**Technical Design Document – Online Movie Ticketing System**

**Functional Requirement Assumptions**

1. Client can avail ticket booking service from anywhere based on geographical locations.
2. Once client selects city, the movie api will allow user access to different movies available in the city.
3. Once user selects movie, our ticketing service will display cinema running the movie.
4. The user can then choose the show at a particular cinema to book tickets.
5. Our service will show seating arrangement from where user can either select single or multiple seats.
6. User will have option to hold on to a seat for 5 minutes lock period after selection before finalizing the booking.
7. Our service will serve the customer in first come first serve manner.

**Non Functional Requirement Assumptions**

1. System/Service should be highly concurrent and available.
2. System should be scalable, concurrent, ACID compliant and secure.

**Technical Stack**

1. Will be having a backbone central Web Api layer that will be secure and can be hosted later in Azure utilizing Azure Api Management for all the Non Functional assumptions made.
2. Will implement data access layer using repository pattern and using Entity Framework database first approach.
3. The service will have a UI for desktop that will be developed in Angular 8 and separate interfaces for apps to be developed for iPhone and android devices.
   1. **High Level Design**

We will initially have separate web servers for managing sessions, application server handling all ticketing management (storing data in database) and cache servers for processing reservation. This entire design will be open for cloud migration where we can have a service bus topic for with subscriptions for catering to online demands, Azure Redis replacing the cache server requirement and Azure Api Management to take care of our central api.

Load Balancers

Clients

Application Server

Web Servers Cache Servers

Database Servers

* 1. **Ticket Booking Workflow**

1. User selects app, searches for a movie.
2. User selects a movie
3. User is shown available list of shows for the movie.
4. User selects the show.
5. User gets to select seats to reserve.
6. If number of seats available user is shown a map for the seats to be booked.
7. Once users selects seats the service will proceed to reserve those seats.
8. If the seats cannot be reserved, then user will move into waiting phase where once a seat is freed the user will be taken onto that from reservation pool.

* If all seats are reserved, the error message will be displayed that all seats are reserved.
* User wants to reserve seats that are no longer available but other seats are available, the user will be redirected to the seating arrangement map to book a differet set of seats.

1. Once seats are successfully reserved, the user has 5 minutes to pay for the reservation. After payment, booking is marked as complete.

User

Application Server

Web Server

Search Movies Search movies

Database

Search results

Search result

Select movie Reserve seats

Display show

Show seats Confirm Reservation

Select seats Reserve seats

Reservation No

Seats available

Seats can be reserved??

Confirmation

No

Waiting service queue

Yes, in waiting queue

Track Reservation

* 1. **Database design and tables**

1. Movie (**MovieId**, Title, Description, Duration, Language, Release date, country)
2. Show (**Show id**, Date, Start time, End time, *Cinema Hall Id, Movie id*)
3. Booking (**bookingId**, Number of seats, timestamp, status, *UserId, ShowId*).
4. User (**UserId**, Name, Password, Email, Phone).
5. Cinema (**CinemaId**, Name, TotalCinemaHalls, *CityId*)
6. CinemaHall (**CinemaHallId**, Name, Total seats, *cinema id*)
7. Show Seat(**Show seat id**, status, price, *CinemaSeatId, ShowId, BookingId*)
8. Payment (**PaymentId**, Amount, Timestamp, DiscountCOuponid, *BookingId*, PaymentMethod)
9. City (**CityId**, Name., State, ZipCode)
10. Cinema Seat (**CinemaSeatId**, SeatNumber, Type, *CinemaHallId*)

All primary keys marked as bold and foreign keys in italic.