MOVIE TICKET BOOKING SYSTEM

A PROJECT REPORT

Submitted by

DHIVYADHARSHINI J
(2303811710422031)

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 bonafide work of **DHIVYADHARSHINI J (2303811710422031)** 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

[illegible]

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. M. SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

[illegible]

SIGNATURE

Mr. M. Saravanan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024

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

[illegible]

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Dr.R.SENTHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.

[illegible]

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

[illegible]

DHIVYADHARSHINI J

Place: Samayapuram

Date: 02.12.2024

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. M. SARAVANAN, 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 comprehensive desktop application developed using Java's Abstract Window Toolkit (AWT) to automate and simplify the process of reserving movie tickets. The system addresses the inefficiencies of manual booking processes by providing an intuitive, user-friendly interface for users to select movies, choose show timings, view real-time seat availability, and book tickets. It integrates various core Java programming concepts such as event handling, data structures, and object-oriented programming to ensure a robust and efficient application.The system maintains movie data, including pricing, seat availability, and show timings, using a combination of HashMaps and arrays, allowing for dynamic updates and scalable performance. Users can easily check the price of tickets, enter the desired number of seats, and confirm their booking. The application features automatic calculations for ticket pricing based on the number of tickets selected, ensuring accurate cost estimation and reducing errors associated with manual calculations. Additionally, it dynamically updates seat availability after every booking, ensuring real-time tracking and preventing double-booking scenarios.The graphical user interface is designed with simplicity in mind, enabling users of all technical proficiencies to interact with the system effortlessly. Key components include dropdown menus for selecting movies and show timings, text fields for inputting user details, and buttons for booking and resetting inputs. The system provides immediate feedback on booking status, such as seat availability and total cost, through result messages displayed on the interface.This project also demonstrates the versatility of Java programming in developing real-world applications by incorporating modular design principles. The modularity makes it easier to maintain and expand the system to include additional features such as graphical seat selection, online payment gateways, and integration with theater management systems.

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 comprehensive desktop application built in Java, leveraging the AWT library to provide an intuitive graphical user interface for booking movie tickets. This program enables users to select a movie from a predefined list, including popular titles like Avatar and Avengers: Endgame, choose a show timing, and view the number of available seats. The system dynamically updates ticket prices and seat availability based on the selected movie and timing. Users can input their details, specify the number of tickets they wish to book, and instantly see the total cost.	**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 1

INTRODUCTION

1.1 Objective

The objective of the Movie Ticket Booking System program is to provide users with an efficient, user-friendly platform for reserving movie tickets in a seamless and hassle-free manner. This system is designed to simulate a real-world booking experience by incorporating essential features such as dynamic selection of movies, show timings, and seat availability, alongside automatic cost calculation based on the selected movie and number of tickets. It ensures data accuracy by validating user inputs and providing real-time updates about seat availability and pricing. The program uses intuitive graphical user interface (GUI) components to allow users to easily navigate through the booking process, offering clear visual feedback and descriptive messages for both successful bookings and error scenarios. Additionally, it supports functionality for resetting the booking interface to accommodate multiple users in quick succession. Overall, the system aims to streamline the ticket booking process, enhance user satisfaction, and replicate a functional and interactive movie reservation system.

1.2 Overview

The Movie Ticket Booking System is a user-centric Java application designed to simulate a functional ticket reservation platform for movie theaters. Built with a graphical user interface (GUI) using the AWT (Abstract Window Toolkit) framework, the program enables users to seamlessly select a movie, choose a preferred show timing, specify the number of tickets, and confirm their booking. It features a dynamic selection mechanism powered by data structures like HashMap to store movie details such as prices and show timings, as well as seat availability for each show. The application ensures real-time seat management by tracking and updating seat availability based on user bookings. Users can input their name, check the cost of tickets, and receive instant feedback on seat availability and booking confirmation. It also includes robust error handling to validate inputs, such as ensuring ticket counts are valid and sufficient seats are available. Additionally, a reset feature clears all inputs and allows new users to start fresh, making it suitable for continuous use in a multi-user environment.

1.3 Java Programming Concepts

The basic concepts of Object-Oriented Programming (OOP) are:

- **Encapsulation:** The program encapsulates data and methods inside the MovieTicketBookingSystem class.
- **Inheritance:** The program uses inheritance through the AWT framework classes like Frame, Label, Choice, and Button, which inherit common properties and methods from parent classes like Component and Container.
- **Polymorphism**: Method Overriding: The WindowAdapter class is extended, and the windowClosing() method is overridden to implement custom behavior for closing the window. This is an example of runtime polymorphism.
- **Encapsulation:** The program encapsulates data and methods inside the MovieTicketBookingSystem class. For instance, variables like moviePrices and movieSeats are kept private to the class and are manipulated through defined methods or event listeners.

Project related concepts

1. Event Handling

- The program responds to user actions, such as selecting a movie or show timing, clicking the "Book Tickets" button, or closing the application window.
- ActionListener and ItemListener are implemented to handle button clicks and item selections dynamically.

2. Error Handling

- Exception handling is used to manage invalid user inputs. For instance, a NumberFormatException is caught if the user enters non-numeric data in the "No. of Tickets" field.

3. Multi-Threading

- Though not explicitly coded, the AWT framework inherently supports event-driven programming, where events like button clicks and item selections are handled in separate threads managed by the framework.

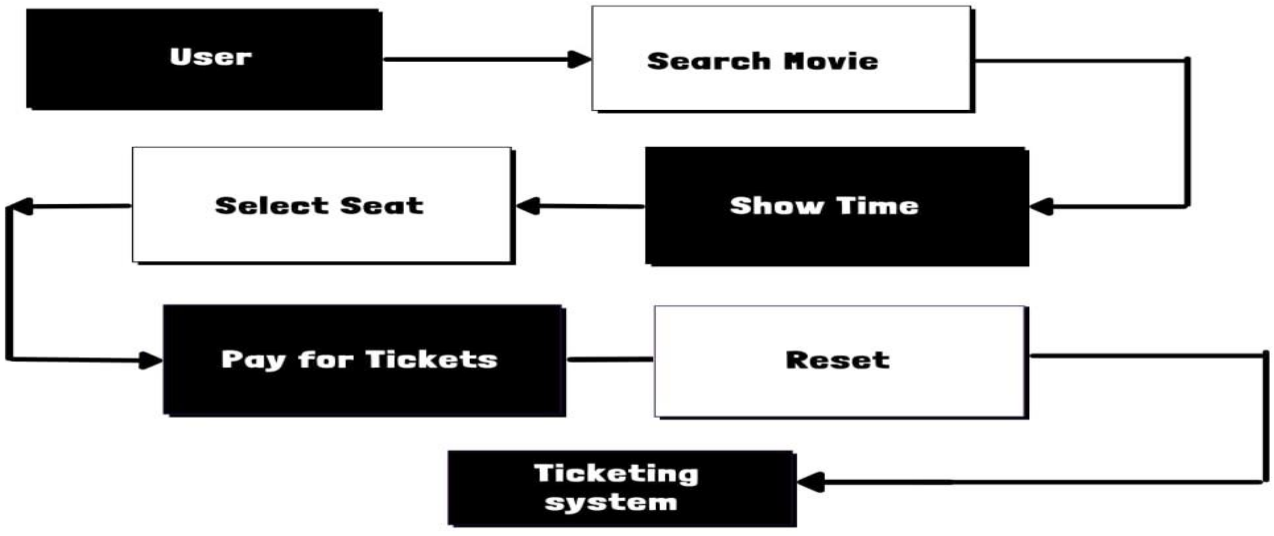
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work involves designing and implementing an interactive Movie Ticket Booking System in Java using the Abstract Window Toolkit (AWT) framework. This system simplifies the process of booking movie tickets by providing users with an intuitive graphical user interface (GUI) that integrates dynamic seat availability, ticket pricing, and show timings for various movies. The application features core functionalities, such as selecting a movie and show timing, entering user details (name and number of tickets), and automatically calculating the ticket cost based on the selected movie. Additionally, it keeps track of real-time seat availability, ensuring that users can only book tickets if sufficient seats are available for the chosen show. The system leverages Object-Oriented Programming (OOP) principles, including encapsulation for data storage and management, abstraction to hide internal complexities, inheritance for GUI components, and polymorphism for event handling and dynamic updates. The backend architecture employs a HashMap to store movie data, prices, and seat availability, organized by show timings.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 User Input Module

This module enables users to input essential booking details, such as their name, the desired movie, show timing, and the number of tickets. It employs text fields and dropdown menus (Choice components) for seamless data entry. The module ensures that all required fields are filled before proceeding with the booking process. Input validation is incorporated to check for valid ticket numbers and prevent errors caused by incomplete or incorrect data.

3.2 Movie Data Module

This module stores and manages the core data of the application, including movie names, ticket prices, and show timings, using HashMap data structures. It dynamically updates seat availability based on the selected movie and timing. The module also provides real-time pricing updates, ensuring users have accurate information about ticket costs for their chosen movie and show timing**.**

3.3 Seat Management Module

This module tracks seat availability using a nested HashMap and Boolean[] arrays, representing the seating layout for each movie and show timing. It efficiently updates the seat status during the booking process by marking reserved seats as unavailable. This module ensures that users always see the latest count of available seats and prevents overbooking by validating seat requests against current availability.

3.4 Booking Module

The booking module handles the core functionality of processing user inputs and confirming reservations. It validates input data, calculates the total ticket cost, and allocates seats by updating the seat availability data. Upon successful booking, it generates a confirmation message, including the user’s name, movie, show timing, booked seat numbers, and total cost. This module ensures accurate and efficient ticket booking.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the Movie Ticket Booking System represents a comprehensive and efficient implementation of a user-friendly ticket reservation platform built using Java's AWT framework. It effectively demonstrates the power of Object-Oriented Programming (OOP) principles such as encapsulation, abstraction, inheritance, and polymorphism, enabling modularity, code reusability, and ease of maintenance. The program combines a visually appealing graphical user interface (GUI) with robust backend logic to handle dynamic ticket pricing, seat availability management, and real-time updates, ensuring a seamless and intuitive booking experience. By leveraging data structures like nested HashMaps, it optimizes data storage and retrieval, allowing the system to manage multiple movies, show timings, and seat configurations efficiently. Furthermore, the system incorporates essential error-handling mechanisms to validate user inputs, prevent invalid operations, and provide clear feedback, enhancing reliability and user satisfaction. The application's modular design ensures that it is highly adaptable for future enhancements, such as integrating online payment systems, adding new movies and show timings, or implementing more advanced features like seat selection and reservation history.

4.2 FUTURE SCOPE

The Movie Ticket Booking System has immense potential for future expansion and enhancements, making it adaptable to meet evolving user demands and technological advancements. In the future, the system can be integrated with a secure online payment gateway to enable cashless transactions, making it more convenient for users. Advanced features like real-time seat selection via an interactive interface, support for multiple languages, and personalized recommendations based on user preferences could significantly enhance the user experience. The application could also be extended to support mobile platforms by integrating it with modern frameworks such as JavaFX or Android, making it accessible on smartphones and tablets. Additionally, incorporating database management systems like MySQL or MongoDB could enable efficient handling of large datasets, such as customer details, booking history, and real-time seat occupancy. Integration with APIs for email or SMS notifications could provide instant booking confirmations or reminders, improving communication.

REFERENCES

Java Books:

1. **"Java: The Complete Reference" by Herbert Schildt**

   Comprehensive coverage of core Java, including AWT and event handling.

2. **"Head First Java" by Kathy Sierra and Bert Bates**

   A beginner-friendly guide to Java programming with a focus on object-oriented Concepts.

Websites:

1. **GeeksforGeeks – JavaTutorials**

   - https://www.geeksforgeeks.org/java/
   - Tutorials and examples of Java programming, including GUI development.

2. **W3Schools - Java Tutorial**

   - URL: https://www.w3schools.com/java/
   - A beginner-friendly resource that offers tutorials on Java programming, including object-oriented principles and core Java concepts.

YouTube Links:

1. Java for Beginners - Java Brains

- URL: https://www.youtube.com/user/koushks
- Offers Java tutorials from the basics to advanced concepts. The channel provides detailed guides on Java programming, including working with objects and classes, which are crucial for building an EPMS.

APPENDIX A

(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.util.HashMap;
import java.util.Map;

public class MovieTicketBookingSystem {
    Frame frame;
    Label title, movieLabel, nameLabel, ticketsLabel, priceLabel, seatLabel, timeLabel,
resultLabel, availableSeatsLabel;
    Choice movieChoice, timeChoice;
    TextField nameField, ticketsField, priceField, availableSeatsField;
    Button bookButton, resetButton;
    // Map to store movie prices, seat availability, and show timings
    Map<String, Double> moviePrices;
    Map<String, Map<String, Boolean[]>> movieSeats;
    public MovieTicketBookingSystem() {
        // Initialize movie data
        moviePrices = new HashMap<>();
        moviePrices.put("Avatar", 300.0);
        moviePrices.put("Avengers: Endgame", 350.0);
        moviePrices.put("The Batman", 280.0);
        moviePrices.put("Spider-Man: No Way Home", 320.0);
        movieSeats = new HashMap<>();
        String[] showTimings = {"10:00 AM", "1:00 PM", "4:00 PM", "7:00 PM"};
        for (String movie : moviePrices.keySet()) {
            Map<String, Boolean[]> timingSeats = new HashMap<>();
            for (String timing : showTimings) {
                Boolean[] seats = new Boolean[100];
                for (int i = 0; i < seats.length; i++) {
                    seats[i] = true;
                }
                timingSeats.put(timing, seats);
    }
            movieSeats.put(movie, timingSeats);
        }
        // Frame setup
        frame = new Frame("Movie Ticket Booking System");
        frame.setLayout(null);
        frame.setSize(800, 700);
        // Title
        title = new Label("Book Your Movie Tickets");
        title.setFont(new Font("Arial", Font.BOLD, 16));
        title.setBounds(260, 40, 280, 30);
        frame.add(title);
        // Movie Label and Choice
        movieLabel = new Label("Select Movie:");
        movieLabel.setBounds(50, 100, 100, 30);
        frame.add(movieLabel);
        movieChoice = new Choice();
        for (String movie : moviePrices.keySet()) {
            movieChoice.add(movie);
        }
        movieChoice.setBounds(150, 100, 180, 30);
        frame.add(movieChoice);
        // Show Timing Label and Choice
        timeLabel = new Label("Show Timing:");
        timeLabel.setBounds(50, 150, 100, 30);
        frame.add(timeLabel);
        timeChoice = new Choice();
        for (String timing : showTimings) {
            timeChoice.add(timing);
        }
        timeChoice.setBounds(150, 150, 180, 30);
        frame.add(timeChoice);
        // Available Seats Label and TextField
        availableSeatsLabel = new Label("Seats Available:");
```

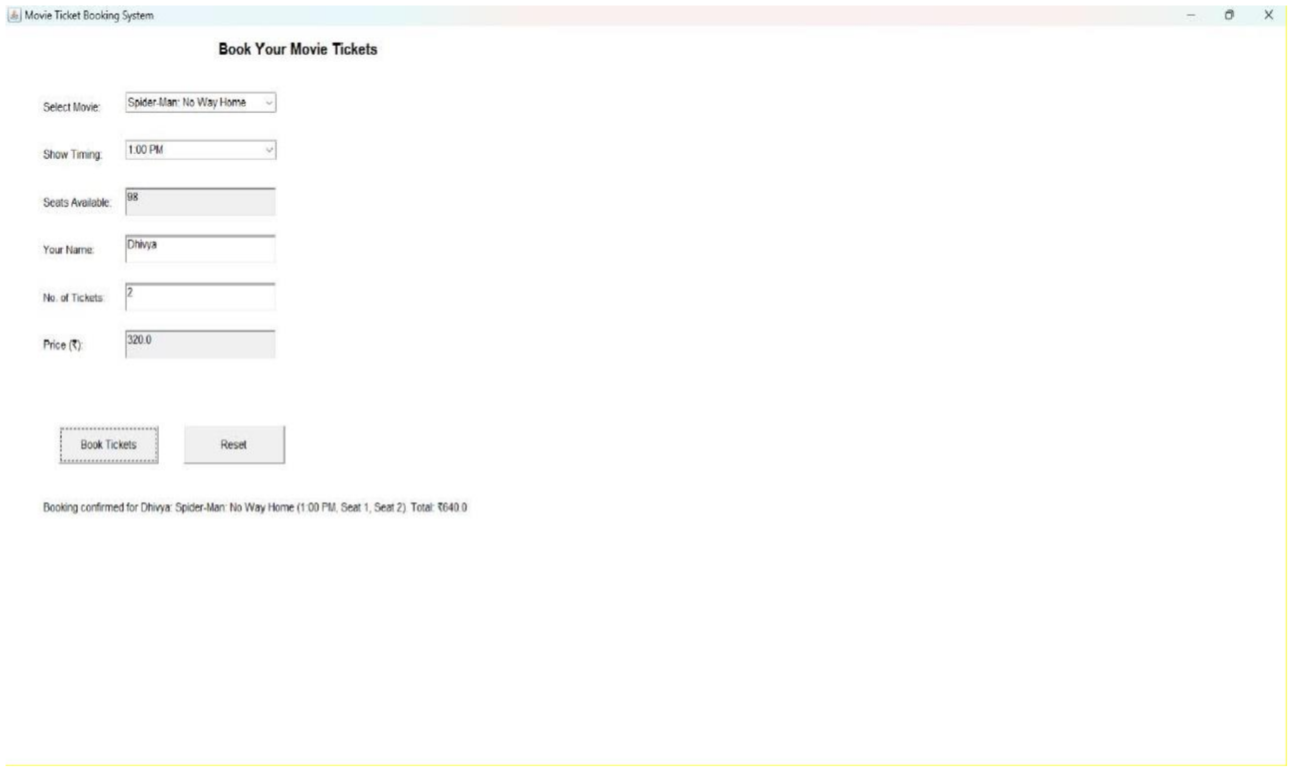
```
availableSeatsLabel.setBounds(50, 200, 100, 30);
frame.add(availableSeatsLabel);
availableSeatsField = new TextField();
availableSeatsField.setBounds(150, 200, 180, 30);
availableSeatsField.setEditable(false);
frame.add(availableSeatsField);
// Name Label and TextField
nameLabel = new Label("Your Name:");
nameLabel.setBounds(50, 250, 100, 30);
frame.add(nameLabel);
nameField = new TextField();
nameField.setBounds(150, 250, 180, 30);
frame.add(nameField);
// Tickets Label and TextField
ticketsLabel = new Label("No. of Tickets:");
ticketsLabel.setBounds(50, 300, 100, 30);
frame.add(ticketsLabel);
ticketsField = new TextField();
ticketsField.setBounds(150, 300, 180, 30);
frame.add(ticketsField);
// Ticket Price Label
priceLabel = new Label("Price (₹):");
priceLabel.setBounds(50, 350, 100, 30);
frame.add(priceLabel);
priceField = new TextField();
priceField.setBounds(150, 350, 180, 30);
priceField.setEditable(false);
frame.add(priceField);
// Book Button
bookButton = new Button("Book Tickets");
bookButton.setBounds(70, 450, 120, 40);
frame.add(bookButton);
// Reset Button
resetButton = new Button("Reset");
```

```
resetButton.setBounds(220, 450, 120, 40);
    frame.add(resetButton);
    // Result Label
    resultLabel = new Label("");
    resultLabel.setBounds(50, 520, 700, 30);
    frame.add(resultLabel);
    // Event Listeners
    ItemListener updateSeatsAndPrice = e -> {
        String movie = movieChoice.getSelectedItem();
        String timing = timeChoice.getSelectedItem();
        // Update ticket price
        priceField.setText(String.valueOf(moviePrices.get(movie)));
        // Update available seats
        Boolean[] seats = movieSeats.get(movie).get(timing);
        int availableSeats = 0;
        for (boolean seat : seats) {
            if (seat) {
                availableSeats++;
            }
        }
        availableSeatsField.setText(String.valueOf(availableSeats));
    };
    movieChoice.addItemListener(updateSeatsAndPrice);
    timeChoice.addItemListener(updateSeatsAndPrice);
    // Book Tickets Button Action
    bookButton.addActionListener(e -> {
        String movie = movieChoice.getSelectedItem();
        String timing = timeChoice.getSelectedItem();
        String name = nameField.getText();
        String tickets = ticketsField.getText();
        if (name.isEmpty() || tickets.isEmpty()) {
            resultLabel.setText("Please fill in all details.");
            return;
             try {
```

```
   int numTickets = Integer.parseInt(tickets);
                if (numTickets < 1) {
                    resultLabel.setText("You must book at least 1 ticket.");
                    return;
                }
                Boolean[] seats = movieSeats.get(movie).get(timing);
                int availableSeats = 0;
                for (boolean seat : seats) {
                    if (seat) {
                        availableSeats++;
                    }
                }
                if (numTickets > availableSeats) {
                    resultLabel.setText("Not enough seats available. Only " + availableSeats + " left.");
                    return;
                }
                // Book seats
                int bookedCount = 0;
                StringBuilder bookedSeats = new StringBuilder();
                for (int i = 0; i < seats.length && bookedCount < numTickets; i++) {
                    if (seats[i]) {
                        seats[i] = false;
                        bookedSeats.append("Seat ").append(i + 1).append(", ");
                        bookedCount++;
                    }
                }
                double totalCost = numTickets * moviePrices.get(movie);
                resultLabel.setText("Booking confirmed for " + name + ": " + movie +
                        " (" + timing + ", " + bookedSeats.substring(0, bookedSeats.length() - 2) + ").
Total: ₹" + totalCost);
                // Update available seats
                availableSeatsField.setText(String.valueOf(availableSeats - numTickets));
            } catch (NumberFormatException ex) {
                resultLabel.setText("Enter a valid number for tickets.");
            }
        });
        // Reset Button Action
        resetButton.addActionListener(e -> {
            nameField.setText("");
            ticketsField.setText("");
            priceField.setText("");
            availableSeatsField.setText("");
            resultLabel.setText("");
            movieChoice.select(0);
            timeChoice.select(0);
        });
        // Close Window Action
        frame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                frame.dispose();
            }
        });
        frame.setVisible(true);
    }
    public static void main(String[] args) {
        new MovieTicketBookingSystem();
    }
}
```

APPENDIX B

(SCREENSHOTS)



Book Your Movie Tickets

Select Movie:	Spider-Man: No Way Home
Show Timing:	1:00 PM
Seats Available:	98
Your Name:	Dhivya
No. of Tickets:	2
Price (₹):	320.0

Booking confirmed for Dhivya: Spider-Man: No Way Home (1:00 PM, Seat 1, Seat 2). Total: ₹640.0