Movie Ticket Booking System with Genre Search and Review System

This mini-project will implement a Movie Ticket Booking system where users can:

- Book movie tickets.
- Search movies based on genres.
- Leave reviews for movies.

We will use Java for the backend and SQL for the database, with a simple front-end interface using Java Swing.

Tools and Technologies Used:

- **Java (for Backend and Frontend)**: Core logic and GUI (Swing).
- **MySQL**: Database to store movie, booking, genre, and review data.
- **JDBC**: Java Database Connectivity to interact with the database.
- **Java Swing**: GUI components for user interaction.

Step-by-Step Implementation

1. Set up MySQL Database

First, set up the MySQL database with the necessary tables.

CREATE DATABASE moviedb;

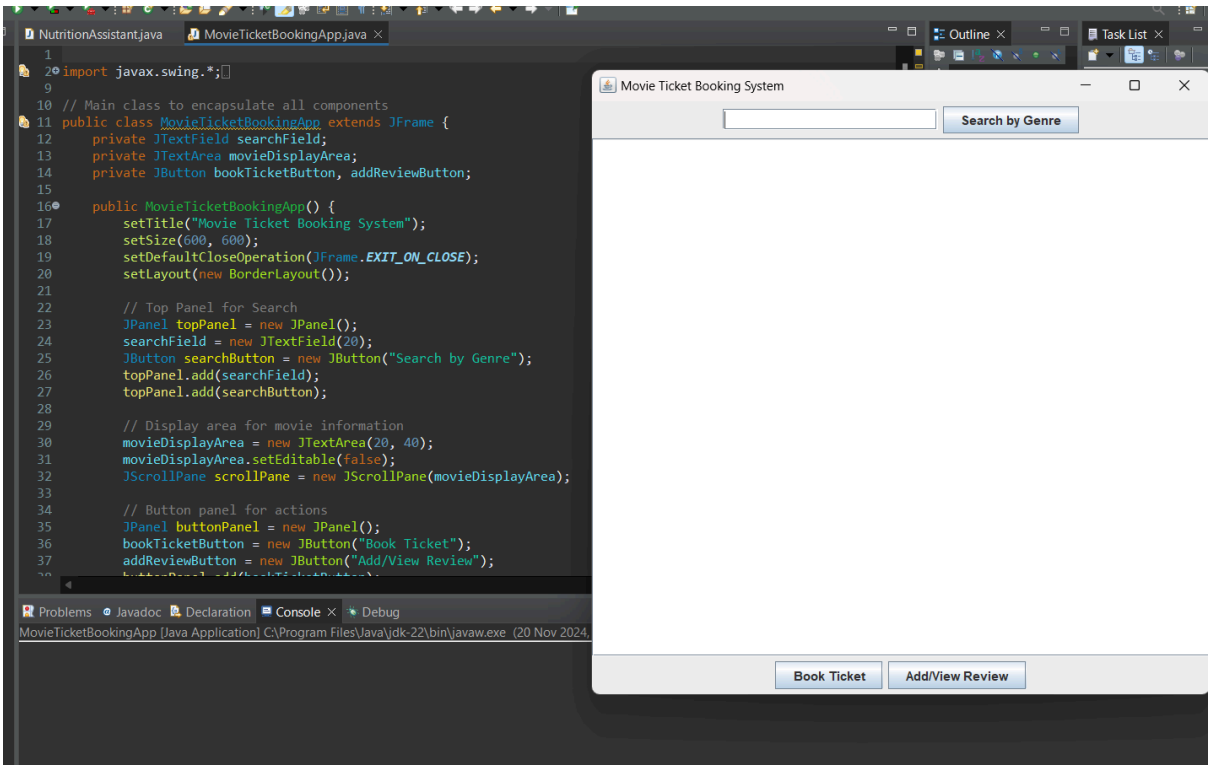
```
USE moviedb;

-- Movies table
CREATE TABLE Movies (
    movie_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(255),
    genre VARCHAR(100),
    director VARCHAR(255),
    duration INT,  -- Duration in minutes
    release_year INT
);

-- Reviews table
CREATE TABLE Reviews (
    review_id INT PRIMARY KEY AUTO_INCREMENT,
    movie_id INT,
    review TEXT,
    rating INT,  -- Rating from 1 to 5
    FOREIGN KEY (movie_id) REFERENCES Movies(movie_id)
);
```

-- Bookings table
CREATE TABLE Bookings (
booking_id INT PRIMARY KEY AUTO_INCREMENT,
user_name VARCHAR(255),
movie_id INT,
seats INT,
booking_time DATETIME,
FOREIGN KEY (movie_id) REFERENCES Movies(movie_id)
);

```
mysql> Desc reviews;
+-------------+-----------+------+-----+---------+----------------+
| Field       | Type      | Null | Key | Default | Extra          |
+-------------+-----------+------+-----+---------+----------------+
| review_id   | int       | NO   | PRI | NULL    | auto_increment |
| user_id     | int       | YES  | MUL | NULL    |                |
| movie_id    | int       | YES  | MUL | NULL    |                |
| rating      | int       | YES  |     | NULL    |                |
| comment     | text      | YES  |     | NULL    |                |
| review_date | timestamp | YES  |     | NULL    |                |
+-------------+-----------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

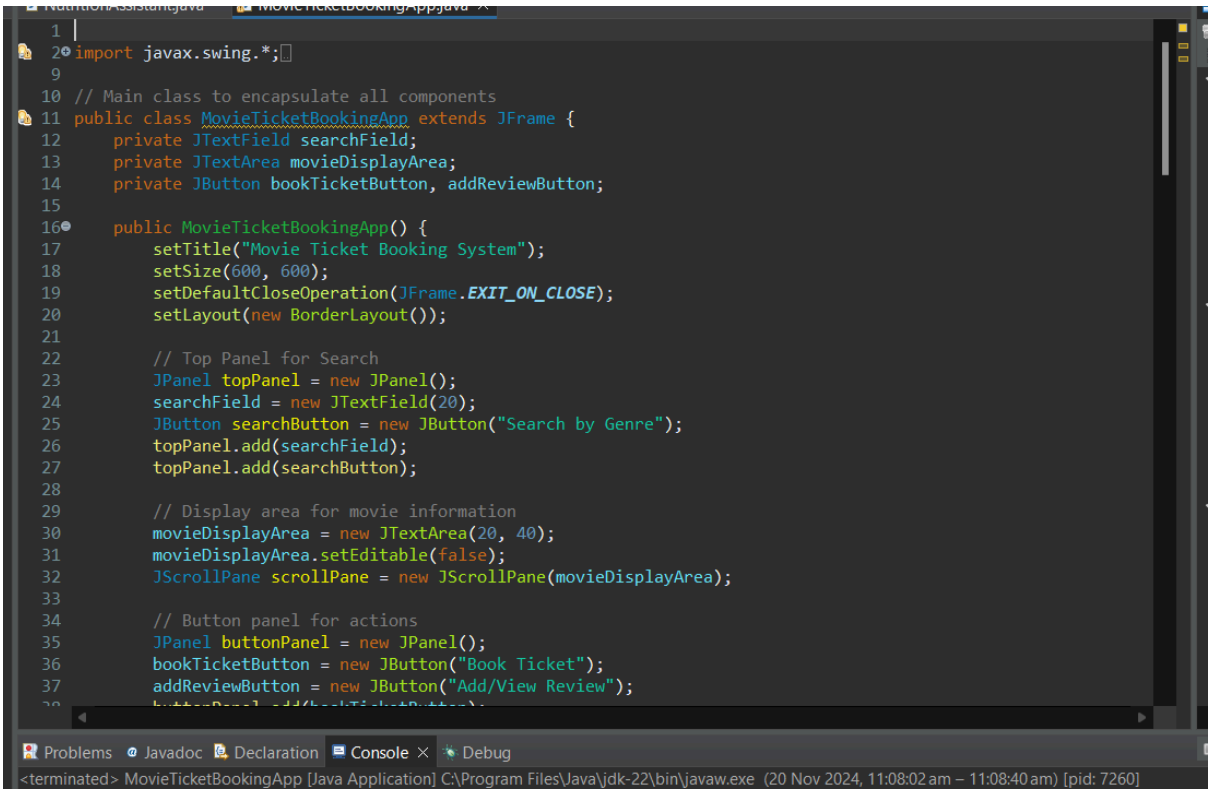
```
mysql> DESC tickets;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| ticket_id    | int         | NO   | PRI | NULL    | auto_increment |
| user_id      | int         | YES  | MUL | NULL    |                |
| movie_id     | int         | YES  | MUL | NULL    |                |
| booking_date | timestamp   | YES  |     | NULL    |                |
| seat_no      | varchar(10) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

```
mysql> DESC users;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| user_id  | int         | NO   | PRI | NULL    | auto_increment |
| username | varchar(50) | YES  | UNI | NULL    |                |
| password | varchar(50) | YES  |     | NULL    |                |
| email    | varchar(50) | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```




```
mysql> show databases;
+--------------------------+
| Database                 |
+--------------------------+
| information_schema       |
| moviedb                  |
| mysql                    |
| nutrition                |
| performance_schema       |
| personalfinancetracker   |
| sakila                   |
| sys                      |
| testdb                   |
| votingdb                 |
| world                    |
+--------------------------+
11 rows in set (0.00 sec)

mysql> use moviedb;
Database changed
mysql>
```

```
mysql> show tables;
+--------------------+
| Tables_in_moviedb  |
+--------------------+
| movies             |
| reviews            |
| tickets            |
| users              |
| vote_count         |
| votes              |
+--------------------+
6 rows in set (0.00 sec)

mysql>
```



```
1
2  import javax.swing.*;
9
10 // Main class to encapsulate all components
11 public class MovieTicketBookingApp extends JFrame {
12     private JTextField searchField;
13     private JTextArea movieDisplayArea;
14     private JButton bookTicketButton, addReviewButton;
15
16     public MovieTicketBookingApp() {
17         setTitle("Movie Ticket Booking System");
18         setSize(600, 600);
19         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         setLayout(new BorderLayout());
21
22         // Top Panel for Search
23         JPanel topPanel = new JPanel();
24         searchField = new JTextField(20);
25         JButton searchButton = new JButton("Search by Genre");
26         topPanel.add(searchField);
27         topPanel.add(searchButton);
28
29         // Display area for movie information
30         movieDisplayArea = new JTextArea(20, 40);
31         movieDisplayArea.setEditable(false);
32         JScrollPane scrollPane = new JScrollPane(movieDisplayArea);
33
34         // Button panel for actions
35         JPanel buttonPanel = new JPanel();
36         bookTicketButton = new JButton("Book Ticket");
37         addReviewButton = new JButton("Add/View Review");
```

Problems Javadoc Declaration Console Debug

<terminated> MovieTicketBookingApp [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (20 Nov 2024, 11:08:02 am – 11:08:40 am) [pid: 7260]

```
mysql> Desc movies;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| movie_id    | int          | NO   | PRI | NULL    | auto_increment |
| title       | varchar(100) | YES  |     | NULL    |                |
| genre       | varchar(50)  | YES  |     | NULL    |                |
| rating      | float        | YES  |     | NULL    |                |
| description | text         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

2. Backend (Java)

Now, let's implement the backend in Java, using JDBC to interact with the MySQL database.

Database Connection Class

This class handles the database connection.

3. Frontend (Java Swing)

MovieBookingApp Class (Main GUI)

Here we implement the main graphical interface for movie booking.

4. Run the Project

1. **Start MySQL Server** and run the SQL commands to create the database and tables.
2. **Compile and Run the Java Classes**:

    - Compile the Java classes: `javac *.java`
    - Run the `MovieBookingApp` class: `java MovieBookingApp`
3. **Test the Application**:
    - Search for movies by genre.
    - Book tickets.
    - Leave and view reviews for movies.

5. Future Enhancements

- Add validation checks (e.g., checking for available seats).
- Improve the GUI

CODE:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.*;

import java.util.ArrayList;

import java.util.List;

// Main class to encapsulate all components

public class MovieTicketBookingApp extends JFrame {

private JTextField searchField;

private JTextArea movieDisplayArea;

private JButton bookTicketButton, addReviewButton;

public MovieTicketBookingApp() {

setTitle("Movie Ticket Booking System");

setSize(600, 600);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

setLayout(new BorderLayout());

// Top Panel for Search
```

```
JPanel topPanel = new JPanel();

searchField = new JTextField(20);

JButton searchButton = new JButton("Search by Genre");

topPanel.add(searchField);

topPanel.add(searchButton);

// Display area for movie information

movieDisplayArea = new JTextArea(20, 40);

movieDisplayArea.setEditable(false);

JScrollPane scrollPane = newJScrollPane(movieDisplayArea);

// Button panel for actions

JPanel buttonPanel = new JPanel();

bookTicketButton = new JButton("Book Ticket");

addReviewButton = new JButton("Add/View Review");

buttonPanel.add(bookTicketButton);

buttonPanel.add(addReviewButton);

// Add panels to the frame

add(topPanel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

add(buttonPanel, BorderLayout.SOUTH);

// Event Listeners

searchButton.addActionListener(e -> searchMoviesByGenre());

addReviewButton.addActionListener(e -> addReview());

// Show the frame

setVisible(true);

}
```

```
// Search movies by genre
private void searchMoviesByGenre() {
String genre = searchField.getText();
List<Movie> movies = MovieService.getMoviesByGenre(genre);
movieDisplayArea.setText("");
for (Movie movie : movies) {
movieDisplayArea.append(movie.toString() + "\n");
}
}
// Adding a review for a movie
private void addReview() {
String movieIdInput = JOptionPane.showInputDialog("Enter Movie ID:");
String userIdInput = JOptionPane.showInputDialog("Enter User ID:");
String ratingInput = JOptionPane.showInputDialog("Enter Rating (1-5):");
String comment = JOptionPane.showInputDialog("Enter Review Comment:");
int movieId = Integer.parseInt(movieIdInput);
int userId = Integer.parseInt(userIdInput);
int rating = Integer.parseInt(ratingInput);
MovieService.addReview(userId, movieId, rating, comment);
JOptionPane.showMessageDialog(this, "Review Added Successfully!");
}
public static void main(String[] args) {
SwingUtilities.invokeLater(MovieTicketBookingApp::new);
}
}
```

```
// Movie Model class
class Movie {
private int movieId;
private String title;
private String genre;
private float rating;
private String description;
public Movie(int movieId, String title, String genre, floatrating, String description) {
this.movieId = movieId;
this.title = title;
this.genre = genre;
this.rating = rating;
this.description = description;
}
@Override
public String toString() {
return "ID: " + movieId + " | Title: " + title + " | Genre: " + genre + " | Rating: " + rating
+ "\nDescription: " + description;
}
}
// Movie Service class for database operations
class MovieService {
private static final String URL = "jdbc:mysql://localhost:3306/moviedb";
private static final String USER = "root";
```

```
private static final String PASSWORD = "Anusivaaakpraz1@"; // Replace with your actual password

// Get Connection

private static Connection getConnection() throwsSQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

// Fetch movies by genre

public static List<Movie> getMoviesByGenre(String genre) {

List<Movie> movies = new ArrayList<>();

String query = "SELECT * FROM Movies WHERE genre = ?";

try (Connection conn = getConnection();

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setString(1, genre);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

movies.add(new Movie(

rs.getInt("movie_id"),

rs.getString("title"),

rs.getString("genre"),

rs.getFloat("rating"),

rs.getString("description")));

}

} catch (SQLException e) {

e.printStackTrace();

}
```

```
return movies;

}

// Add review to a movie

public static void addReview(int userId, int movieId, intrating, String comment) {

String query = "INSERT INTO Reviews (user_id, movie_id, rating, comment, review_date)
VALUES (?, ?, ?, ?, NOW())";

try (Connection conn = getConnection();

PreparedStatement stmt = conn.prepareStatement(query)) {

stmt.setInt(1, userId);

stmt.setInt(2, movieId);

stmt.setInt(3, rating);

stmt.setString(4, comment);

stmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

}
```

Conclusion

This Movie Ticket Booking system, built using Java and MySQL, provides a functional prototype for users to book movie tickets, search movies by genre, and submit reviews. By following a step-by-step approach, we were able to:

1. **Set up the database**: Created and populated a MySQL database with tables for storing movie details, reviews, and bookings.
2. **Develop the backend**: Implemented Java classes with JDBC to interact with the database, enabling users to search movies, book tickets, and leave reviews.
3. **Create the frontend**: Used Java Swing to design a simple and interactive user interface for users to perform tasks like searching for movies by genre and booking tickets.

The project demonstrates key concepts such as:

- **Database connectivity** with JDBC.
- **CRUD operations** (Create, Read, Update, Delete) to interact with movie, review, and booking data.
- **GUI programming** using Java Swing for user interaction.

This mini-project serves as a solid foundation for more advanced ticketing systems, with potential extensions such as adding payment integration, user authentication, and dynamic seat reservations. It also highlights how Java and SQL can work together to create functional applications with real-world use cases.