# **ONLINE MOVIE TICKET SYSTEM**

# CAPGEMINI TRAINING PROGRAMME

# JEE CLOUD

# PREPARED BY

Divya

Tushar

Umang

**INTRODUCTION**

This is an online movie ticket system having an admin role and a customer role where admin has access to add city, the theatres in the city, the movies each theatre will showcase and the show timings for each movie. The customer can be an unregistered user or a registered user. Unregistered user can register itself or can view movie shows without registering while the registered user can view movie shows and book tickets for the particular show by paying online. Registered users will have the option to cancel its booking in which case he will get a refund on some specific terms. Users can select the language of movie and view the seat map and choose seats for themselves.

**CLASSES**

**Class: User**

**Attributes:**

* private Integer userId
* private String userName
* private String password

**Methods:**

* **registerNewUser(): User**

This method will prompt user to enter all the necessary details required to register the user as a customer.

* **signIn(String, String): Boolean**

It will sign in registered users and admin into the system.

* **signOut: Boolean**

It will return true once the user logs out of the system successfully.

**Class: Admin**

**Attributes**

* private adminId: String
* private adminName: String
* private adminPassword: String
* private dateOfBirth: LocalDate
* private adminContact: String

**Methods:**

* **addTheater(Theater): Theater**

It will return the object of theater once a new theater is added in the system successfully.

* **deleteTheater(Integer): Boolean**

This method will prompt admin to delete a theater’s data by passing its Idfrom the system and on successful operation it will return true..

* **addMovie(Movie): Movie**

It will allow admin to add a movie to be showcased by the theater in the system.

* **deleteMovie(Integer): Boolean**

It will return the object of theater once a new theater is added in the system successfully.

* **addScreen(Screen): Screen**

This method will prompt admin to add new screen in the theater in the system.

* **deleteScreen(Integer):Boolean**

It will allow admin to delete a screen of theater from the system.

* **addShow(Show): Show**

It will return the object of show once a new show is added for a movie in the system successfully.

* **deleteShow(Integer):Boolean**

It will allow admin to delete a show for a movie from the system.

**Class: Customer**

**Attributes:**

* private customerId: String
* private customerName: String
* private customerPassword: String
* private dateOfBirth: LocalDate
* private myTickets: List<Ticket>
* private customerContact: String

**Methods:**

* **chooseCity(String): Boolean**

Selects the city and returns the list of theaters in that particular city.

* **bookMovieTicket(Theater): Boolean**

Invokes the entire booking process, which will return true once the booking is completed successfully.

* **cancelMovieTicket(Ticket): Boolean**

Initiates the cancellation of the movie ticket. Returns true on completion.

**Class: Theater**

**Attributes:**

* theaterId: Integer
* theaterName: String
* theaterCity: String
* movies: Movie[\*]
* listOfScreens: Screen[\*]
* managerName: String
* managerContact: String

**Methods**

* **searchMovie(String) : Movie**

This method is used to search for a movie in a particular theater. If the user requested movie is present in that theater then the details of the movie will be shown to the user.

* **searchScreen(Integer): Screen**

This method is used to search for a screen in a particular theater. If the requested screen is present in the theater then the details of the screen will be shown. This operation is used by Admin to upLocalDate the screen details.

**Class: Movie**

**Attributes**

* movieId: Integer
* movieName: String
* movieGenre: Show[\*]
* movieDirector: LocalDate
* movieLength: Integer
* languages: String[\*]
* movieReleaseDate: LocalDate

**Class: Booking**

**Attributes**

* bookingId: Integer
* movieId: Integer
* showId: Integer
* showRef: Show
* bookingDate: LocalDate
* transactionId: Integer
* totalCost: Double
* seatList: Seat[\*]
* ticket: Ticket

**Methods**

* **chooseSeats(): Seat[\*]**

For a particular show the customer chooses seats he is interested in booking. This will return a list of seats he has chosen as well as block the seats from booking.

* **calculateTotalCost(): Double**

This function will fetch the price assigned to each seat from the list of seats the user has selected for booking and add them up.

* **choosePaymentMethod(): PaymentMethod**

The customer now chooses the method he wants use for confirming his purchase of seats for the show.

* **makePayment(PaymentMethod, Double): Boolean**

In this method the mode of payment user has selected is passed along with the total cost of current booking. It will invoke merchant side transaction process. For now we are considering this method only returns the confirmation of payment, later we will also consider an object will be returned containing transaction information.

* **showTicket(): Ticket**

Upon confirmation of payment the ticket will be generated containing reference of current object and ticket parameters.

* **cancelBookings(Seats…): Boolean**

This method will take as input the seats from the current booking session and upLocalDate their status as Available in corresponding show.

**Class: Show**

**Attributes**

* showId: Integer
* showStartTime: Time
* showEndTime: Time
* seats: Seat[\*]
* showName: String
* movieName: Movie
* screenId: Integer
* theaterId: Integer

**Methods**

* **upLocalDateSeatsBooked(in String[\*], in boolean): Seats[\*]**

This method is used to search for a show in a theatre. If the requested show is present in the theatre then the detail of the show will be shown to the user.

**VALIDATION RULES**

* All Id should not be null or 0.

Admin Id will be 4 digit starting with 1.

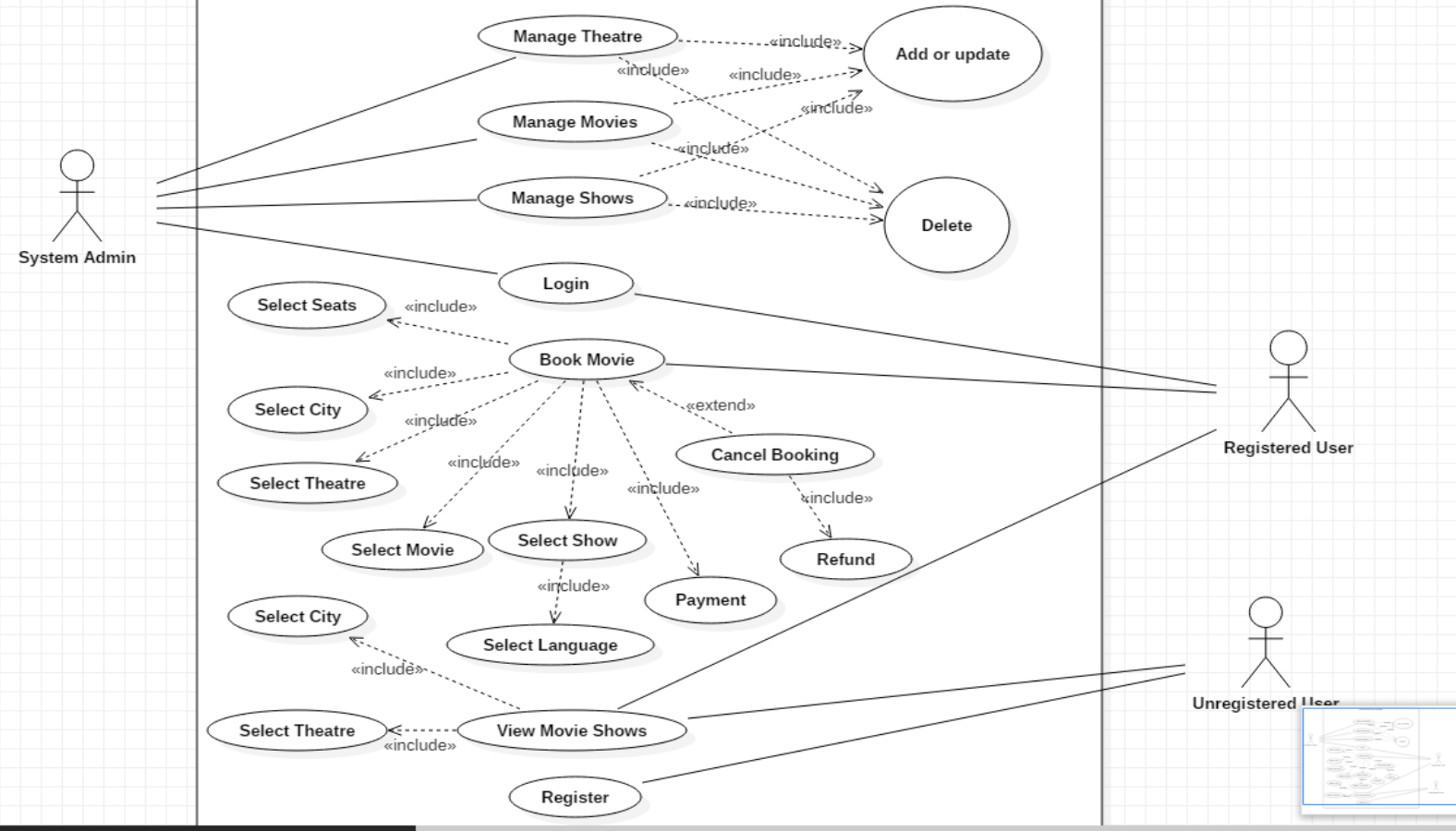
Customer Id will be min 6 digit.

Theater Id will be min 4 digit starting with 2  
 Movie Id will be min 4 digit starting with 3.  
 Screen Id will be min 4 digit

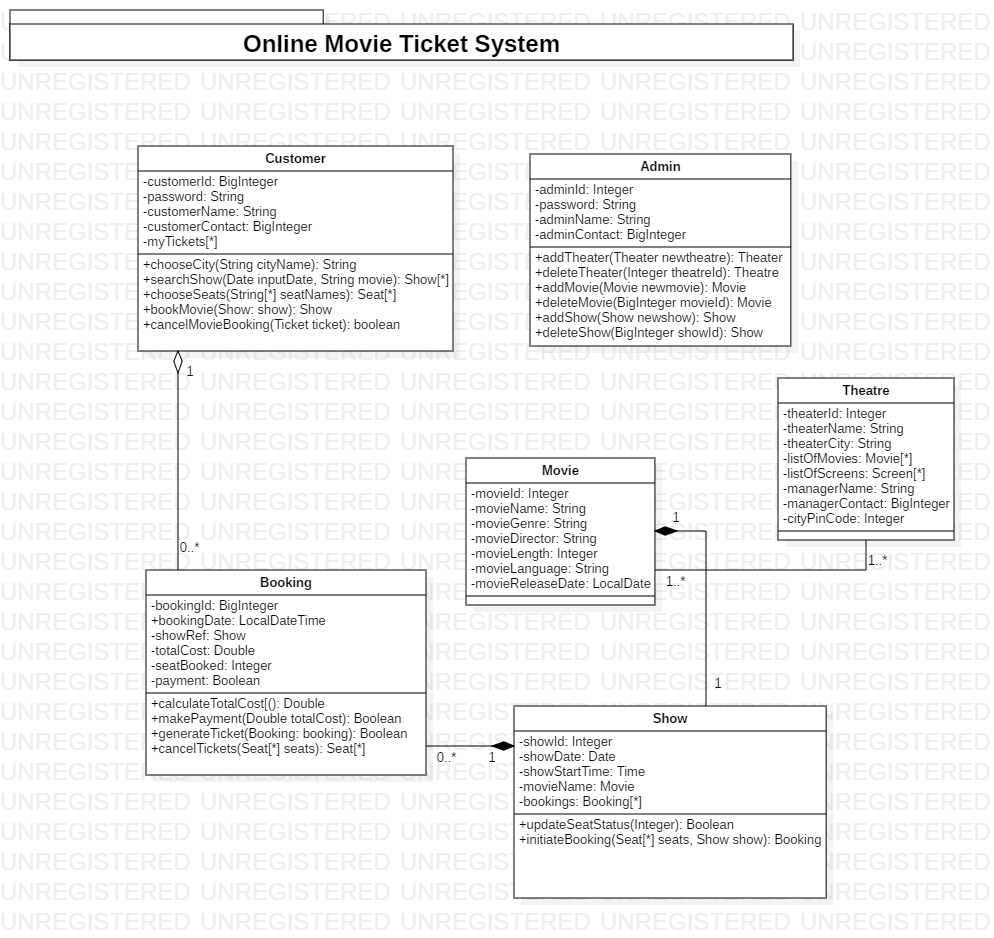
ShowId will be min 8 digit having theaterId, movieId and screenId concatenated.  
 SeatId will be min 6 digit with theaterId and screenId concatenated.  
 BookingId will be 10 digit with customerId, theaterId , movieId , showId concatenated.  
Ticket Id will be min 8 digit and will have booking and transactionId combination.

* Name field in any class should not be null and must be character.
* Password should be of minimum 8 characters.
* Contact Number must be of 10 digits.
* Date of Birth should not be null and should be previous from present Date.
* Booking Date should be either present day or previous not the future.
* Movie show booked Date and time should be either present Date or future.
* Any new show added in the screen of a particular theater should not be already existing for that screen.

## **USE CASE DIAGRAM**

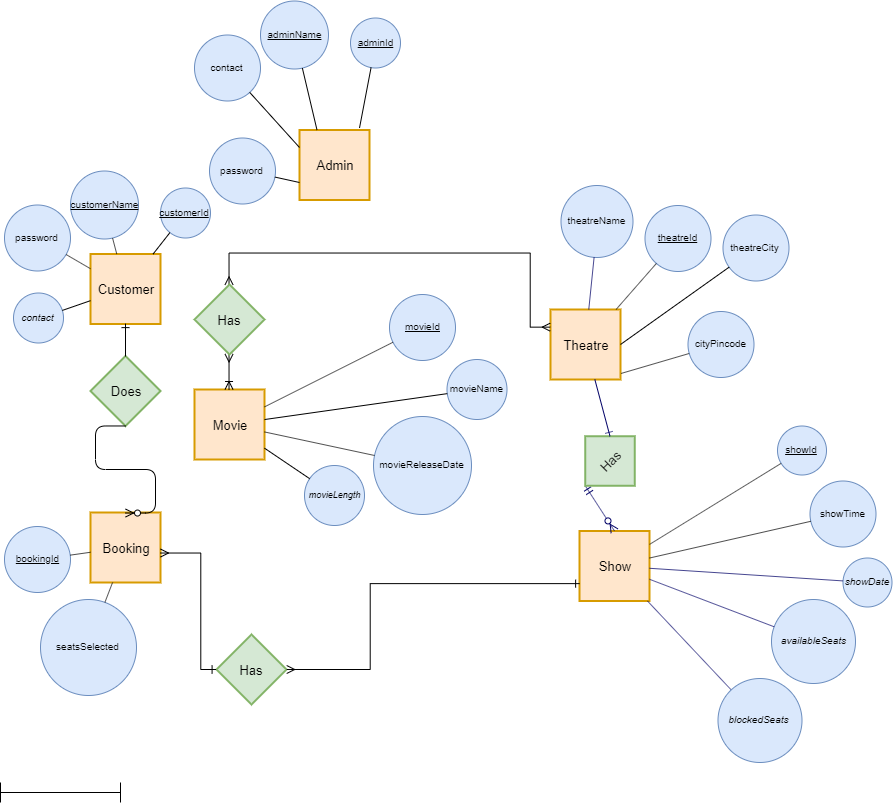


## **CLASS DIAGRAM**

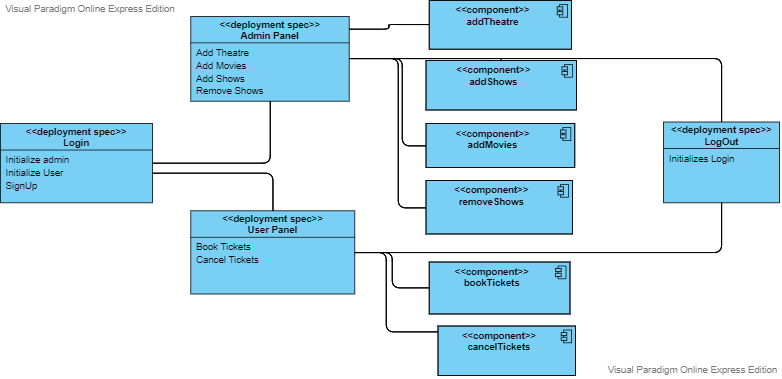


­­

ER Model Diagram



Deployment Diagram



SPRINT – 1

Tasks:

* Class Diagram
* Use case Diagram
* Project Flow with introduction

SPRINT – 2

Tasks: Implemented collections

* **Unregistered User** : Can view movies,theatres,shows,Register
* **Registered User**: Can Book Movie, View movies and shows in selected theatres, cancel Booking
* **Admin** : Can add theatre, add/update movie, add/update show,View customers

SPRINT – 3

Tasks: Implemented Jdbc

* **Unregistered User** : Can view movies,theatres,shows,Register
* **Registered User**: Can login, Book Movie, View movies and shows in selected theatres, cancel Booking
* **Admin** : Can login, add theatre, add/update movie, add/update show,View customers

SPRINT – 4

Tasks: Implement Jpa with spring mvc

* **Unregistered User** : Can view movies,theatres,shows,Register
* **Registered User**: Can login,Book Movie, View movies and shows in selected theatres, cancel Booking
* **Admin** : Can login, add theatre, add/update movie, add/update show,View customers

­­­­­­