# Online Movie Booking Platform

## **Introduction**

This design document outlines the strategy to build online movie ticket booking platform that caters to both B2B (theatre partners) and B2C ( end customer ) clients.

## **Business Drivers and Goals**

This project aims to accomplish following

* Enable theatre partners to onboard their theatres over this platform and get access to a bigger customer while going digital.
* Enable end customers to browse the platform to get access to movies across different cities, languages, and genres, as well as bool tickets in advance with a seamless experience

## **Proposed Architecture**

A screenshot of a computer

Description automatically generated

## **Proposed Technology Stacks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No** | **Technology Stack** | **Software** | **Major Version** | **Comments** |
| 1 | Backend Language | Java | 21(LTS) | Java 21 is the latest Long-Term Support release |
| 2 | Front End (UI) | Angular | 17 | Latest Long-Term Support release |
| 3 | Cloud Provider | GCP/AWS/Azure |  | Any cloud provider can be selected based on organization preference and standard |
| 4 | RDBMS | Cloud SQL – PostgreSQL | PostgreSQL 16 | Release on 7th June’ 2024 |
| 5 | Document DB | MongoDB | 6.0 | Latest LTS released on July 2022 |
| 6 | Spring | Spring Framework | 6 | Supported by Spring Boot 3 |
| 7 | Spring Boot | Spring Boot | 3 |  |
| 8 | Reactive Programming | Spring Web Flux | Spring Framework 6 |  |
| 9 | Reactive Repository | R2DBC | 1 |  |
| 10 | Identity Provider | Azure AD |  |  |
| 11 | API Gateway | Apigee |  |  |

## **Platform Architecture Overview**

The platform will be built using microservice architecture, enabling scalability flexibility, and ease of maintenance.

## 5.1 Microservice Components

### 5.1.1 Theatre Partner Service

Manage theatre onboarding and details . Allow theatre partners to manage screenings, seat availability, pricing, etc.

### Theatre APIs

### 5.1.1.1.1 Onboard Theatre

* Method: POST /theatre-service/v1/theatres
* Description: Onboards a new theatre to the platform.
* Request Body:

A computer screen with green text

Description automatically generated

* Response: 201 Created
* Response Body:

A computer screen with green text

Description automatically generated

### 5.1.1.1.2 Get Theatre by ID

* Method: GET /theatre-service/v1/theatres/{id}
* Description: Retrieves details of a specific theatre by its ID.
* Response: 200 OK
* Response Body:

A computer screen with green text

Description automatically generated

### 5.1.1.1.3 Get All Theatres

* Method: GET /theatre-service/v1/theatres
* Description: Retrieves a list of all theatres onboarded onto the platform.
* Response: 200 OK
* Response Body:



### 5.1.1.1.4 Update Theatre

* Method: PUT /theatre-service/v1/theatres/{id}
* Description: Updates the details of an existing theatre.
* Request Body: Similar to POST.
* Response: 200 OK
* Response Body: Updated theatre details.

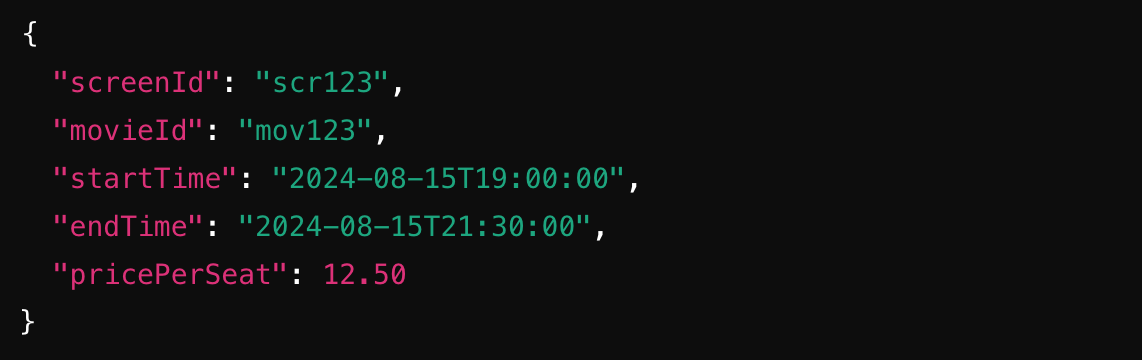
### 5.1.1.1.5 Delete Theatre

* Method: DELETE /theatre-service/v1/theatres/{id}
* Description: Deletes a theatre from the platform.
* Response: 200 OK
* Response: 204 No Content

### Show APIs

### Schedule Show

* Method: POST /theatre-service/v1/shows
* Description: Schedules a new show on a specific screen.
* Request Body:



* Response: 201 Created
* Response Body: Details of the schedule show with showId.

### Get Show by ID

* Method: GET /theatre-service/v1/shows/{id}
* Description: Retrieves details of a specific show by its ID.
* Response: 200 OK
* Response Body: Show details with show ID.

### Update Show

* Method: PUT /theatre-service/v1/shows/{id}
* Description: Updates the details of an existing show.
* Response: 200 OK
* Response Body: Updated show details.

### Delete Show

* Method: DELETE /theatre-service/v1/shows/{id}
* Description: Deletes a show from the platform.
* Similar to other DELETE API.

## **Non-Functional Requirements**

This section outlines the non-functional requirements (NFRs) that the platform adhere to. These requirements define the desired qualities and characteristic of the cloud environment, ensuring it meets the business report.

* **Target Uptime:** Achieve a minimum uptimeof 99.95% for Movie Ticket Booking platform.
* **Disaster Recovery:** Implement a robust DR strategy utilizing GCP’s built in redundancy features to minimize downtime in case of outage.
* **Automatic Failover:** Configure automatic failover mechanism to ensure application seamlessly switch to redundant resource in case of failure.
* **Scalability:** The cloud environment should be able to scale resource automatically.
* **Load Balancing:** Implement load balancing techniquesto distribute incoming trafficefficiently across application instances.
* **Automated Deployment:** Leverage DevOps practice and automation tool for infrastructure and application deployments to streamline the release process and minimize manual intervention.
* **Monitoring and Logging:** Implement comprehensive Google monitoring and logging solutions to track the application health, resource utilization and identity potential issue proactively.

## **MVP1 Project Planning**

