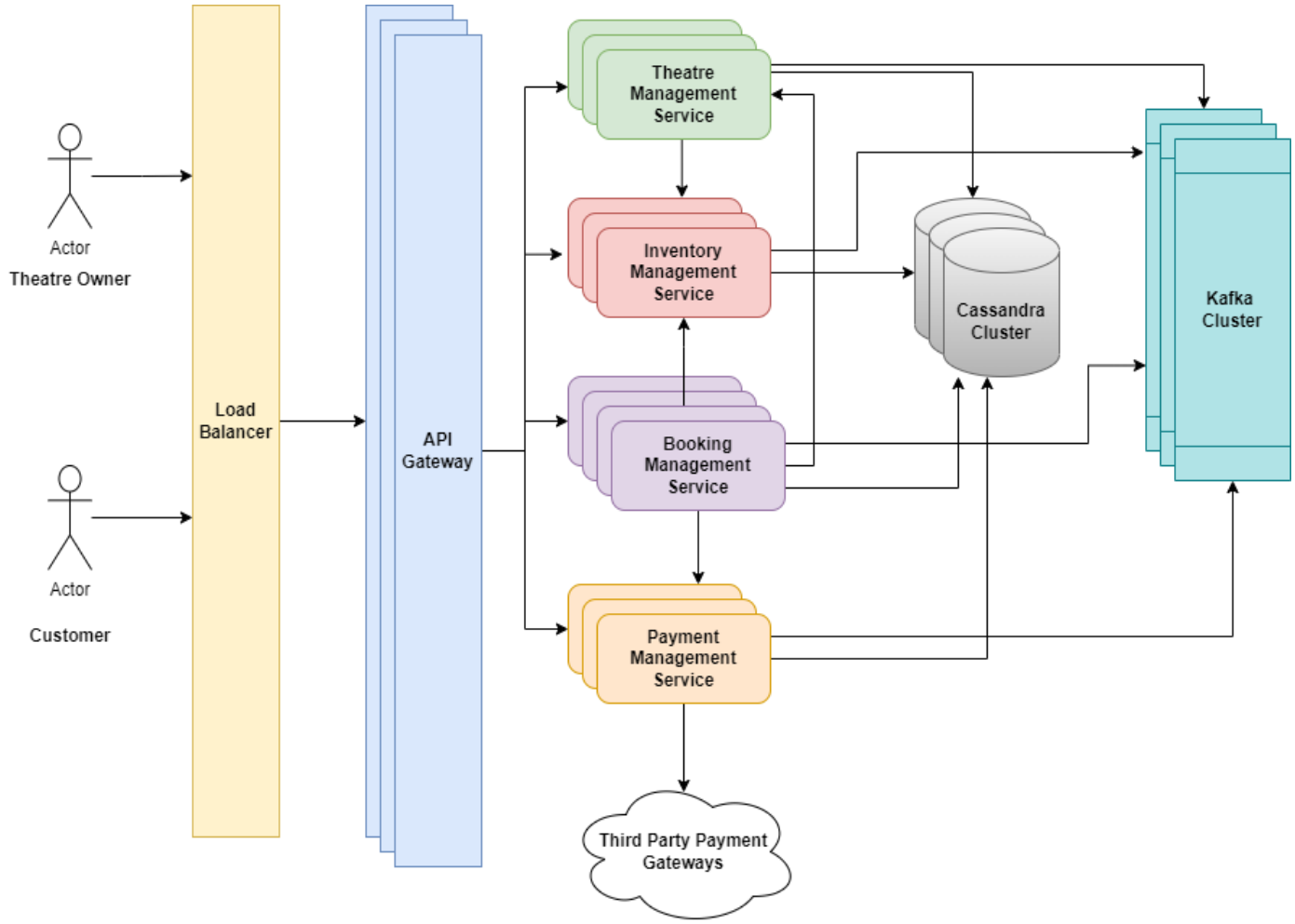
High Level Design – Ticket Booking Platform

1. Goal

Need to build an online movie ticket booking platform that caters to both B2B (theatre partners) and B2C (end customers) clients. Key goals that we want accomplished as part of this 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.

2. High Level Architecture Diagram



3. Functional Requirements

- **Browse theatres currently running the show (movie selected) in the town, including show timing by a chosen date (Implemented)**
- Booking platform offers in selected cities and theatres
    - 50% discount on the third ticket
    - Tickets booked for the afternoon show get a 20% discount
- Book movie tickets by selecting a theatre, timing, and preferred seats for the day
- **Theatres can create, update, and delete shows for the day. (Implemented)**
- Bulk booking and cancellation
- Theatres can allocate seat inventory and update them for the show

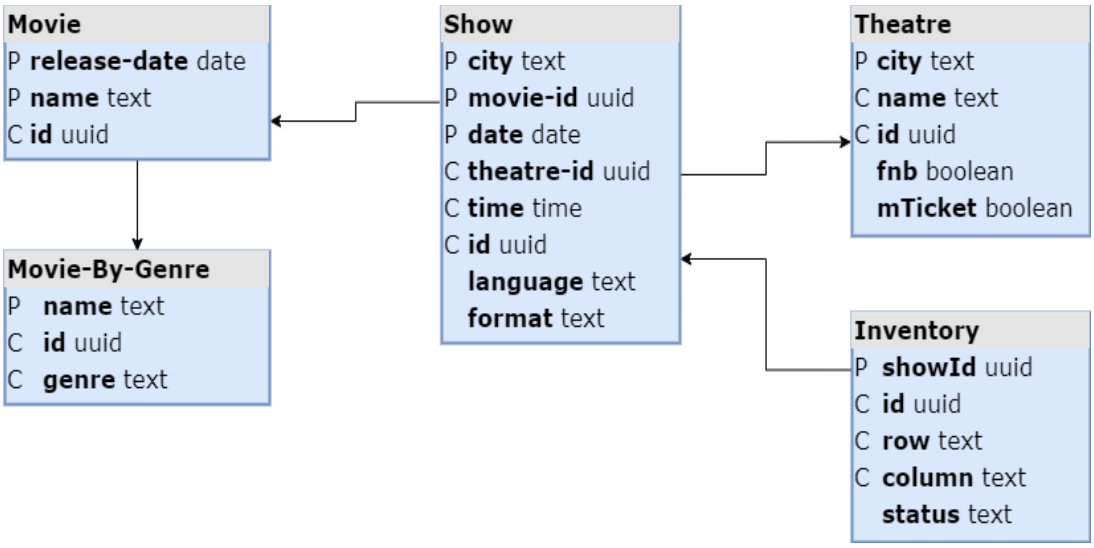
4. Non-Functional Requirements

- **Describe transactional scenarios and design decisions to address the same.**
    1. When customer books tickets, payment should be processed and inventory should be updated.
    2. When theatre owner cancels a show, refund should be initiated for customers who have booked same show.
    3. When theatre owner changes show timing or seating chart or language or format or movie, customers should be notified/provided with refund.
    4. These scenarios will require distributed transaction management and we need to use Saga design pattern to manage these transactions atomicity.
- **Integrate with theatres having existing IT system and new theatres and localization(movies)**
    1. For theatres which have their IT system for ticket booking, we should expose APIs which will keep data (bookings and customer details) in sync.
- **How will you scale to multiple cities, countries and guarantee platform availability of 99.99%?**
    1. To support for multiple cities/countries, we should handle high volume of data. For this we should use data storage which will scale up well for higher volumes. Cassandra can be a good choice for this. It is highly available as well.
    2. To cater customer from different geographic locations, we should have data centers at multiple locations. Any public cloud can support this and platform availability would be guaranteed.
- **Integration with payment gateways**
    1. We should develop separate payment service / module, which will interact with third party payment gateways.
    2. This would make adding support for new payment gateways easy and will no impact other services.
- **How do you monetize platform?**
    1. ADs can be put up on website.
    2. Collaboration with various banks for offers on credit/debit cards.
    3. Collaboration with various payment gateways for offers.
    4. Introducing corporate offers on bulk bookings.

- **How to protect against OWASP top 10 threats.**
    1. We should integrate CI/CD pipeline with DevSecOps tools such as – SonarQube, OWASP Dependency Track, Coverity, HPFortify or Synopsys.
    2. Developer should address CRITICAL, HIGH and MEDIUM issues immediately. LOW priority issues can be planned as per developer's bandwidth.
    3. Before release, penetration testing should also be performed.

5. Platform provisioning, sizing and Release Requirements

- **Discuss your technology choices and decisions through key drivers**
    1. Use of microservice architecture – which will segregate services as per business domain/functionality. This will help with development, scalability, fault isolation and ease of deployment.
    2. Use of Spring Boot for rapid development of microservices
    3. As the data volume will grow as we expand to different cities/countries, we should use database which will be scalable and highly available. Cassandra is good fit for this.
    4. For faster searches across platform, ElasticSearch should be used.
    5. There will be some scenarios which would require asynchronous processing. Apache Kafka should be used for these requirements as its highly scalable, durable and offers high performance.
- **Discuss database, transactions, and data modelling.**
    1. As the data volume will grow as we expand to different cities/countries, we should use database which will be scalable and highly available. Cassandra is good fit for this.
    2. Cassandra has query-first design model, so we should analyze the queries that we would be needing frequently and do data modelling as per that.
    3. Data Model: P – Partition Key, C – Clustering Key



- **Discuss enterprise systems that you may need to manage specific areas**.
    1. We need to integrate with third party payment gateways

- **Discuss hosting solution and sizing (Cloud / Hybrid/ Multi cloud)- Any**
    1. As we need scalable and highly available platform across various geographical locations, we should host our service on any public cloud.

- **Discuss release management across cities, languages etc -** Will cover this section in design discussion
- **Provide details on monitoring solution**
    1. Monitoring tools such as Prometheus, Grafana can be used.
- **Discuss overall KPIs -** Will cover this section in design discussion
- **Create a high-level project plan and estimates breakup.** Will cover this section in design discussion

6. 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

Will cover this section in design discussion.