# Purpose

Purpose of this document to capture requirement, design, db models for the below problem.

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.

# Functional Requirements

|  |  |
| --- | --- |
| SN | Requirement |
| FR1 | Browse theatres currently running the show (movie selected) in the town, including show timing by a chosen date |
| FR2 | Booking platform offers in selected cities and theatres  • 50% discount on the third ticket  • Tickets booked for the afternoon show get a 20% discount |
| FR3 | Book movie tickets by selecting a theatre, timing, and preferred seats for the day |
| FR4 | Theatres can create, update, and delete shows for the day. |
| FR5 | Bulk booking and cancellation |
| FR6 | Theatres can allocate seat inventory and update them for the show. |

# Non Functional Requirements

|  |  |  |
| --- | --- | --- |
| **SN** | **Requirement** | **Comments** |
| NF1 | Describe transactional scenarios and design decisions to address the same. | XYZ apps will be connecting to registered Theatre using REST and will fetch data from there. |
| NF2 | Integrate with theatres having existing IT system and new theatres and localization(movies) | Using REST/SOAP will fetch Theatre, Movie and timing |
| NF3 | How will you scale to multiple cities, countries and guarantee platform availability of 99.99%? | Setting up more containers, or Amazon Cloud Front(CDN) using AWS cloud |
| NF4 | Integration with payment gateways | Don’t have much idea |
| NF5 | How do you monetize platform? |  |
| NF6 | How to protect against OWASP top 10 threats. | Applying security patches for threats and vulnerabilities, scheduled scans etc |

# Main Objects

As per my understanding there will be the following main objects in database.

|  |
| --- |
| **Users** |
| userId  name  dateOfBirth  mobNo  emailId  sex  typeoflogin |

|  |
| --- |
| **Theatre** |
| theaterId  theaterName  city  address |

|  |
| --- |
| **Movie** |
| movieId  movieName  movieType  language  status  moviePic  related movies |

|  |
| --- |
| **MovieTheatre** |
| movieId  TheaterId  Timing  Status  Seats  finalRating |

|  |
| --- |
| **Booking** |
| bookingId  userId  movieId  bookedSeats  amount  booked\_date  movie\_timing  payment\_info |

|  |
| --- |
| **ReviewComments** |
| reviewComments  movieID  userID  reviewDate  rating |

|  |
| --- |
| **AvailableScheme** |
| schemeID  schemeName  schemeStartDate  schemeEndDate  isActive |

# Design Flow

We will be creating different microservices for this overall solution, and can scale these services on the basis of load/need using containers orchestration available tools in AWS-

1. Booking Platform will offer its services to Theatres, by creating microservice say **UpdateDetailsService,** which will provide services like**:**
   1. To add/delete/update any movie in the platform.
   2. Will enable theatre to update their seats inventory for a specific movie.

Technology: Spring, Oracle, REST

1. Second microservice **BrowsingService**, which will do tasks like following:
   1. User authentication, we can keep password in the DB in encrypted form
   2. User can browse the movie and book the seat and ticket for selected time.
      1. Call BillingService to show the final cost for the tickets.
      2. Call PaymentService to make payment
      3. If both BillingService and PaymentService done then update inventory
      4. If user has clicked cancel in any moment of above process then show booking page
   3. Updating/showing review comments and final rating, on the input of rating, update finalRating of the movie.

Technology: Spring, Oracle, REST, jsps

1. Third microservice, **BillingService**, which will be doing following:
   1. This microservice will be called mostly from **BrowsingService**, for which input will be user id, order details and it will calculate the final bill.
   2. While calculating the final bill, it will check the <<**AvailableScheme**>> db object and will calculate break up of final bill and send the response.
   3. **BillingService** will support predefined schemes, which can be made inactive/active any time and support the logic of following schemes as start:
      1. schemeID: sc01, schemeName: buy2get50 schemestartDate:01/01/2023 schemeEndDate: 28/02/2023 active: yes
      2. schemeID: sc02, schemeName: after12free20 schemestartDate:01/01/2023 schemeEndDate: 28/02/2023 active: yes
      3. schemeID: sc03, schemeName: buy3get50 schemestartDate:01/01/2023 schemeEndDate: 28/02/2023 active: no

Technology: Spring, Oracle, REST

1. Fourth microservice, **PaymentService** which will be responsible for making payment and talking to payment gateways

Technology: Spring, Oracle, REST, Security

1. **AdminService**, to do some maintenance and clearance stuff like:
   1. Enable/disabling scheme
   2. Facility to communicate to any service like **UpdateDetailsService**
   3. To facilitate backend work, to update inventory, clearing logs etc

Technology: Spring, Oracle, REST, Security

# Platform provisioning, sizing & Release requirements

|  |  |  |
| --- | --- | --- |
| SN | Query | Comment |
| 1 | Discuss your technology choices and decisions through key drivers | Spring, microservices, Oracle, REST and DB will be used as a mode of communication between two micro services |
| 2 | Discuss database, transactions, and data modelling. | Distributed transactions |
| 3 | Discuss enterprise systems that you may need to manage specific areas | Cannot think of |
| 4 | Discuss hosting solution and sizing (Cloud / Hybrid/ Multi cloud)- Any | Multi cloud it depends upon of geography of the theatres we want to manage. |
| 5 | Discuss release management across cities, languages etc | NA |
| 6 | Provide details on monitoring solution | Can have Grafana dashboards for monitoring various containers etc |
| 7 | Discuss overall KPIs | NA |
| 8 | Create a high-level project plan and estimates breakup. | Attached |

# Project Plan

