Github: <https://github.com/ponnam86/BookMyShowApp>

**Technologies recommended**

* Language – **Java17**
* Frameworks- **SpringBoot 3.x & MicroServices**
* AI- Suggest – **Github copilot & Github Actions**
* Database - **MySQL**
* Integration and Data technologies- **Spring REST, Lombok, UnitTest, OpenAPI**
* Cloud technologies- **Spring Cloud**
* Preferred editor to build and present solution - **Intellij**

# **Functional features to implement:**

Implemented below endpoints using which we can operate on different services.

**Browse Movies**: Users can view a list of available movies.

**Search and Filter**: Users can search for movies based on title, date, location, genre and movie ID.

**Movie Details**: Users can view detailed information about a specific movie, including title, director, description, genre, date, location, total seats, available seats, and price of ticket.

**Booking Tickets**: Users can book tickets for a movie by specifying the movie ID, quantity of tickets, and total price.

**Booking History**: Users can view their booking history, which includes details of movies they have booked, such as title, director, description, genre, date, location, booked tickets, and total price.

**Add/Delete/Update Movies**: Admin users can add new movies, delete existing movies, and update movie information.

**Technologies**: Java, Spring Boot, Spring Data JPA, Postman, MySQL Database and OpenAPI

**API Endpoints**

GET /movie: Get a list of movies.

GET /movie/{id}: Get detailed information about a specific movie.

GET /movie?genre=Drama: Get a list of movies based on genre.

GET /movie?date=2024-01-22: Get a list of movies based on date.

GET /movie?movieName="avatar": Get a list of movies based on title.

GET /movie?location="pvr": Get a list of movies based on location.

GET /movie/order-history: Get booking history.

POST /movie: Add a new movie.

POST /movie/book-seat?movieId=91&&showDate=2024-01-22&&showTime="06:00 pm": Create a new booking.

DELETE /movie/{movieId}: Delete a movie.

PUT /movie/{movieId}: Update movie information.

# **Non-functional requirements:**

1. Transactional Scenarios: If we want to integrate with any third party payment gateways, we should be properly configured with **transaction management using different transaction propagation and isolation** techniques. << we can extend my application, if required>>
2. Scaling and platform availability:

**Horizontal**: add new containers or virtual machines to distribute the load

**Vertical**: increasing the resources (CPU, memory, etc.)

By using Kubernetes/docker, we can orchestrate the application pods with configuration to set some threshold limits. << We can extend My application with docker configuration>>

Used Caching for now in our application. However we can extend to **Redis caching**/**HazleCast** to increase performance

1. **Integration with payment gateways :** From my experience, payment gateways integration has to do with transaction management and maintain lot of PII information. << we can extend my application, if required. I was integrated with master/visa payment gateways>>
2. **How do you monetize platform:**  I configured log4j currently. However we can monitor the application by integrating with **ELK ( by writing logstash configuration) OR Splunk**.

I was using **AppDynamics tool** to distributed tracking . We can also configure Dynaconfig/Grafana

1. protect against OWASP: I just did basic user validation. However we can extend our application for **Basic JWT authentication or OAuth 2** integration by assigning some secret application key.

We can configure **CORS** as well by defining roles and authentication endpoints.

One more way is, white listing Ips. And configuring a API gateway.

1. Compliance: As always ensure PII related stuff, should be encrypton/decryption logic used.

# **Platform provisioning, sizing & Release requirements:**

1. technology choices and decisions through key drivers:

Based on application complexity as usage goes on, **Split the application into various microservices** (Userservice, MovieService, PaymentService,NotificationService)

I prefer **to apply feature flagging**(Ex:Lanuch Darkly) tools to the application which provides flexibility of release on demand.

We can use **NOSQL for SearchService** by storing data as json data type.

1. database, transactions, and data modelling:

As mentioned, I used RDBMS as I did basic draft of spring boot application. However, if we split into different microservices, extend to **NOSQL(for search)**, Transaction Management(for Payment service) and Json data type. We can integrate **with AWS S3 to store images.**

1. hosting solution and sizing:

Can be hosted in docker and sizing(scalling) can be done through Kubernetes orchestration.

We have to create a CI/CD pipeline to this happened. I **prefer Github&Jenkins.**

1. monitoring solution and log analysis:

I will suggest to integrate wit ELK by writing logstash configuration. OR Splunk.

Use distributed tracing mechanisms like Grafana or AppDynamics tool(cisco)

1. KPIs:

This platform is for entertainment enthusiasts, where users can discover, share, and discuss their favourite movies. We should get customer reviews and act on it.

Work towards the competitors and increase traffic hits by providing customer rating is high. ( Probably we should maintain/publish some reports) << Not exactly sure on this>>

1. high-level project plan and estimates breakup:

should come up with HLD having below details.

**Define goals & requirements**

**Which technology should be bring for application**

**Any Searching technology**

**Any caching technology**

**Type of databases**

**How Async communication (kafka/pub-sub..etc)**

**Log management**

**Monitoring and tracking**

**Release management**

Estimates breakup.

This depends on proper Agile implementation. Based on requirements and product owner alignment, we should create different features.

Based on team’s capacity, we should groom and estimate on the PI plan and prepare sprint backlog.

By doing grooming, assign story points and tag to appropriate sprint.

Having daily standups and touch base with stakeholders/product-team .

Use different templates as part of Safe agile like gant charts/ velocity estimates/ HLD/LLD..etc

# **Product management and Stakeholder management:**

Again this comes under agile.

Stakeholders are individuals (or groups) that can either impact the success and execution or are impacted by a product. Everyone who must contribute to or approve the activities required to design, build, and bring the product to market.

I feel there are lot of steps like **Identification and prioritizing stakeholder, Identifying Risks and Opportunities and addressing them, Proper relationship maintenance** till project closure.

I assume, as a good scrum master, we should keep this in mind to soft execution of project road map.