

MOVIE TICKET BOOKING SYSTEM



A PROJECT REPORT

Submitted by

SHAMIRTHA J (2303811710421142)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**MOVIE TICKET BOOKING SYSTEM**” is the bona fide work of **SHAMIRTHA J (2303811710422142)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mr. A. MALARMANNAN A, M.E.,
ASSISTANT PROFESSOR

SIGNATURE

Mr. A. Malarmannan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on ...06/12/2024.....

CGB1201-JAVA PROGRAMMING
Mr. M. RAVANAN, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

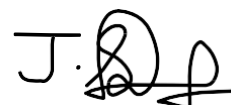
CGB1201-JAVA PROGRAMMING
Mr. R. KANAKHIK, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8138-SCE, TRICHY.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**MOVIE TICKET BOOKING SYSTEM**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

Signature



SHAMIRTHA J

Place: Samayapuram

Date: 06/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Movie Ticket Booking System is a Java-based graphical user interface (GUI) application that allows users to select movies, time slots, and seats for booking tickets. The system displays a list of available movies with different prices, and users can choose a time slot for the selected movie. A 5x5 grid represents the available seats, where users can select or deselect seats by clicking on them. The total cost of the tickets is dynamically calculated based on the number of seats selected and the movie's price. Once the user confirms their selection, a booking confirmation message is displayed. Additionally, the system includes functionality to reset the selections or exit the application. Using Java's AWT framework, the program handles user inputs through various components like labels, text fields, and buttons. The system also ensures that seats once booked cannot be selected again, providing a realistic simulation of a movie ticket booking process.

ABSTRACT WITH POs AND PSOs MAPPING**CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.**

ABSTRACT	POs MAPPED	PSOs MAPPED
The Movie Ticket Booking System is a Java GUI application that lets users select movies, time slots, and seats to book tickets. It calculates the total cost based on selected seats and movie price, tracks seat availability, and shows a booking confirmation. The system also allows users to reset selections or exit the application.	PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3	PSO1 -3 PSO2 -3 PSO3 -3

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	ix
1	INTRODUCTION	
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	1
2	PROJECT METHODOLOGY	
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	MODULE DESCRIPTION	
	3.1 User Interface (UI) Module	4
	3.2 Movie and Time Slot Selection Module	4
	3.3 Seat Selection Module	4
	3.4 Total Cost Calculation Module	4
	3.5 Booking Confirmation Module	4
	3.6 Reset and Exit Module	4
4	CONCLUSION & FUTURE SCOPE	
	4.1 Conclusion	5
	4.2 Future Scope	5
	APPENDIX A (SOURCE CODE)	7
	APPENDIX B (SCREENSHOTS)	12
	REFERENCES	13

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the **Movie Ticket Booking System** is to provide an interactive and user-friendly platform for booking movie tickets. The system aims to:

1. Enable users to select movies, time slots, and seats easily.
2. Calculate the total cost of tickets based on the movie and number of selected seats.
3. Display seat availability and visually indicate selected or booked seats.
4. Provide a booking confirmation upon successful ticket selection.
5. Allow users to reset their selections or exit the application.
6. Demonstrate the use of Java's AWT framework to create a functional GUI-based application.

1.2 Overview

The objective of the Movie Ticket Booking System is to develop a user-friendly and interactive application that allows users to easily book movie tickets. The system enables users to select movies, time slots, and seats, while dynamically calculating the total cost based on their choices. It provides a visual interface for seat selection, indicating available, selected, and booked seats. The system aims to ensure a smooth booking experience by allowing users to confirm their bookings, reset their selections, or exit the application. Additionally, it demonstrates the application of Java's AWT framework to create a functional graphical user interface for real-world ticket booking scenarios.

1.3 Java Programming Concepts

Object-Oriented Programming (OOPS) Concepts:

The Movie Ticket Booking System uses key OOP concepts in Java:

1. **Classes and Objects:** The system is structured using classes like Movie Ticket Booking System, with objects representing UI components (e.g., buttons, text fields).
2. **Encapsulation:** Internal details, like seat availability and pricing, are hidden, allowing users to interact with a clean interface.

3. Inheritance: The system extends Java classes like Frame for window creation and implements event listeners like ActionListener for button actions.
4. Polymorphism: Methods like action Performed are used for different actions but with different behaviors based on the button clicked.
5. Abstraction: Complex logic (e.g., seat booking and cost calculation) is hidden from the user, providing a simple interface for interaction.

These concepts ensure the system is modular, maintainable, and flexible.

Java Methods:

In the Movie Ticket Booking System, Java methods are used to perform specific tasks. The constructor initializes the GUI components and sets up event listeners. The action Performed method handles user actions like button clicks. The calculate Total () method computes the total ticket price, while book Tickets () simulates the booking process and shows a confirmation. The reset Seat Selection () method clears the seat selection, and update Seat Price () adjusts the ticket price based on the selected movie. These methods structure the program, ensuring modularity and efficient handling of user interactions.

Java Concepts:

In the Movie Ticket Booking System, several key Java concepts are used to build the application. Classes and objects define the structure of the program, with components like buttons and labels representing UI elements. Event handling is managed through ActionListener to respond to user actions like button clicks and seat selections. Arrays store seat availability, while control flow (if-else statements and loops) manages logic for seat selection and movie pricing. Exception handling ensures valid inputs, and methods are used to organize tasks like calculating costs and booking tickets. These concepts ensure a functional and efficient ticket booking system.

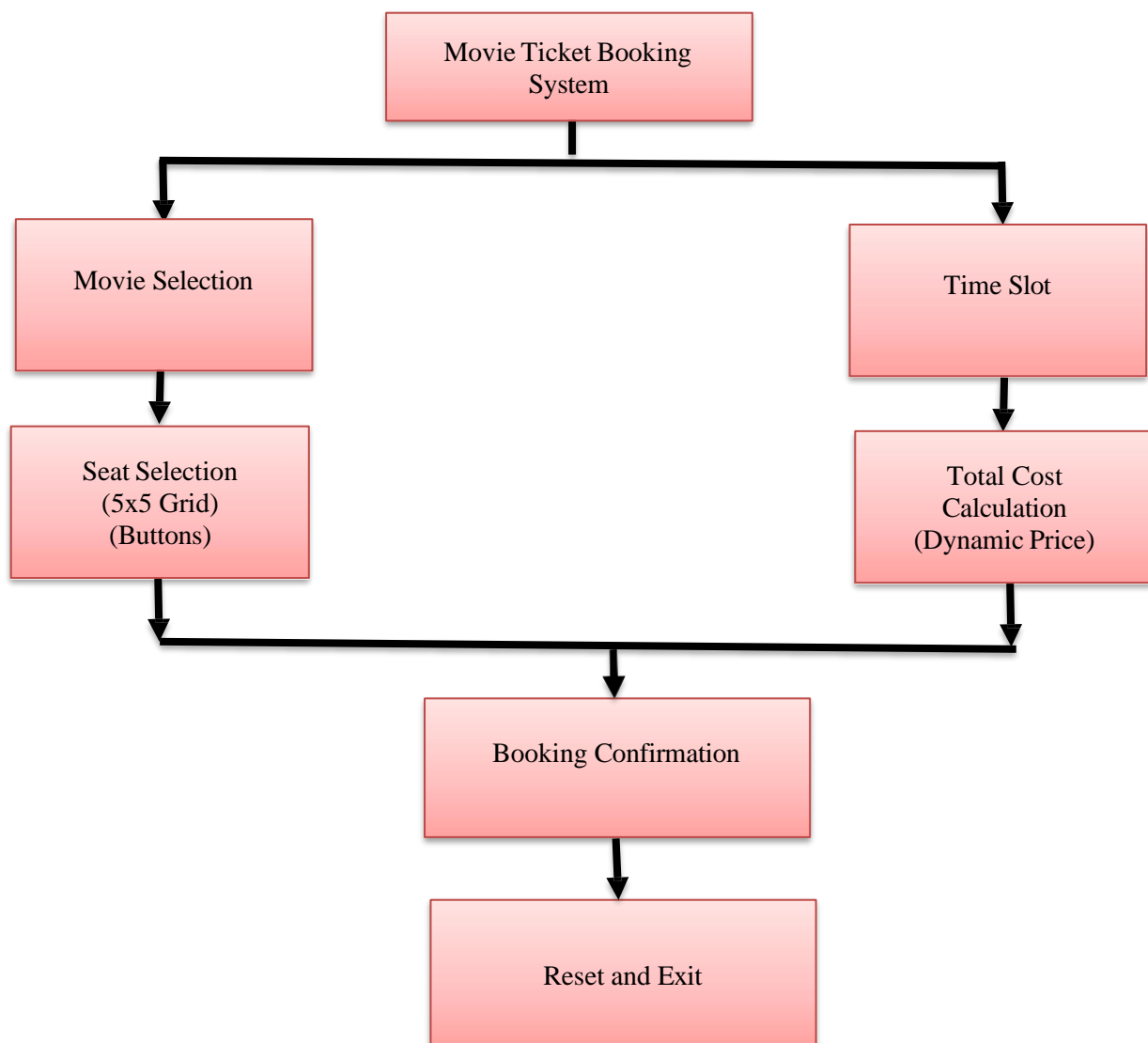
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

- Movie and Time Slot Selection
- Seat Selection
- Dynamic Pricing and Cost Calculation
- Booking Confirmation
- Reset and Exit Options

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Module 1: User Interface (UI) Module

The UI Module is responsible for constructing and displaying the graphical user interface (GUI) using Java's Abstract Window Toolkit (AWT). This module provides a visual representation of the Movie Ticket Booking System, allowing users to interact with the application. It includes key components like labels, buttons, text fields, drop-down choice lists, and panels for organizing the layout.

3.2 Module 2: Movie and Time Slot Selection Module

This module handles the selection of a movie and its corresponding time slot. The user selects a movie from a dropdown list, and the system retrieves the associated movie details, such as the name and price. Similarly, the user can choose a time slot from a list of available options, and the system records the choice.

3.3 Module 3: Seat Selection Module

The Seat Selection Module is responsible for displaying a grid of available seats and allowing the user to select or deselect them. Each seat is represented by a button that toggles between an "available" (green) state and a "selected" (red) state. Users can click on the buttons to select seats, and the total number of selected seats is updated dynamically.

3.4 Module 4: Total Cost Calculation Module

The Total Cost Calculation Module calculates the total cost of the movie tickets based on the selected movie, time slot, and number of seats. When the user presses the "Calculate" button, this module computes the cost by multiplying the number of selected seats with the price of the chosen movie. The total cost is displayed in the totalField text field.

3.5 Module 5: Booking Confirmation Module

The Booking Confirmation Module handles the final stage of the booking process. After the user has selected seats and calculated the total cost, this module simulates the booking process and displays a confirmation message with the details of the booking. It shows the movie name, time slot, number of tickets, and total price.

3.6Module 6: Reset and Exit Module

This module provides functionality to either reset the application or exit the program. The "Reset" button clears all selections, including movie choice, time slot, seat selection, ticket count, and total cost. The "Exit" button allows the user to close the application.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 Conclusion

The Movie Ticket Booking System successfully demonstrates how a simple Java application can be used to manage movie ticket bookings in an interactive and efficient manner. Through the use of core Java concepts like event handling, object-oriented programming, and GUI development with AWT, the system allows users to select a movie, choose a time slot, pick seats, and calculate the total cost seamlessly. Additionally, features like booking confirmation and seat reset enhance the user experience by providing clear feedback and the ability to start fresh if needed.

The modular design of the system ensures that each component, from seat selection to cost calculation, is easily manageable and can be extended or modified in the future. Overall, the project showcases the practical application of Java in building user-friendly, event-driven applications that simulate real-world scenarios effectively.

4.2 Future Scope

The Movie Ticket Booking System has significant potential for future enhancement. These improvements include dynamic pricing based on factors like movie popularity, time of day, and demand, as well as real-time seat availability to ensure no double bookings and support multiple users simultaneously. Integration with payment gateways will enable secure transactions, and user authentication could personalize the experience, storing booking history, preferences, and payment methods for convenience. Additionally, developing a mobile app version and supporting multiple theaters with different showtimes will increase the system's reach and usability. The introduction of advanced search filters, like genre or rating, will help users find movies more efficiently. Notifications for booking confirmations and reminders will keep users informed, while a rating and review system will allow users to share their feedback and help others make informed decisions. Furthermore, social media integration can enhance visibility and engagement by allowing users to share bookings and experiences with their networks.

APPENDIX A

(Project Source Code)

```
import java.awt.*;
import java.awt.event.*;

public class MovieTicketBookingSystem {

    // Declare components
    Frame frame;
    Label lblMovie, lblTickets, lblTime, lblSeats, lblTotal;
    Choice movieChoice, timeChoice;
    TextField ticketField, totalField;
    Button btnCalculate, btnBook, btnReset, btnExit;
    Button[][] seats; // 2D array to represent seats
    int selectedSeatsCount = 0;
    int seatPrice = 10; // Default price per seat (can be dynamic based on the movie)

    public MovieTicketBookingSystem() {
        // Create the frame
        frame = new Frame("Movie Ticket Booking System");

        // Create labels
        lblMovie = new Label("Select Movie:");
        lblTickets = new Label("Number of Tickets:");
        lblTime = new Label("Select Time Slot:");
        lblSeats = new Label("Select Seats:");
        lblTotal = new Label("Total Amount:");

        // Create choices (dropdowns) for movie selection and time
        movieChoice = new Choice();
        // Add movie names with their respective prices to the dropdown
        movieChoice.add("Movie 1: The Avengers - $100");
        movieChoice.add("Movie 2: Spider-Man - $150");
        movieChoice.add("Movie 3: Batman - $200");
        movieChoice.add("Movie 4: Iron Man - $250");
        movieChoice.add("Movie 5: Captain America - $300");

        timeChoice = new Choice();
        timeChoice.add("Morning (9:00 AM)");
        timeChoice.add("Afternoon (1:00 PM)");
        timeChoice.add("Evening (6:00 PM)");

        // Create text fields for ticket count and total amount
        ticketField = new TextField();
        totalField = new TextField();
        totalField.setEditable(false); // Total should not be editable

        // Create buttons for calculation, booking, reset, and exit
        btnCalculate = new Button("Calculate");
        btnBook = new Button("Book Tickets");
        btnReset = new Button("Reset");
        btnExit = new Button("Exit");
    }
}
```

```

// Create a panel for seat selection (grid layout)
Panel seatPanel = new Panel(new GridLayout(5, 5, 5, 5)); // 5x5 grid of seats
seats = new Button[5][5]; // Create a 5x5 array of buttons for the seats

// Initialize the seat buttons and add them to the seat panel
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
        seats[i][j] = new Button("Seat " + (i * 5 + j + 1));
        seats[i][j].setBackground(Color.GREEN); // Green indicates available
        seats[i][j].addActionListener(new SeatSelectionHandler(i, j));
        seatPanel.add(seats[i][j]);
    }
}

// Set Layout for the frame
frame.setLayout(new FlowLayout());

// Add components to the frame
frame.add(lblMovie);
frame.add(movieChoice);
frame.add(lblTickets);
frame.add(ticketField);
frame.add(lblTime);
frame.add(timeChoice);
frame.add(lblSeats);
frame.add(seatPanel);
frame.add(lblTotal);
frame.add(totalField);
frame.add(btnCalculate);
frame.add(btnBook);
frame.add(btnReset);
frame.add(btnExit);

// Set size and visibility of the frame
frame.setSize(500, 450);
frame.setVisible(true);

// Add event listeners
btnCalculate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        calculateTotal();
    }
});

btnBook.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        bookTickets();
    }
});

btnReset.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        resetSeatSelection();
    }
}

```

```

    });

    btnExit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.exit(0); // Exit the application
        }
    });

    // Close the frame when the user closes the window
    frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we) {
            System.exit(0);
        }
    });
}

// Method to handle seat selection/deselection
class SeatSelectionHandler implements ActionListener {
    int row, col;

    // Constructor
    SeatSelectionHandler(int row, int col) {
        this.row = row;
        this.col = col;
    }

    public void actionPerformed(ActionEvent e) {
        Button seatButton = seats[row][col];
        if (seatButton.getBackground() == Color.GREEN) {
            // If the seat is available, select it
            seatButton.setBackground(Color.RED); // Red indicates selected
            selectedSeatsCount++;
        } else {
            // If the seat is already selected, deselect it
            seatButton.setBackground(Color.GREEN); // Green indicates available
            selectedSeatsCount--;
        }
        // Update the number of tickets in the text field
        ticketField.setText(String.valueOf(selectedSeatsCount));
    }
}

// Method to calculate the total cost based on the selected seats
public void calculateTotal() {
    try {
        int ticketCount = Integer.parseInt(ticketField.getText());
        String selectedMovie = movieChoice.getSelectedItemId();
        int price = 0;

        // Extract price based on selected movie
        if (selectedMovie.contains("$100")) {
            price = 100;
        } else if (selectedMovie.contains("$150")) {
            price = 150;
        }
    }
}

```

```

    } else if (selectedMovie.contains("$200")) {
        price = 200;
    } else if (selectedMovie.contains("$250")) {
        price = 250;
    } else if (selectedMovie.contains("$300")) {
        price = 300;
    }

    // Calculate total
    int total = ticketCount * price;
    totalField.setText("$" + total);
} catch (NumberFormatException e) {
    totalField.setText("Invalid input");
}
}

// Method to simulate ticket booking
public void bookTickets() {
    String selectedMovie = movieChoice.getSelectedItemAt();
    String selectedTime = timeChoice.getSelectedItemAt();
    String tickets = ticketField.getText();
    String total = totalField.getText();

    if (tickets.isEmpty() || total.isEmpty() || total.equals("Invalid input")) {
        totalField.setText("Please fill in all fields!");
    } else {
        // Simulate booking by showing a confirmation message
        Dialog bookingConfirmation = new Dialog(frame, "Booking Confirmation", true);
        bookingConfirmation.setLayout(new FlowLayout());
        bookingConfirmation.setSize(300, 150);
        Label confirmationMessage = new Label("Booking Confirmed for " + selectedMovie +
" at " + selectedTime);
        bookingConfirmation.add(confirmationMessage);
        bookingConfirmation.setVisible(true);

        // Reset seat selection and ticket count
        resetSeatSelection();
    }
}

// Method to reset the seat selection
public void resetSeatSelection() {
    selectedSeatsCount = 0;
    ticketField.setText("");
    totalField.setText("");

    // Reset seat buttons to available
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (seats[i][j].getBackground() == Color.RED) {
                seats[i][j].setBackground(Color.GREEN); // Reset to available
            }
        }
    }
}

```

```
}  
  
public static void main(String[] args) {  
    new MovieTicketBookingSystem();  
}  
  
}
```

APPENDIX B (SCREENSHOTS)

Movie Ticket Booking System

Select Movie: Number of Tickets:

Select Time Slot: Select Seats:

Seat 1	Seat 2	Seat 3	Seat 4	Seat 5
Seat 6	Seat 7	Seat 8	Seat 9	Seat 10
Seat 11	Seat 12	Seat 13	Seat 14	Seat 15
Seat 16	Seat 17	Seat 18	Seat 19	Seat 20
Seat 21	Seat 22	Seat 23	Seat 24	Seat 25

Total Amount:

Movie Ticket Booking System

Select Movie: Number of Tickets:

Select Time Slot: Select Seats:

Seat 1	Seat 2	Seat 3	Seat 4	Seat 5
Seat 6	Seat 7	Seat 8	Seat 9	Seat 10
Seat 11	Seat 12	Seat 13	Seat 14	Seat 15
Seat 16	Seat 17	Seat 18	Seat 19	Seat 20
Seat 21	Seat 22	Seat 23	Seat 24	Seat 25

Total Amount:

Booking Confirmation

Booking Confirmed for Movie 3: Batman - \$200 at Afternoon (1:00 PM)

REFERENCES

1. Herbert Schildt, "Java: The Complete Reference" McGraw-Hill Education, 11th Edition, 2018. A comprehensive guide to Java, covering AWT and GUI development.
2. Ken Arnold, James Gosling, "The Java Programming Language" Addison-Wesley, 4th Edition, 2005. Detailed reference for Java programming and event-driven programming.
3. Joshua Bloch, "Effective Java" Addison-Wesley, 3rd Edition, 2018. Best practices and efficient coding strategies for Java developers.