**Design Document**

**Problem Statement**

Build an online movie ticket booking platform that caters to both B2B (theatre partners)

and B2C (end customers) clients.

**Functional Requirements**

**User Personas:**

1. **Theater Partner:**
   1. Theatre partners be able to onboard their theatres
   2. Theatres can create, update, and delete shows for the day.
   3. Theatres can allocate seat inventory and update them for the show.
2. **Customer**
3. Browse theatres currently running the show (movie selected) in the city, including show timing by a chosen date.
4. Book movie tickets by selecting a theatre, timing, and preferred seats.
5. Booking platform offers in selected cities and theatres
   1. 50% discount on the third ticket
   2. Tickets booked for the afternoon show get a 20% discount
6. Bulk booking and cancellation.

**Non Functional Requirements**

1. High availability
2. Transactional scenarios identification and management.
   1. Avoid duplicate seat booking.

**Services**

1. UserManagementService
   1. User
   2. UserACL
2. TheatreService
   1. Theater
   2. Seat
   3. Show
3. MovieService
   1. Movie
4. TicketBookingService
   1. Booking
   2. ShowSeatBookingStatus
5. PaymentService
   1. Discount

**Entity Model**

User

* UserdId (CustomerId, TheatreId)
* Name
* Age
* City
* Longitude
* Latitude
* Type (Customer, TheatrePartner)

UserACL

* RoleId
* UserId
* Role

Movie

* MovieId
* Name
* ReeleaseType (National, International, Regional)
* ReleaseDate
* Language
* Cast

Theatre

* TheatreId
* Name
* City
* Location (Longitude & Latitude)
* ScreenType (Single/Multiscreen)

Seat

* SeatId
* TheatreId
* ScreenNumber
* SeatNumber

Show

* ShowId
* TheatreId
* MovieId
* ShowDateTime
* Status (Confirmed, Cancelled)

ShowSeatStatus

* ShowSeatStatusId
* ShowId
* SeatId
* Status (Available, BookingInProgress, Booked)
* BookingId
* LastUpdatedAt

BookingStatus

* BookingId
* UserId
* BookingDate
* BookingAmount
* Status (In\_Progress, Completed, Cancelled)

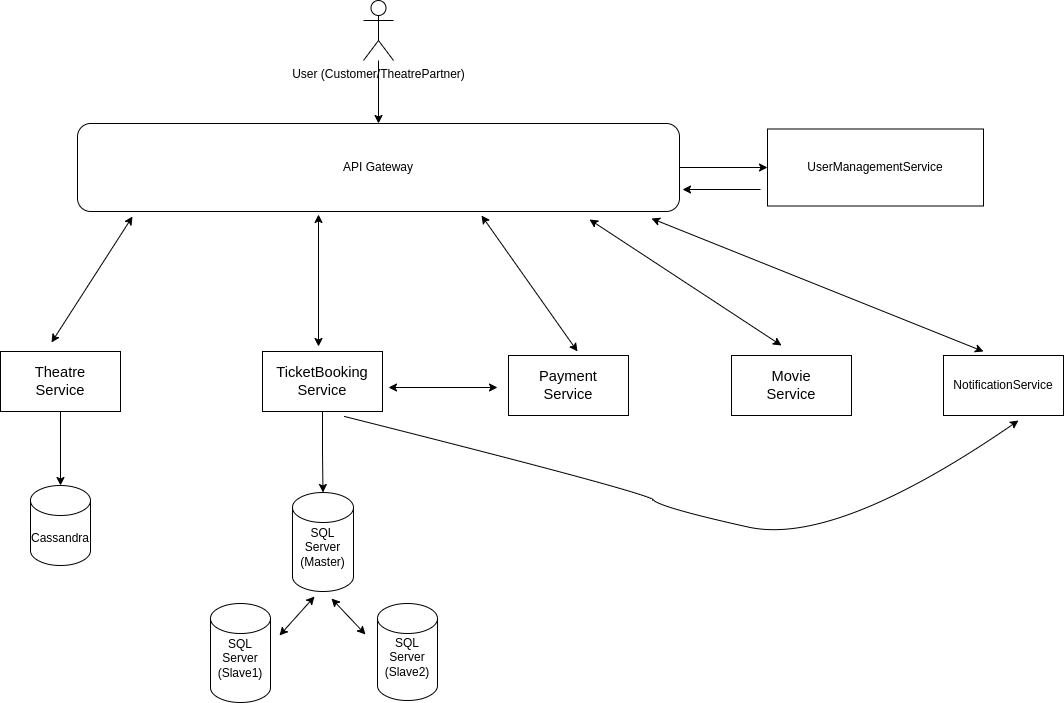
DiscountOffer

* DiscountOfferId
* DiscountPercent
* TheatreId
* StartDate
* EndDate
* MinTickets

**APIs**

| S.No. | Functionality | Service | Contract | Roles |
| --- | --- | --- | --- | --- |
| 1. | Theatre partners be able to onboard their theatres | TheatreService | PUT  /theater  {  “Name”: “”,  “Location”:  .  .  .  .  .  } | THEATRE\_ADMIN |
| 2. | Theatres can create, update, and delete shows for the day | TheatreService | PUT  /threater/{theaterId}/show  {  }  —--------  PATCH  /threater/{theaterId}/show  {  }  —--------  DELETE  /threater/{theaterId}/show/{showId} | THEATRE\_ADMIN |
| 3. | Theatres can allocate seat inventory and update them for the show. | TheatreService | PUT  /theater/{theatreId}  {seats: []} | THEATRE\_ADMIN |
| 4. | Browse theatres currently running the show (movie selected) in the city, including show timing by a chosen date. | TheatreService | GET /show?city={city}&showDateTime={yyyy-MM-dd} | CUSTOMER\_READ |
| 5. | Book movie tickets by selecting a theatre, timing, and preferred seats | TicketBookingService | PUT  /booking  { theatreId: "", showId: “”,  seats: []  } | CUSTOMER\_WRITE |
| 6. | Booking platform offers in selected cities and theatres | PaymentService | GET /discountOffer?city={city}&theatre={theatreId} | CUSTOMER\_READ |

**High Level Design**



**Ticket Booking Flow**

1. User selects the seats and make the api call to ticketBookingService.
2. The underlying SQL store updates the status for the selected seats as ‘BookingInProgress’ (Txn management needed).
3. If the step 2 above completes successfully, then request is redirected to paymentService and user can make the payment.
4. On successful payment payment service makes api call to ticketBookiungService to confirm the booking with txnReferenceId and seats status is changed to ‘Booked’
5. User is redirected to booking status page.

**Failure Scenarios Handling**

1. If unable to update seat status in step #1 as ‘BookingInProgress’ due to a concurrent booking, user is asked to reselect other seats.
2. If there is a timeout or failure in payment service the request is accepted for retry in the background and in case of succes/failure txn it will notify the ticketBookingService to complete or cancel the booking accordingly and trigger a notification to user about the same.

**Edge Cases**

1. In case the payment retry is taklng longer than lets say an SLA of 5 - 10 mins to confirm the payment status, the seats still in ‘BookingInProgress’ can be made available for the user for the selection.