PROJECT REPORT

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

– DATA SCIENCE WITH MACHINE LEARNING

By

S DHANUSHRAGAV

Registration No: 12207881

Section: K22UN

Roll No: A27

Supervisor

SHUBHAM SHARMA

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

October 2023

ACKNOWLEDGEMENT

I hereby declare that the work reported in the Assignment Project entitled "MOVIE TICKET BOOKING SYSTEM" in partial fulfilment of the requirement for the award of Degree for Bachelor of Technology in Computer Science and Engineering – Data Science with Machine Learning at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Shubham Sharma. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

S Dhanushragav

R. No: 12207881

TABLE OF CONTENTS

INTRODUCTION

The primary goal of this project was to design and implement a console-based movie ticket booking system using the Java programming language. The intention behind this endeavour was to create an intuitive, user-friendly interface that allows users to seamlessly interact with the system, providing them with the flexibility to choose movies, theatres, and specific seats for booking. The project aimed not only to fulfil these functional requirements but also to uphold essential software engineering principles such as modularity, simplicity, and educational value.

In the realm of software development, creating an interactive movie ticket booking system represents a quintessential challenge. It necessitates the harmonious integration of various programming concepts, ranging from object-oriented design principles to user input handling and data management. The significance of this project lies not only in its functionality but also in its potential educational value. By embracing best practices, employing modular coding techniques, and ensuring an intuitive user experience, the system can serve as a valuable learning resource for aspiring developers and enthusiasts alike.

OBJECTIVES AND SCOPE OF THE PROJECT

Objectives:

- Develop a user-friendly movie ticket booking system to simplify the process of buying tickets for movies.
- Implement a secure user authentication system to ensure that only authorized users can access the booking platform.
- Provide a diverse selection of movies for users to choose from, including details such as movie title, genre, and duration.
- Enable users to view the available seats for a selected movie and choose their preferred seating arrangement.
- Simulate a payment gateway to allow users to make payments securely and confirm their bookings.
- Create a booking history feature to track and display users' past movie bookings.
- Ensure the system's scalability, allowing for easy addition of new movies and seats in the future.

- Implement error handling and validation mechanisms to enhance the robustness of the system.
- Provide a seamless and intuitive user experience, optimizing the system's usability and accessibility.

Scope:

- The project encompasses the development of a Java-based movie ticket booking application.
- The system will support user registration and login functionalities, ensuring secure access to the platform.
- Users can browse a catalog of available movies, displaying essential details about each film.
- The application will provide real-time information on seat availability for each movie screening.
- Users will have the flexibility to choose specific seats based on their preferences and availability.
- The system will simulate a payment gateway, allowing users to confirm their bookings securely.
- A booking history feature will be implemented to store and display users' past bookings.
- The project focuses on the backend logic and does not include extensive user interface design.
- The application will handle errors gracefully, providing appropriate feedback to users in case of invalid inputs or failed transactions.
- Future enhancements, such as integrating real payment gateways, can be considered but are not within the current scope of the project.

APPLICATION TOOLS

In a typical software development project like a movie ticket booking system, various tools and technologies are used to facilitate the development, testing, and deployment processes. Here are some common application tools that could be used for developing the project:

Programming Language:

Java: Java is the chosen programming language for this project due to its simplicity, portability, and extensive libraries, making it well-suited for building robust backend systems.

Integrated Development Environment (IDE):

IntelliJ IDEA: A powerful and user-friendly Java IDE that offers intelligent coding assistance, robust debugging capabilities, and excellent support for Java development. It is widely used by Java developers and provides a seamless development experience.

Version Control:

Git: Git is a distributed version control system that allows multiple developers to collaborate on a project. It tracks changes, enables branching and merging, and ensures version history is maintained effectively.

GitHub: GitHub is a web-based hosting service for Git repositories. It provides a platform for collaborative software development, offering features like issue tracking, pull requests, and project management tools.

Using Java as the programming language, IntelliJ IDEA as the IDE, and Git/GitHub for version control, developers can efficiently collaborate, write code, and maintain version history throughout the movie ticket booking system project.

METHODOLOGY

1. Requirement Analysis:

Identified core features: User registration, user authentication, movie selection, seat selection, booking, and booking history management.

2. Development:

User Management: Implemented user registration and authentication logic. Created the User and UserManager classes for managing user data and authentication processes.

Movie and Seat Management: Developed classes for managing movie data (Movie class) and seat availability (Seat class). Implemented seat selection and booking logic (SeatSelection class).

Booking and History: Created the Booking class to represent a booking, and BookingHistory class to manage booking history. Implemented logic to record and display user bookings.

Input Validation: Implemented input validation mechanisms to handle user inputs securely. Ensured the application can handle various input scenarios without crashing.

3. Execution and Output:

User Interaction: Enabled user interaction through the console interface. Users can register, log in, select movies, choose seats, and view booking history.

User Feedback: Provided feedback to users regarding successful operations, invalid inputs, and booking status. Ensured clear communication with the users throughout the booking process.

Error Handling: Implemented robust error handling to gracefully manage unexpected inputs and errors. Ensured the application does not crash and provides helpful error messages.

5. Documentation:

Code Documentation: Documented code extensively using comments to ensure readability and understanding for future developers.

6. Testing and Validation:

Conducted testing to validate individual components' functionality, ensuring each class works as intended. And made user(classmates) to test, gathered feedback and made necessary adjustments..

FLOWCHART

1. **Start**

2. **User Registration**:
   - Get username and password from the user
   - Register user in the system

3. **User Login**:
   - Get username and password from the user
   - Validate user credentials
   - If valid, proceed to movie selection; else, end the program

4. **Movie Selection**:
   - Display available movies
   - Get user's choice of movie

5. **Seat Selection**:
   - Display available seats for the selected movie
   - Get user's choice of seat
   - If the seat is available, book the seat; else, ask for another choice

6. **Booking Confirmation**:
   - Confirm the booking
   - Store the booking details

7. **Display Booking History**:
   - Display user's booking history

8. **End**

Algorithm Implementation (Pseudocode):

1. **<u>User Registration</u>**:

```
function registerUser():
   input username
   input password
   create new User object with username and password
   add User object to UserManager
   display "User registered successfully"
```

2. **<u>User Login:</u>**

```
function loginUser():
   input username
   input password
   if UserManager.loginUser(username, password) is true:
      display "Login successful. Welcome, " + username + "!"
      return true
   else:
      display "Invalid username or password. Login failed."
      return false
```

3. **<u>Movie Selection:</u>**

```
function selectMovie():
   display "Available Movies:"
   for each movie in movies:
      display movie.title
   input selectedMovieIndex
   return movies[selectedMovieIndex - 1]
```

4. **<u>Seat Selection:</u>**

```
function selectSeat(seats):
   display "Available Seats:"
   for each seat in seats:
      if seat.isAvailable:
         display "Seat Number: " + seat.getSeatNumber() +
               ", Row: " + seat.getRow() +
               ", Column: " + seat.getColumn()
   input selectedSeatNumber
   if seats[selectedSeatNumber - 1].isAvailable:
      book the seat
      display "Seat booked successfully!"
      return seats[selectedSeatNumber - 1]
   else:
      display "Seat not available or invalid seat number."
      return null
```

5. **<u>Booking Confirmation:</u>**

```
function confirmBooking(username, selectedMovie, selectedSeat):
    create new Booking object with username, selectedMovie, and selectedSeat
    add Booking object to BookingHistory
```

6. **<u>Display Booking History:</u>**

```
function displayBookingHistory(bookingHistory):
    bookings = bookingHistory.getBookings()
    for each booking in bookings:
        display "Username: " + booking.getUsername()
        display "Movie: " + booking.getMovie().getTitle()
        display "Seat Number: " + booking.getSeat().getSeatNumber() +
                ", Row: " + booking.getSeat().getRow() +
                ", Column: " + booking.getSeat().getColumn()
        display "Booking Time: " + booking.getTimestamp()
```

SCREENSHOTS

MovieTicketBookingSystem.java

```java
import java.util.*;

class User {
    private final String username;
    private final String password;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }
}
```

MovieTicketBookingSystem.java

```java
class UserManager {
    3 usages
    private final Map<String, User> users;

    1 usage
    public UserManager() {
        users = new HashMap<>();
    }

    1 usage
    public void registerUser(String username, String password) {
        users.put(username, new User(username, password));
    }

    1 usage
    public boolean loginUser(String username, String password) {
        User user = users.get(username);
        return user != null && user.getPassword().equals(password);
    }
}
```

MovieTicketBookingSystem.java

```java
9 usages
class Movie {
    2 usages
    private final String title;

    3 usages
    public Movie(String title) {
        this.title = title;
    }

    2 usages
    public String getTitle() {
        return title;
    }
}
```

MovieTicketBookingSystem.java

```java
17 usages
class Seat {
    2 usages
    private final int seatNumber;
    2 usages
    private final int row;
    2 usages
    private final int column;
    3 usages
    private boolean isAvailable;

    8 usages
    public Seat(int seatNumber, int row, int column) {
        this.seatNumber = seatNumber;
        this.row = row;
        this.column = column;
        this.isAvailable = true;
    }

    3 usages
    public int getSeatNumber() {
        return seatNumber;
    }

    2 usages
    public int getRow() {
        return row;
    }
```

MovieTicketBookingSystem.java

```java
        return row;
    }

    2 usages
    public int getColumn() {
        return column;
    }

    2 usages
    public boolean isAvailable() {
        return isAvailable;
    }

    1 usage
    public void book() {
        isAvailable = false;
    }
}
```

MovieTicketBookingSystem.java

```java
2 usages
class SeatSelection {
    3 usages
    private final List<Seat> seats;

    1 usage
    public SeatSelection(List<Seat> seats) {
        this.seats = seats;
    }

    2 usages
    public void displayAvailableSeats() {
        System.out.println("\nAvailable Seats:");
        for (Seat seat : seats) {
            if (seat.isAvailable()) {
                System.out.println("Seat Number: " + seat.getSeatNumber() +
                        ", Row: " + seat.getRow() +
                        ", Column: " + seat.getColumn());
            }
        }
    }
```

MovieTicketBookingSystem.java

```java
    }

    1 usage
    public boolean bookSeat(int seatNumber) {
        for (Seat seat : seats) {
            if (seat.getSeatNumber() == seatNumber && seat.isAvailable()) {
                seat.book();
                System.out.println("Seat booked successfully!");
                return true;
            }
        }
        System.out.println("Seat not available or invalid seat number.");
        return false;
    }
}
```

MovieTicketBookingSystem.java

```java
class Booking {
    private final String username;
    private final Movie movie;
    private final Seat seat;
    private final long timestamp;

    public Booking(String username, Movie movie, Seat seat) {
        this.username = username;
        this.movie = movie;
        this.seat = seat;
        this.timestamp = System.currentTimeMillis();
    }

    public String getUsername() {
        return username;
    }

    public Movie getMovie() {
        return movie;
    }
```

MovieTicketBookingSystem.java

```java
    }

    public long getTimestamp() {
        return timestamp;
    }
}

class BookingHistory {
    private final List<Booking> bookings;

    public BookingHistory() {
        bookings = new ArrayList<>();
    }

    public void addBooking(String username, Movie movie, Seat seat) {
        bookings.add(new Booking(username, movie, seat));
    }

    public List<Booking> getBookings() {
        return bookings;
    }
}
```

MovieTicketBookingSystem.java

```java
}

public class MovieTicketBookingSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // SignUp
        System.out.print("SignUp to start your booking --->");
        UserManager userManager = new UserManager();
        System.out.print("\nEnter username: ");
        String username = scanner.nextLine();

        System.out.print("Enter password: ");
        String password = scanner.nextLine();
        userManager.registerUser(username, password);
        System.out.println("User registered successfully.");

        // Login
        System.out.println();
        System.out.println("Login to your account --->");
        System.out.print("Enter your username to login: ");
        String loginUsername = scanner.nextLine();
        System.out.print("Enter your password to login: ");
        String loginPassword = scanner.nextLine();

        if (userManager.loginUser(loginUsername, loginPassword)) {
            System.out.println("Login successful. Welcome, " + loginUsername + "!");
        } else {
            System.out.println("Invalid username or password. Login failed.");
```

MovieTicketBookingSystem.java

```java
        if (userManager.loginUser(loginUsername, loginPassword)) {
            System.out.println("Login successful. Welcome, " + loginUsername + "!");
        } else {
            System.out.println("Invalid username or password. Login failed.");
            System.exit( status: 0);
        }

        // Available Movies
        List<Movie> movies = new ArrayList<>();
        movies.add(new Movie( title: "Jailer"));
        movies.add(new Movie( title: "Leo"));
        movies.add(new Movie( title: "Pathaan"));

        // Movie Selection
        System.out.println("\nAvailable Movies:");
        for (int i = 0; i < movies.size(); i++) {
            System.out.println((i + 1) + ". " + movies.get(i).getTitle());
        }
        System.out.print("Enter the number of the movie you want to watch: ");
        int selectedMovieIndex = scanner.nextInt();
        Movie selectedMovie = movies.get(selectedMovieIndex - 1);

        // Seat Selection
        List<Seat> seats = new ArrayList<>();
        // Create seats and add them to the list
        System.out.println("\nRow 1 Starts from the Screen.");
        seats.add(new Seat( seatNumber: 1,  row: 1,  column: 1));
        seats.add(new Seat( seatNumber: 2,  row: 1,  column: 2));
```

MovieTicketBookingSystem.java

```java
            seats.add(new Seat( seatNumber: 1,  row: 1,  column: 1));
            seats.add(new Seat( seatNumber: 2,  row: 1,  column: 2));
            seats.add(new Seat( seatNumber: 3,  row: 2,  column: 2));
            seats.add(new Seat( seatNumber: 4,  row: 2,  column: 3));
            seats.add(new Seat( seatNumber: 5,  row: 3,  column: 3));
            seats.add(new Seat( seatNumber: 6,  row: 3,  column: 4));
            seats.add(new Seat( seatNumber: 7,  row: 4,  column: 1));
            seats.add(new Seat( seatNumber: 8,  row: 4,  column: 4));

            SeatSelection seatSelection = new SeatSelection(seats);

            // Display available seats
            seatSelection.displayAvailableSeats();

            // Get user input for seat selection
            System.out.print("Enter the seat number you want to book: ");
            int selectedSeatNumber = scanner.nextInt();

            // Book the selected seat
            seatSelection.bookSeat(selectedSeatNumber);

            // Add the booking to the booking history
            BookingHistory bookingHistory = new BookingHistory();
            bookingHistory.addBooking(loginUsername, selectedMovie, seats.get(selectedSeatNu

            // Display updated available seats
            seatSelection.displayAvailableSeats();
```

MovieTicketBookingSystem.java

```java
            // Add the booking to the booking history
            BookingHistory bookingHistory = new BookingHistory();
            bookingHistory.addBooking(loginUsername, selectedMovie, seats.get(selectedSeatNu

            // Display updated available seats
            seatSelection.displayAvailableSeats();

            // Display Booking History
            System.out.println("\nBooking History:");
            List<Booking> userBookings = bookingHistory.getBookings();
            for (Booking userBooking : userBookings) {
                System.out.println("Username: " + userBooking.getUsername());
                System.out.println("Movie: " + userBooking.getMovie().getTitle());
                System.out.println("Seat Number: " + userBooking.getSeat().getSeatNumber() +
                        ", Row: " + userBooking.getSeat().getRow() +
                        ", Column: " + userBooking.getSeat().getColumn());
                System.out.println("Booking Time: " + userBooking.getTimestamp());
                System.out.println();
            }
        }
    }
```

```
Project
MovieTicketBookingSystem.java
Run
MovieTicketBookingSystem
C:\Users\erdha\.jdks\openjdk-21\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.2\lib\idea_rt.jar=59744
SignUp to start your booking --->
Enter username: Dhanush
Enter password: Dha123
User registered successfully.

Login to your account --->
Enter your username to login: Dhanush
Enter your password to login: Dha123
Login successful. Welcome, Dhanush!

Available Movies:
1. Jailer
2. Leo
3. Pathaan
Enter the number of the movie you want to watch: 2

Row 1 Starts from the Screen.

Available Seats:
Seat Number: 1, Row: 1, Column: 1
Seat Number: 2, Row: 1, Column: 2
Seat Number: 3, Row: 2, Column: 2
Seat Number: 4, Row: 2, Column: 3
Seat Number: 5, Row: 3, Column: 3
Seat Number: 6, Row: 3, Column: 4
```

```
Project
MovieTicketBookingSystem.java
Run
MovieTicketBookingSystem
Seat Number: 4, Row: 2, Column: 3
Seat Number: 5, Row: 3, Column: 3
Seat Number: 6, Row: 3, Column: 4
Seat Number: 7, Row: 4, Column: 1
Seat Number: 8, Row: 4, Column: 4
Enter the seat number you want to book: 7
Seat booked successfully!

Available Seats:
Seat Number: 1, Row: 1, Column: 1
Seat Number: 2, Row: 1, Column: 2
Seat Number: 3, Row: 2, Column: 2
Seat Number: 4, Row: 2, Column: 3
Seat Number: 5, Row: 3, Column: 3
Seat Number: 6, Row: 3, Column: 4
Seat Number: 8, Row: 4, Column: 4

Booking History:
Username: Dhanush
Movie: Leo
Seat Number: 7, Row: 4, Column: 1
Booking Time: 1697980973621


Process finished with exit code 0
```

GITHUB LINK

https://github.com/S-Dhanushragav/Movie-Ticket-Booking-System

SUMMARY

The Movie Ticket Booking System project is a Java-based console application designed to streamline the process of booking movie tickets. This system focuses on providing users with a seamless and secure experience, enabling them to register, log in, choose movies, select seats, confirm bookings, and view their booking history. The project emphasizes modularity, user interaction, and data management, showcasing fundamental principles of object-oriented programming and user-centric design.

Key Features and Functionalities:

User Management: Implemented user registration and authentication functionalities, ensuring secure access to the system.

Movie Selection: Users can browse available movies, selecting their preferred film for booking.

Seat Selection: Users can view and select available seats for the chosen movie, with real-time updates on seat availability.

Booking and Confirmation: Integrated logic for seat booking, enabling users to confirm their selections securely.

Booking History: Implemented a booking history feature, allowing users to review their past bookings.

Highlights:

Modular Design: The project exhibits a well-organized and modular structure, emphasizing the principles of object-oriented programming, making it easily extendable for future enhancements.

User Interaction: The console interface provides a user-friendly experience, guiding users through the booking process with clear instructions and feedback.

Error Handling: Robust error handling mechanisms are in place, ensuring the system gracefully handles invalid inputs and unexpected scenarios, providing informative messages to users.

Conclusion and Future Enhancements:

The Movie Ticket Booking System project successfully demonstrates core functionalities essential for an online ticket booking platform. The project lays a strong foundation for future enhancements, such as integrating databases for persistent data storage, adding payment processing capabilities, and developing a user-friendly web interface for wider accessibility.

The implementation reflects a keen focus on user experience, system robustness, and adherence to programming best practices. The project stands as a testament to the team's dedication to creating an efficient, secure, and user-friendly movie ticket booking system.