Internship Report

Internship report submitted in fulfillment of the requirement for the certificate of the internship

By

Your Name: **ARPAN KUNDU**

College Name: **SANAKA EDUCATIONAL TRUST GROUP OF INSTITUTIONS**

Domain Name: **JAVA DEVELOPER**

Batch (JUNE-JULY)

Date of Submission: 30th JULY

2024

DECLARATION

I hereby declare that the projects entitled "**MOVIE TICKET BOOKING SYSTEM**" and submitted for the completion of my internship at InternJunction, are my original work. These projects have not been submitted for any degree, diploma, or other internship programs at any other organizations.

Name:	Arpan Kundu
Place:	Durgapur
Date:	20th july

ACKNOWLEDGEMENT

It is my proud privilege to express my heartfelt gratitude to several individuals who have assisted me directly or indirectly in completing this project. First and foremost, I would like to extend my deepest appreciation to my internship guides at InternJunction. Their sincere guidance, unwavering support, and invaluable inspiration have been instrumental in the successful completion of this internship.

The insights and knowledge I gained during this period have been profound and enlightening, significantly broadening my understanding of the topics at hand. This experience has not only deepened my academic and professional knowledge but also prepared me for future challenges and opportunities.

I am genuinely grateful for the opportunities provided by InternJunction, which have allowed me to explore and engage with new and exciting avenues of knowledge. This internship has been an enriching journey, and I am confident that the skills and insights I have gained will be of immense benefit in my future endeavours.

To all those who have contributed to this project, whether directly or indirectly, I extend my sincere thanks. Your support and encouragement have been invaluable, and I am truly indebted to you all.

TABLE OF CONTENTS

INTRODUCTION

Overview:

The Movie Ticket Booking System is a console-based application designed to streamline the process of booking and managing movie tickets. The system consists of two main classes: `MovieTicketBookingSystem` and `Customer`. The `Customer` class includes various methods for interacting with the system, such as displaying movie lists, booking tickets, canceling reservations, and allowing user selections.

Scope of the Project:

The project aims to develop a robust and user-friendly ticket booking system that operates within a console environment. It focuses on providing essential functionalities for managing movie bookings, including:

- Viewing available movies and showtimes.
- Booking tickets for selected movies.
- Canceling existing bookings.
- Facilitating user interactions through a simple text-based interface.

Objectives:

1. *Develop a Functional Booking System:* Create a working application that allows users to book and manage movie tickets through a console interface.

2. *Implement Core Features:* Provide functionalities such as displaying movie lists, booking tickets, canceling reservations, and handling user selections.

3. *Ensure Usability:* Design the system to be intuitive and easy to navigate, even within a console-based environment.

4. *Enhance User Experience:* Improve user interaction by offering clear prompts and managing inputs effectively.

Result:

The completed Movie Ticket Booking System successfully provides a console-based solution for booking and managing movie tickets. The system enables users to:

- Browse available movies and their showtimes.
- Reserve tickets for desired shows.
- Cancel or modify existing reservations.
- Enjoy a straightforward and functional interface for all booking-related tasks.

The Movie Ticket Booking System project employs two key classes: `MovieTicketBookingSystem`, which handles the overall functionality and operations of the booking system, and `Customer`, which manages user interactions and ticket management processes. The `Customer` class is equipped with methods for displaying movie options, facilitating bookings, and handling cancellations. The project showcases practical application of object-oriented programming principles and provides a solid foundation for understanding and developing console-based applications.

METHODOLOGY

<u>Movie Ticket Booking System</u>

The Movie Ticket Booking System is designed to streamline the process of booking and canceling movie tickets. This system consists of two primary classes: `MovieTicketBookingSystem` and `Customer`.

Class Structure

1. MovieTicketBookingSystem

- This class serves as the main entry point for the system. It contains the main method, which displays options to the user and redirects the flow of control based on the user's selection.

2. Customer

- This class encapsulates the logic for handling customer interactions, such as adding movies and canceling bookings. It includes methods for displaying movie lists, handling bookings, and processing cancellations.

Flow of Control

When the application starts, the main method in the `MovieTicketBookingSystem` class presents the user with options to either book or cancel a ticket. Based on the user's choice:

Booking: If the user selects the booking option, control is passed to the `addMovies()` method in the `Customer` class. This method facilitates the process of selecting and booking a movie.

Canceling: If the user selects the cancel option, control is passed to the `cancel()` method in the `Customer` class. This method manages the cancellation of existing bookings.

The Movie Ticket Booking System is designed to be user-friendly, providing straightforward options for booking and canceling movie tickets. The separation of concerns between the `MovieTicketBookingSystem` and `Customer` classes ensures that the system is modular and easy to maintain. By following this structured approach, the system can be easily extended to include additional features in the future.

MovieTicketBookingSystem Class

The `MovieTicketBookingSystem` class serves as the main entry point for the system. It contains the main method, which displays options to the user and redirects the flow of control based on the user's selection. The key responsibilities and flow of the `MovieTicketBookingSystem` class are as follows:

1. Displaying Options:

- When the application starts, the main method in the `MovieTicketBookingSystem` class presents the user with options to either book a ticket or cancel a ticket.

2. Handling User Choice:

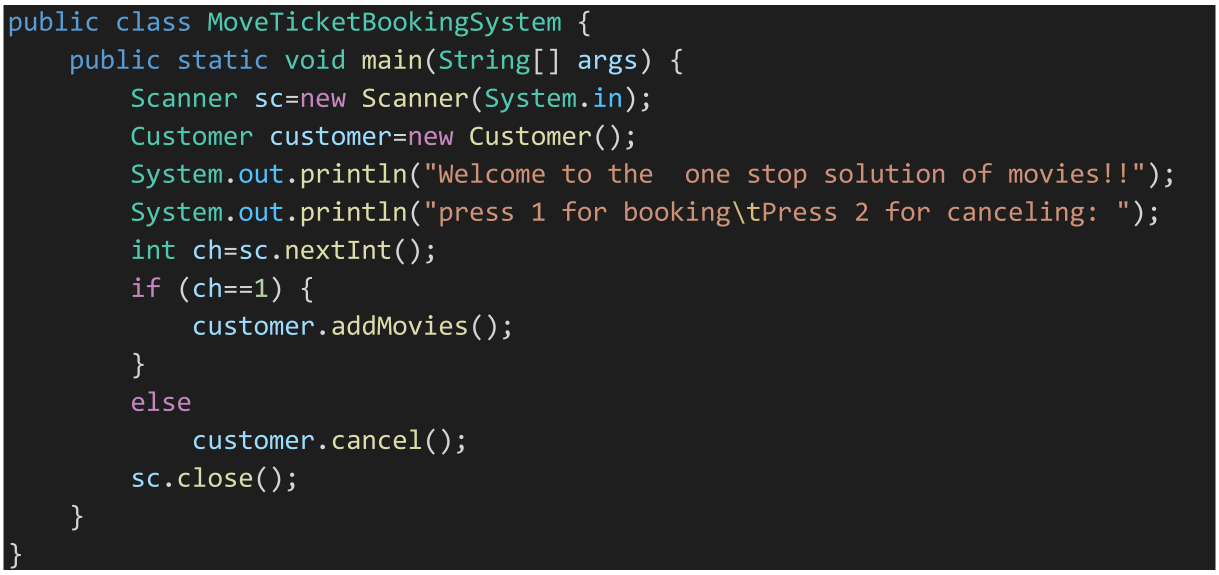
Based on the user's choice, the system redirects control to the appropriate method in the `Customer` class:

Booking: If the user selects the booking option, control is passed to the `addMovies()` method in the `Customer` class.

Canceling: If the user selects the cancel option, control is passed to the `cancel()` method in the `Customer` class.

3. Error Handling:

The system employs error handling mechanisms to manage any unexpected errors gracefully, ensuring a smooth user experience.



```
public class MoveTicketBookingSystem {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        Customer customer=new Customer();
        System.out.println("Welcome to the  one stop solution of movies!!");
        System.out.println("press 1 for booking\tPress 2 for canceling: ");
        int ch=sc.nextInt();
        if (ch==1) {
            customer.addMovies();
        }
        else
            customer.cancel();
        sc.close();
    }
}
```

By following this structured approach, the `MovieTicketBookingSystem` class effectively manages the primary user interactions and ensures a seamless flow within the movie ticket booking system. This design allows for easy maintenance and future extensions of the system.

Customer Class

The `Customer` class is responsible for handling the logic related to customer interactions in the Movie Ticket Booking System. It includes methods for adding movies, displaying available movie genres, processing movie selections, handling ticket collections, making payments, and canceling tickets. The class uses six `LinkedList` objects to store movies categorized by genre: action, comedy, detective, thrillers, suspense, and horror.

Key Methods and Their Functionality

1. LinkedLists for Movie Storage

- The `Customer` class contains six `LinkedList` objects to store movies of different genres: action, comedy, detective, thrillers, suspense, and horror.

2. addMovies()

- This void method is used to add new movies to a particular section. It performs the necessary logic to add movies and then calls the `section()` method for further processing.

3. section()

- This method displays the types of movies available in the theatre, such as action, comedy, detective, thrillers, suspense, and horror. After displaying the options, it prompts the user to choose a genre and then calls the `movieList(char choice)` method, passing the user's choice as an argument.

4. movieList()

- This method takes a character parameter representing the selected movie genre. It uses a switch-case statement to display the movies in the selected genre. For each case, it calls the `movieSelection(LinkedList<String> genreList)` method, passing the appropriate `LinkedList` for the selected genre. Error handling is managed using a try-catch block.

5. movieSelection()

- This method takes a `LinkedList` parameter representing the selected genre list. It allows the user to select a movie from the list, stores the selected movie name in a string variable, and then calls the `ticketCollection()` method for further processing. Error handling is managed using a try-catch block.

6. ticketCollection()

- This method manages the ticket collection process. It prompts the user to select the type of theatre and ticket class (first class, second class, third class) along with their respective prices. It also asks for the time slot (10:00 AM, 2:00 PM, 7:00 PM) and the number of tickets the user wants to buy. Finally, it calls the `makePayment(int ticketCount, String ticketType)` method, passing the ticket count and ticket type as parameters.

7. makePayment()

- This method calculates the total amount the user needs to pay based on the ticket count and ticket type. It provides options for payment modes (online and offline) and handles the payment process. The user is asked to enter the exact amount, and upon successful payment, the tickets are confirmed. If the payment fails, the user is prompted to try again. The method also provides an option to cancel the tickets by pressing the 'c' button, which redirects to the `cancel()` method.

8. cancel()

- This method handles the cancellation of tickets. If no tickets are available, it displays an appropriate message. Otherwise, it sets the ticket count to zero and declares that the tickets are canceled.

```java
class Customer
{
    LinkedList<String> action=new LinkedList<String>();
    LinkedList<String> comedy=new LinkedList<String>();
    LinkedList<String> detective=new LinkedList<String>();
    LinkedList<String> thrillers=new LinkedList<String>();
    LinkedList<String> suspence=new LinkedList<String>();
    LinkedList<String> horror=new LinkedList<String>();
    public void addMovies()
    {
        action.add("Avenger");action.add("Attack"); action.add("Wolvarine");
action.add("Moon_Kinght"); action.add("Kalki");

        comedy.add("Hera
Phire");comedy.add("Golmal");comedy.add("Welcome");comedy.add("Welcome_back");
comedy.add("HosueFull-4");

        detective.add("Bomkeysh");detective.add("Agent_Roy");detective.add("Fe
luDa");

        thrillers.add("Simle");thrillers.add("Katputli");thrillers.add("Hitman
");thrillers.add("Yodha");

        suspence.add("Feluda");suspence.add("chup");suspence.add("365_Days");

        horror.add("Conjuring");horror.add("Anabela");horror.add("Exorcist");
        section();
    }
    public void movieList(char ch)
    {
        switch (ch) {
            case 'A':for(int i=0;i<action.size();i++)
                        System.out.println((i+1)+" : "+action.get(i));
                    movieSelection(action);
                break;
            case 'C':for(int i=0;i<comedy.size();i++)
                        System.out.println((i+1)+" : "+comedy.get(i));
                    movieSelection(comedy);
                break;
            case 'D':for(int i=0;i<detective.size();i++)
                            System.out.println((i+1)+" : "+thrillers.get(i));
                    movieSelection(detective);
                break;
            case 'T':for(int i=0;i<thrillers.size();i++)
                        System.out.println((i+1)+" : "+thrillers.get(i));
                    movieSelection(thrillers);
                break;
            case 'S':for(int i=0;i<suspence.size();i++)
```

```
                        System.out.println((i+1)+" : "+suspence.get(i));
                    movieSelection(suspence);
                break;
            case 'H':for(int i=0;i<horror.size();i++)
                        System.out.println((i+1)+" : "+horror.get(i));
                    movieSelection(horror);
                break;
            default:
                break;
        }
    }
    public void movieSelection(LinkedList<String> choice)
    {
        Scanner sc=new Scanner(System.in);
        try {
            int fev=0;
        System.out.println("Select your movie: ");
        fev=sc.nextInt();
        if(fev<=choice.size()+1){
            String movieName=choice.get(fev-1);
            System.out.println("You choose: "+movieName+" movie.");
            ticketCollection();
        }
        else
            movieSelection(choice);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        finally
        {
            sc.close();
        }
    }
    int ticketcount=0;
    public void ticketCollection()
    {
        int fc=250,sc=200,tc=150;
        Scanner scan=new Scanner(System.in);
        System.out.println("Now choose type of the theater: \nPress '1' for
NORMAL\tPress '2' for IMAX 3D");
        int theater=scan.nextInt();
        if(theater==1)
            System.out.println("You select NORMAL");
        else
            System.out.println("You select IMAX 3D");
        System.out.println("What type of ticket are you looking for: ");
        System.out.println("Press '1' for FIRST CLASS(Price:"+fc+")\tPress '2'
for SECOND CLASS(Price:"+sc+")\tPress '3' for THIRD CLASS(Price:"+tc+")");
```

```java
        int c=scan.nextInt();
        if(c==1)
            System.out.println("You choose FIRST CLASS");
        else if(c==2)
            System.out.println("You choose SECOND CLASS");
        else
            System.out.println("You choose THIRD CLASS");
        System.out.println("We have three slot: \n1 : 10.00AM\n2 : 02.00PM\n3 : 07.00PM");
        int slot=scan.nextInt();
        if(slot==1)
            System.out.println("You choose 10.00AM slot.");
        else if(slot==2)
            System.out.println("You choose 02.00PM slot.");
        else
            System.out.println("You choose 07.00PM slot");
        System.out.println("How many ticket do you want to book? ");
        ticketcount=scan.nextInt();
        makePayment(ticketcount,c);
        scan.close();

    }
    public void makePayment(int ticketcount,int c)
    {
        Scanner scan=new Scanner(System.in);
        int total=0;
        if(c==1)
            total=ticketcount*250;
        else if(c==2)
            total=ticketcount*200;
        else
            total=ticketcount*150;
        System.out.println("So, you total amount is: "+total);
        System.out.println("Press '1' for Online\tPress '2' for Cash: ");
        String mode=scan.next();
        if(mode.equals("1")){
            System.out.println("Enter your amount: ");
            int amount=scan.nextInt();
            if(amount==total)
                System.out.println("You money is accepted.");
            else
                System.out.println("Please enter actual amount to confirm.");
        }
        else
        {
            System.out.println("Enter your amount: ");
            int amount=scan.nextInt();
            if(amount==total)
```

```java
                System.out.println("You money is accepted.");
            else
                System.out.println("Please enter actual amount to confirm.");
        }
        System.out.println("You "+ticketcount+"tickets are confirmed!!");
        System.out.println("If you want to cancel then press 'C' or press
anything: ");
        int can=scan.next().toUpperCase().charAt(0);
        if(can=='C')
            cancel();
        else
            System.out.println("Thank you!!\tVisit Agian!!");
        scan.close();
}
    public void section()
    {
        Scanner sc=new Scanner(System.in);
        char ch=' ';
        try{
        System.out.println("Hey, Welcome to the Customer Desk......\nAt that
time our collections
are:\nComedy,Thrillers,Action,Detective,Suspence,Horror,etc");
        System.out.println("So,What type of movie are you looking for:
\nPlease Select from the below list: ");
        System.out.println("Press 'A' for Action\tPress 'C' for Comedy\tPress
'T' for Thrillers\tPress 'D' for Detective\tPress 'S' for Suspence\tPress 'H'
for Horror: ");
        ch=sc.next().toUpperCase().charAt(0);
        }
        catch(Exception e){
            e.printStackTrace();
        }
        switch (ch) {
            case 'A':movieList(ch);
                break;
            case 'C':movieList(ch);
                break;
            case 'T':movieList(ch);
                break;
            case 'D':movieList(ch);
                break;
            case 'S':movieList(ch);
                break;
            case 'H':movieList(ch);
                break;
            default:System.out.println("Please enter the actual one!!!");
                     section();
                break;
        }
        sc.close();
    }
    public void cancel()
    {
        Scanner sc=new Scanner(System.in);
        String confirm=null;
        if(ticketcount==0)
            System.out.println("You do not book any ticket from any show!!");
        else
        {
            System.out.println("Are you sure yo want to cancel your booking: 
yes/no");
            confirm=sc.next();
            if(confirm.equals("yes"))
            {
                ticketcount=0;
                System.out.println("Your tickets are successfully canceled.");
            }
            else
                System.out.println("Ok!!");
        }
        sc.close();
    }
}
```

Summary of the Working Procedure

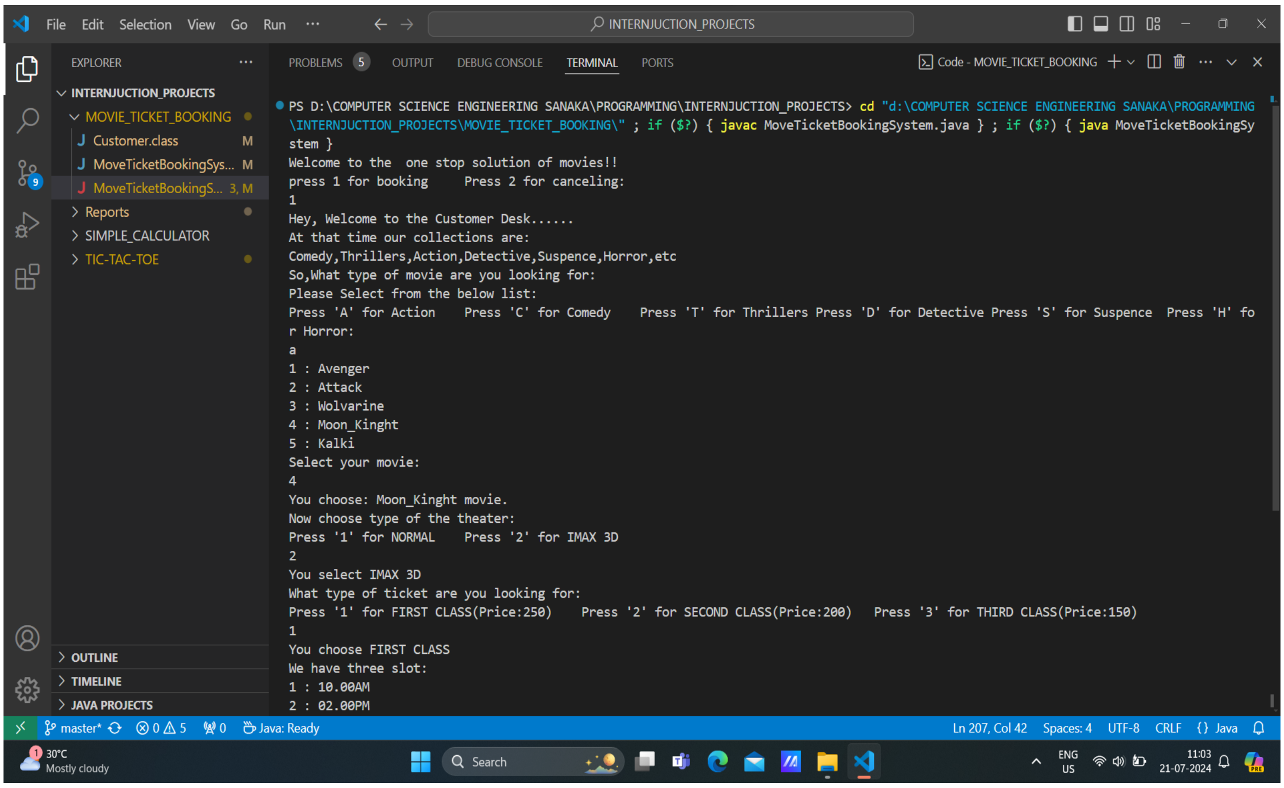
Movie Addition and Selection: The process starts with adding movies to different genres using the `addMovies()` method. The `section()` method displays the available genres, and the user selects a genre. The `movieList()` method processes the user's selection and calls the `movieSelection()` method to select a specific movie.

Ticket Collection and Payment: The `ticketCollection()` method manages the selection of theatre type, ticket class, and time slot, and collects the number of tickets. The `makePayment()` method handles the payment process, confirming the tickets upon successful payment or prompting the user to try again in case of failure.

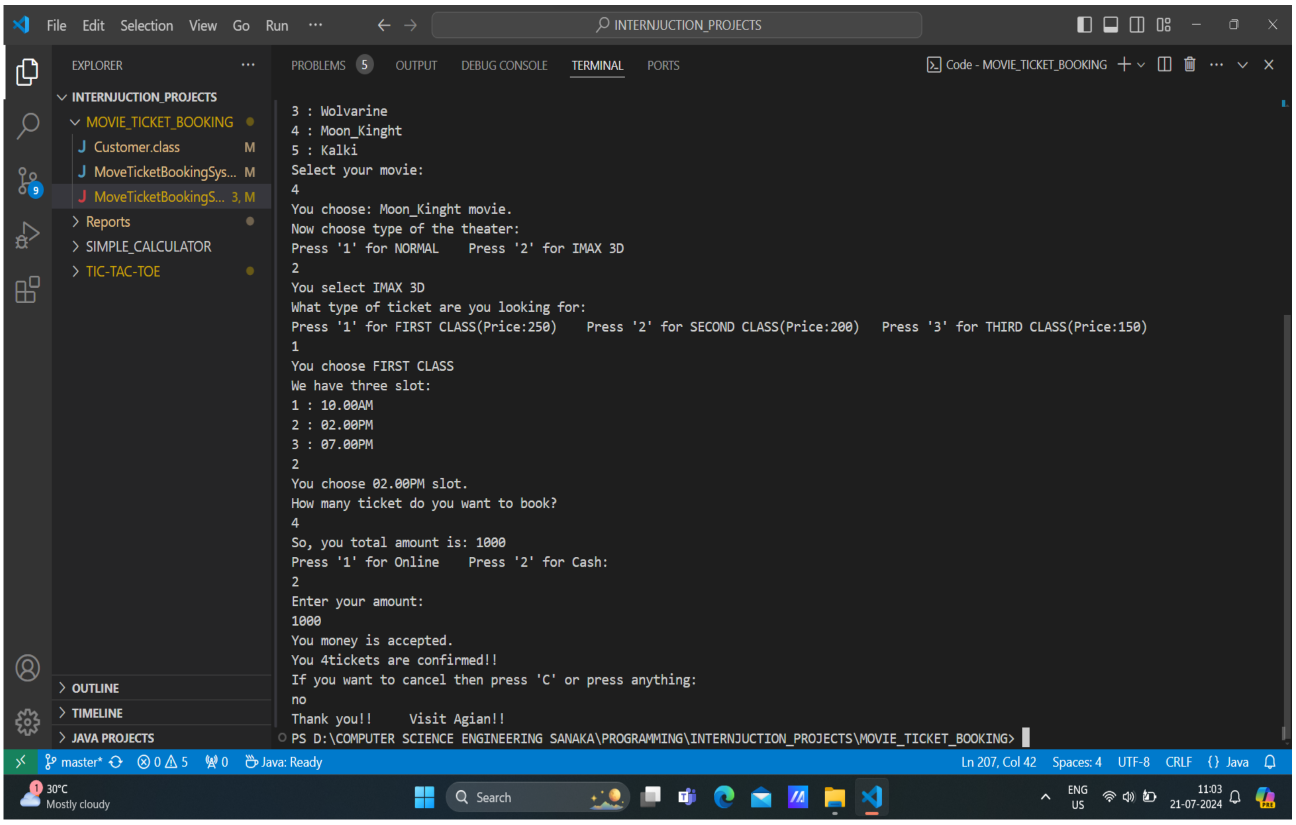
Ticket Cancellation: If the user decides to cancel the tickets, the `cancel()` method is invoked to set the ticket count to zero and confirm the cancellation.

By following this structured approach, the `Customer` class effectively manages movie selection, ticket collection, payment, and cancellation processes, ensuring a user-friendly and maintainable system.

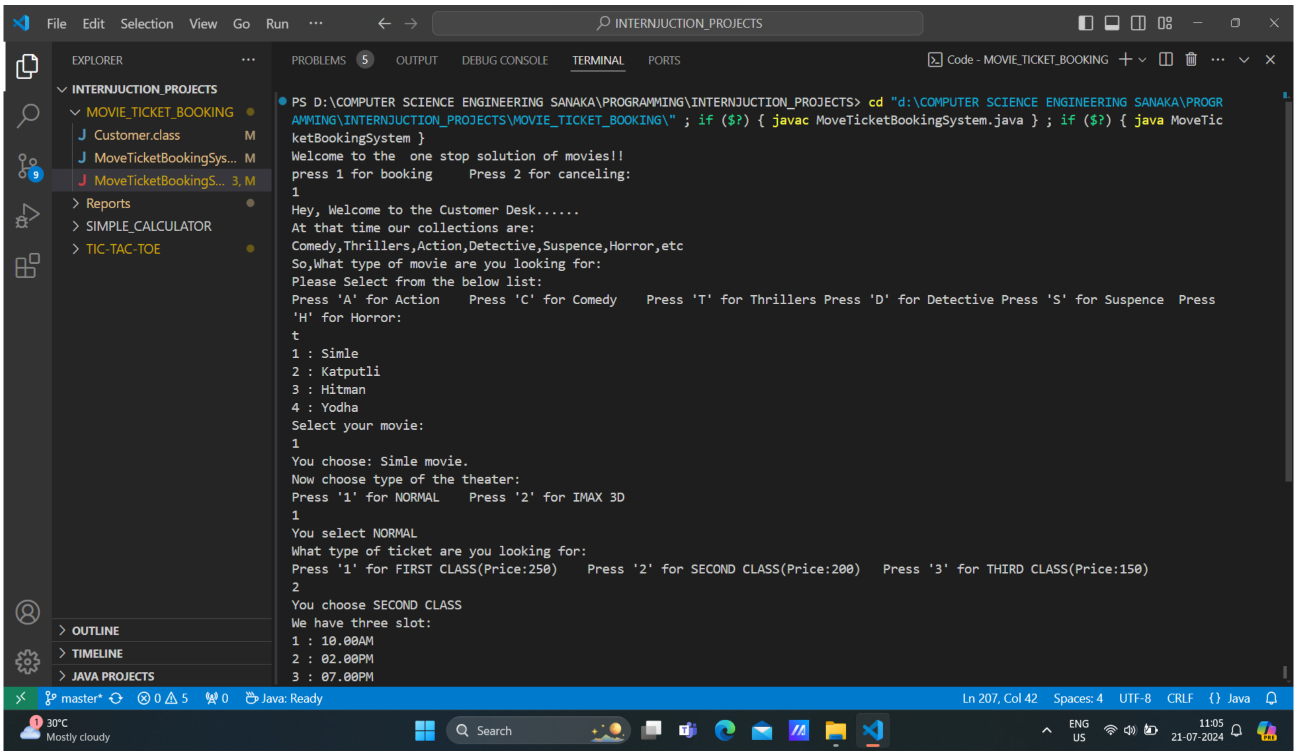
RESULTS



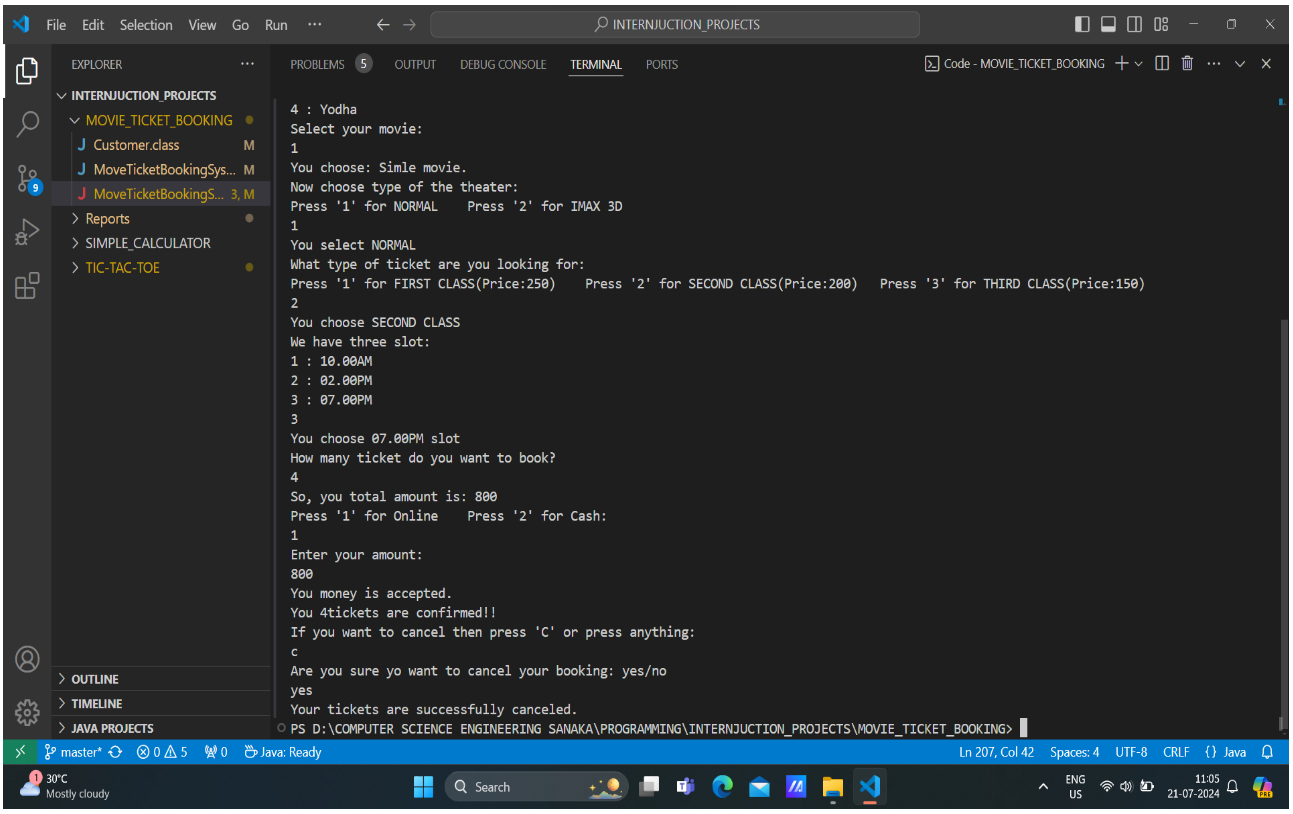
```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNJUCTION_PROJECTS> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING
\INTERNJUCTION_PROJECTS\MOVIE_TICKET_BOOKING\" ; if ($?) { javac MoveTicketBookingSystem.java } ; if ($?) { java MoveTicketBookingSy
stem }
Welcome to the  one stop solution of movies!!
press 1 for booking     Press 2 for canceling:
1
Hey, Welcome to the Customer Desk......
At that time our collections are:
Comedy,Thrillers,Action,Detective,Suspence,Horror,etc
So,What type of movie are you looking for:
Please Select from the below list:
Press 'A' for Action    Press 'C' for Comedy    Press 'T' for Thrillers Press 'D' for Detective Press 'S' for Suspence  Press 'H' fo
r Horror:
a
1 : Avenger
2 : Attack
3 : Wolvarine
4 : Moon_Kinght
5 : Kalki
Select your movie:
4
You choose: Moon_Kinght movie.
Now choose type of the theater:
Press '1' for NORMAL    Press '2' for IMAX 3D
2
You select IMAX 3D
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250)    Press '2' for SECOND CLASS(Price:200)    Press '3' for THIRD CLASS(Price:150)
1
You choose FIRST CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
```



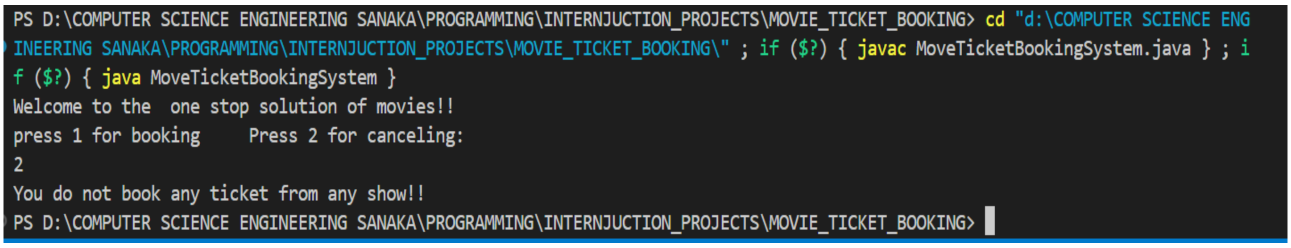
```
3 : Wolvarine
4 : Moon_Kinght
5 : Kalki
Select your movie:
4
You choose: Moon_Kinght movie.
Now choose type of the theater:
Press '1' for NORMAL    Press '2' for IMAX 3D
2
You select IMAX 3D
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250)    Press '2' for SECOND CLASS(Price:200)    Press '3' for THIRD CLASS(Price:150)
1
You choose FIRST CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
3 : 07.00PM
2
You choose 02.00PM slot.
How many ticket do you want to book?
4
So, you total amount is: 1000
Press '1' for Online    Press '2' for Cash:
2
Enter your amount:
1000
You money is accepted.
You 4tickets are confirmed!!
If you want to cancel then press 'C' or press anything:
no
Thank you!!     Visit Agian!!
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNJUCTION_PROJECTS\MOVIE_TICKET_BOOKING>
```



```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNJUCTION_PROJECTS> cd "d:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGR
AMMING\INTERNJUCTION_PROJECTS\MOVIE_TICKET_BOOKING\" ; if ($?) { javac MoveTicketBookingSystem.java } ; if ($?) { java MoveTic
ketBookingSystem }
Welcome to the  one stop solution of movies!!
press 1 for booking     Press 2 for canceling:
1
Hey, Welcome to the Customer Desk......
At that time our collections are:
Comedy,Thrillers,Action,Detective,Suspence,Horror,etc
So,What type of movie are you looking for:
Please Select from the below list:
Press 'A' for Action    Press 'C' for Comedy    Press 'T' for Thrillers Press 'D' for Detective Press 'S' for Suspence  Press
'H' for Horror:
t
1 : Simle
2 : Katputli
3 : Hitman
4 : Yodha
Select your movie:
1
You choose: Simle movie.
Now choose type of the theater:
Press '1' for NORMAL    Press '2' for IMAX 3D
1
You select NORMAL
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250)    Press '2' for SECOND CLASS(Price:200)   Press '3' for THIRD CLASS(Price:150)
2
You choose SECOND CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
3 : 07.00PM
```



```
4 : Yodha
Select your movie:
1
You choose: Simle movie.
Now choose type of the theater:
Press '1' for NORMAL    Press '2' for IMAX 3D
1
You select NORMAL
What type of ticket are you looking for:
Press '1' for FIRST CLASS(Price:250)    Press '2' for SECOND CLASS(Price:200)   Press '3' for THIRD CLASS(Price:150)
2
You choose SECOND CLASS
We have three slot:
1 : 10.00AM
2 : 02.00PM
3 : 07.00PM
3
You choose 07.00PM slot
How many ticket do you want to book?
4
So, you total amount is: 800
Press '1' for Online    Press '2' for Cash:
1
Enter your amount:
800
You money is accepted.
You 4tickets are confirmed!!
If you want to cancel then press 'C' or press anything:
c
Are you sure yo want to cancel your booking: yes/no
yes
Your tickets are successfully canceled.
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNJUCTION_PROJECTS\MOVIE_TICKET_BOOKING>
```



```
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNJUCTION_PROJECTS\MOVIE_TICKET_BOOKING> cd "d:\COMPUTER SCIENCE ENG
INEERING SANAKA\PROGRAMMING\INTERNJUCTION_PROJECTS\MOVIE_TICKET_BOOKING\" ; if ($?) { javac MoveTicketBookingSystem.java } ; i
f ($?) { java MoveTicketBookingSystem }
Welcome to the  one stop solution of movies!!
press 1 for booking     Press 2 for canceling:
2
You do not book any ticket from any show!!
PS D:\COMPUTER SCIENCE ENGINEERING SANAKA\PROGRAMMING\INTERNJUCTION_PROJECTS\MOVIE_TICKET_BOOKING>
```

CONCLUTION

Successful Completion of the Movie Ticket Booking System Project

The successful completion of the Movie Ticket Booking System project marks a significant achievement in our practical programming journey.

Practical Application of Knowledge

We effectively applied our theoretical knowledge to this project, implementing modularity and object-oriented programming concepts to develop a functional and user-friendly movie ticket booking system. The project allowed us to manage user input, handle data, and integrate various functionalities in a real-world context.

Achievement of Objectives

The project met its primary objective of creating a robust platform for managing and booking movie tickets. It also served as a valuable learning resource, demonstrating practical applications of coding principles and providing a comprehensive understanding of system design and data management.

Skill Enhancement

Our programming skills were significantly enhanced through hands-on experience with Java. We gained proficiency in user input handling, error management, and modularity programming. The complexity of the Movie Ticket Booking System further strengthened our ability to develop scalable and maintainable software solutions.

Positive Impact on Professional Growth

This project positively impacted our professional growth by improving our problem-solving abilities, deepening our understanding of OOP principles, and offering insights into industry-level software development. It provided a practical experience in developing and managing complex systems, preparing us for future challenges in the field.

Overall Experience

Completing the Movie Ticket Booking System was both enriching and fulfilling. It solidified our technical skills and boosted our confidence in tackling real-world programming tasks. The knowledge and experience gained from this project will be invaluable for our future endeavors in software development. We are grateful for this opportunity and look forward to applying these skills in future projects.

FUTURE WORK

1. Enhanced User Interface:

- Develop a more intuitive and interactive user interface to improve the overall user experience. Consider incorporating features like color coding, better formatting of output, and more detailed prompts.

2. Advanced Search and Filtering:

- Implement advanced search and filtering options to help users find movies based on various criteria such as genre, rating, or showtimes more easily.

3. Reservation Management:

- Add functionality for users to view, modify, or cancel their existing reservations. This could include updating ticket quantities, changing showtimes, or processing refunds.

4. Error Handling and Validation:

- Enhance error handling and input validation to ensure robustness and reliability. Implement more comprehensive checks for invalid input, such as incorrect movie IDs or invalid dates.

5. Database Integration:

- Integrate a database system to manage movie listings, showtimes, and bookings persistently. This will allow for data storage and retrieval even after the console application is closed.

6. User Authentication:

- Add user authentication features to allow for personalized experiences, such as saving user profiles, tracking booking history, and managing multiple accounts.

7. Multi-User Support:

- Implement multi-user capabilities to allow multiple users to interact with the system simultaneously. This would include concurrent booking handling and session management.

8. Automated Reports and Analytics:

- Develop features to generate reports and analytics on ticket sales, popular movies, and user preferences. This could help in analyzing system performance and making informed decisions.

9. Integration with External APIs:

- Explore integrating with external APIs to fetch real-time movie data, such as current showtimes, ratings, and reviews. This can enhance the system's functionality and provide up-to-date information.

10. Testing and Optimization:

- Conduct thorough testing to identify and fix bugs or performance issues. Optimize the code for better efficiency and responsiveness, especially when handling large volumes of data or concurrent requests.

11. Documentation and User Guide:

- Create comprehensive documentation and a user guide to assist users in navigating and utilizing the system effectively. This will also be valuable for future maintenance and development.

12. Scalability and Future Enhancements:

- Plan for scalability to accommodate future growth and additional features. Consider how the system can be expanded to support more complex scenarios or integrated with other applications.