MOVIE TICKET BOOKING SYSTEM

A MINI-PROJECT REPORT

Submitted by

NISHANTH B	**240701364**
NIMALAN M	**240701362**

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **“MOVIE TICKET BOOKING SYSTEM”** is the bonafide work of **“NISHANTH B, NIMALAN M, ”** who carried out the project work under my supervision.

SIGNATURE

Mrs. Deepa.B

ASSISTANT PROFESSOR SG

Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on

____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The *Movie Ticket Booking System* is a desktop-based application developed using **Java Swing** for the graphical interface and **MySQL** for the backend database. The system allows users to view available movies, showtimes, and book tickets conveniently. It also provides functionalities for viewing and canceling previously booked tickets. This application automates the manual ticket booking process by managing movie details, theater information, and user bookings efficiently. The project ensures ease of use, real-time seat availability updates, and secure database operations using JDBC.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. E.M. MALATHY** and our Deputy Head Of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Mrs. B. Deepa ,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. NISHANTH B

2. NIMALAN M

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1) INTRODUCTION

In the modern entertainment industry, online and computerized movie ticket booking has become essential to manage large volumes of users efficiently. Traditional manual booking systems often lead to issues like overbooking, data inconsistency, and lack of transparency. The *Movie Ticket Booking System* aims to overcome these challenges by automating the process through a user-friendly graphical interface and reliable database management.

The project provides an interactive platform where users can:

- Select a movie from the available list.
- View detailed showtime and theater information.
- Book, view, or cancel tickets easily.

This system ensures that data such as available seats, user bookings, and showtimes remain accurate and updated.

1.2) SCOPE OF THE WORK

The scope of this project includes:

- Developing a GUI-based system using Java Swing.
- Integrating with a MySQL database for persistent data storage.
- Providing real-time updates of seat availability and bookings.

- Allowing users to manage their bookings (view or cancel).
- Enabling administrators (optionally) to update movie or showtime information.

The system can be extended in the future to support online payment, user authentication, and admin dashboards.

1.3) PROBLEM STATEMENT

Manual or semi-automated ticket booking systems often result in:

- Inefficient data handling.
- Difficulty tracking available seats in real-time.
- Inconvenient user experience when booking or canceling tickets.
- Human errors and lack of synchronization between bookings and showtime data.

To overcome these issues, a digital ticket booking system is required that provides **accuracy**, **speed**, and **ease of use** through automation and database integration.

1.4) AIM AND OBJECTIVES OF THE PROJECT

Aim:
To design and implement an interactive desktop-based *Movie Ticket Booking System* that allows users to book, view, and cancel tickets efficiently.

Objectives:

- To provide an intuitive graphical user interface for movie ticket booking.
- To integrate a database for managing movies, showtimes, and user bookings.
- To ensure accurate seat availability tracking and real-time updates.

- To allow users to manage their tickets (book, view, cancel) securely.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	AMD Ryzen 5
Memory Size	:	8 GB
HDD	:	256 GB

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 11
Front – End	:	Java
Back - End	:	MySql
Language	:	Java, MySql

CHAPTER 3

MODULE DESCRIPTION

1.**Movie Management Module**

- Retrieves and displays the list of movies available from the database.
- Allows users to refresh and view updated movie lists dynamically.
- Connected to the Movies table in the database.

2. Showtime Management Module

- Displays showtime details for the selected movie, including theater, date, time, and available seats.
- Fetches real-time information from the Showtimes table.
- Ensures users can only book if enough seats are available.

3. Booking Module

- Allows users to input their name and number of seats to book.
- Inserts booking details into the Bookings table.
- Automatically updates available seat count in the Showtimes table.
- Implements **transaction control** (commit/rollback) to prevent inconsistent data during booking failures.

4. Ticket Display Module

- Displays all bookings made by a specific user.
- Fetches detailed information (movie, theater, date, time, seats) using SQL joins.
- Provides easy access for users to view their confirmed bookings.

5. Ticket Cancellation Module

- Enables users to cancel previously booked tickets.
- On cancellation, the corresponding booking record is removed from the Bookings table.
- Updates available seats in the Showtimes table accordingly.
- Maintains data consistency using database transactions.

CHAPTER 4

SAMPLE CODING

```
import javax.swing.*;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableModel;

import java.awt.*;

import java.awt.event.*;

import java.sql.*;

import java.util.Vector;

class MovieTicketBookingSystem extends JFrame {

   private Connection conn;

   private JComboBox<String> movieCombo;

   private JTable showtimeTable, ticketTable;

   private JTextField userField, seatField, ticketUserField;

   private JButton bookBtn, refreshBtn, showTicketsBtn, cancelTicketBtn;

   private DefaultTableModel showtimeModel, ticketModel;



   public MovieTicketBookingSystem() {
```

```
setTitle("Movie Ticket Booking System");

setSize(900, 600);

setDefaultCloseOperation(EXIT_ON_CLOSE);

setLocationRelativeTo(null);



try {

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

  conn = DriverManager.getConnection(

    "jdbc:mysql://localhost:3306/sample1", "root", "BNishanth060407$");

} catch (Exception e) {

  JOptionPane.showMessageDialog(this, "DB Connection Failed: " +
e.getMessage());

  System.exit(1);

}

JPanel main = new JPanel(new BorderLayout(10, 10));

main.setBorder(new EmptyBorder(10, 10, 10, 10));

add(main);
```

```
    JLabel title = new JLabel("REC Movie Booking System",
SwingConstants.CENTER);

    title.setFont(new Font("Segoe UI", Font.BOLD, 26));

    title.setOpaque(true);

    title.setBackground(new Color(70, 130, 180));

    title.setForeground(Color.WHITE);

    main.add(title, BorderLayout.NORTH);

    JPanel top = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 5));

    movieCombo = new JComboBox<>();

    refreshBtn = new JButton("Refresh");

    top.add(new JLabel("Select Movie:")); top.add(movieCombo);
top.add(refreshBtn);

    main.add(top, BorderLayout.BEFORE_FIRST_LINE);

    showtimeModel = new DefaultTableModel(

      new String[]{"ID", "Theater", "Date", "Time", "Seats"}, 0);

    showtimeTable = new JTable(showtimeModel);

    main.add(new JScrollPane(showtimeTable), BorderLayout.CENTER);

    JPanel bottom = new JPanel(new GridLayout(1, 2, 10, 10));

    main.add(bottom, BorderLayout.SOUTH);
    // Booking Panel

    JPanel book = new JPanel(new GridLayout(3, 2, 5, 5));

    book.setBorder(BorderFactory.createTitledBorder("Book Tickets"));

    userField = new JTextField(); seatField = new JTextField();

    bookBtn = new JButton("Book Ticket");

    book.add(new JLabel("Name:")); book.add(userField);

    book.add(new JLabel("Seats:")); book.add(seatField);

    book.add(new JLabel("")); book.add(bookBtn);

    bottom.add(book);


    // Ticket Panel

    JPanel ticket = new JPanel(new BorderLayout());

    ticket.setBorder(BorderFactory.createTitledBorder("Your Bookings"));

    JPanel tTop = new JPanel();

    ticketUserField = new JTextField(10);

    showTicketsBtn = new JButton("Show"); cancelTicketBtn = new
JButton("Cancel");

    tTop.add(new JLabel("User:")); tTop.add(ticketUserField);
    tTop.add(showTicketsBtn); tTop.add(cancelTicketBtn);

    ticket.add(tTop, BorderLayout.NORTH);

    ticketModel = new DefaultTableModel(

        new String[]{"Booking ID", "Movie", "Theater", "Date", "Time", "Seats"}, 0);

    ticketTable = new JTable(ticketModel);

    ticket.add(new JScrollPane(ticketTable), BorderLayout.CENTER);

    bottom.add(ticket);

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

    movieCombo.addActionListener(e -> loadShowtimes());

    bookBtn.addActionListener(e -> bookShow());

    showTicketsBtn.addActionListener(e -> loadTickets());

    cancelTicketBtn.addActionListener(e -> cancelTicket());

    loadMovies();

    setVisible(true);

}

private void loadMovies() {

    movieCombo.removeAllItems();
```

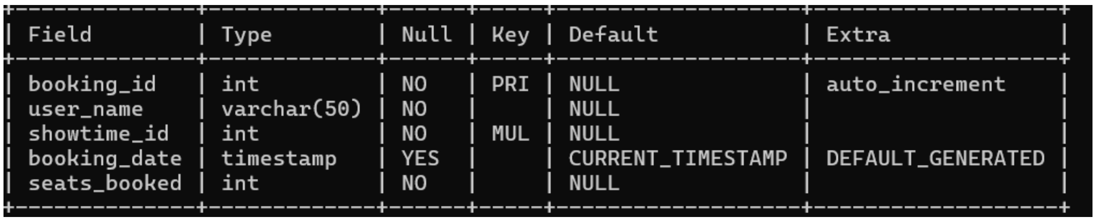
```
        try (PreparedStatement ps = conn.prepareStatement("SELECT title FROM
Movies")) {
            ResultSet rs = ps.executeQuery();
            while (rs.next()) movieCombo.addItem(rs.getString(1));
        } catch (Exception e) { JOptionPane.showMessageDialog(this, "Error: " +
e.getMessage()); }
    }

    private void loadShowtimes() {
        showtimeModel.setRowCount(0);
        String movie = (String) movieCombo.getSelectedItem();
        if (movie == null) return;
        try (PreparedStatement ps = conn.prepareStatement(
            "SELECT s.showtime_id, t.name, s.show_date, s.show_time,
s.available_seats " +
            "FROM Showtimes s JOIN Movies m ON s.movie_id=m.movie_id " +
            "JOIN Theaters t ON s.theater_id=t.theater_id WHERE m.title=?")) {
            ps.setString(1, movie);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) showtimeModel.addRow(new Object[]{
                rs.getInt(1), rs.getString(2), rs.getDate(3),
                rs.getTime(4), rs.getInt(5)});
        } catch (Exception e) { JOptionPane.showMessageDialog(this, "Load Error: " +
e.getMessage()); }
    }


    private void bookShow() {
        int row = showtimeTable.getSelectedRow();
        if (row < 0) return;
        String name = userField.getText().trim();
        int seats = Integer.parseInt(seatField.getText().trim());
        int available = (int) showtimeModel.getValueAt(row, 4);
        if (seats > available) { JOptionPane.showMessageDialog(this, "Not enough
seats!"); return; }
        int id = (int) showtimeModel.getValueAt(row, 0);


        try {
            conn.setAutoCommit(false);
            PreparedStatement ps1 = conn.prepareStatement(
                "INSERT INTO Bookings(user_name,showtime_id,seats_booked) VALUES(?,?,?)");
            ps1.setString(1, name); ps1.setInt(2, id); ps1.setInt(3, seats);
            ps1.executeUpdate();
            PreparedStatement ps2 = conn.prepareStatement(
                "UPDATE Showtimes SET available_seats=available_seats-? WHERE showtime_id=?");
            ps2.setInt(1, seats); ps2.setInt(2, id);
            ps2.executeUpdate();
            conn.commit();
            JOptionPane.showMessageDialog(this, "Booking successful!");
            loadShowtimes();
        } catch (Exception e) {
            try { conn.rollback(); } catch (Exception ignored) {}
            JOptionPane.showMessageDialog(this, "Booking failed: " + e.getMessage());
        }
    }
```

```
  private void loadTickets() {
    String user = ticketUserField.getText().trim();
    if (user.isEmpty()) return;
    ticketModel.setRowCount(0);
    try (PreparedStatement ps = conn.prepareStatement(
      "SELECT b.booking_id, m.title, t.name, s.show_date, s.show_time, b.seats_booked " +
      "FROM Bookings b JOIN Showtimes s ON b.showtime_id=s.showtime_id " +
      "JOIN Movies m ON s.movie_id=m.movie_id JOIN Theaters t ON s.theater_id=t.theater_id " +
      "WHERE b.user_name=?")) {
      ps.setString(1, user);
      ResultSet rs = ps.executeQuery();
      while (rs.next()) ticketModel.addRow(new Object[]{
        rs.getInt(1), rs.getString(2), rs.getString(3),
        rs.getDate(4), rs.getTime(5), rs.getInt(6)});
    } catch (Exception e) { JOptionPane.showMessageDialog(this, "Ticket load error"); }
  }
  private void cancelTicket() {
    int row = ticketTable.getSelectedRow();
    if (row < 0) return;
    int bookingId = (int) ticketModel.getValueAt(row, 0);
    int seats = (int) ticketModel.getValueAt(row, 5);
    try {
      conn.setAutoCommit(false);
      PreparedStatement ps = conn.prepareStatement(
        "SELECT showtime_id FROM Bookings WHERE booking_id=?");
      ps.setInt(1, bookingId);
      ResultSet rs = ps.executeQuery();
      if (!rs.next()) return;
      int showtimeId = rs.getInt(1);
      conn.prepareStatement("DELETE FROM Bookings WHERE booking_id=" +
bookingId).executeUpdate();
      PreparedStatement ps2 = conn.prepareStatement(
        "UPDATE Showtimes SET available_seats=available_seats+? WHERE
showtime_id=?");
```

```
            ps2.setInt(1, seats); ps2.setInt(2, showtimeId);

            ps2.executeUpdate();

            conn.commit();

            JOptionPane.showMessageDialog(this, "Booking cancelled.");

            loadTickets(); loadShowtimes();

        } catch (Exception e) {

            try { conn.rollback(); } catch (Exception ignored) {}

            JOptionPane.showMessageDialog(this, "Cancel failed: " + e.getMessage());

        }

    }



    public static void main(String[] args) {

        SwingUtilities.invokeLater(MovieTicketBookingSystem::new);

    }

}
```

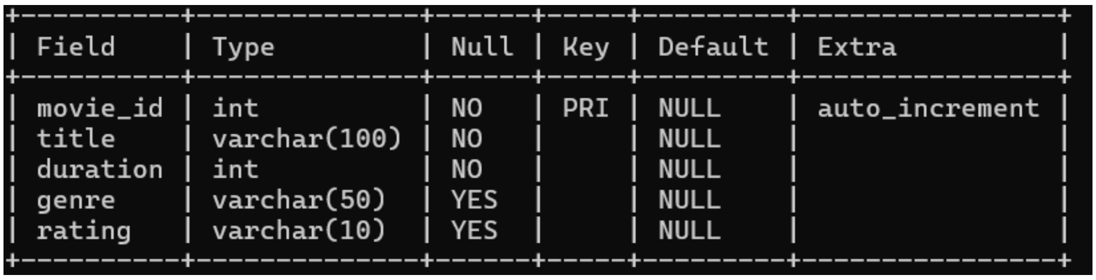
TABLES USED

1) Bookings:



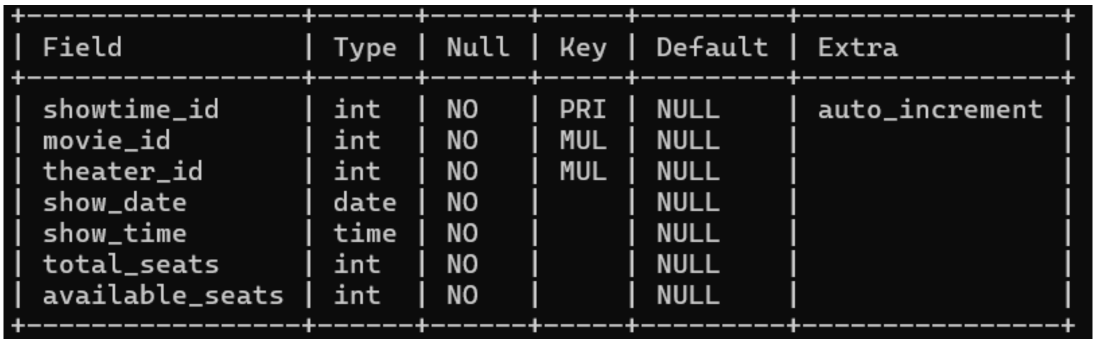
```
+--------------+-------------+------+-----+-------------------+-------------------+
| Field        | Type        | Null | Key | Default           | Extra             |
+--------------+-------------+------+-----+-------------------+-------------------+
| booking_id   | int         | NO   | PRI | NULL              | auto_increment    |
| user_name    | varchar(50) | NO   |     | NULL              |                   |
| showtime_id  | int         | NO   | MUL | NULL              |                   |
| booking_date | timestamp   | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| seats_booked | int         | NO   |     | NULL              |                   |
+--------------+-------------+------+-----+-------------------+-------------------+
```

2) Movies:



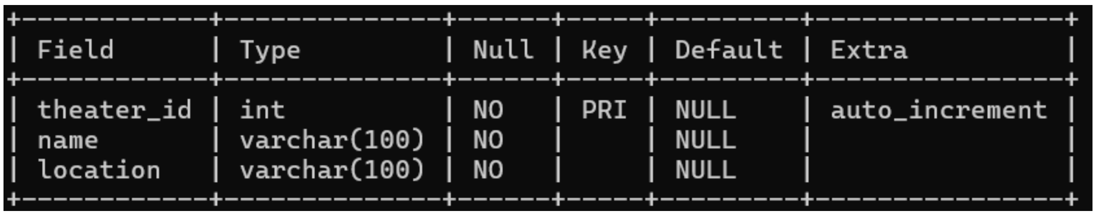
```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| movie_id | int          | NO   | PRI | NULL    | auto_increment |
| title    | varchar(100) | NO   |     | NULL    |                |
| duration | int          | NO   |     | NULL    |                |
| genre    | varchar(50)  | YES  |     | NULL    |                |
| rating   | varchar(10)  | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

3)Showtimes:



```
+-----------------+------+------+-----+---------+----------------+
| Field           | Type | Null | Key | Default | Extra          |
+-----------------+------+------+-----+---------+----------------+
| showtime_id     | int  | NO   | PRI | NULL    | auto_increment |
| movie_id        | int  | NO   | MUL | NULL    |                |
| theater_id      | int  | NO   | MUL | NULL    |                |
| show_date       | date | NO   |     | NULL    |                |
| show_time       | time | NO   |     | NULL    |                |
| total_seats     | int  | NO   |     | NULL    |                |
| available_seats | int  | NO   |     | NULL    |                |
+-----------------+------+------+-----+---------+----------------+
```

4)Theatres:

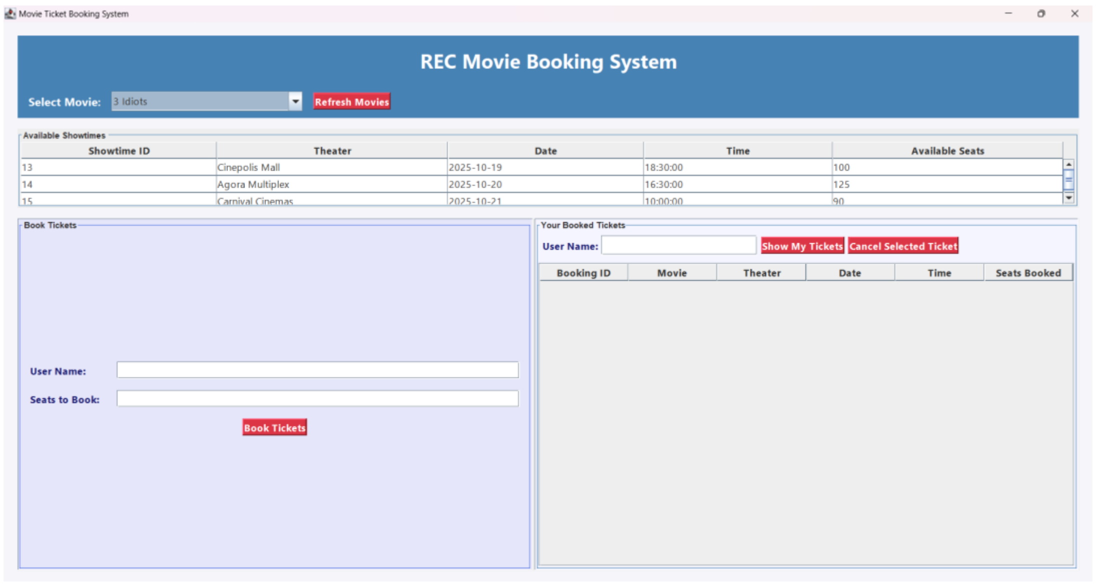


```
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| theater_id | int          | NO   | PRI | NULL    | auto_increment |
| name       | varchar(100) | NO   |     | NULL    |                |
| location   | varchar(100) | NO   |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
```

CHAPTER 5

SCREEN SHOTS

5.1) Introduction page

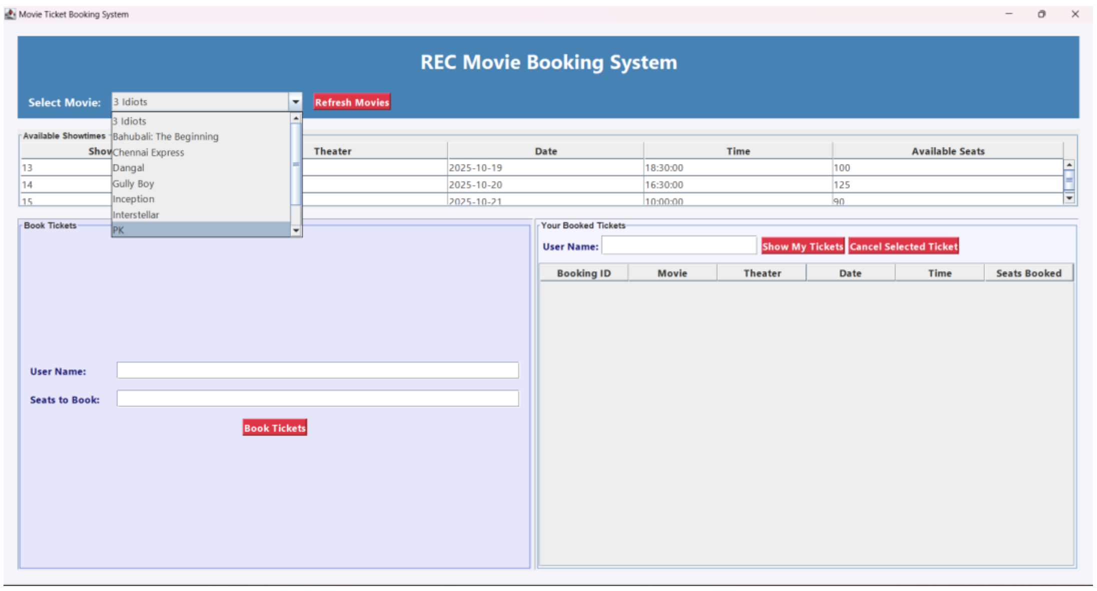


Available Showtimes

Showtime ID	Theater	Date	Time	Available Seats
13	Cinepolis Mall	2025-10-19	18:30:00	100
14	Agora Multiplex	2025-10-20	16:30:00	125
15	Carnival Cinemas	2025-10-21	10:00:00	90

Booking ID	Movie	Theater	Date	Time	Seats Booked

5.2) MOVIES AVAILABLE

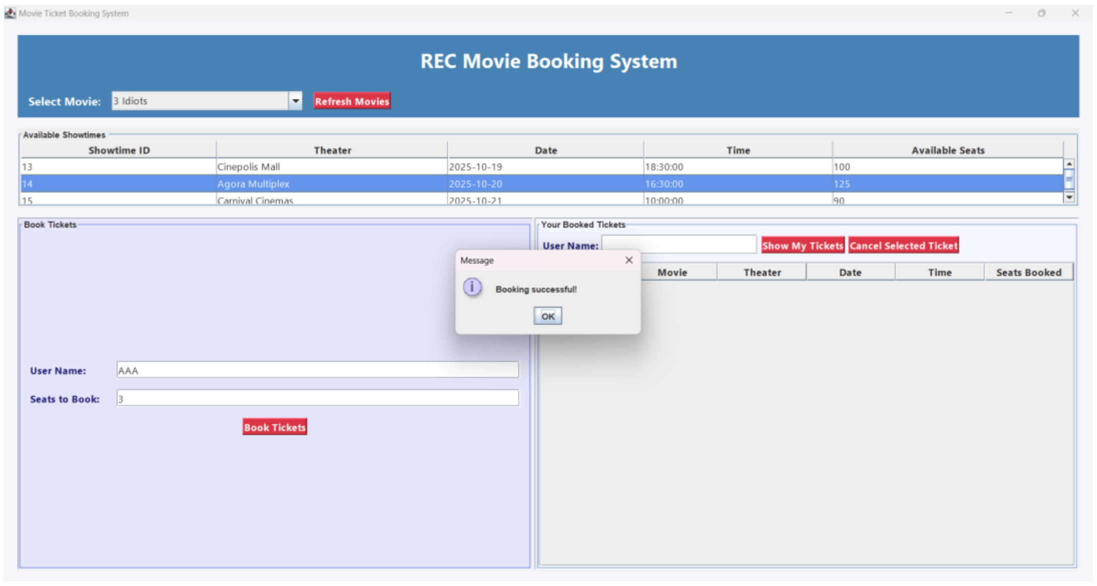


Available Showtimes

Show	Theater	Date	Time	Available Seats
13		2025-10-19	18:30:00	100
14		2025-10-20	16:30:00	125
15		2025-10-21	10:00:00	90

Booking ID	Movie	Theater	Date	Time	Seats Booked

5.3) BOOKING TICKETS



Showtime ID	Theater	Date	Time	Available Seats
13	Cinepolis Mall	2025-10-19	18:30:00	100
14	Agora Multiplex	2025-10-20	16:30:00	125
15	Carnival Cinemas	2025-10-21	10:00:00	90

Message

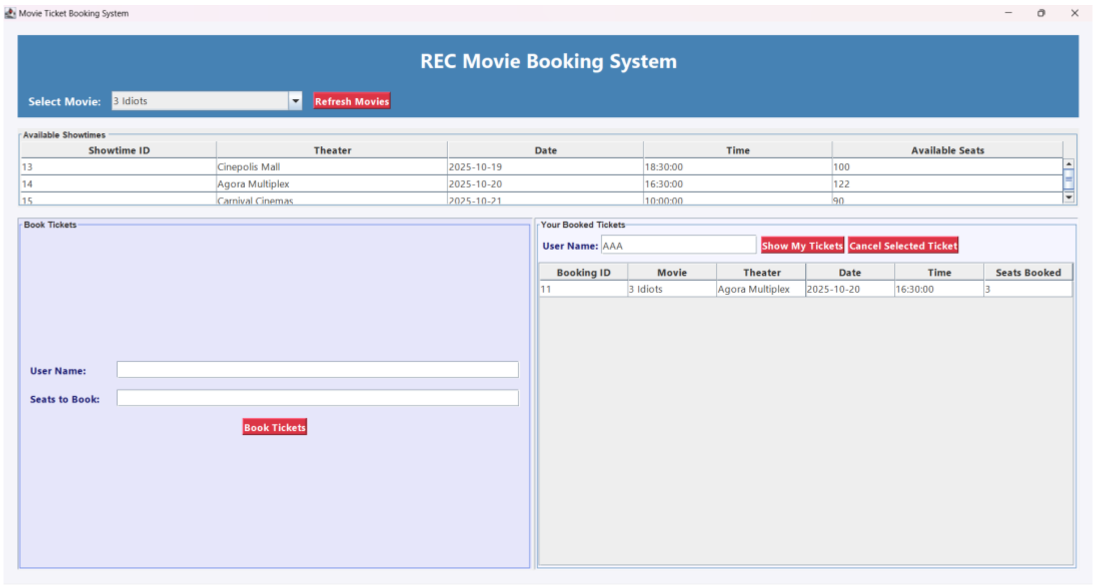
Booking successful!

OK

User Name: AAA

Seats to Book: 3

5.4) BOOKED TICKETS DISPLAYED

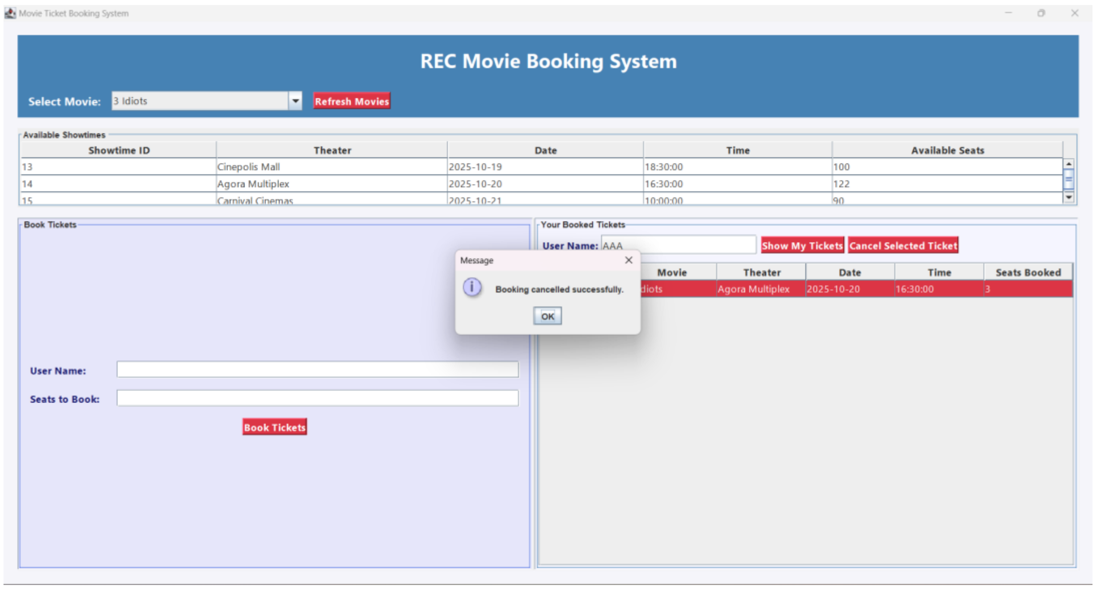


Showtime ID	Theater	Date	Time	Available Seats
13	Cinepolis Mall	2025-10-19	18:30:00	100
14	Agora Multiplex	2025-10-20	16:30:00	122
15	Carnival Cinemas	2025-10-21	10:00:00	90

User Name: AAA

Booking ID	Movie	Theater	Date	Time	Seats Booked
11	3 Idiots	Agora Multiplex	2025-10-20	16:30:00	3

5.5) CANCELLING TICKETS



Showtime ID	Theater	Date	Time	Available Seats
13	Cinepolis Mall	2025-10-19	18:30:00	100
14	Agora Multiplex	2025-10-20	16:30:00	122
15	Carnival Cinemas	2025-10-21	10:00:00	90

Booking cancelled successfully.

Movie	Theater	Date	Time	Seats Booked
diots	Agora Multiplex	2025-10-20	16:30:00	3

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The *Movie Ticket Booking System* successfully provides a digital platform for booking and managing movie tickets. It simplifies the traditional manual process by automating data retrieval, booking operations, and ticket management through an efficient GUI. The integration of **Java Swing** and **MySQL** ensures a smooth user experience and secure data handling.

The system ensures:

- Real-time seat updates.
- Error-free transactions.
- Convenient ticket management for users.

This project lays a strong foundation for further enhancements like online payment integration, seat mapping visualization, and admin-level management features, making it a scalable and practical solution for cinema booking systems.

CHAPTER 7 - REFERENCES

1. https://www.w3schools.com/sql/
2. https://www.wikipedia.org/
3. https://www.learnpython.org/
4. SUMITHA ARORA TEXTBOOK