to browse available movies and view details of that movie.
  + The system must provide a search functionality allowing users to search for movies based on movie title.
* **Ticket Booking:**
  + Users must be able to select a movie, choose a showtime, and book tickets.
  + The system must provide a seat selection feature allowing users to pick their preferred seats based on availability.
  + Users must be able to review their booking details before confirming.
* **Booking Management:**
  + Users must be able to view their booking history and upcoming bookings.
  + Users must be able to modify their bookings, such as changing seats or the number of tickets, based on availability.
  + Users must have the option to cancel bookings, subject to the cancellation policy.
* **Admin Movie Management:**
  + Admins must be able to add new movies to the system, including all relevant details.
  + Admins must be able to update movie details, such as showtimes or seat availability.
  + Admins must be able to delete movies that are no longer available for booking.
* **Booking List for Admin:**
  + Admins must be able to view a comprehensive list of all bookings made by users.
  + The system must provide filtering and sorting options for admins to manage bookings efficiently.

# Non-Functional Requirements

It defines how the system should behave rather than what it should do.

* **Security:**
  + The system must ensure secure authentication and authorization, using **Spring Security 6** with JWT tokens for user sessions.
  + All user data, especially payment information, must be encrypted and securely stored.
* **Compatibility:**
  + The system must be compatible with various web browsers and devices, ensuring accessibility for all users.
* **Usability:**
  + The user interface must be easy to navigate.
  + The system must provide clear instructions and feedback at each step of the booking process.

# Methodology

* Waterfall method
  + Short time span in creating the project.
  + Focusing more and narrow down the goals, task, and features in achieving project completion.
* Planning
  + Deciding which features to prioritize and to be implemented.

# UML (High Level)

A diagram of a user

Description automatically generated

* Role to User:
  + User is associated with one Role, but Role can be assigned to many Users.
* User to Movie:
  + User can interact with one or more movies, and movie can be interacted with many users.
* Movie to Booking:
  + One Booking is associated with a movie, movie can have many Bookings.
* User To Booking:
  + User can have many Bookings; Booking can only be made by one User.

# UML (Low Level)

A screenshot of a computer program

Description automatically generated

# Sequence DiagramA diagram of a program Description automatically generated

# Backend Routes

## User

|  |  |  |
| --- | --- | --- |
| * GET | /users | Get all users ( Admin Only ) |
| * GET | /users/{id} | Get user by id |
| * POST | /users | Post a user |
| * PUT | /users/{id} | Update a user |
| * DELETE | /users/{id} | Delete a user by id |

## Movie

|  |  |  |
| --- | --- | --- |
| * GET | /movies | Get all movies |
| * GET | /movies/{id} | Get movie by id |
| * POST | /movies | Post a movie ( Admin Only ) |
| * PUT | /movies/{id} | Update a movie ( Admin Only ) |
| * DELETE | /movies/{id} | Delete a movie by id ( Admin Only ) |

## Booking

|  |  |  |
| --- | --- | --- |
| * GET | /bookings | Get all bookings ( Admin Only ) |
| * GET | /bookings/user/{userId} | Get all bookings from a specific user |
| * GET | /bookings/{id} | Get a booking by id |
| * POST | /bookings | Create a new booking |
| * DELETE | /bookings/{id} | Cancels a booking |

## Authentication

|  |  |  |
| --- | --- | --- |
| * POST | /auth/login | User login and receive JWT Token |
| * POST | /auth/register | User registration |
| * POST | /auth/logout | User logout |

# Frontend Routes

## Public Routes

|  |  |
| --- | --- |
| * / | Home page (Displays the movie list and movie information) |
| * /login | Login Page |
| * /register | Registration Page |
| * /movies/{id} | /users/{id} |

## User Routes

|  |  |
| --- | --- |
| * /profile | User profile page(Viewing and Editing user information) - *OPTIONAL* |
| * /bookings | User’s bookings page(List of bookings) |
| * /bookings/{movieId} | Booking Page(Selection of seats) |

## Admin Routes

|  |  |
| --- | --- |
| * /admin/dashboard | Admin management page |
| * /admin/bookings | Admin booking management page |

# Routing Arrangement

## Home (*PUBLIC*)

* + - ***/***  (Home, Displays all movies)
    - */login* (Login)
    - */register* (Register)
    - */movies/{id}* (Movie Detail)

## User (*AUTHENTICATED*)

* + - */profile* (User Profile)
    - */bookings* (Booking detail)
    - */booking/{movieId}* (New Booking)

## Admin (*ADMIN AUTHENTICATED*)

* + - */admin/dashboard* (Update, Create, Delete movie)
    - */bookings/{id}* (Edit movie)

# Planning

A screen shot of a computer

Description automatically generated