**Problem statement (Version 1.3.3)**

XYZ wants to build an online movie ticket booking platform that caters to both B2B (theatre partners) and B2C (end customers) clients.

Key goals it wants accomplished as part of its solution:

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

**Technologies recommended**

* Language -Java and other add-on languages
* Frameworks- Any
* AI- Suggest
* Database - Any
* Integration and Data technologies- Any
* Cloud technologies- Any
* Preferred editor to build and present solution

*PS:* ***The given exercise will help you get prepared for technical discussion and demonstrate your current understanding on key architectural artifacts. It’s expected that you may not all sections hence can ignore them.***

**Evaluation criteria**

* Code artifacts (**APIs Contract, Design Patterns,** Any one Scenario Implementation)
* Design principles to address functional requirement and non-functional requirement
* DB & Data model
* Platform solutions detailing
* Solution completeness, presentation, and discussion.
* Solution coverage uniqueness and extensibility.

Note: Incomplete solution component would be discussed during discussion round. All sections are not mandatory.

*You can skip solution areas that you are not comfortable by making a note of it.*

1. **Functional features to implement** Good to have - Code Implementation (Read scenario)):

**Anyone of the following read scenarios:** (Only Service Implementation needed/ No UI required)

* Browse theatres currently running the show (movie selected) in the town, including show timing by a chosen date.
* Booking platform offers in selected cities and theatres
  + 50% discount on the third ticket
  + Tickets booked for the afternoon show get a 20% discount

**Anyone of the following write scenarios:** Good to have - Code Implementation (write scenario):

* Book movie tickets by selecting a theatre, timing, and preferred seats for the day
* Theatres can create, update, and delete shows for the day.
* Bulk booking and cancellation
* Theatres can allocate seat inventory and update them for the show

**Solution: Functional Requirements**

**B2B (Theatre Partners)**

**Theatre Management**

Register theatre with details details - name, location, contact information.

Retrieve, update and delete theatre information

Upload and manage movie screenings, show timings, and seat layouts.

Set ticket pricing and discount offers.

Allocate and manage seat inventory.

**Show Management**

Add, update, and delete movie shows.

Update pricing based on demand - add discount or increase prices

**Booking System**

User can book multiple seats for single show

Set preference for seat selection dependence on availability

**Revenue Tracking**

Track real-time ticket sales and revenue analytics.

Export reports for reconciliation and auditing.

Handle bulk bookings and cancellations.

**Customer Support Integration**

Provide support through chat, email, and ticket-based issue resolution.

**B2C (End Customers)**

**User Registration & Profile Management**

Create and manage user profiles.

Save preferences for movie genres, favourite theatres, and languages.

Access booking history

Create, update or delete bookings.

**Movie Catalog and Search**

Search movies by title, genre, language, and location.

Filter by rating, popularity, and availability.

Browse theatres and showtimes in a selected city.

Browse movies based on movie\_title, show\_date, show\_time, theatre, city

**Ticket Booking & Payments**

Select seats and book tickets for a preferred show.

Avail discounts (e.g., 50% discount on the third ticket, 20% discount on afternoon shows).

Integrate multiple payment methods (credit/debit cards, UPI, wallets).

Receive digital tickets and booking confirmations via email/SMS.

**Cancellation & Refunds**

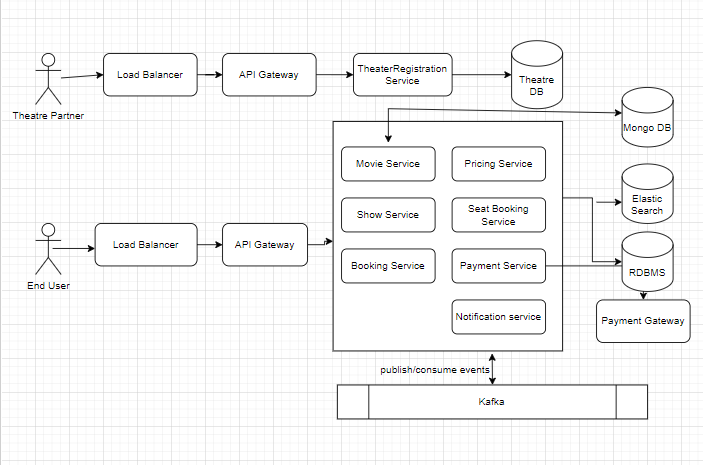
Cancel tickets before a show with automated refund processing.

Modify bookings based on availability and theatre policies.

**AI-based**

AI-powered recommendation engine for delivering personalized offers. Our system can seamlessly integrate to suggest movies and shows tailored to individual user preferences and viewing behavior.

**Technical Architecture -** <https://app.diagrams.net/#G1TAkHGLXPoDnQ1Ak57i3UMxQo7pnPbT2c#%7B%22pageId%22%3A%22QrkNUFReH4stlNR6YxVi%22%7D>



DB and Data Model

Users - user\_id, name, email, phone, password, role

Role (customer, ticket\_agent, admin, theatre\_partner)

Theatres - theatre\_id, name, city, address, email, phone

Movies - movie\_id, title, language, genre, duration, release\_date

Shows - show\_id, theatre\_id, movie\_id, show\_date, show\_time, price

foreign key Theatres(theatre\_id)

foreign key Movies(movie\_id)

Seats - seat\_id, theatre\_id, seat\_number, type\_of\_seat

foreign key Theatres(theatre\_id)

Type\_Of\_Seat (Regular, Gold, Premium)

ShowSeats - show\_seat\_id, show\_id, seat\_is, status, price

Foreign key Shows(show\_id)

Foreign key Seats(seat\_id)

Status (Available, Booked, Reserved)

Bookings - booking\_id, user\_id, show\_id, booking\_time, total\_amount, status);

foreign key Users(user\_id)

foreign key Shows(show\_is)

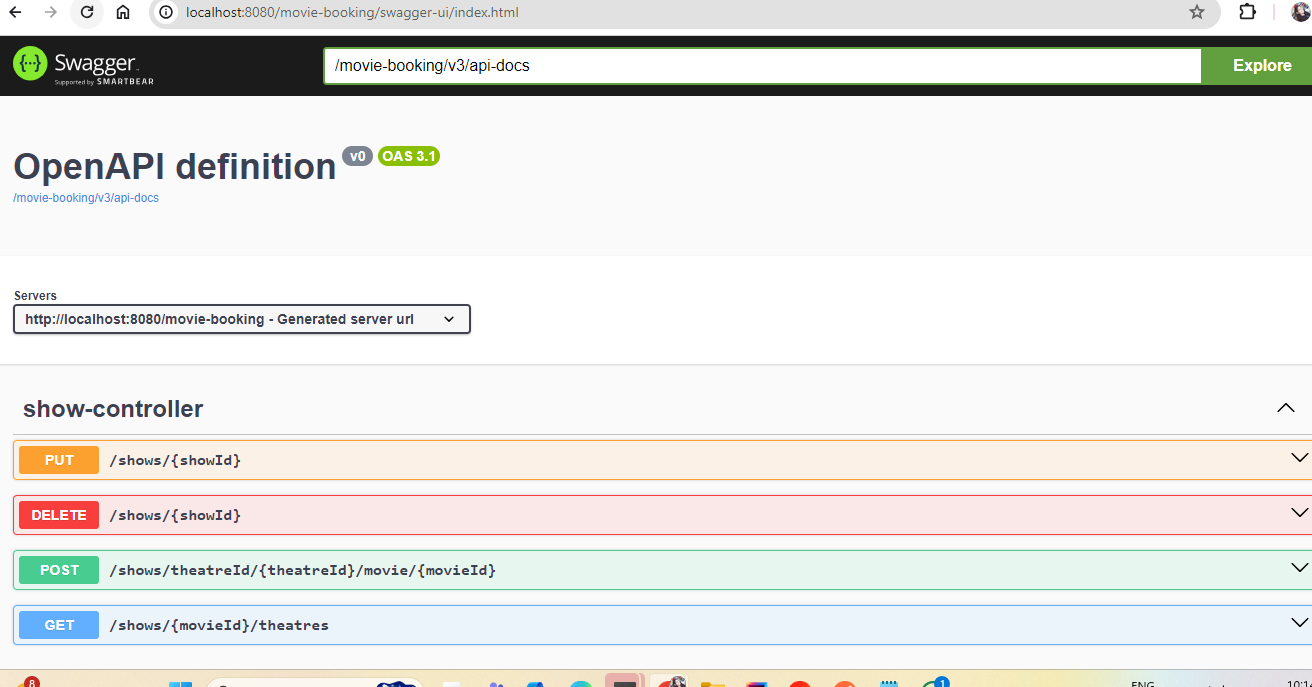
Status (Confirmed, Cancelled)

Payments - payment\_id, booking\_id, amount, payment\_status, payment\_date);

foreign key reference Bookings(booking\_id)

PaymentStatus (Success, Failed, Pending)

**OpenAPI integration**



**Candidate Solution:** *Refer to applicant solution.*

**Panel Feedback**

**Summary:** max three lines

**Strengths:** max two lines

**Gaps:** Max two lines

**Proficiency Level:** Low/ Med/High:

1. **Non-functional requirements-(**Mandatory **-**Design/Arch solution & Optional Implementation**):**

* Describe transactional scenarios and design decisions to address the same.
* Integrate with theatres having existing IT system and new theatres and localization(movies)
* How will you scale to multiple cities, countries and guarantee platform availability of 99.99%?
* Integration with payment gateways
* How do you monetize platform?
* How to protect against OWASP top 10 threats.
* Any Compliance

**Solution: Non-Functional Requirements**

* **Distributed Transctions -** using saga pattern for long-running tasks. Eg. Booking + payment
* **Localization** - Store multi-language metadata (movies, descriptions) in MongoDB, and use i18n support in Spring for localization.
* **Scalability -** Kafka-based async event processing, Redis caching, database read replicas, and CDN for static content.
* **Availability -** 99.99% uptime via multi-region cloud deployment (AWS/GCP). Auto-scaling, disaster recovery and failover support.
* **Payment gateways -** Integrate UPI, cards, Razorpay or PayPal SDKs/APIs in Spring Boot.
* **Monetization strategies -** commition model, advertisement
* **Security -** Spring Security with OAuth2 & JWT, OWASP protection, input validation and CSRF protection, rate limiting, secure HTTP headers, CORS
* **Compliance -** GDPR & PCI-DSS readiness, Logging of consent, PII encryption, audit trails.

**Candidate Solution:** *Refer to applicant solution.*

**Panel Feedback**

**Summary:** max three lines

**Strengths:** max two lines

**Gaps:** Max two lines

**Proficiency Level:** Low/ Med/High:

1. **Platform provisioning, sizing & Release requirements:** (Mandatory-Architecture artifacts)

* Discuss your technology choices and decisions through key drivers.
* Discuss database, transactions, and data modelling.
* Discuss COTS enterprise systems that you may need to manage specific areas.
* Discuss hosting solution and sizing (Cloud / Hybrid/ Multi cloud)- Any
* Discuss release management across Geos and internationalization.
* Provide details on monitoring solution and log analysis
* Discuss overall KPIs
* Create a high-level project plan and estimates breakup.

**Solution:**

**Technology choices and decisions through key drivers**

* Scalability: Microservices architecture (Spring Boot + Spring Cloud)
* Developer Productivity: Java 17+, Gradle/Maven, Spring Boot auto-configuration
* Fast Time to Market: Use of open-source tools, RESTful APIs, Swagger
* Security: Spring Security, OAuth2, JWT, HTTPS
* Observability: Prometheus + Grafana, ELK stack
* Asynchronous processing: Kafka for booking events and payment handling

**Database, Transactions, and Data Modelling**

* Relational DB,NoSQL, Elasticsearch
* Transactions are managed using @Transactional, which handles rollbacks, commits, and isolation levels.
* Elasticsearch and caching are useful for performance optimization

**COTS Enterprise Systems**

* Payment processing via Paypal, Stripe or Razorpay.
* SMS/email notifications via AWS SMS or SES.
* Logging and monitoring with ELK Stack (Elasticsearch, Logstash, Kibana).

**Hosting Solution and Sizing**

* Cloud Solution: AWS with Kubernetes for container orchestration.
* Hybrid Option: On-prem integration for legacy systems.
* Multi-Cloud: Azure/GCP expansion for redundancy and failover.
* Auto-scaling groups manage peak traffic loads.

**Release Management Across Geos and Internationalization**

* Global Deployment: Multi-region support in AWS for latency reduction.
* Internationalization: Multi-language support with localization.
* Phased Releases: Canary deployments for controlled feature rollouts.
* CI/CD Pipelines: Jenkins and GitHub Actions for automated deployments.

**Key Performance Indicators (KPIs)**

* System Uptime: 99.99% availability target.
* Response Time: API response within 200ms.
* Booking Success Rate: 98%+ successful transactions.
* Scalability: highly scalable system

**Monitoring**

* Centralized logging with ELK, metrics via Prometheus & Grafana

**High-Level Project Plan & Estimates**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | **Milestone** | **Duration** | **Resources** |
| 1 | Requirement Analysis | 2 Weeks | 2 |
| 2 | Architecture Design | 4 weeks | 2 |
| 3 | Development | 12 weeks | 2 |
| 5 | Testing & QA | 4 weeks | 1 |
| 6 | Deployment | 2 weeks | 2 |

**Candidate Solution:** *Refer to applicant solution.*

**Panel Feedback**

**Summary:** max three lines

**Strengths:** max two lines

**Gaps:** Max two lines

**Proficiency Level:** Low/ Med/High:

1. **Product management and Stakeholder management**

* Please talk about stakeholder management instances
  + What decisions and actions were taken for decision closure?
* Overall technology management
* Enabling team and introducing efficiencies
* Delivery planning and estimates

**Candidate Solution:** *Refer to applicant solution.*

**Stakeholder Management Instances & Decision Closure**

* Decision-Making Process: Regular sprint reviews and stakeholder meetings to gather feedback.
* Monthly stakeholder steering committee to align roadmap with business strategy.
* Actions Taken for Decision Closure:
* Prioritization of features based on user impact and business needs.
* Collaborative discussions with theatre partners to align show management requirements.
* Data-driven decision-making using analytics and customer feedback.
* Conflict resolution through structured discussions and compromise strategies.

**Overall Technology Management**

* Tech Stack Evolution/Consistency: Regular updates based on emerging trends and security patches.
* Scalability Planning: Ensuring infrastructure supports peak loads.
* Disaster Recovery & Backup: Regular backups and failover strategies.
* Code Quality & Best Practices: Continuous integration and code reviews.

**Enabling Team and Introducing Efficiencies**

* Agile Methodology: Implementing Scrum with bi-weekly sprints.
* Automation: CI/CD pipelines, automated testing, and deployment strategies.
* Knowledge Sharing: Conducting regular tech talks and training sessions.
* Cross-functional Collaboration: Encouraging communication between engineering, product, and business teams.

**Delivery planning and estimates**

* Delivery Methodology:Agile Scrum with 2-week sprints
* Initial MVP scope defined collaboratively with stakeholders
* Epics broken into user stories tracked in JIRA
* Release via blue-green deployment and toggled features (LaunchDarkly)
* Estimation Strategy:T-shirt sizing (S/M/L/XL) → Converted to story points (Fibonacci series)

**Panel Feedback**

**Summary:** max three lines

**Strengths:** max two lines

**Gaps:** Max two lines

**Proficiency Level:** Low/ Med/High:

**Final Summary**

Panel Name:

Account Name:

Summary:

mention any domain understanding :

Ability to Solution (Low / Med/ High):

Ability to lead and manage (Low/Med/High):

Communication: Poor/Good/Very Good

Concerns:

Suggested title?

Suggested role?

Disclaimer:

*This document is meant to assess your technical skills and is classified as "Sapient confidential". This document by any means shall not be used/shared without permission from Sapient, non-adherence to this can get your candidature blocked for employment with Sapient.*