

PROJECT REPORT ON MOVIE TICKET BOOKING SYSTEM



L OVELY
P ROFESSIONAL
U NIVERSITY

SUBMITTED BY:-

NAME

Raushan Kumar

Kumar Sourav

Priyanshu Chaurasiya

REG NO:-

12109076

12112610

12113013

SUBMITTED TO:-

Surbhi Ma'am 23540

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who contributed to the successful completion of this project.

Primarily, we would like to thank our teacher for providing us with guidance, support, and feedback throughout the entire project. Their expertise and insight were invaluable in ensuring that we were on the right track and meeting all project requirements.

We would also like to thank our team members for their dedication, hard work, and cooperation. Each member brought unique skills and perspectives to the project, and together, we were able to create a comprehensive and effective Movie Ticket Booking system.

In this project movie ticket is booked using movie Ticket booking system. We enter into Web page by logging with User Name and Password. Then we select the Movie and later in which Theatre movie is running. Later choose Show Timings and enter no of tickets you want . Finally it displays the details of the procedure and print the form to show at respective ticket counter to get ticket.

Introduction

The main purpose of our online ticket booking system is to provide an alternate and convenient way for a customer to buy cinema tickets. It is an automatic system. After the data has been fed into the database, the staff does not need to do anything with the order once it is received through the system

Objective:- The objective of this code is to provide a basic movie ticket booking system that allows users to select a movie from a list, choose seats, pay for their booking, and receive a confirmation message. The code also includes exception handling to handle various input errors and prevent the program from crashing. The user is prompted to enter their username and password, and the code validates the input before proceeding to the booking process. The system aims to provide a user-friendly interface and error-free booking process for users.

Abstract:-

A movie ticket booking system is an online platform that enables customers to browse, select and book movie tickets for their desired showtimes and seats. The system includes a website or mobile app that displays available movies, showtimes, seating plans, and ticket prices. Customers can select their desired movie, showtime, and seat, and then purchase the tickets online using various payment methods such as credit/debit card, net banking, or mobile wallet.

The system generally consists of two main parts, the customer-facing front-end and the back-end, which manages the database, reservations, and payment processing. The front-end allows customers to search for available movies and showtimes, select their preferred seats, and pay for their tickets. The back-end of the system is responsible for storing and managing the data of all movies, showtimes, seats, and payments.

In addition to basic ticket booking functionality, a movie ticket booking system may offer various features such as user account creation, seat selection, seat availability status, ratings and reviews, discounts and promotions, and social media sharing. Some advanced systems may also include analytics and reporting tools to help cinema operators track ticket sales, customer preferences, and other performance metrics.

Overall, a movie ticket booking system streamlines the ticket booking process, making it easier and more convenient for customers to purchase tickets online. The system provides real-time updates on movie schedules and seat availability, reducing the need for customers to queue up at the box office or call the cinema for ticket reservations. This allows cinema operators to focus on providing a better movie experience for their customers, while increasing their revenue and reducing operational costs.

This code is a simple movie ticket booking system implemented in Java. The program allows the user to select a movie from a pre-defined list, choose the number of seats, and make a payment using one of four pre-defined banks. The program also includes

exception handling to ensure that user inputs are valid and to handle errors that may occur during the booking process.

About The code:-

This code simulates a movie booking system where the user can select a movie, choose the number of seats they want, and pay for them using one of the four available banks. The code has several methods that handle different parts of the booking process, and it also includes exception handling to handle errors gracefully.

The main method creates an instance of the User class and calls the login method, which prompts the user to enter their username and password. If the username or password is empty, an exception is thrown.

Once the user is logged in, the getname method is called, which prompts the user to enter their name. If the name is empty, an exception is thrown. Then, the getmovie method is called, which displays a list of movies and prompts the user to select one. If the user enters an invalid choice, an exception is thrown.

After the user selects a movie, the getseat method is called, which prompts the user to choose the number of seats they want and then select the seats. The code calculates the total amount to be paid and prompts the user to select a bank for payment. If the user enters an invalid bank selection or the payment amount does not match the total amount, an exception is thrown.

Overall, this code demonstrates how to use exception handling to make your code more robust and handle errors gracefully, as well as how to break a complex task into smaller, more manageable parts using different methods.

This is a Java program for a simple movie ticket booking system that allows the user to select a movie from a list, choose seats, and pay for the tickets via an online banking system. The program is designed to be run on the command line.

The code starts with importing the "Scanner" class, which is used to get input from the user. Then it defines a class "MovieTicketBooking" with four static methods: "main", "getname", "getmovie", and "getseat". The "main" method creates an instance of the

"User" class, calls its "login" method to get the user's credentials, and then calls the "getname" method to start the ticket booking process.

The "getname" method prompts the user to enter their name, validates it, and then calls the "getmovie" method. The "getmovie" method displays a list of movies, prompts the user to select one, validates the input, and then calls the "getseat" method. The "getseat" method prompts the user to enter the number of seats they want and to choose their seats. It then calculates the total cost and prompts the user to select a bank to pay with. Once the user selects a bank, the program prompts them to enter the payment amount, validates it, and then displays a success message if the payment is successful.

The "User" class defines two private variables "username" and "password" and a public method "login" to get the user's credentials. It validates the inputs and throws an exception if they are empty.

Overall, this program demonstrates how to use basic Java programming concepts like input/output, control structures, arrays, exceptions, and classes to create a simple movie ticket booking system.

main() method: This is the entry point of the program. It creates an instance of the User class and calls its login() method to authenticate the user. After successful authentication, it calls the getname() method to get the name of the user.

getname() method: This method prompts the user to enter their name and checks whether the name is empty or not. If the name is empty, it throws an exception. Otherwise, it greets the user and calls the getmovie() method to get the movie selection.

getmovie() method: This method displays a list of movies and prompts the user to select a movie from the list. It then checks whether the user's choice is valid or not. If the choice is invalid, it throws an exception. Otherwise, it displays the selected movie and calls the getseat() method to get the number of seats and seat selection.

getseat() method: This method prompts the user to enter the number of seats they want to book and checks whether the input is valid or not. If the input is invalid, it throws an exception. Otherwise, it asks the user to select the seats and calculates the total amount to pay based on the number of seats booked. It then prompts the user to select a bank to make the payment and checks whether the selection is valid or not. If the selection is invalid, it throws an exception. Otherwise, it asks the user to enter the payment amount

and checks whether the payment amount matches the total amount to be paid or not. If the payment amount does not match, it throws an exception. Otherwise, it displays a success message indicating that the payment is completed and the seat is booked.

The User class contains only one method, login(). This method prompts the user to enter their username and password and checks whether they are empty or not. If either the username or password is empty, it throws an exception. Otherwise, it validates the username and password (in a real-world application, this would involve checking against a database of registered users) and grants access if they are correct.

Source Code:-

```
import java.util.Scanner;

public class MovieTicketBooking {
    static Scanner sc = new Scanner(System.in);

    static String[] movieList = {"1)Avatar", "2)Robot 2.0", "3)Transformer",
    "4)Avenger", "5)Ice age"};

    public static void main(String[] args) {
        try {
```

```
User user = new User();  
user.login();  
getname();  
} catch (Exception e) {  
    System.out.println("An error occurred: " + e.getMessage());  
}  
}
```

```
public static void getname() throws Exception{  
    String name;  
    System.out.println("Enter your name:");  
    name = sc.nextLine();  
    if (name.isEmpty()) {  
        throw new Exception("Name cannot be empty");  
    }  
    System.out.println("WELCOME " + name);  
    getmovie();  
}
```

```
public static void getmovie() throws Exception {  
    System.out.println("Select the movie from the list:");  
    for (int i = 0; i < movieList.length; i++) {  
        System.out.println(movieList[i]);  
    }  
    int choice = sc.nextInt();  
    if (choice < 1 || choice > 5) {  
        throw new Exception("Invalid movie selection");  
    }  
    System.out.println("You selected: " + movieList[choice - 1]);  
}
```

```

    getseat();
}

public static void getseat() throws Exception {
    int n;
    System.out.println("How many seats do you want?");
    n = sc.nextInt();
    if (n <= 0) {
        throw new Exception("Number of seats must be positive");
    }
    int[] arr = new int[n];
    System.out.println("Choose your seats:");
    for (int i = 0; i < n; i++) {
        arr[i] = sc.nextInt();
    }
    int amount = n * 150; //1 ticket cost is 150
    System.out.println("Total amount to pay: " + amount);
    System.out.println("Please select your bank to pay:");
    System.out.println("1) Axis Bank");
    System.out.println("2) ICICI Bank");
    System.out.println("3) Indian Bank");
    System.out.println("4) State Bank");

    int bank = sc.nextInt();
    switch (bank) {
        case 1:
            System.out.println("Welcome to Axis bank");
            break;
        case 2:

```



```

        System.out.println("Welcome to ICICI bank");
        break;
    case 3:
        System.out.println("Welcome to Indian bank");
        break;
    case 4:
        System.out.println("Welcome to State bank");
        break;
    default:
        throw new Exception("Invalid bank selection");
    }
    System.out.println("Enter the amount:");
    int amountpay = sc.nextInt();
    if (amountpay == amount) {
        System.out.println("Your payment is successfully completed.");
        System.out.println("Your seat has been successfully booked.");
        System.out.println("Thank you!");
    } else {
        throw new Exception("Payment amount does not match the total amount");
    }
}
}

class User {
    private String username;
    private String password;

    public void login() throws Exception {
        System.out.println("Enter your username:");

```

```
username = MovieTicketBooking.sc.nextLine();  
if (username.isEmpty()) {  
    throw new Exception("Username cannot be empty");  
}  
  
System.out.println("Enter your password:");  
password = MovieTicketBooking.sc.nextLine();  
if (password.isEmpty()) {  
    throw new Exception("Password cannot be empty");  
}  
}  
}
```


OUTPUTS :-

```
PS C:\Users\HP> cd desktop
PS C:\Users\HP\desktop> javac MovieTicketBooking.java
PS C:\Users\HP\desktop> java MovieTicketBooking
Enter your username:
RSP123
Enter your password:
RSP@123
Enter your name:
Raushan
WELCOME Raushan
Select the movie from the list:
1)Avatar
2)Robot 2.0
3)Transformer
4)Avenger
5)Ice age
3
You selected: 3)Transformer
How many seats do you want?
3
Choose your seats:
9
8
7
Total amount to pay: 450
Please select your bank to pay:
1) Axis Bank
2) ICICI Bank
3) Indian Bank
4) State Bank
4
Welcome to State bank
Enter the amount:
450
Your payment is successfully completed.
Your seat has been successfully booked.
Thank you!
```

Output with Exception:-

```
Enter your username:
RSP123
Enter your password:
RSP@123
Enter your name:
Raushan
WELCOME Raushan
Select the movie from the list:
1)Avatar
2)Robot 2.0
3)Transformer
4)Avenger
5)Ice age
2
You selected: 2)Robot 2.0
How many seats do you want?
1
Choose your seats:
2
Total amount to pay: 150
Please select your bank to pay:
1) Axis Bank
2) ICICI Bank
3) Indian Bank
4) State Bank

1
Welcome to Axis bank
Enter the amount:
100
An error occurred: Payment amount does not match the total amount
PS C:\Users\HP\desktop> |
```



```
PS C:\Users\HP> cd desktop
PS C:\Users\HP\desktop> javac MovieTicketBooking.java
PS C:\Users\HP\desktop> java MovieTicketBooking
Enter your username:
RSP@123
Enter your password:
RSP@123
Enter your name:
Raushan
WELCOME Raushan
Select the movie from the list:
1)Avatar
2)Robot 2.0
3)Transformer
4)Avenger
5)Ice age
2
You selected: 2)Robot 2.0
How many seats do you want?
.
An error occurred: null
PS C:\Users\HP\desktop> |
```

Comments:-

- The first line imports the Scanner class to get user input.
- The Movie1 class is defined, which is the main class of the program.
- A Scanner object is created to get user input.
- An array of movie names is created.
- The main method is defined, which calls the login and getname methods and catches any exceptions that may occur.

- The getname method is defined, which asks the user for their name and checks if the name is empty. If the name is not empty, the method proceeds to ask the user to select a movie.
- The getmovie method is defined, which displays the movie list and asks the user to select a movie by its index number. The method checks if the index number is valid and proceeds to ask the user to select a seat.
- The getseat method is defined, which asks the user to select the number of seats and seat numbers. The method calculates the total amount to pay based on the number of seats and asks the user to select a bank for payment. The method checks if the bank selection is valid and proceeds to ask the user to enter the payment amount.
- If the payment amount matches the total amount, the method confirms the payment and booking of the seats. Otherwise, an exception is thrown.
- The User class is defined, which has a login method to get the user's username and password and validate them. If the username or password is empty, an exception is thrown.

Advantages:-

Convenience: Customers can book their movie tickets from the comfort of their own homes or anywhere they have access to the internet, saving them time and effort.

Easy to use: The project is designed to be user-friendly, making it easy for customers to navigate and book their tickets.

Fast and efficient: Customers can quickly select their movie, seats, and payment method without having to wait in long queues.

Real-time availability: The system provides real-time information about available seats, making it easier for customers to choose their preferred seats.

Secure: The system ensures the security of customer information and transactions, protecting users' privacy and preventing fraud.

Centralized data management: The project provides a centralized platform for managing movie information, ticket booking, and payment records, making it easier for theater operators to track and manage their business.

Disadvantages:-

Lack of error handling: Although the code includes some basic error handling, such as checking for empty inputs and invalid movie selections, it does not cover all possible scenarios. For example, if the user enters a non-integer value when prompted for the number of seats they want, the program will crash.

Limited functionality: The program only allows the user to select a movie, choose seats, and make a payment. It does not include other common features of movie booking systems, such as the ability to select specific showtimes, choose between different seating options, or view a seating chart.

Security vulnerabilities: The program does not include any security measures to protect user data, such as encryption or password hashing. Additionally, the login process is very basic and could be vulnerable to brute force attacks or other forms of hacking.

Lack of scalability: The program is designed to handle only one user at a time and does not include any features for managing multiple bookings simultaneously. This could be a problem if the system were to be scaled up to handle a larger user base.

Lack of integration with external systems: The program does not include any integration with external systems, such as payment gateways or customer databases. This limits its usefulness in a real-world setting, where such integrations are often necessary.

Features Lists :-

1. Users Login.
2. User Register.
3. User enter their name
4. They select no of tickets with their won choice
5. Selection of Movie module.
6. Selecting Theaters module.
7. Number of Tickets booking module.
8. Display booked Tickets with deatials
9. Booked from anywhere

Hardware and So>ware Requirement

1. Architecture: - 32bit or 64bit.
 2. Processor type: - Intel Core, AMD Ryzen (any of these).
 3. Cores: - 8 Minimum
 4. Threads: - 16 Minimum
 5. RAM: - 2GB Minimum
 6. Processor Speed: - 1.20 GHZ or higher
 7. Hard disk: - 40GB or more.
 8. SoCware Required: Any text editor (like notepad) and command prompt, Intellie.9.
- Compiler Required: JDK 1.8 or higher.

CONCLUSION

The project was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project

Automation of the entire system improves the efficiency

We can provide the communication between Customer and Cinema Hall.

Can also create Registration for Customer so that Cinema Hall can contactthem about exiting offers.

Can also make Customer to Post their View on Website.

The System has adequate scope for modification in future if it is necessary.

Creating a Movie Ticket booking System Project helped us to gain experience on how multiple classes, Function works together and how exception handling works, and Show

sms to user. Every member of the group contributed in making different classes. We hope this project will help user to save their time, and increase versatility of the system.

REFERENCE:-

www.w3schools.com

www.coderanch.com

www.apache.org

www.javasoft.com

*Thank
you*

