**MOVIE TICKET BOOKING**

**INTRODUCTION**

The **Movie Ticket Booking System** is an automated system designed to facilitate the booking of movie tickets in a theatre efficiently. Traditionally, booking movie tickets involves long queues, manual record-keeping, and human error. This project eliminates these issues by providing a **graphical user interface (GUI)** for customers to select movies, show timings, dates, and seats, and allows the admin to view all bookings instantly.

This system ensures **real-time seat availability**, **booking confirmation**, and **dynamic updates** to the booking database. It also uses **Java Swing for GUI** and **MySQL for backend storage**, ensuring a robust and scalable solution.

**OBJECTIVES**

* To provide an **interactive interface** for booking movie tickets.
* To display **available and booked seats** in different colors for clarity.
* To allow customers to choose **movie, show time, date, and seats**.
* To maintain a **record of all bookings** in a database.
* To provide an **admin interface** to view all customer bookings.
* To reduce manual work and errors associated with ticket booking.

**SYSTEM REQUIREMENTS AND SETUP**

**Software Requirements**

* Minimum 4 GB RAM
* Processor: Intel i3 or above
* Hard Disk: 100 MB free space

**Hardware Requirements**

* Operating System: Windows 10 or above
* Java Development Kit (JDK) 1.8 or above
* MySQL 8.0 or above
* IDE: Eclipse
* MySQL Connector JAR for Java

**Setup Instructions**

1.Create MySQL Database:

**CREATE DATABASE movie\_booking\_db;**

**USE movie\_booking\_db;**

2. Create Bookings Table:

**CREATE TABLE IF NOT EXISTS bookings (**

**id INT AUTO\_INCREMENT PRIMARY KEY,**

**customer\_name VARCHAR(100) NOT NULL,**

**theatre VARCHAR(50) NOT NULL,**

**movie VARCHAR(100) NOT NULL,**

**show\_date DATE NOT NULL,**

**show\_time VARCHAR(20) NOT NULL,**

**seats VARCHAR(100) NOT NULL,**

**booking\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**

**);**

3.Import the table structure using the provided SQL script.

4.Add mysql-connector-java.jar to the project libraries.

5.Run the MovieServer.java first to start the server.

6.Run CustomerUI.java for booking tickets.

7.Run AdminUI.java to view the booked tickets.

8. Launch **MovieServer**, listening on **port 12345**.

**PROCEDURE**

1. **Customer Booking:**

* Open the Customer GUI.
* Enter customer name.
* Select **Movie**, **Date**, and **Show Time**.
* Available seats are shown in **green**, booked seats in **red**.
* Select desired seats (they turn **yellow**) and click **Confirm Booking**.
* Booking is sent to the server and stored in the MySQL database.

2. **Admin View:**

* Open Admin GUI.
* Click **Refresh Bookings** to see all bookings.
* The admin can view **customer name, movie, theatre, date, time, and seats**.

3. **Server Handling:**

* Server runs continuously, accepting client connections.
* Handles seat booking and updates the database in real-time.
* Sends booked seat information to Customer GUI to mark them as unavailable.

**FLOW DIAGRAM**

CUSTOMERUI

ADMINUI

MYSQL DATABASE

SERVER

**JDBC**

**CODE**

**1. model/Booking.java**

package model;

import java.io.Serializable;

import java.time.LocalDate;

public class Booking implements Serializable {

private String customerName;

private String theatre;

private String movie;

private String showTime;

private LocalDate showDate;

private String seats;

public Booking(String customerName, String theatre, String movie, String showTime, LocalDate showDate, String seats) {

this.customerName = customerName;

this.theatre = theatre;

this.movie = movie;

this.showTime = showTime;

this.showDate = showDate;

this.seats = seats;

}

public String getCustomerName() { return customerName; }

public String getTheatre() { return theatre; }

public String getMovie() { return movie; }

public String getShowTime() { return showTime; }

public LocalDate getShowDate() { return showDate; }

public String getSeats() { return seats; }

@Override

public String toString() {

return "Name: " + customerName + " | Theatre: " + theatre +

" | Movie: " + movie + " | Date: " + showDate +

" | Time: " + showTime + " | Seats: " + seats;

}

}

**2.database/DBConnection.java**

package database;

import java.sql.\*;

public class DBConnection {

private static Connection *con*;

public static Connection getConnection() {

try {

if (*con* == null || *con*.isClosed()) {

Class.*forName*("com.mysql.cj.jdbc.Driver");

*con* = DriverManager.*getConnection*(

"jdbc:mysql://localhost:3306/movie\_booking\_db",

"root",

"harsh@123456"

);

}

} catch (Exception e) {

e.printStackTrace();

}

return *con*;

}

}

**3.server/MovieServer.java**

package server;

import model.Booking;

import database.DBConnection;

import java.io.\*;

import java.net.\*;

import java.sql.\*;

import java.time.LocalDate;

import java.util.\*;

public class MovieServer {

private static final int *PORT* = 12345;

public static void main(String[] args) {

try (ServerSocket serverSocket = new ServerSocket(*PORT*)) {

System.*out*.println("🎬 Movie Booking Server running on port " + *PORT*);

while(true) {

Socket clientSocket = serverSocket.accept();

new Thread(new ClientHandler(clientSocket)).start();

}

} catch(Exception e) { e.printStackTrace(); }

}

}

class ClientHandler implements Runnable {

private Socket socket;

public ClientHandler(Socket socket) { this.socket = socket; }

@Override

public void run() {

try(

ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());

ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

) {

Object obj = ois.readObject();

if(obj instanceof Booking) {

Booking booking = (Booking) obj;

saveBookingToDB(booking);

oos.writeObject("✅ Booking Confirmed: " + booking.toString());

oos.flush();

} else if(obj instanceof String) {

String request = (String) obj;

if("ADMIN\_VIEW".equals(request)) {

List<Booking> bookings = getAllBookings();

oos.writeObject(bookings);

oos.flush();

} else if("GET\_BOOKED\_SEATS".equals(request)) {

// Receive details

String movie = (String) ois.readObject();

String time = (String) ois.readObject();

LocalDate date = (LocalDate) ois.readObject();

List<String> bookedSeats = getBookedSeats(movie, time, date);

oos.writeObject(bookedSeats);

oos.flush();

}

}

} catch(Exception e) { e.printStackTrace(); }

}

private void saveBookingToDB(Booking booking) {

try(Connection conn = DBConnection.*getConnection*()) {

String sql = "INSERT INTO bookings(customer\_name, theatre, movie, show\_time, show\_date, seats) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement ps = conn.prepareStatement(sql);

ps.setString(1, booking.getCustomerName());

ps.setString(2, booking.getTheatre());

ps.setString(3, booking.getMovie());

ps.setString(4, booking.getShowTime());

ps.setDate(5, java.sql.Date.*valueOf*(booking.getShowDate()));

ps.setString(6, booking.getSeats());

ps.executeUpdate();

ps.close();

} catch(SQLException ex) { ex.printStackTrace(); }

}

private List<Booking> getAllBookings() {

List<Booking> list = new ArrayList<>();

try(Connection conn = DBConnection.*getConnection*()) {

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM bookings");

while(rs.next()) {

Booking b = new Booking(

rs.getString("customer\_name"),

rs.getString("theatre"),

rs.getString("movie"),

rs.getString("show\_time"),

rs.getDate("show\_date").toLocalDate(),

rs.getString("seats")

);

list.add(b);

}

rs.close();

stmt.close();

} catch(SQLException ex) { ex.printStackTrace(); }

return list;

}

private List<String> getBookedSeats(String movie, String time, LocalDate date){

List<String> seats = new ArrayList<>();

try(Connection conn = DBConnection.*getConnection*()) {

String sql = "SELECT seats FROM bookings WHERE movie=? AND show\_time=? AND show\_date=?";

PreparedStatement ps = conn.prepareStatement(sql);

ps.setString(1, movie);

ps.setString(2, time);

ps.setDate(3, java.sql.Date.*valueOf*(date));

ResultSet rs = ps.executeQuery();

while(rs.next()) {

String[] booked = rs.getString("seats").split(",");

seats.addAll(Arrays.*asList*(booked));

}

rs.close();

ps.close();

} catch(SQLException ex) { ex.printStackTrace(); }

return seats;

}

}

**4. client/CustomerUI.java**

package client;

import model.Booking;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.time.LocalDate;

import java.util.List;

import java.util.Set;

import java.util.HashSet;

public class CustomerUI extends JFrame {

private JTextField nameField;

private JComboBox<String> movieList, timeList;

private JComboBox<LocalDate> dateList;

private JButton[][] seatButtons = new JButton[5][6];

private JButton confirmButton;

private JTextArea bookingDetails;

private Set<String> selectedSeats = new HashSet<>();

private Set<String> bookedSeats = new HashSet<>();

private final String theatreName = "Tamil";

public CustomerUI() {

setTitle("🎬 Movie Ticket Booking");

setSize(700, 600);

setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);

setLocationRelativeTo(null);

setLayout(new BorderLayout(10,10));

// Top Panel

JPanel topPanel = new JPanel(new GridLayout(4,2,10,10));

topPanel.add(new JLabel("Customer Name:"));

nameField = new JTextField();

topPanel.add(nameField);

topPanel.add(new JLabel("Select Movie:"));

String[] movies = {"Avengers", "Inception", "Interstellar", "Titanic"};

movieList = new JComboBox<>(movies);

topPanel.add(movieList);

topPanel.add(new JLabel("Select Time:"));

String[] times = {"10:00 AM","1:00 PM","4:00 PM","7:00 PM"};

timeList = new JComboBox<>(times);

topPanel.add(timeList);

topPanel.add(new JLabel("Select Date:"));

LocalDate today = LocalDate.*now*();

dateList = new JComboBox<>(new LocalDate[]{today,today.plusDays(1),today.plusDays(2),today.plusDays(3)});

topPanel.add(dateList);

add(topPanel, BorderLayout.*NORTH*);

// Seats Panel

JPanel seatPanel = new JPanel(new GridLayout(5,6,5,5));

char row = 'A';

for(int i=0;i<5;i++){

for(int j=0;j<6;j++){

String seatCode = row+String.*valueOf*(j+1);

JButton btn = new JButton(seatCode);

btn.setOpaque(true);

btn.setBorderPainted(false);

btn.setForeground(Color.*WHITE*);

btn.setBackground(Color.*GREEN*);

btn.addActionListener(e -> {

JButton b = (JButton) e.getSource();

String seat = b.getText();

if(bookedSeats.contains(seat)){

JOptionPane.*showMessageDialog*(this, "Seat " + seat + " already booked!");

} else if(selectedSeats.contains(seat)){

selectedSeats.remove(seat);

b.setBackground(Color.*GREEN*);

} else{

selectedSeats.add(seat);

b.setBackground(Color.*YELLOW*);

}

bookingDetails.setText("Selected Seats: " + selectedSeats);

});

seatButtons[i][j] = btn;

seatPanel.add(btn);

}

row++;

}

add(seatPanel, BorderLayout.*CENTER*);

// Bottom Panel

JPanel bottomPanel = new JPanel(new BorderLayout());

bookingDetails = new JTextArea(5,30);

bookingDetails.setEditable(false);

bottomPanel.add(new JScrollPane(bookingDetails), BorderLayout.*CENTER*);

confirmButton = new JButton("Confirm Booking");

bottomPanel.add(confirmButton, BorderLayout.*SOUTH*);

add(bottomPanel, BorderLayout.*SOUTH*);

// Listeners

confirmButton.addActionListener(e -> sendBookingToServer());

movieList.addActionListener(e -> loadBookedSeats());

timeList.addActionListener(e -> loadBookedSeats());

dateList.addActionListener(e -> loadBookedSeats());

loadBookedSeats(); // initial

}

private void loadBookedSeats() {

bookedSeats.clear();

selectedSeats.clear();

for(int i=0;i<5;i++){

for(int j=0;j<6;j++) seatButtons[i][j].setBackground(Color.*GREEN*);

}

try(Socket socket = new Socket("localhost",12345);

ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

ObjectInputStream ois = new ObjectInputStream(socket.getInputStream())){

oos.writeObject("GET\_BOOKED\_SEATS");

oos.flush();

oos.writeObject((String)movieList.getSelectedItem());

oos.writeObject((String)timeList.getSelectedItem());

oos.writeObject((LocalDate)dateList.getSelectedItem());

oos.flush();

List<String> booked = (List<String>) ois.readObject();

bookedSeats.addAll(booked);

for(int i=0;i<5;i++){

for(int j=0;j<6;j++){

String seat = seatButtons[i][j].getText();

if(bookedSeats.contains(seat)) seatButtons[i][j].setBackground(Color.*RED*);

}

}

} catch(Exception e){ JOptionPane.*showMessageDialog*(this, "Server Error: "+e.getMessage()); }

}

private void sendBookingToServer(){

String name = nameField.getText().trim();

if(name.isEmpty() || selectedSeats.isEmpty()){

JOptionPane.*showMessageDialog*(this, "Enter name and select seats!");

return;

}

Booking booking = new Booking(

name,

theatreName,

(String)movieList.getSelectedItem(),

(String)timeList.getSelectedItem(),

(LocalDate)dateList.getSelectedItem(),

String.*join*(",",selectedSeats)

);

try(Socket socket = new Socket("localhost",12345);

ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

ObjectInputStream ois = new ObjectInputStream(socket.getInputStream())){

oos.writeObject(booking);

oos.flush();

String response = (String)ois.readObject();

JOptionPane.*showMessageDialog*(this, response);

selectedSeats.clear();

bookingDetails.setText("");

nameField.setText("");

loadBookedSeats();

} catch(Exception e){ JOptionPane.*showMessageDialog*(this, "Server Error: "+e.getMessage()); }

}

public static void main(String[] args){

SwingUtilities.*invokeLater*(() -> new CustomerUI().setVisible(true));

}

}

**5.client/AdminUI.java**

package client;

import model.Booking;

import javax.swing.\*;

import java.awt.\*;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.util.List;

public class AdminUI extends JFrame {

private JTextArea bookingsArea;

private JButton refreshBtn;

public AdminUI(){

setTitle("🎟️ Admin - Booked Customers");

setSize(700,500);

setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);

setLocationRelativeTo(null);

setLayout(new BorderLayout(10,10));

bookingsArea = new JTextArea();

bookingsArea.setFont(new Font("Monospaced",Font.*PLAIN*,12));

bookingsArea.setEditable(false);

add(new JScrollPane(bookingsArea),BorderLayout.*CENTER*);

refreshBtn = new JButton("Refresh Bookings");

add(refreshBtn,BorderLayout.*SOUTH*);

refreshBtn.addActionListener(e -> loadBookings());

loadBookings();

}

private void loadBookings(){

try(Socket socket = new Socket("localhost",12345);

ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());

ObjectInputStream ois = new ObjectInputStream(socket.getInputStream())){

oos.writeObject("ADMIN\_VIEW");

oos.flush();

List<Booking> bookings = (List<Booking>) ois.readObject();

bookingsArea.setText(String.*format*("%-15s %-10s %-20s %-12s %-12s %-15s\n",

"Customer","Theatre","Movie","Date","Show Time","Seats"));

bookingsArea.append("-------------------------------------------------------------------------------\n");

for(Booking b : bookings){

bookingsArea.append(String.*format*("%-15s %-10s %-20s %-12s %-12s %-15s\n",

b.getCustomerName(),

b.getTheatre(),

b.getMovie(),

b.getShowDate(),

b.getShowTime(),

b.getSeats()));

}

} catch(Exception e){ JOptionPane.*showMessageDialog*(this,"Server Error: "+e.getMessage()); }

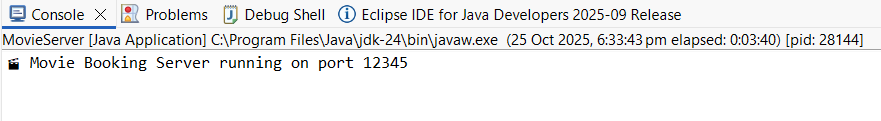
}

public static void main(String[] args){ SwingUtilities.*invokeLater*(() -> new AdminUI().setVisible(true)); }

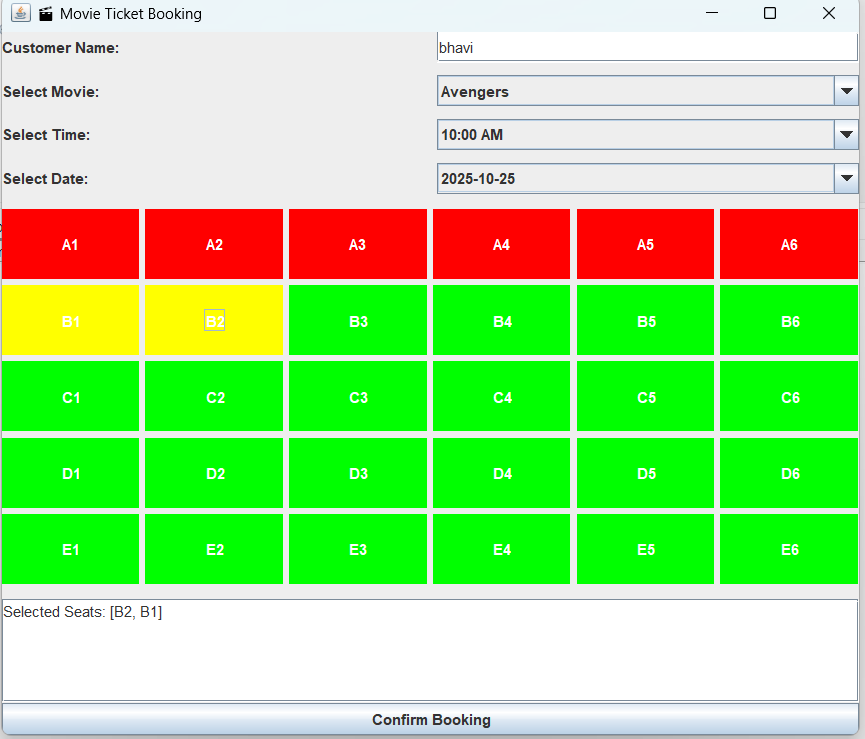
}

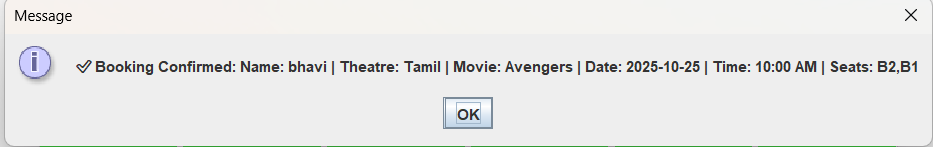
**SCREENSHOTS**

**1.server/MovieServer.java**

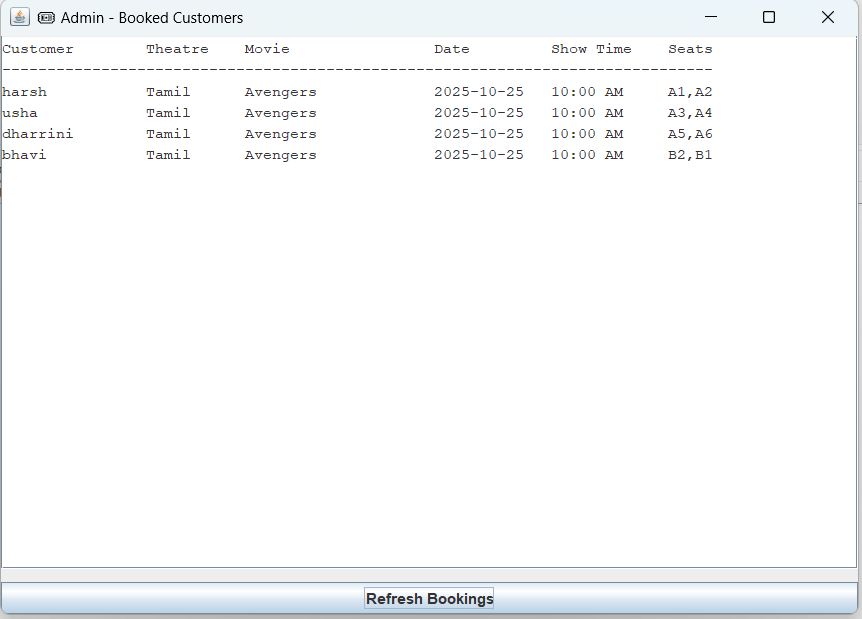


**2. client/CustomerUI.java**





**3.client/AdminUI.java**



**GITHUB REFERENCE**