# **LLD for a "Book My Show" system**

### Basic Features

1. **User Management**: Users can register, login, and manage their profiles.
2. **Search Movies**: Users can search for movies by location, language, and genre.
3. **View Showtimes**: Users can view showtimes for a selected movie.
4. **Book Tickets**: Users can select a showtime, choose seats, and book tickets.
5. **Payment**: Users can make payments for their bookings.
6. **Notifications**: Users receive booking confirmation notifications.

### Key Classes and Their Interactions

#### 1. User Management

* **User**
* **UserService**
* **AuthenticationService**

#### 2. Movie and Showtime Management

* **Movie**
* **Showtime**
* **Cinema**
* **CinemaHall**
* **Seat**

#### 3. Booking and Payment

* **Booking**
* **Payment**
* **PaymentService**

#### 4. Notification

* **NotificationService**

### Design Patterns Involved

1. **Singleton Pattern**: For services like AuthenticationService and NotificationService.
2. **Factory Pattern**: For creating payment objects (CreditCardPayment, PayPalPayment, etc.).
3. **Observer Pattern**: For sending notifications.

### Class Design

#### User Management

public class User {

private String userId;

private String name;

private String email;

private String password;

// getters and setters

}

public class UserService {

private Map<String, User> users = new HashMap<>();

public void registerUser(String userId, String name, String email, String password) {

// Implementation

}

public User loginUser(String email, String password) {

// Implementation

return null;

}

}

#### Authentication Service (Singleton)

public class AuthenticationService {

private static AuthenticationService instance;

private AuthenticationService() {}

public static AuthenticationService getInstance() {

if (instance == null) {

instance = new AuthenticationService();

}

return instance;

}

public boolean authenticate(String email, String password) {

// Implementation

return false;

}

}

#### Movie and Showtime Management

public class Movie {

private String movieId;

private String title;

private String language;

private String genre;

// getters and setters

}

public class Showtime {

private String showtimeId;

private Movie movie;

private CinemaHall cinemaHall;

private Date startTime;

private List<Seat> availableSeats;

// getters and setters

}

public class Cinema {

private String cinemaId;

private String name;

private String location;

private List<CinemaHall> cinemaHalls;

// getters and setters

}

public class CinemaHall {

private String hallId;

private List<Seat> seats;

// getters and setters

}

public class Seat {

private String seatId;

private boolean isAvailable;

// getters and setters

}

#### Booking and Payment

public class Booking {

private String bookingId;

private User user;

private Showtime showtime;

private List<Seat> bookedSeats;

private Payment payment;

// getters and setters

}

public class Payment {

private String paymentId;

private double amount;

// getters and setters

}

public interface PaymentService {

void processPayment(Payment payment);

}

public class CreditCardPaymentService implements PaymentService {

@Override

public void processPayment(Payment payment) {

// Implementation

}

}

public class PayPalPaymentService implements PaymentService {

@Override

public void processPayment(Payment payment) {

// Implementation

}

}

#### Payment Factory

public class PaymentServiceFactory {

public static PaymentService getPaymentService(String type) {

if (type.equalsIgnoreCase("CreditCard")) {

return new CreditCardPaymentService();

} else if (type.equalsIgnoreCase("PayPal")) {

return new PayPalPaymentService();

}

return null;

}

}

#### Notification Service (Observer)

public class NotificationService {

private static NotificationService instance;

private NotificationService() {}

public static NotificationService getInstance() {

if (instance == null) {

instance = new NotificationService();

}

return instance;

}

public void sendNotification(User user, String message) {

// Implementation

}

}

### Interactions

1. **User Registration/Login**:
   * User registers or logs in via UserService.
   * Authentication is handled by AuthenticationService.
2. **Search Movies**:
   * User searches for movies, and UserService interacts with a repository or database to fetch movie details.
3. **View Showtimes**:
   * User selects a movie and views showtimes, UserService fetches showtime details from Showtime.
4. **Book Tickets**:
   * User selects a showtime and seats, UserService creates a Booking and processes payment through PaymentServiceFactory.
5. **Payment**:
   * PaymentServiceFactory creates appropriate PaymentService (e.g., CreditCardPaymentService) to process the payment.
6. **Notification**:
   * On successful booking, NotificationService sends confirmation to the user.