# High Level Design

Movie Ticket Booking System

| | |
|---|---|
| Written By | Gaurav Anil Nalawade |
| Date | 07–July-2024 |

## Contents

## Abstract

In today's fast-paced world, the movie-going experience has been revolutionized by our innovative Movie Ticket Booking System. Combining cutting-edge technology with a user-friendly interface, our platform simplifies ticket booking with seamless navigation, interactive seat maps, and personalized recommendations. Real-time updates keep users informed with live showtimes, dynamic seat availability, and instant booking confirmations. The system also fosters social interactions with features like friends and family bookings, shared wish lists, and group discounts. Prioritizing security and privacy, we ensure encrypted transactions and user-controlled privacy settings, offering a safe and personalized movie booking experience.

# 1. Introduction

## 1.1. What is High-Level design document?

The goal of a High-Level Design (HLD) document is to provide an overall architecture of the Movie Ticket Booking System, outlining the system's structure and components. An HLD document gives a bird's-eye view of the system, focusing on the architecture, modules, interfaces, and data flow between various components. It serves as a blueprint for the system, offering a framework for understanding how the system meets business and technical requirements.

## 1.2. Scope

High-Level Design (HLD) is a system-level design process that provides an abstract overview of the system architecture and its components. This process includes defining the architecture, modules, interfaces, and data flow. It helps in understanding the system's structure and its interaction with external systems. The HLD document includes:

- System Architecture: Defines the overall structure of the system, including the main components and their interactions.
- Modules and Components: Describes the major modules of the system and their responsibilities. Interfaces: Specifies the interfaces between modules and external systems, detailing how data flows between them.
- Data Flow: Illustrates the flow of data through the system, identifying key data processing points and storage components.
- Technology Stack: Outlines the technologies and platforms to be used for implementing the system.
- Security Considerations: Provides an overview of the security measures to protect user data and ensure secure transactions.
- Performance Requirements: Highlights the performance criteria that the system must meet, including response times and scalability.

## 2. General Description

### 2.1 Project Perspective

Our movie ticket booking system is a comprehensive platform that allows users to browse and book tickets for their favorite films. It offers a user-friendly interface with real-time updates on showtimes and seat availability. The system is equipped with collaborative features that enable users to plan movie outings with friends and family effortlessly.

### 2.2 Problem Statement

The Movie Ticket Booking System aims to enhance the movie-booking  experience by providing a seamless, user-friendly platform for discovering, booking, and enjoying films. Despite advancements in digital technology, the current market lacks an integrated system that combines ease of use, real-time updates, and robust security features. Existing solutions often struggle with fragmented user interfaces, inefficient seat selection processes, and inadequate real-time information updates, leading to customer frustration and missed opportunities for seamless bookings.

### 2.3 Proposed Solution

The Movie Ticket Booking System aims to modernize and streamline the process of booking movie tickets, ensuring a seamless experience for users. This system needs to provide an intuitive user interface for exploring movie listings, viewing showtimes, and selecting seats. Users should be able to receive personalized recommendations based on their viewing history and access real-time updates on showtimes and seat availability.

The front-end must be developed using Angular to offer a responsive and interactive user experience. The back-end should utilize Spring Boot and MySQL to manage data and handle business logic securely and efficiently. Key features include secure user authentication, dynamic seat selection, and the ability to handle high traffic volumes.

### 2.4 Furthermore Improvement

Additionally, the system should support group bookings, shared wishlists, and offer group discounts for corporate and school outings. To ensure reliability, the deployment will be on AWS, using EC2 instances for the application, RDS for the MySQL database, and S3 for storing the Angular project's production files. Security, performance, and scalability are paramount, requiring comprehensive testing using JUnit and Mockito to validate functionality and performance. This system aims to enhance user satisfaction by offering a convenient, efficient, and secure movie ticket booking experience.

## 2.5 Technical Requirement

- Technical Requirements

The Movie Ticket Booking System requires a robust and scalable architecture to ensure seamless user experiences and efficient operations. The front-end will be developed using Angular, featuring components, services, and HTTP clients to facilitate dynamic user interactions and communication with the back-end via RESTful APIs.

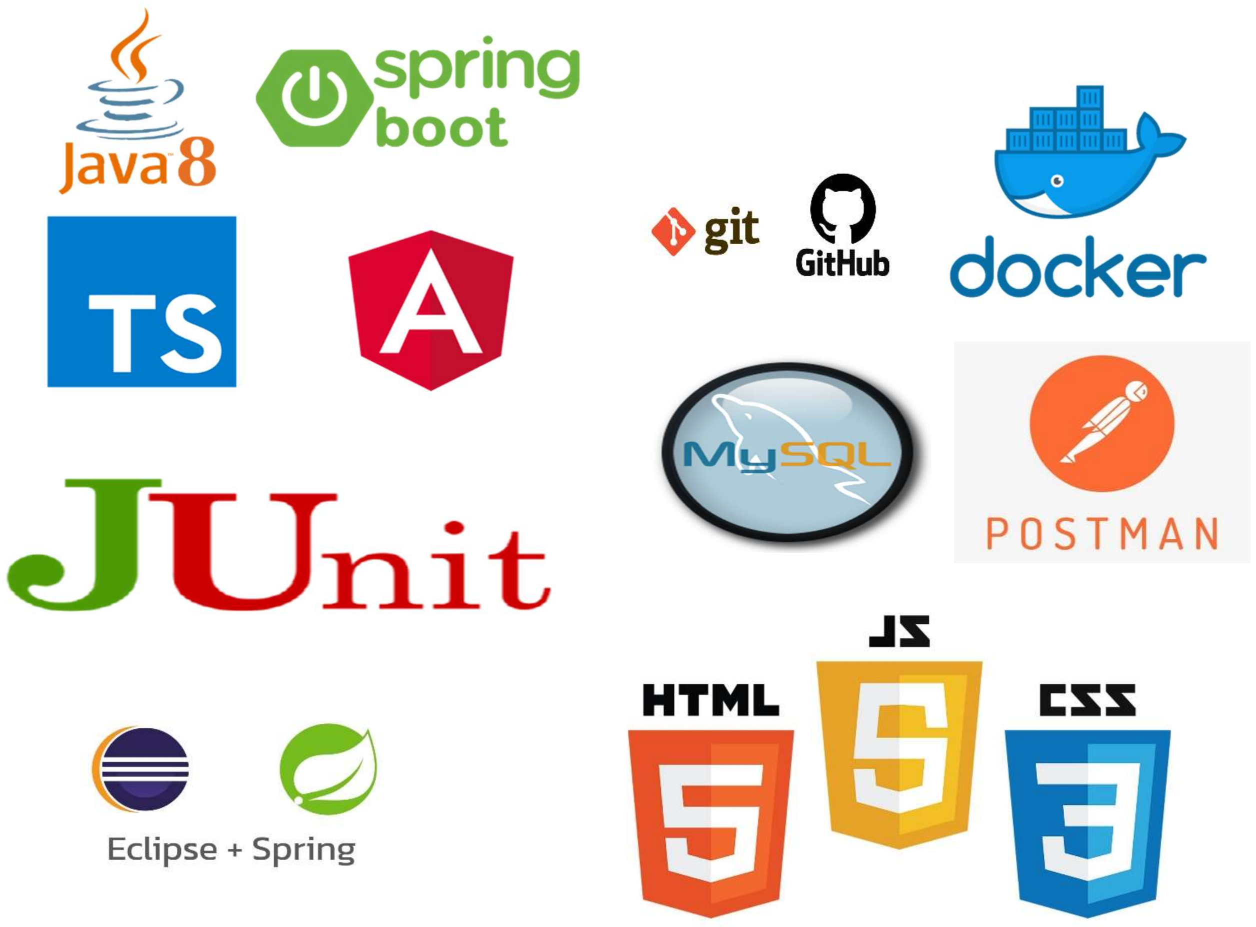
The back-end will utilize Spring Boot to manage business logic and interactions with a MySQL database, leveraging Spring Data JPA for data access and manipulation. The system must implement Spring Security with JWT authentication to secure user data and API endpoints.

Deployment will be on AWS, utilizing:

- EC2 instances: Hosting the Spring Boot application.
- RDS : Managing the MySQL database.
- S3: Storing and serving the Angular application's production files.

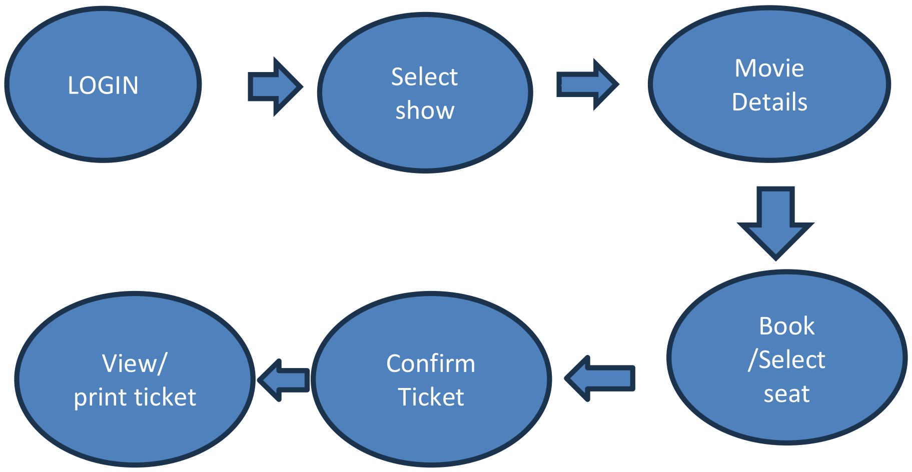
Comprehensive testing is required, using JUnit and Mockito for unit and integration tests, ensuring all functionalities meet performance and security standards. The system must also support scalability to handle high traffic volumes and ensure high availability, with load balancing and backup strategies in place.

## 2.6 Tools Used:

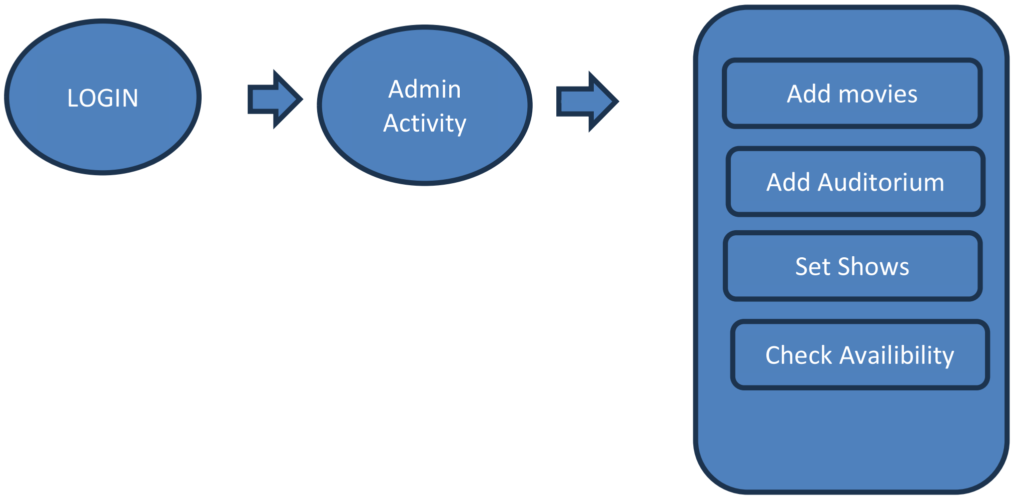


## 3. Design Details

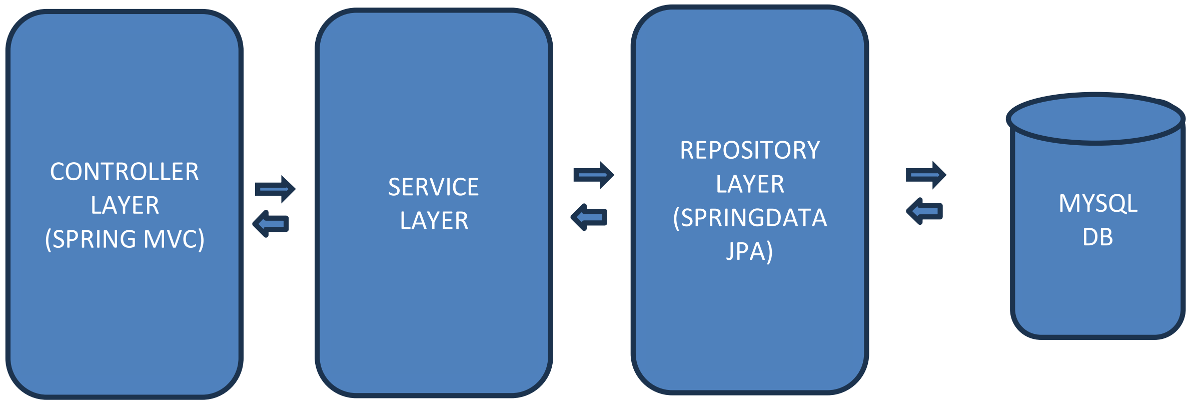
### 3.1. Process Flow (To Book Ticket)



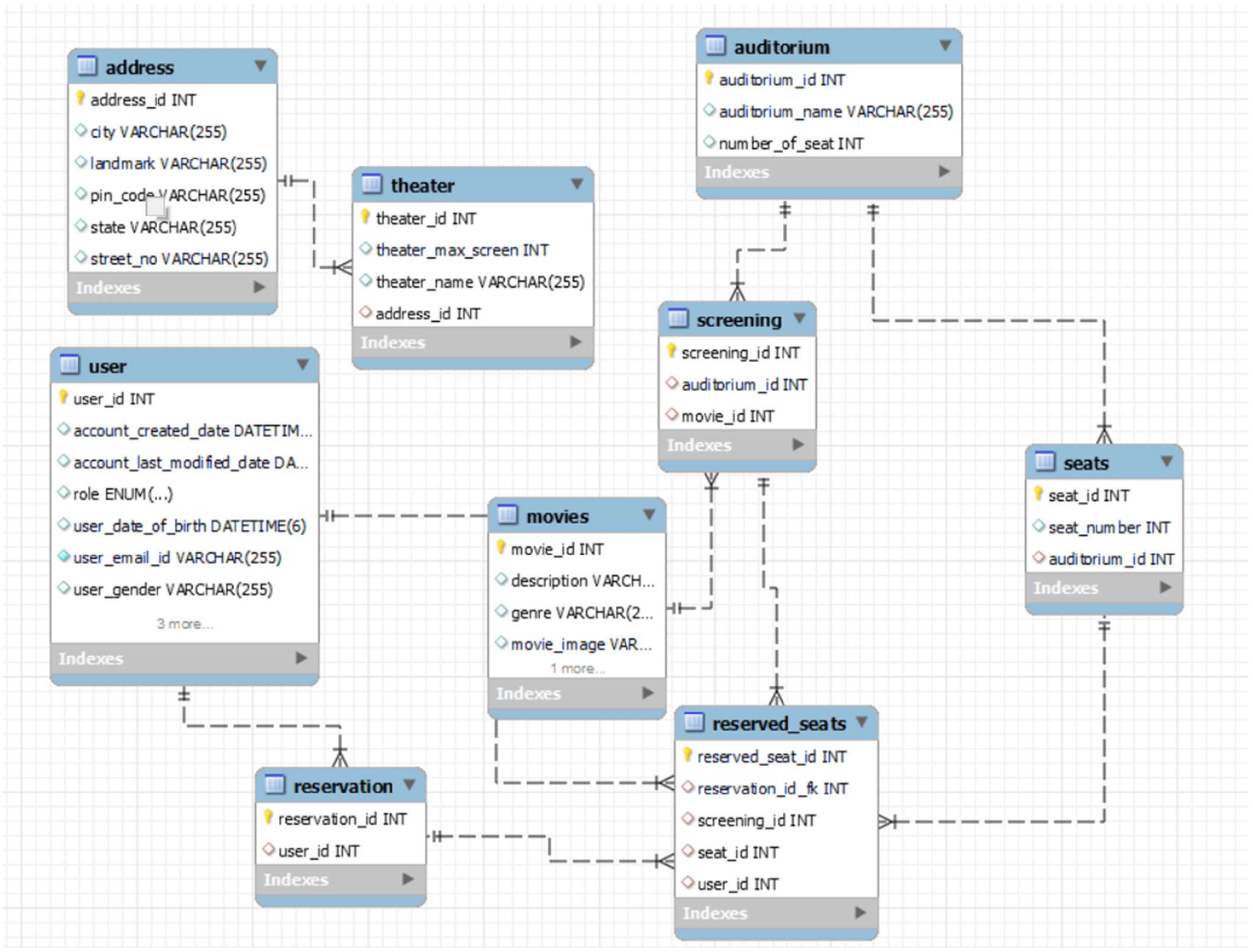
### 3.2. Admin Panel



## 3.3. Back-End Layer (SpringBoot and Mysql) Design


CONTROLLER LAYER (SPRING MVC)
SERVICE LAYER
REPOSITORY LAYER (SPRINGDATA JPA)
MYSQL DB

## 3.4 Database Design



## 3.5 JUnit & Mockito Test Cases

| Test Case Description | Pre-Requisite | Expected Result |
|---|---|---|
| Testing Api's for Admin Services | | |
| Test Functionality of addMovie() Api: *public void testAddMovie()* | Movie Object For Testing And Mock Object of Respective Repository | Return true if Movie Object Saved successfully else false |
| Test Functionality of *getMovieWithId* () Api: *public void testGetMovieWithId ()* | Movie Object and Id For Testing And Mock Object of Respective Repository | Test true when data with respective id exist |
| Test Functionality of addAuditorium () Api: *public void testAddAuditorium ()* | AuditoriumObj ect For Testing And Mock Object of Respective Repository 1. | Return true if Auditorium Object Saved successfully else false |
| Test Functionality of getAllAuditorium () Api: *public void testGetAllAuditorium ()* | Auditorium Object For Testing And Mock Object of Respective Repository | Return true if Auditorium list is fetched successfully else false |

## 4. Conclusion

The development of the Movie Ticket Booking System leverages a comprehensive suite of modern tools and technologies to create a robust, scalable, and user-friendly application. Utilizing Angular for the front-end, we ensure an intuitive and dynamic user experience with responsive layouts and seamless interactions. The back-end, powered by Spring Boot and MySQL, provides secure and efficient data management and business logic processing.

Deployment on AWS using EC2, RDS, and S3 guarantees high availability, scalability, and reliable performance. Security is enhanced through Spring Security with JWT authentication, ensuring that user data and interactions are protected. The use of Docker and Docker Compose ensures consistent and reproducible environments, facilitating smoother development and deployment processes.

Comprehensive testing with JUnit and Mockito ensures that all components function correctly and meet performance standards, while Git and GitHub support effective version control and collaboration.