DIPLOMA OF INFORMATION TECHNOLOGY

CENTRE FOR DIPLOMA STUDIES

FINAL PROJECT REPORT

COURSE NAME	**OBJECT ORIENTED PROGRAMMING**
COURSE CODE	**DAT20303**
SECTION	**6**
PROJECT TITLE	**CINEMA TICKETS**
GROUP MEMBERS	• **WAN MUHAMMAD ALIF FIRDAUS BIN WAN HANAPI (AA210292)** • **WAN ALIF DANIAL BIN WAN KAMARULFARID (AA212909)** • **WAN MOHAMAD ZARIN NAIF BIN WAN MOHAMAD NIZAM (AA212643)** • **QAIREL QAYYUM BIN MUHAMAD RIDHUAN (AA212363)** • **SYAZA INSYIRAH BINTI MD NIZAM (AA212628)**
NAME OF LECTURER	**ABDUL HALIM BIN OMAR**

TABLE OF CONTENT

BIL	TITLE	PAGE
1	Introduction	3-4
2	Objective	5
3	Use case diagram	6
4	Documentation	7-13
5	Appendix	14-90
6	Rubric	91-95

INTRODUCTION

1. Project Background

An online cinema booking system is an excellent choice for a programming project for several reasons:

Relevance: In today's digital age, people are looking for convenience and efficiency in their everyday activities. With the increasing popularity of streaming services, cinemas are looking for ways to enhance the customer experience and increase ticket sales. An online booking system can make it easier for customers to purchase tickets and choose their seats, leading to a more convenient and enjoyable cinema experience. It also reduces the time and effort required for customers to purchase tickets, freeing them up to focus on other things.

Complexity: An online cinema booking system involves several complex systems and processes, such as seat selection, payment processing, and real-time availability updates. The system must also be able to handle a large volume of transactions and user data, making it an ideal platform for students to develop and demonstrate their technical skills, such as database design, web development, and payment processing.

Real-world application: An online cinema booking system is a practical and real-world application that students can use to apply their programming knowledge and skills. This can help them better understand how their skills can be applied in the real world and can make their learning more engaging and relevant. Moreover, students will also have the opportunity to learn about user experience design, as well as the importance of making the system intuitive, user-friendly and accessible to a wide range of users.

Versatility: An online cinema booking system can be implemented in many different ways, with varying levels of complexity and functionality. This provides students with the opportunity to tailor their project to their skills and interests, making it a flexible and engaging project. For example, some students might focus on creating a simple and straightforward booking system, while others might choose to add additional features such as a rewards program, real-time seat selection, or integration with popular payment providers.

Opportunity for improvement: With the rapid development of technology and the changing needs of consumers, there is always room for improvement in an online cinema booking system. This means that students have the opportunity to continually innovate and improve their system, making it a long-term and challenging project that can help them grow as programmers.

Overall, an online cinema booking system is a great choice for a programming project because it is relevant, complex, practical, versatile and provides opportunities for improvement. The project can help students develop a range of technical skills, learn about user experience design, and gain a better understanding of the real-world application of their programming knowledge.

2. PROBLEM BACKGROUND

The increasing popularity of streaming services has put pressure on cinemas to find new ways to enhance the customer experience and increase ticket sales. The traditional methods of purchasing cinema tickets, such as standing in line or calling to reserve seats, can be time-consuming and inconvenient for customers. The aim of this project is to develop a web-based application that will solve these problems by enabling customers to easily purchase cinema tickets and choose their seats online.

The online cinema booking system should be user-friendly and accessible to a wide range of customers, regardless of their technical abilities. The system should also be efficient and secure, with real-time availability updates and seat selection capabilities. The seat selection process should allow customers to view a map of the cinema, choose their preferred seats, and purchase their tickets in a single transaction. The system should also support various payment methods, such as credit cards and online payment services, to ensure a convenient and seamless purchase experience.

In addition to providing a better experience for customers, the online cinema booking system should also provide a platform for students to demonstrate their technical skills. This includes database design, web development, payment processing, and user experience design. The project should challenge students to develop a system that can handle a large volume of transactions and user data, while also maintaining a high level of security and reliability.

PROJECT OBJECTIVE

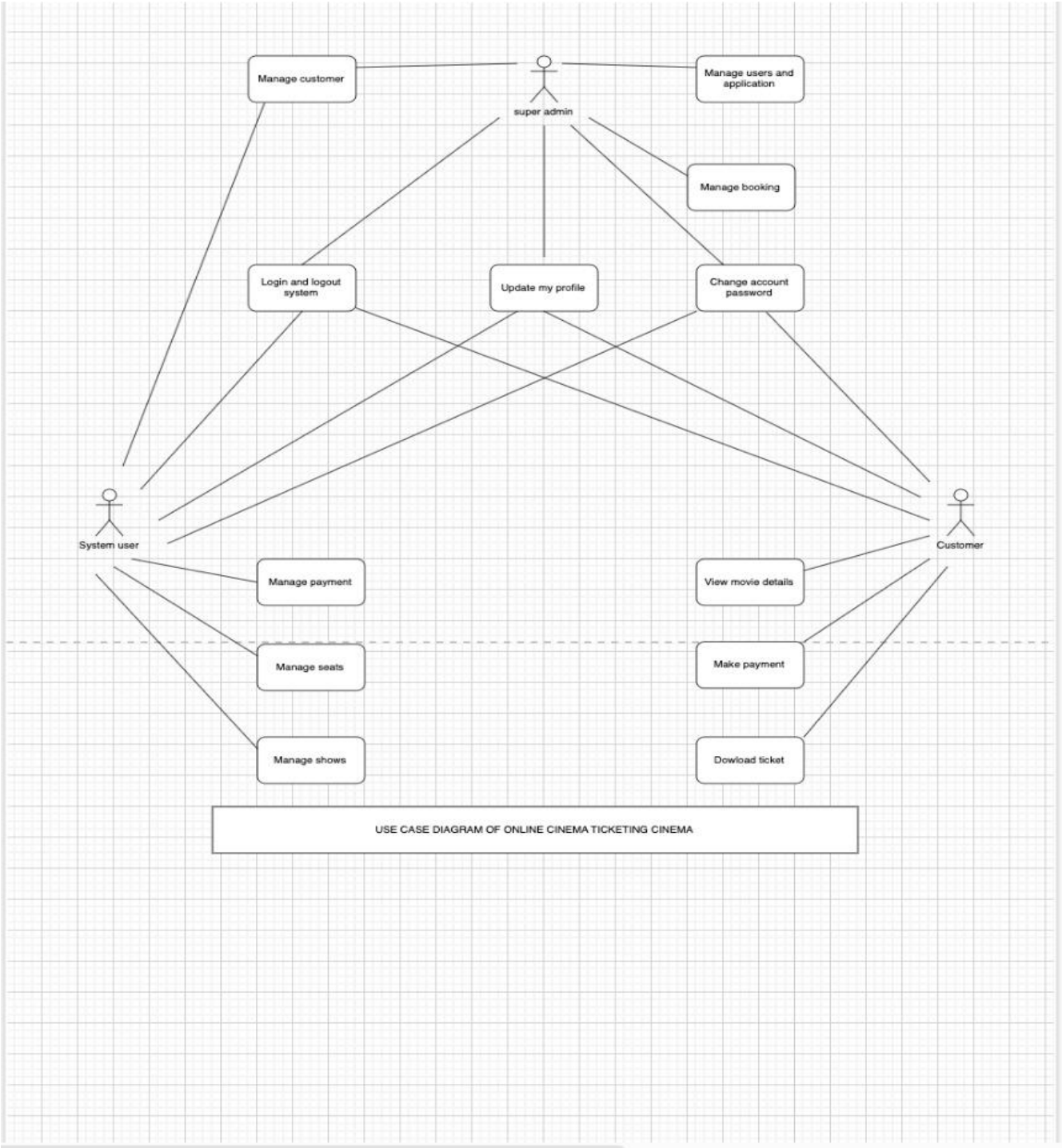
The goal of an online cinema booking system is to provide a fast, secure, and convenient platform for users to search and book cinema tickets. The system automates the traditional manual process and enhances the movie-going experience by allowing users to select preferred seats, showtimes, and make secure payments. It aims to automate the traditional manual process of buying tickets, reducing waiting times, and increasing customer satisfaction. This project aims to implement features such as movie listings, seat selection, payment processing, and ticket confirmation. The ultimate goal is to simplify the ticket booking process and enhance the cinema-going experience for users.

- To provide a user-friendly platform for movie-goers to browse and book tickets for movies at their preferred cinema.

- To streamline the booking process, reducing the time and effort required for purchasing tickets.

- To offer real-time availability of tickets and seat selection options.

USE CASE DIAGRAM



Documentation of program

```
-----------------------------------Welcome to Golden Screen Cinema-----------------------------------

[1]-Log in
[2]-Sign up
[3]-Show available Movies
[4]-Quit the program
Select an option:
```

Here is the homepage or main menu of Golden Screen Cinema

```
-----------------------------------Welcome to Golden Screen Cinema-----------------------------------

[1]-Log in
[2]-Sign up
[3]-Show available Movies
[4]-Quit the program
Select an option: 5
Invalid input. Please enter within the range (1-4):
```

There can be only 1 until 4 need to be selected only if user enter out of the number range there will be input validation pop out and they need to key in only that range of number only

```
-------------------------------------------------------------------------------------------
                                            Log in
-------------------------------------------------------------------------------------------

Enter mobile number: 01162293464

Enter your password: 19757979
Invalid password.
Password must contain:
- At least 1 digit (0-9)
- At least 1 lowercase letter (a-z)
- At least 1 uppercase letter (A-Z)
- At least 1 special character
- No whitespace characters
- Length between 6 and 20 characters

Enter your password:
```

When a user attempts to log in to Golden Screen Cinema without first creating an account. To log in to their account, users must enter their mobile number and password. If the password meets all of the requirements, the user will be taken to the next step.

```
[2]-Sign up
[3]-Show available Movies
[4]-Quit the program
Select an option: 5
Invalid input. Please enter within the range (1-4): 1




-------------------------------------------------------------------------------------------------------
                                        Log in
-------------------------------------------------------------------------------------------------------

Enter mobile number: 01162293464

Enter your password: 19757979
Invalid password.
Password must contain:
- At least 1 digit (0-9)
- At least 1 lowercase letter (a-z)
- At least 1 uppercase letter (A-Z)
- At least 1 special character
- No whitespace characters
- Length between 6 and 20 characters

Enter your password: Qairel@19757979
Incorrect mobile number or password. (1-try again or 2-sign up):
```

Even if the user enters the correct password requirements, they cannot access the Golden Screen Cinema because they do not yet have an account. There will be an input validation that says to press 1 to try again if the user key contains incorrect information and 2 to sign up and create a new account as a new user.

```


-------------------------------------------------------------------------------------------
                                    Sign up
-------------------------------------------------------------------------------------------

Enter your full name: QAIREL QAYYUM BIN MUHAMAD RIDHUAN

Enter mobile number: 01162293464

Enter your email: qayyumqairel1811@gmail.com

Enter your password: 181103Qairel
```

These are the requirements and user needs to fill before creating a new account.

```
-------------------------------------------- Now Showing --------------------------------------------
No   Name                                          Genre       Duration     Language  Classification
[1]  Megan                                         Horror      1 h 42 mins     ENG       P13
[2]  Puss In Boots: The Last Wish                  Animation   1 h 29 mins     ENG       P13
[3]  Avatar: The Way of Water                      Action      3 h 12 mins     ENG       P13
[4]  Mechamato Movie                               Animation   2 h 2 mins      ENG       P13
[5]  Sword Art Online The Movie - Progressive - SODN Animation   1 h 40 mins     JPN       P13
Enter any movie number:
```

After creating a new account, the user will see a menu that shows what movies are currently showing and available, as well as all of the details for each movie.

```
5]  Sword Art Online The Movie - Progressive - SODN Animation   1 h 40 mins     JPN       P13
nter any movie number: df
nvalid input. Please enter an integer (1-5):
```

User need to enter that range of number only to select any movies

```
Invalid input. Please enter an integer (1-5): 3



Kluang -Kluang Mall
Avatar: The Way of Water
Showtimes and dates:
1. 11:15 AM 29 Jan
2. 11:15 AM 30 Jan
Enter the showtime number: 2



Cinema: Kluang -Kluang Mall
Movie Title: Avatar: The Way of Water
Showtime and date: 11:15 AM 30 Jan
Hall: 1   - 2D Digital
```

The user has entered number 3, indicating that the movie will be avatar the way of water, and the user must also select the date and time of the selected movie.

```
------------------------------
          S C R E E N
  0  1  2  3  4  5  6  7  8  9
A A0 A1 A2 A3 A4 A5 A6 A7 A8 A9
B B0 B1 B2 B3 B4 B5 B6 B7 B8 B9
C C0 C1 C2 C3 C4 C5 C6 C7 C8 C9
D D0 D1 D2 D3 D4 D5 D6 D7 D8 D9
E E0 E1 E2 E3 E4 E5 E6 E7 E8 E9
F F0 F1 F2 F3 F4 F5 F6 F7 F8 F9
G G0 G1 G2 G3 G4 G5 G6 G7 G8 G9
H H0 H1 H2 H3 H4 H5 H6 H7 H8 H9
I I0 I1 I2 I3 I4 I5 I6 I7 I8 I9
J J0 J1 J2 J3 J4 J5 J6 J7 J8 J9

Enter seat row (e.g. A): c
Invalid row selected. Please choose a valid row.

Enter seat row (e.g. A): A

Enter seat column (e.g. 0): 7
Do you want to add more seats (1-Yes 2-No): 1

Enter seat row (e.g. A): A

Enter seat column (e.g. 0): 7
Selected seat is booked. Please enter the available seat only.

Enter seat row (e.g. A):
```

This is the section where the user must select their seat number based on the row of seats. If the user has already booked a seat and wishes to book another, but enters the same seat number as before, there will be input validation that says the selected seat is booked, please enter the available seat only, so the user must enter another seat that has not yet been booked.

```
Enter seat row (e.g. A): A

Enter seat column (e.g. 0): 8
Do you want to add more seats (1-Yes 2-No): 2

Seat(s): A7 A8

Confirm the selected seat(s) (1-Yes 2-No): 1

Seat(s): A7 A8
```

```
How many adults (13+): 2

How many Children (3-12): 0

How many have disabilities (OKU): 0

How many seniors (55+): 0

Cinema: Kluang -Kluang Mall
Movie Title: Avatar: The Way of Water
Showtime and date: 11:15 AM 30 Jan
Hall: 1  - 2D Digital

Seat(s): A7 A8
Adult:     2 x RM12.0
```

So now comes the part where the user is asked how many children, adults, disabled people, and seniors are present based on the number of seats reserved.

```
************************************************
                Transcation Detail


Cinema: Kluang -Kluang Mall
Movie Title: Avatar: The Way of Water
Showtime and date: 11:15 AM 30 Jan
Hall: 1  - 2D Digital

Seat(s): A7 A8
Trans Date and Time: 06-02-2023 14:17:44
Trans ID: 1676424122675200899
Transaction amount(RM): 24.0
Name: QAIREL QAYYUM BIN MUHAMAD RIDHUAN
Email: qayyumqairel1811@gmail.com
************************************************

Enter the payment method (1-Credit Card/Debit Card 2-Online Banking): 2



Making payment of RM24.0 using online banking
Transfer to: Golden Screen Cinema Sdn Bhd

[1]-Maybank
[2]-CIMB Bank
[3]-Public Bank
[4]-Hong Leong Bank
[5]-RHB Bank
[6]-AmBank
[7]-Alliance Bank
[8]-Bank Islam
[9]-Bank Rakyat
[10]-Affin Bank
Select your bank (1-10):
```

Before entering the full payment, user must select one of two payment methods which are credit card or online banking.

```
Enter username: qairel181103
Enter password: 19757979qairel
Invalid password.
Password must contain:
- At least 1 digit (0-9)
- At least 1 lowercase letter (a-z)
- At least 1 uppercase letter (A-Z)
- At least 1 special character
- No whitespace characters
- Length between 6 and 20 characters
Enter password: Qayyum@19757979
Enter payment amount: RM24
```

If the user chooses online banking, these requirements must be met before proceeding with the payment process.

```
**************************************************

                    RECEIPT

Reference number:     1676425799101
Transaction status:   Successful
Trans Date and Time:  06-02-2023 14:17:44
From:                 1674749373301899 Saving account
To:                   Golden Screen Cinema Sdn Bhd

--------------------------------------------------
Total (RM):                          24.0
--------------------------------------------------

**************************************************




**************************************************
Transaction status: Successful
Name:               QAIREL QAYYUM BIN MUHAMAD RIDHUAN
Email:              qayyumqairel1811@gmail.com
Mobile:             01162293464
Amount Paid(RM):    24.0
Payment Method:     Online banking
--------------------------------------------------
```

Here is the preferred receipt after paying for the chosen movie.

```
**************************************************
|
                        RECEIPT

Reference number:     1676425799101
Transaction status:   Successful
Trans Date and Time:  06-02-2023 14:17:44
From:                 1674749373301899 Saving account
To:                   Golden Screen Cinema Sdn Bhd

--------------------------------------------------
Total (RM):                              24.0
--------------------------------------------------


**************************************************




**************************************************
Transaction status:  Successful
Name:                QAIREL QAYYUM BIN MUHAMAD RIDHUAN
Email:               qayyumqairel1811@gmail.com
Mobile:              01162293464
Amount Paid(RM):     24.0
Payment Method:      Online banking
--------------------------------------------------
```

```
--------------------------------------------------
               Order details:
--------------------------------------------------

Cinema: Kluang -Kluang Mall
Movie Title: Avatar: The Way of Water
Showtime and date: 11:15 AM 30 Jan
Hall: 1  - 2D Digital

Seat(s): A7 A8
      You can show this receipt at the
          counter to enter the cinema.
```

Appendix

- **Admin.java**

```
package cinemaProject;


import java.util.Scanner;


import cinemaProject.Cinema.Hall;


import java.util.ArrayList;
import java.util.Comparator;
import java.util.Map;


public class Admin {
   private String adminNo = "admin";
   private String adminPword = "Admin@010203";


   public String getAdminNo() {
      return adminNo;
   }


   public String getAdminPword() {
      return adminPword;
   }


   public static boolean isAdmin(String input_mobile, String input_pword) {
       Admin admin = new Admin();
      if(input_mobile.equals(admin.getAdminNo()) && input_pword.equals(admin.adminPword))
```

```
            return true;
        return false;
    }

    public void updateCustomer(ArrayList<Customer> custlist, Scanner scan) {
        System.out.print("Enter the customer name to update: ");
        String name = scan.nextLine();

        Customer customer = null;
        for (Customer c : custlist) {
            if (c.getfName().equalsIgnoreCase(name)) {
                customer = c;
                break;
            }
        }

        if (customer == null) {
            System.out.println("No customer found with name " + name);
            return;
        }

        System.out.print("Enter new name: ");
        String newName = scan.nextLine();
        customer.setfName(newName);

        System.out.print("Enter new mobile number: ");
        String newMobile = scan.nextLine();
```

```
    customer.setMobileNo(newMobile);


    System.out.print("Enter new email: ");
    String newEmail = scan.nextLine();
    customer.setEmail(newEmail);


    System.out.print("Enter new password: ");
    String newPassword = scan.nextLine();
    customer.setPassword(newPassword);


    System.out.println("Customer details updated successfully!");
}



public void insertMovie(ArrayList<Cinema> cinema, ArrayList<Movie> movies, ArrayList<SalesReport>
  sales, Scanner scan) {


    System.out.print("Enter the movie title: ");
    String titleMovie = scan.nextLine();


    System.out.print("Enter the movie genre: ");
    String genreMovie = scan.nextLine();


    System.out.print("Enter the movie duration: ");
    String durationMovie = scan.nextLine();


    System.out.print("Enter the movie language: ");
```

```
String langMovie = scan.nextLine();


System.out.print("Enter the movie classification: ");
String classifMovie = scan.nextLine();


System.out.print("Enter the number of showtime and showdate: ");
int numberShowtime = Booking.getValidInt(1,20, scan);


String[] showtime = new String[numberShowtime];
String[] showdate = new String[numberShowtime];
for(int i=0;i<numberShowtime;i++) {
   System.out.print("Enter the movie showtime "+(i+1)+" (hh:mm (AM/PM)): ");
   showtime[i] = scan.nextLine();
   System.out.print("Enter the movie showdate "+(i+1)+" (dd month): ");
   showdate[i] = scan.nextLine();
}


movies.add(new Movie(titleMovie,genreMovie,durationMovie,langMovie,classifMovie));


System.out.print("Enter the branch name of the cinema that show this movie: ");
String cinemaName = scan.nextLine();
Cinema newCinema = new Cinema(cinemaName);
newCinema.addMovie(new Movie(titleMovie,genreMovie,durationMovie,langMovie,classifMovie), showtime,
showdate);
cinema.add(newCinema);


System.out.print("Enter the hall number that show this movie: ");
```

```
    int number = Booking.getValidInt(1,20, scan);

    System.out.print("Enter the number of row inside the hall "+number+" : ");
    int row = Booking.getValidInt(1,10, scan);

    System.out.print("Enter the number of column inside the hall "+number+" : ");
    int column = Booking.getValidInt(1,10, scan);

    System.out.print("Enter the name of the hall "+number+" : ");
    String hallName = scan.nextLine();
    Hall newHall = newCinema.new Hall(cinemaName, number, row, column, hallName);
    newHall.addMovie(newCinema.getMovie(titleMovie));
    newCinema.addHall(newHall);

    SalesReport newSales = new SalesReport();
    newSales.addSaleData(cinemaName, titleMovie, 0);
    sales.add(newSales);
}

public void updateMovie(ArrayList<Cinema> cinema, ArrayList<Movie> movies, Scanner scan) {
    System.out.print("Enter the title of the movie to update: ");
    String title = scan.nextLine();
    int movieIndex = -1;
    int cinemaIndex = -1;

    // search for the movie in the list of movies
    for (int i = 0; i < movies.size(); i++) {
```

```
        if (movies.get(i).getMovieName().equalsIgnoreCase(title)) {
            movieIndex = i;
            break;
        }
    }

    // if the movie is not found, return
    if (movieIndex == -1) {
        System.out.println("Movie not found!");
        return;
    }

    // search for the cinema that shows the movie
    for(Cinema cinemas: cinema) {
        for (Map.Entry<Movie, Map<String[], String[]>> entry : cinemas.getMovieAndShowtimes().entrySet()) {
            Movie movie = entry.getKey();
            if (movie.getMovieName().equals(movies.get(movieIndex).getMovieName())) {
                cinemaIndex = cinema.indexOf(cinemas);
                break;
            }
        }
    }



    // if the cinema is not found, return
    if (cinemaIndex == -1) {
        System.out.println("Cinema not found!");
```

```
        return;
    }

    System.out.print("Enter the updated movie genre: ");
    String genreMovie = scan.nextLine();
    movies.get(movieIndex).setMovieGenre(genreMovie);

    System.out.print("Enter the updated movie duration (hh mm): ");
    String durationMovie = scan.nextLine();
    movies.get(movieIndex).setDuration(durationMovie);

    System.out.print("Enter the updated movie language: ");
    String langMovie = scan.nextLine();
    movies.get(movieIndex).setLanguage(langMovie);

    System.out.print("Enter the updated movie classification: ");
    String classifMovie = scan.nextLine();
    movies.get(movieIndex).setClassif(classifMovie);

    // update the showtimes and showdates
    System.out.print("Enter the number of updated showtimes and showdates: ");
    int numberShowtime = Booking.getValidInt(1,20, scan);

    String[] showtime = new String[numberShowtime];
    String[] showdate = new String[numberShowtime];
    for (int i = 0; i < numberShowtime; i++) {
        System.out.print("Enter the updated movie showtime " + (i + 1) + " (hh:mm (AM/PM)): ");
```

```
        showtime[i] = scan.nextLine();
        System.out.print("Enter the updated movie showdate " + (i + 1) + " (dd mm): ");
        showdate[i] = scan.nextLine();
    }

    // update the movie details in the cinema
    cinema.get(cinemaIndex).addMovie(movies.get(movieIndex), showtime, showdate);

}


public void deleteMovie(ArrayList<Movie> movieList, Scanner scan) {
    System.out.print("Enter the movie name to delete: ");
    String name = scan.nextLine();

    for (Movie movie : movieList) {
      if (movie.getMovieName().equals(name)) {
          movieList.remove(movie);
          System.out.println("Movie deleted successfully.");
          break;
      } else {
        System.out.println("There is no movie "+ name +" in the system.");
      }
    }
  }

public void updateCinema(ArrayList<Cinema> cinemaList, Scanner scan) {
```

```
    System.out.print("Enter the cinema name to update: ");
    String name = scan.nextLine();

    for (Cinema cinema : cinemaList) {
      if (cinema.getBranchName().equals(name)) {
        System.out.println("Updating details for " + cinema.getBranchName());
        System.out.print("Enter new name: ");
        cinema.setBranchName(scan.nextLine());
        System.out.println("Cinema details updated successfully.");
        break;
      }
    }
  }

public void showReportToday(ArrayList<SalesReport> sales) {
    System.out.println("\n\n");
    System.out.println("-----------------------------------------------------------------------------------------");
    System.out.println("                                Summary");
    System.out.println("-----------------------------------------------------------------------------------------\n");
    for (SalesReport sale : sales) {
        System.out.println("Cinema Name:              " + sale.getCinemaName());
        System.out.println("Movie Name:               " + sale.getMovieName());
        System.out.println("Total Sale (RM):          " + sale.getTotalSale());
        System.out.println("Quantity Adult:           " + sale.getQuantityAdult());
        System.out.println("Quantity Child:           " + sale.getQuantityChild());
        System.out.println("Quantity Oku:             " + sale.getQuantityOku());
        System.out.println("Quantity Senior:          " + sale.getQuantitySenior());
```

```
            System.out.println("");
        }



        System.out.println("-----------------------------------------------------------------------------------------------");
        System.out.println("                                    Ranking");
        System.out.println("-----------------------------------------------------------------------------------------------\n");
        sales.sort(new Comparator<SalesReport>() {
            public int compare(SalesReport sr1, SalesReport sr2) {
                return Integer.compare(sr2.getTotalTicket(), sr1.getTotalTicket());
            }
        });

        for (SalesReport sale : sales) {
            System.out.println("Movie Name:                 " + sale.getMovieName());
            System.out.println("Total Ticket Sold:          " + sale.getTotalTicket());
            System.out.println("");
        }
    }



}
```

- **Booking.java**

```
package cinemaProject;



import cinemaProject.Cinema.Hall;
```

```
import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.Scanner;

import java.util.Map;

import java.util.Random;

import java.util.ArrayList;

import java.util.InputMismatchException;


public class Booking {


   public static void main(String[] args) {

      Scanner scan = new Scanner(System.in);


      ArrayList<Customer> custList = new ArrayList<Customer>();

      ArrayList<Movie> movieList = new ArrayList<Movie>();

      ArrayList<Cinema> cinemaList = new ArrayList<Cinema>();

      ArrayList<SalesReport> salesCinema = new ArrayList<SalesReport>();


      //initialize existed customer

      custList.add(new Customer("Wan Zarin Wan
     Nizam","0192891001","wanzarin@gmail.com","001122_Aa"));
```

```
//add movie to array movie list

movieList.add(new Movie("Megan","Horror","1 h 42 mins","ENG","P13"));

movieList.add(new Movie("Puss In Boots: The Last Wish","Animation","1 h 29 mins","ENG","P13"));

movieList.add(new Movie("Avatar: The Way of Water","Action","3 h 12 mins","ENG","P13"));

movieList.add(new Movie("Mechamato Movie","Animation","2 h 2 mins","ENG","P13"));

movieList.add(new Movie("Sword Art Online The Movie – Progressive - SODN","Animation","1 h 40 mins","JPN","P13"));



//initialize the cinema object

Cinema cinemaMidV = new Cinema("Johor Bahru - The Mall, Mid Valley Southkey");

Cinema cinemaKluang = new Cinema("Kluang -Kluang Mall");

Cinema cinemaParadigm = new Cinema("Johor Bahru – Paradigm Mall");



//add movie to cinema object

cinemaMidV.addMovie(new Movie("Megan","Horror","1 h 42 mins","ENG","P13"), new String[]{"02:45 PM","02:45 PM"}, new String[] {"28 Jan","29 Jan"});

cinemaMidV.addMovie(new Movie("Puss In Boots: The Last Wish","Animation","1 h 29 mins","ENG","P13"), new String[]{"05:30 PM","05:30 PM"}, new String[] {"27 Jan","28 Jan"});

cinemaKluang.addMovie(new Movie("Avatar: The Way of Water","Action","3 h 12 mins","ENG","P13"), new String[]{"11:15 AM","11:15 AM"}, new String[] {"29 Jan","30 Jan"});

cinemaKluang.addMovie(new Movie("Mechamato Movie","Animation","2 h 2 mins","ENG","P13"), new String[]{"10:30 AM","10:30 AM"}, new String[] {"27 Jan","28 Jan"});

cinemaParadigm.addMovie(new Movie("Sword Art Online The Movie – Progressive - SODN","Animation","1 h 40 mins","JPN","P13"), new String[]{"12:25 PM","12:25 PM"}, new String[] {"26 Jan","27 Jan"});
```

```
//initialize hall in each cinema

Hall hallMidv1 = cinemaMidV.new Hall("Johor Bahru - The Mall, Mid Valley Southkey",1, 10, 10, "Play Plus");

hallMidv1.addMovie(cinemaMidV.getMovie("Megan"));

Pricing megan = new Pricing(21.00,10.00,10.00,10.00);

cinemaMidV.setPrice(megan);

cinemaMidV.addHall(hallMidv1);


Hall hallMidv2 = cinemaMidV.new Hall("Johor Bahru - The Mall, Mid Valley Southkey",2, 10, 10, "2D Digital");

hallMidv2.addMovie(cinemaMidV.getMovie("Puss In Boots: The Last Wish"));

Pricing pussInBoot = new Pricing(22.00,13.00,13.00,13.00);

cinemaMidV.setPrice(pussInBoot);

cinemaMidV.addHall(hallMidv2);


Hall hallKluang1 = cinemaKluang.new Hall("Kluang -Kluang Mall",1, 10, 10, "2D Digital");

hallKluang1.addMovie(cinemaKluang.getMovie("Avatar: The Way of Water"));

Pricing avatar = new Pricing(12.00,9.00,9.00,9.00);

cinemaKluang.setPrice(avatar);

cinemaKluang.addHall(hallKluang1);


Hall hallKluang2 = cinemaKluang.new Hall("Kluang -Kluang Mall",2, 10, 10, "2D Digital");

hallKluang2.addMovie(cinemaKluang.getMovie("Mechamato Movie"));

Pricing mechamato = new Pricing(12.00,9.00,9.00,9.00);
```

```
cinemaKluang.setPrice(mechamato);

cinemaKluang.addHall(hallKluang2);


Hall hallPara1 = cinemaParadigm.new Hall("Johor Bahru - The Mall, Mid Valley Southkey",1, 10, 10, "2D Digital");

hallPara1.addMovie(cinemaParadigm.getMovie("Sword Art Online The Movie – Progressive - SODN"));

Pricing sword = new Pricing(20.00,9.00,9.00,9.00);

cinemaParadigm.setPrice(sword);

cinemaParadigm.addHall(hallPara1);


//add cinema to array cinema list

cinemaList.add(cinemaMidV);

cinemaList.add(cinemaKluang);

cinemaList.add(cinemaParadigm);


//create SalesReport object for each cinema

SalesReport salesMidv1 = new SalesReport();

salesMidv1.addSaleData(cinemaMidV.getBranchName(), cinemaMidV.getMovie("Megan").getMovieName(), 0);

salesCinema.add(salesMidv1);


SalesReport salesMidv2 = new SalesReport();

salesMidv2.addSaleData(cinemaMidV.getBranchName(), cinemaMidV.getMovie("Puss In Boots: The Last Wish").getMovieName(), 0);
```

```
salesCinema.add(salesMidv2);


SalesReport salesKluang1 = new SalesReport();

salesKluang1.addSaleData(cinemaKluang.getBranchName(), cinemaKluang.getMovie("Avatar: The Way of Water").getMovieName(), 0);

salesCinema.add(salesKluang1);


SalesReport salesKluang2 = new SalesReport();

salesKluang2.addSaleData(cinemaKluang.getBranchName(), cinemaKluang.getMovie("Mechamato Movie").getMovieName(), 0);

salesCinema.add(salesKluang2);


SalesReport salesParadigm = new SalesReport();

salesParadigm.addSaleData(cinemaParadigm.getBranchName(), cinemaParadigm.getMovie("Sword Art Online The Movie – Progressive - SODN").getMovieName(), 0);

salesCinema.add(salesParadigm);


int next;

do {

    int loop;


    do {


        //main menu
```

```
        System.out.println("----------------------------------Welcome to Golden Screen
Cinema----------------------------------\n");

        System.out.print("[1]-Log in\n[2]-Sign up\n[3]-Show available Movies\n[4]-Quit the program\nSelect an
option: ");

        int option = getValidInt(1,4,scan);



        //instantiate related object for the booking process

        Customer cust = null;

        ArrayList<String> selectedSeats = new ArrayList<>();

        Movie selectedMovie = null;

        Cinema selectedCinema = null;

        Hall selectedHall = null;



        //log in for existed customer

        if(option == 1) {

            Admin admin = new Admin();

            Customer existedCust = Customer.log_in(custList,scan,admin);



            //check if admin is logging in

            if(existedCust.getMobileNo().equals(admin.getAdminNo()) &&
existedCust.getPassword().equals(admin.getAdminPword())) {



                int adminChoice;
```

```
            do {

            System.out.println("\nWelcome Admin!\n1.Update the customer detail\n2.Update movie
detail\n3.Delete Movie\n4.Update cinema detail\n5.Show report today");

            System.out.print("Select an option: ");

            adminChoice = getValidInt(1,5,scan);



            if(adminChoice == 1)

               admin.updateCustomer(custList, scan);



            else if(adminChoice == 2)

               admin.updateMovie(cinemaList, movieList, scan);



            else if(adminChoice == 3)

               admin.deleteMovie(movieList, scan);



            else if(adminChoice == 4)

               admin.updateCinema(cinemaList, scan);



            else

               admin.showReportToday(salesCinema);



            System.out.print("Back to menu? (1-Yes 2-No): ");

            adminChoice = getValidInt(1, 2, scan);
```

```
        if(adminChoice == 2)
            break;

        }while(adminChoice == 1);
    }

    cust = existedCust;
}

//sign in for new customer
if(option == 2) {
    //sign in for new customer
    Customer newCust = Customer.sign_in(custList,scan);
    cust = newCust;
}

//continue to booking process for existed customer, new customer or unknown customer
if(option == 3 || cust != null){

    //display available movie that contained in movieList
    System.out.println("\n\n");
    Movie.displayMovie(movieList);
```

```
            option = getValidInt(1,movieList.size(), scan);


            //instantiate selected movie

            selectedMovie = Movie.getMovie(movieList,(option-1));


            for(Cinema cinemas: cinemaList) {

                //get each movie and showtime that contained inside the cinema

                for (Map.Entry<Movie, Map<String[], String[]>> entry :
cinemas.getMovieAndShowtimes().entrySet()) {

                    Map<String[], String[]> showtimes = entry.getValue();

                    Movie movie = entry.getKey();


                    //search cinema that contained the selected movie

                    if (movie.getMovieName().equals(selectedMovie.getMovieName())) {

                        System.out.println("\n\n");

                        System.out.println("\n"+cinemas.getBranchName());

                        System.out.println(movie.getMovieName());

                        System.out.println("Showtimes and dates:");


                        for (Map.Entry<String[], String[]> showtime : showtimes.entrySet()) {


                            //get showtime and date for that particular movie inside the cinema

                            String[] time = showtime.getKey();
```

```
                    String[] date = showtime.getValue();



                    for(int i=0; i<time.length; i++ ) {

                       System.out.println((i+1) + ". " + time[i] + " " + date[i]);

                    }



                    //get selected showtime and date from user

                    System.out.print("Enter the showtime number: ");

                    option = getValidInt(1,time.length, scan);

                    selectedCinema = new Cinema(cinemas.getBranchName());



                    //instantiate and initialize selected cinema

                    selectedHall = cinemas.getHall(selectedMovie);

                    selectedCinema.addMovie(movie, new String[] {"time: ",time[option-1]}, new String[]
{"Date: ",date[option-1]});

                    selectedHall.addMovie(movie);

                    selectedCinema.addHall(selectedHall);

                    selectedCinema.setPrice(cinemas.getPrice());

                }



            }

        }

    }
```

```
//check if selected cinema object instantiated successfully
if (selectedCinema != null) {
    System.out.println("\n\n");
    System.out.println(selectedCinema.toString());
} else {
    System.out.println("Invalid option!");
}

//declare related variable to booking seat process
int child=0, adult=0, senior=0, oku=0;
int numberOfSeat=0;

//check if selected hall instantiated successfully
if(selectedHall != null) {
    do {
        System.out.println("\n\n\n");
        System.out.println("-----------------------------");
        System.out.println("         S C R E E N");
        selectedHall.displaySeats();

        //declare row and column that would be entered by user
        int row, column;
```

```
char rowChar;
String rowInput;

do {

    do {
        System.out.print("\nEnter seat row (e.g. A): ");
        rowInput = scan.next();

        //check if row is not a single character
        if (rowInput.length() != 1) {
            System.out.println("Invalid input. Please enter a single character.");
        } else {
            //convert string row to int
            rowChar = rowInput.charAt(0);
            row = rowChar - 'A';

            //check if the row within the row range
            if(row < 0 || row >= selectedHall.getSeatRow().length ) {
                System.out.println("Invalid row selected. Please choose a valid row.");
            } else {
                break;
            }
```

```
        }


    }while(true);


    System.out.print("\nEnter seat column (e.g. 0): ");

    column = getValidInt(0,selectedHall.getSeatColumn().length, scan);


    //check if the seat was booked or not

    if (selectedHall.getSeat(row, column) != 1) {

        selectedHall.setSeat(row, column, 1);


        Cinema.setSeatInCinemaArray(cinemaList,selectedMovie.getMovieName(),row,column,1);


        //collect the entered seat by concatinating row and column

        String userSeat = rowInput+""+column;

        selectedSeats.add(userSeat);

        numberOfSeat ++;


        System.out.print("Do you want to add more seats (1-Yes 2-No): ");

        option = getValidInt(1,2, scan);

    } else {

        System.out.println("Selected seat is booked. Please enter the available seat only.");

    }
```

```
            }while(option == 1);


            //display selected seats
            Hall.displaySelectedSeat(selectedSeats);


            System.out.print("\n\nConfirm the selected seat(s) (1-Yes 2-No): ");
            option = getValidInt(1,2, scan);


            if(option == 1 ) {
               break;
            } else {
               //Loop through all the selected seats and set their value back to 0 (i.e. not booked)
               for(String seat : selectedSeats) {
                  int rowToDel = seat.charAt(0) - 'A';
                  int columnToDel = Integer.parseInt(seat.substring(1));
                  selectedHall.setSeat(rowToDel, columnToDel, 0);

Cinema.setSeatInCinemaArray(cinemaList,selectedMovie.getMovieName(),rowToDel,columnToDel,0);


               }


               //Clear the selected seats list
```

```
                    selectedSeats.clear();


                    //Reset the number of selected seats to 0
                    numberOfSeat = 0;


                    System.out.print("Seats have been unbooked. Do you want to book the seat again (1-Yes 2-No):
");
                    option = getValidInt(1,2, scan);
                    if(option == 2)
                        selectedSeats = null;
                }


            }while(option == 1);


        }


        if(selectedSeats != null) {
            //display selected seats
            Hall.displaySelectedSeat(selectedSeats);


            do {
                System.out.print("\n\nHow many adults (13+): ");
                adult = getValidInt(0,numberOfSeat,scan);
```

```
            System.out.print("\nHow many Children (3-12): ");
            child = getValidInt(0,numberOfSeat-adult,scan);

            System.out.print("\nHow many have disabilities (OKU): ");
            oku = getValidInt(0,numberOfSeat-adult-child,scan);

            System.out.print("\nHow many seniors (55+): ");
            senior = getValidInt(0,numberOfSeat-adult-child-oku,scan);

            //check if number of people equals
            if(adult+child+oku+senior != numberOfSeat)
                System.out.print("The total of people is not corresponds to the number of selected
seats.\nPlease ensure that the total number of each category corresponds to the number of selected seats.");

        }while(adult+child+oku+senior != numberOfSeat);

        //display selected cinema, movie and seats
        System.out.println(selectedCinema.toString());
        Hall.displaySelectedSeat(selectedSeats);

        double childPrice = child*selectedCinema.getPrice().getChildPrice();
        double adultPrice = adult*selectedCinema.getPrice().getAdultPrice();
```

```
            double okuPrice = oku*selectedCinema.getPrice().getOkuPrice();
            double seniorPrice = senior*selectedCinema.getPrice().getSeniorPrice();

            System.out.println("");

            if(adultPrice > 0)
                System.out.println("Adult:     "+adult+" x RM"+selectedCinema.getPrice().getAdultPrice());

            if(childPrice > 0)
                System.out.println("Children:  "+child+" x RM"+selectedCinema.getPrice().getChildPrice());

            if(okuPrice > 0)
                System.out.println("OKU:       "+oku+" x RM"+selectedCinema.getPrice().getOkuPrice());

            if(seniorPrice > 0)
                System.out.println("Senior:    "+senior+" x RM"+selectedCinema.getPrice().getSeniorPrice());

            System.out.print("\nWould you like to proceed with the payment process? (1-Yes 2-No): ");
            option = getValidInt(1,2,scan);

            Payment pay = null;

            if(option == 1) {
```

```
                if(cust == null) {

                    System.out.print("Please sign in/log in to proceed with the payment process (1-Sign in 2-Log
in): ");

                    option = getValidInt(1,2,scan);



                    if(option == 1)

                        cust = Customer.sign_in(custList,scan);

                    else

                        cust = Customer.log_in(custList, scan, null);

                }



                double amount =
senior*selectedCinema.getPrice().getSeniorPrice()+oku*selectedCinema.getPrice().getOkuPrice()+child*selec
tedCinema.getPrice().getChildPrice()+adult*selectedCinema.getPrice().getAdultPrice();



                LocalDateTime currentTime = LocalDateTime.now();

                DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

                String formattedTime = currentTime.format(formatter);

                Random random = new Random();

                long rand_int = System.currentTimeMillis() + random.nextInt(1000000000);



                System.out.println("\n\n\n");

                System.out.println("**************************************************");

                System.out.println("               Transcation Detail\n");
```

```
                System.out.println(selectedCinema.toString());

                Hall.displaySelectedSeat(selectedSeats);

                System.out.println("\nTrans Date and Time: " + formattedTime);

                System.out.println("Trans ID: " + rand_int*999999);

                System.out.println("Transaction amount(RM): "+amount);

                System.out.println("Name: "+cust.getfName());

                System.out.println("Email: "+cust.getEmail());

                System.out.println("************************************************");


                System.out.print("\nEnter the payment method (1-Credit Card/Debit Card 2-Online Banking):
");

                option = getValidInt(1,2,scan);


                pay = new Payment(cust.getfName(),cust.getEmail(),cust.getMobileNo(),amount,option);

                System.out.println("\n\n\n");

                pay.makePayment();


                if(option == 1) {


                   String cardNumber, expiryDate, nameCard, cvv;

                   boolean valid;


                   do {
```

```
                    do {
                        System.out.print("Enter your card number:");
                        cardNumber = scan.nextLine();

                        if(!Customer.validateStringNumber(16, 16, cardNumber))
                            System.out.println("Invalid card number. The card number must contains 16
number.");

                    }while(!Customer.validateStringNumber(16, 16, cardNumber));

                    do {
                        System.out.print("Enter the expiry date (MM/YY):");
                        expiryDate = scan.nextLine();

                        valid = Payment.isValidExpirationDate(expiryDate);
                        if(valid)
                            break;

                    }while(!valid);

                    do {
                        System.out.print("Enter your name on card: ");
                        nameCard = scan.nextLine();
```

```
    if (!Customer.validateString(nameCard))
        System.out.println("Invalid name. The name must contains alphabet only.");

}while(!Customer.validateString(nameCard));

do {
    System.out.print("Enter the CVC/CVV2: ");
    cvv = scan.nextLine();

    valid = Payment.isValidCvv(cvv);
    if(valid)
        break;

}while(!valid);

if(pay.cardPayment(scan)) {
    System.out.println("\n\n\n");
    System.out.println("*************************************************");
    System.out.println("");
    System.out.println("                  RECEIPT\n");
    System.out.println("Reference number:    "+rand_int);
    System.out.println("Transaction status:  Successful");
```

```
                System.out.println("Trans Date and Time: " + formattedTime);
                System.out.println("Name:              " + nameCard);
                System.out.println("From:              " + cardNumber);
                System.out.println("To:              Golden Screen Cinema Sdn Bhd\n");
                System.out.println("-----------------------------------------------");
                System.out.println("Total (RM):                    " + amount);
                System.out.println("-----------------------------------------------");
                System.out.println("\n\n***********************************************");

                break;
            }
            else {
                System.out.print("Try again or cancel the transaction (1-Try again 2-Cancel): ");
                option = getValidInt(1, 2, scan);
            }

        }while(option == 1);

    } else {
        do {
            String accNumb, username, password;

            System.out.println("Transfer to: Golden Screen Cinema Sdn Bhd\n");
```

```
System.out.println("[1]-Maybank\r\n"
        + "[2]-CIMB Bank\r\n"
        + "[3]-Public Bank\r\n"
        + "[4]-Hong Leong Bank\r\n"
        + "[5]-RHB Bank\r\n"
        + "[6]-AmBank\r\n"
        + "[7]-Alliance Bank\r\n"
        + "[8]-Bank Islam\r\n"
        + "[9]-Bank Rakyat\r\n"
        + "[10]-Affin Bank");
System.out.print("Select your bank (1-10): ");
int bank = getValidInt(1,10,scan);


do {
   System.out.print("\nEnter username: ");
   username = scan.nextLine();


   if(username == null)
      System.out.println("Invalid username. Please enter valid username.");


}while(username == null);


do {
```

```
            System.out.print("Enter password: ");
            password = scan.nextLine();

            if(!Customer.validatePassword(password)) {
                System.out.println("Invalid password.");
                System.out.println("Password must contain:");
                System.out.println("- At least 1 digit (0-9)");
                System.out.println("- At least 1 lowercase letter (a-z)");
                System.out.println("- At least 1 uppercase letter (A-Z)");
                System.out.println("- At least 1 special character");
                System.out.println("- No whitespace characters");
                System.out.println("- Length between 6 and 20 characters");
            }
        }while(!Customer.validatePassword(password));

        if(rand_int < 0)
            rand_int*=-99999999;
        if(pay.onlinePayment(username, password, scan)) {
            System.out.println("\n\n\n");
            System.out.println("************************************************");
            System.out.println("");
            System.out.println("                    RECEIPT\n");
            System.out.println("Reference number:    "+rand_int);
```

```
            System.out.println("Transaction status:  Successful");
            System.out.println("Trans Date and Time: " + formattedTime);
            System.out.println("From:              " + rand_int*999 + " Saving account");
            System.out.println("To:              Golden Screen Cinema Sdn Bhd\n");
            System.out.println("-----------------------------------------------");
            System.out.println("Total (RM):                    " + amount);
            System.out.println("-----------------------------------------------");
            System.out.println("\n\n***********************************************");
            System.out.println("\n\n\n\n\n\n\n");

        }
        else {
            System.out.print("Try again or cancel the transaction (1-Try again 2-Cancel): ");
            option = getValidInt(1, 2, scan);
        }
    }while(option == 1);
}

//display the transaction details and order details
pay.paymentSucceed(selectedCinema, selectedSeats);

//assignment for each customer category quantity to object salesReport
SalesReport chosenCinema = new SalesReport();
```

```
                chosenCinema.addSaleData(selectedCinema.getBranchName(), selectedMovie.getMovieName(),
amount);

                chosenCinema.addTicketData(adult, child, oku, senior);



                for(SalesReport sales: salesCinema) {

                   if(sales.getCinemaName().equals(chosenCinema.getCinemaName())) {

                      if(sales.getMovieName().equals(chosenCinema.getMovieName())) {

                         sales.addTicketData(adult, child, oku, senior);

                         sales.setTotalSale(amount);

                      }

                   }


                }




             }

          }

       } else

          break;


       System.out.print("Back to main menu? (1-Yes 2-No): ");

       loop = getValidInt(1,2,scan);
```

```
            System.out.println("\n\n\n");
        }while(loop == 1);


        System.out.print("Next customer? (1-Yes 2-No): ");
        next = getValidInt(1,2,scan);


    }while(next == 1);


}


public static int getValidInt(int min, int max, Scanner scan) {
        int number;
        while (true) {
            try {
                number = scan.nextInt();
                if (number >= min && number <= max) {
                    scan.nextLine();
                    return number;
                } else {
                    if(min != max) {
                        System.out.print("Invalid input. Please enter within the range ("+ min +"-" + max+"): ");
                        scan.nextLine();
                    } else if(min == max) {
```

```
                    System.out.print("Invalid input. Please enter within the range ("+ min +"): ");
                    scan.nextLine();
                }
            }
        } catch (InputMismatchException e) {
            if(min != max) {
                System.out.print("Invalid input. Please enter an integer ("+min+"-"+max+"): ");
                scan.nextLine();
            } else if(min == max) {
                System.out.print("Invalid input. Please enter this integer ("+ min +"): ");
                scan.nextLine();
            }
        }
    }
}
```

- **Cinema.java**

```
package cinemaProject;


import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
```

```
public class Cinema {
    private String branchName;
    private Map<Movie, Map<String[], String[]>> showtime;
    private ArrayList<Hall> hall;
    private Pricing price;

    public Cinema(String branchName) {
        this.branchName = branchName;
        this.showtime = new HashMap<>();
        this.hall = new ArrayList<>();
    }

    public void addMovie(Movie movie, String[] showtime, String[] showDate) {
        if (showtime.length != showDate.length) {
            System.out.println("The number of showtime and date should be the same");
            return;
        }

        Map<String[], String[]> movieShowtimes = new HashMap<>();
        movieShowtimes.put(showtime, showDate);

        this.showtime.put(movie, movieShowtimes);
```

```
}


public Movie getMovie(String movieTitle) {
    for(Map.Entry<Movie, Map<String[],String[]>> entry : showtime.entrySet()) {
        Movie movie = entry.getKey();
        if (movie.getMovieName().equals(movieTitle)) {
            return movie;
        }
    }
    return null;
}


public String getBranchName() {
    return branchName;
}


public Map<Movie, Map<String[], String[]>> getMovieAndShowtimes() {
    return showtime;
}


public void setBranchName(String name) {
    this.branchName = name;
}
```

```
public void addHall(Hall halls) {
    this.hall.add(halls);
  }

public Hall getHall(Movie movie) {
    for(Map.Entry<Movie, Map<String[],String[]>> entry : showtime.entrySet()) {
        Movie movies = entry.getKey();
        if(movies.getMovieName().equals(movie.getMovieName())) {
            for (Hall halls : hall) {
                if (halls.getMovie().getMovieName().equals(movie.getMovieName())) {
                    return halls;
                }
            }
        }
    }
    return null;
}

public static void setSeatInCinemaArray(ArrayList<Cinema> cinema, String movieTitle, int rows, int columns,
    int value) {
    for(Cinema cinemalist: cinema) {
        Movie movie = cinemalist.getMovie(movieTitle);
```

```
        if (movie != null && movie.getMovieName().equals(movieTitle)) {

            cinemalist.getHall(movie).setSeat(rows, columns, value);

        }

    }



}



public void setPrice(Pricing price) {

    this.price = price;

}



public Pricing getPrice() {

    return price;

}



public String toString() {

    StringBuilder result = new StringBuilder();

    for(Map.Entry<Movie, Map<String[],String[]>> entry : showtime.entrySet()) {

        Map<String[], String[]> show = entry.getValue();

        Movie movie = entry.getKey();



        for (Map.Entry<String[], String[]> entryShow : show.entrySet()) {

            String[] movieShowtimes = entryShow.getKey();
```

```
            String[] movieDate = entryShow.getValue();
            String time = movieShowtimes[1], date = movieDate[1];
            result.append("\nCinema: ").append(branchName);
            result.append("\nMovie Title: ").append(movie.getMovieName());
            result.append("\nShowtime and date: ").append(time).append(" ").append(date).append("\n");
            for(Hall halls: hall) {
                result.append("Hall: ").append(halls.getNumber()).append("  - ").append(halls.getName());
            }
        }
    }
    return result.toString();
}



class Hall extends Cinema {
    private int number;
    private String name;
    private int[][] seats;
    private Movie movie;

    public Hall(String branchName, int number, int rows, int columns, String name) {
        super(branchName);
        this.number = number;
```

```
        this.name = name;
        this.seats = new int[rows][columns];
    }

    public int getNumber() {
        return this.number;
    }

    public String getName() {
        return this.name;
    }

    public Movie getMovie() {
        return this.movie;
    }

    public int[][] getSeatRow(){
        return this.seats;
    }

    public int[] getSeatColumn(){
        int[] column = this.seats[0];
        return column;
```

```
    }

    public int getSeat(int row, int column) {
        return this.seats[row][column];
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setMovie(Movie movie) {
        this.movie = movie;
    }

    public void addMovie(Movie movie) {
         this.movie = movie;
    }

    public void displaySeats() {
```

```
        System.out.print("   ");
        for (int i = 0; i < seats.length; i++) {
            System.out.print(i + "   ");
        }
        System.out.println();

        for (int i = 0; i < seats[0].length; i++) {
            System.out.print((char)('A' + i) + "  ");
            for (int j = 0; j < seats.length; j++) {
                if (seats[i][j] == 1) {
                    System.out.print("X   ");
                } else {
                    System.out.print((char)('A' + i) + "" + j + "  ");
                }
            }
            System.out.println();
        }
    }

    public void setSeat(int row, int column, int value) {
        if (row >= 0 && row < seats.length && column >= 0 && column < seats[0].length) {
            seats[row][column] = value;
```

```
            } else {
                System.out.println("Invalid seat. Please enter available seats only.");
            }
        }

        public static void displaySelectedSeat(ArrayList<String> selectedSeats) {
            System.out.print("\nSeat(s): ");
            for(int i = 0; i < selectedSeats.size(); i++) {
              System.out.print(selectedSeats.get(i) + " ");
            }
          }

      }
}
```

- **Customer.java**

```
package cinemaProject;


import java.util.ArrayList;
import java.util.Scanner;
import java.util.regex.Pattern;


public class Customer {
```

```
private String fName, mobileNo, email, password;


Customer(String value_fName, String value_mobileNo, String value_email, String value_pword)
{
    this.fName = value_fName;
    this.mobileNo = value_mobileNo;
    this.email = value_email;
    this.password = value_pword;
}


public String getfName() {
    return fName;
}
public String getMobileNo() {
    return mobileNo;
}
public String getEmail() {
    return email;
}
public String getPassword() {
    return password;
}
```

```
public void setfName(String name) {

    this.fName = name;

}

public void setMobileNo(String mobile) {

    this.mobileNo = mobile;

}

public void setEmail(String mail) {

    this.email = mail;

}

public void setPassword(String pword) {

    this.password = pword;

}



public static Customer sign_in(ArrayList<Customer> custList, Scanner scan)

{

    String input_fName, input_mobileNo, input_email, input_pword;



    System.out.println("\n\n\n\n\n\n\n");

    System.out.println("-----------------------------------------------------------------------------------------");

    System.out.println("                                  Sign up");

    System.out.println("-----------------------------------------------------------------------------------------");



    do {
```

```
    System.out.print("\nEnter your full name: ");
    input_fName = scan.nextLine();


    if (validateString(input_fName) == false) {
        System.out.println("Invalid name. Name must contains alphabet only.");
    }
} while(validateString(input_fName) == false);


do {
    System.out.print("\nEnter mobile number: ");
    input_mobileNo = scan.nextLine();
    if (validateStringNumber(10,11,input_mobileNo) == false) {
        System.out.println("Invalid mobile number. Mobile number must contain 10-11 digits.");
    }
}while(validateStringNumber(10,11,input_mobileNo) == false);


do {
    System.out.print("\nEnter your email: ");
    input_email = scan.nextLine();
    if(validateEmail(input_email) == false) {
        System.out.println("Invalid email address. Please enter a valid email address in the format
example@domain.com.");
    }
```

```
    }while(validateEmail(input_email) == false);


    Admin admin = new Admin();
    do {
        System.out.print("\nEnter your password: ");
        input_pword = scan.nextLine();
        if(validatePassword(input_pword) == false || input_pword.equals(admin.getAdminPword())) {
            System.out.println("Invalid password.");
            System.out.println("Password must contain:");
            System.out.println("- At least 1 digit (0-9)");
            System.out.println("- At least 1 lowercase letter (a-z)");
            System.out.println("- At least 1 uppercase letter (A-Z)");
            System.out.println("- At least 1 special character");
            System.out.println("- No whitespace characters");
            System.out.println("- Length between 6 and 20 characters");
        }
    }while(validatePassword(input_pword) == false || input_pword.equals(admin.getAdminPword()));



    Customer custInfo = new Customer(input_fName,input_mobileNo,input_email,input_pword);
    custList.add(custInfo);
    return custInfo;
}
```

```
public static Customer log_in(ArrayList<Customer> custList, Scanner scan, Admin admin) {
    int ans;
    do {
        String input_mobileNo, input_pword;


        System.out.println("\n\n\n\n\n\n\n");
        System.out.println("-----------------------------------------------------------------------------------------");
        System.out.println("                                  Log in");
        System.out.println("-----------------------------------------------------------------------------------------");



        do {
            System.out.print("\nEnter mobile number: ");
            input_mobileNo = scan.nextLine();
            if (validateStringNumber(10,11,input_mobileNo) == false &&
    !input_mobileNo.equals(admin.getAdminNo())) {
                System.out.println("Invalid mobile number. Mobile number must contain 10-11 digits.");
            }
        }while(validateStringNumber(10,11,input_mobileNo) == false &&
    !input_mobileNo.equals(admin.getAdminNo()));


        do {
            System.out.print("\nEnter your password: ");
```

```
        input_pword = scan.nextLine();
        if(validatePassword(input_pword) == false) {
            System.out.println("Invalid password.");
            System.out.println("Password must contain:");
            System.out.println("- At least 1 digit (0-9)");
            System.out.println("- At least 1 lowercase letter (a-z)");
            System.out.println("- At least 1 uppercase letter (A-Z)");
            System.out.println("- At least 1 special character");
            System.out.println("- No whitespace characters");
            System.out.println("- Length between 6 and 20 characters");
        }
    }while(validatePassword(input_pword) == false);

    for (Customer c : custList) {
        if (c.mobileNo.equals(input_mobileNo) && c.password.equals(input_pword)) {
            Customer cust = new Customer(c.getfName(),c.getMobileNo(),c.getEmail(),c.getPassword());
            return cust;
        }
    }
    if(Admin.isAdmin(input_mobileNo, input_pword)) {
        Customer cust = new Customer("Admin",input_mobileNo,"admin@dmin.com",input_pword);
        return cust;
    }
```

```
        System.out.print("Incorrect mobile number or password. (1-try again or 2-sign up): ");
        ans = Booking.getValidInt(1,2, scan);

        if(ans == 2) {
            Customer cust = Customer.sign_in(custList,scan);
            return cust;
        }

    }while(ans == 1);
    return null;
}

public static boolean validateEmail(String email) {
    String emailRegex = "^[a-zA-Z0-9_+&*-]+(?:\\."+
                        "[a-zA-Z0-9_+&*-]+)*@" +
                        "(?:[a-zA-Z0-9-]+\\.)+[a-z" +
                        "A-Z]{2,7}$";

    Pattern pat = Pattern.compile(emailRegex);
    if (email == null)
        return false;
    return pat.matcher(email).matches();
```

```
}


public static boolean validatePassword(String password) {
    String passwordRegex = "^(?=.*[0-9])"
                         + "(?=.*[a-z])(?=.*[A-Z])"
                         + "(?=.*[~!@#$%^&*-_+=])"
                         + "(?=\\S+$).{6,20}$";
    Pattern pat = Pattern.compile(passwordRegex);
    if (password == null)
        return false;
    return pat.matcher(password).matches();
}


public static boolean validateStringNumber(int min, int max, String number) {
    if (number == null)
        return false;


    if (number.length() == 0)
        return false;


    if (number.length() < min || number.length() > max )
        return false;
```

```
    for (int i = 0; i < number.length(); i++) {
        if (!Character.isDigit(number.charAt(i)))
            return false;
    }
    return true;
}

public static boolean validateString(String string) {
    if (string == null) {
        return false;
    }

    if (string.length() == 0)
        return false;

    for (int i = 0; i < string.length(); i++) {
        if (!Character.isLetter(string.charAt(i)) && string.charAt(i) != ' ') {
            return false;
        }
    }
    return true;
}
```

```
}
```

- **Movie.java**

```
package cinemaProject;

import java.util.ArrayList;



public class Movie {

   private String movieName, movieGenre, duration, language, classification;



   Movie(String title, String genre, String timeDuration, String movieLang, String classif){

      this.movieName = title;

      this.movieGenre = genre;

      this.duration = timeDuration;
```

```
    this.language = movieLang;
    this.classification = classif;
}

public static Movie getMovie(ArrayList<Movie> movie, int index) {
    if(index < movie.size() && index >= 0){
        return movie.get(index);
    }
    return null;
}

public String getMovieName() {
    return movieName;
}

public String getMovieGenre() {
    return movieGenre;
}

public String getDuration() {
    return duration;
}
```

```
public String getLanguage() {

    return language;

}



public String getClassif() {

    return classification;

}



public void setMovieName(String name) {

    this.movieName = name;

}



public void setMovieGenre(String genre) {

    this.movieGenre = genre;

}



public void setDuration(String time) {

    this.duration = time;

}



public void setLanguage(String lang) {

    this.language = lang;

}
```

```
public void setClassif(String classif) {
    this.classification = classif;
}

@Override
public String toString() {
    return movieName+"\n"+movieGenre+" | "+duration+" | "+language+" | "+classification;
}

public static void displayMovie(ArrayList<Movie> movie) {
    System.out.println("---------------------------------------- Now Showing ----------------------------------------");
    System.out.printf("%-5s%-48s%-12s%-13s%-10s%-5s\n", "No", "Name", "Genre", "Duration",
   "Language", "Classification");
    int i = 0;
    for(Movie list: movie) {
        System.out.printf("%-5s%-48s%-12s%-16s%-10s%-5s\n", "["+(i+1)+"]", list.movieName,
    list.movieGenre, list.duration, list.language, list.classification);
        i++;
    }
    System.out.print("Enter any movie number: ");
}

}
```

- **Payment.java**

```
package cinemaProject;


import java.util.ArrayList;
import java.util.Calendar;
import java.util.Scanner;


import cinemaProject.Cinema.Hall;


public class Payment {
   private String name;
   private String email;
   private String mobile;
   private double amount;
   private int paymentMethod;


   public Payment(String name, String email, String mobile, double amount, int paymentMethod) {
      this.name = name;
      this.email = email;
      this.mobile = mobile;
      this.amount = amount;
      this.paymentMethod = paymentMethod;
```

```
}

public String getName() {
    return name;
}

public String getEmail() {
    return email;
}

public String getMobile() {
    return mobile;
}

public double getAmount() {
    return amount;
}

public void setAmount(double amount) {
    this.amount = amount;
}

public int getPaymentMethod() {
```

```
    return paymentMethod;
}

public void setPaymentMethod(int paymentMethod) {
    this.paymentMethod = paymentMethod;
}

public void makePayment() {
    if(paymentMethod == 1) {
        System.out.println("Making payment of RM" + amount + " using credit card/debit card");
    } else {
        System.out.println("Making payment of RM" + amount + " using online banking");
    }
}

public boolean cardPayment(Scanner scan) {

    double amountUser;

    do {
        System.out.print("Enter payment amount: RM");
        amountUser = scan.nextDouble();
```

```
        if(amountUser < amount)
            System.out.println("Insufficient amount. Please enter RM"+amount);

        if(amountUser > amount)
            System.out.println("Excessive amount. Please enter RM"+amount);

    }while(amountUser != amount);

    return true;

}

public boolean onlinePayment(String username, String password, Scanner scan) {

    double amountUser;

    do {
        System.out.print("Enter payment amount: RM");
        amountUser = scan.nextDouble();

        if(amountUser < amount)
            System.out.println("Insufficient amount. Please enter RM"+amount);
```

```
        if(amountUser > amount)
            System.out.println("Excessive amount. Please enter RM"+amount);

    }while(amountUser != amount);

    if (username != null && password != null) {
        return true;
    } else {
        System.out.println("Transaction failed. Please check your email and password and try again.");
        return false;
    }
}

public void paymentSucceed(Cinema cinema, ArrayList<String> seats) {
    System.out.println("************************************************");
    System.out.println("Transaction status: Successful");
    System.out.println("Name:              " + name);
    System.out.println("Email:             " + email);
    System.out.println("Mobile:            " + mobile);
    System.out.println("Amount Paid(RM):   " + amount);
    if(paymentMethod == 1)
        System.out.println("Payment Method:    Credit card/ Debit card");
```

```
    else
        System.out.println("Payment Method:     Online banking");
    System.out.println("-----------------------------------------------");
    System.out.println("               Order details:");
    System.out.println("-----------------------------------------------");
    System.out.println(cinema.toString());
    Hall.displaySelectedSeat(seats);
    System.out.println("\n      You can show this receipt at the\n         counter to enter the cinema.\n");
    System.out.println("***********************************************");


}


public static boolean isValidCvv(String cvv) {
    if (cvv.length() == 3 || cvv.length() == 4) {
        try {
            int cvvValue = Integer.parseInt(cvv);
            return true;
        } catch (NumberFormatException e) {
            System.out.println("Error: CVV must be a number.");
            return false;
        }
    } else {
        System.out.println("Error: CVV must contains 3 or 4 digits.");
```

```
        return false;
    }
}



public static boolean isValidExpirationDate(String expirationDate) {
    // Check if the length of the expiration date is not 5 or 7 characters
    if (expirationDate.length() != 5 && expirationDate.length() != 7) {
        return false;
    }

    // Split the expiration date by "/" character
    String[] dateParts = expirationDate.split("/");

    // Check if the split array is not of length 2
    if (dateParts.length != 2) {
        System.out.println("Error: Invalid format. Expiration date should be in the format of MM/YY or
    MM/YYYY.");
        return false;
    }

    int month;
    int year;
```

```
try {
    // Try to parse the first part of the split array as integer for month
    month = Integer.parseInt(dateParts[0]);
    // Try to parse the second part of the split array as integer for year
    year = Integer.parseInt(dateParts[1]);
} catch (NumberFormatException e) {
    System.out.println("Error: Invalid format. Month and year should be numbers.");
    return false;
}

// Check if the month is less than 1 or greater than 12
if (month < 1 || month > 12) {
    System.out.println("Error: Invalid month. Month should be between 1 and 12.");
    return false;
}

// If the length of the expiration date is 5 characters, add 2000 to year
if (expirationDate.length() == 5)
    year += 2000;

// Get the current date
Calendar currentDate = Calendar.getInstance();
```

```
    int currentYear = currentDate.get(Calendar.YEAR);


    // Check if the year is less than the current year
    if (year < currentYear) {
        System.out.println("Error: Your card has been expired.");
        return false;
    }


    int currentMonth = currentDate.get(Calendar.MONTH) + 1;


    // Check if the year is same as the current year and the month is less than the current month
    if (year == currentYear && month < currentMonth) {
        System.out.println("Error: Your card has been expired.");
        return false;
    }


    // If all checks pass, return true
    return true;
    }


}
```

- **Pricing.java**

```
package cinemaProject;


public class Pricing {

   private double childPrice;

   private double adultPrice;

   private double seniorPrice;

   private double okuPrice;


   public Pricing(double adultPrice, double childPrice, double seniorPrice, double okuPrice) {

      this.childPrice = childPrice;

      this.adultPrice = adultPrice;

      this.seniorPrice = seniorPrice;

      this.okuPrice = okuPrice;

   }


   public double getChildPrice() {

      return childPrice;

   }


   public double getAdultPrice() {

      return adultPrice;
```

```
    }


    public double getSeniorPrice() {
        return seniorPrice;
    }


    public double getOkuPrice() {
        return okuPrice;
    }
}
```

- **SalesReport.java**

```
package cinemaProject;


import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;


public class SalesReport {
//    private Map<Movie, Map<String[], Double>> salesData;
    private String cinemaName;
```

```
    private String movieName;
    private double totalSale;
    private int quantityChild;
    private int quantityAdult;
    private int quantityOku;
    private int quantitySenior;

    public SalesReport() {
        this.quantityChild = 0;
        this.quantityAdult = 0;
        this.quantityOku = 0;
        this.quantitySenior = 0;
    }

    public void addSaleData(String cinemaName, String movieName, double totalAmount) {
        this.cinemaName = cinemaName;
        this.movieName = movieName;
        this.totalSale = totalAmount;
    }

    public void addTicketData(int qttAdult, int qttChild, int qttOku, int qttSenior) {
        this.quantityAdult += qttAdult;
        this.quantityChild += qttChild;
```

```
    this.quantityOku += qttOku;
    this.quantitySenior += qttSenior;
}

public double getTotalSale() {
    return totalSale;
}

public int getQuantityChild() {
    return quantityChild;
}

public int getQuantityAdult() {
    return quantityAdult;
}

public int getQuantityOku() {
    return quantityOku;
}

public int getQuantitySenior() {
    return quantitySenior;
}
```

```
public int getTotalTicket() {
    return quantityAdult+quantityChild+quantityOku+quantitySenior;
}

public String getCinemaName() {
    return cinemaName;
}

public String getMovieName() {
    return movieName;
}

public void setTotalSale(double totalSale) {
    this.totalSale += totalSale;
}

public void setQuantityChild(int quantityChild) {
    this.quantityChild = quantityChild;
}

public void setQuantityAdult(int quantityAdult) {
    this.quantityAdult = quantityAdult;
```

```
    }

    public void setQuantityOku(int quantityOku) {
        this.quantityOku = quantityOku;
    }

    public void setQuantitySenior(int quantitySenior) {
        this.quantitySenior = quantitySenior;
    }

    public void setCinemaName(String cinemaName) {
        this.cinemaName = cinemaName;
    }

    public void setMovieName(String movieName) {
        this.movieName = movieName;
    }

}
```

- **Ticket.java**

```
package cinemaProject;
```

```
public class Ticket {


   private int child = 0;

   private int adult = 0;

   private int senior = 0;

   double cost = 0.00;

   private  double SeatPrice = 5.00;


   public Ticket (int child, int adult, int senior){

      this.child = child;

      this.adult = adult;

      this.senior = senior;




   }

   public void setSeatPrice(double SeatPrice){

      this.SeatPrice = SeatPrice;

   }


   public double getSeatPrice(){

      return SeatPrice;

   }
```

```
  public void calcChargeDue(){

     cost = (child * 0.7 * getSeatPrice()) + (adult * 1.5 * getSeatPrice()) + (senior * 1 * getSeatPrice());

  }

  public double getChargeDue(){
     return cost;
  }
}
```

Rubric

LOD 4, PLO 2, CLO1: Investigation *Apply the fundamental concepts of object-oriented programming in problem solving.* **(C3)** **REPORT EVALUATION (10 %)**							
Criterion	**Cognitive Level**	**Poor 1**	**Weak 2**	**Satisfactory 3**	**Good 4**	**Excellent 5**	**Score**
Messaging and Sequence Fragment: *Identify matching control structure and class behavior for sequence diagram*	Remembering C1	Unable to identify correctly matching control structure and class behavior at all.	Able to identify correctly matching control structure and class behavior. But, much incorrect	Able to identify correctly matching control structure and class behavior, but in general	Able to identify correctly matching control structure and class behavior but lack minor detail in the sequence diagram	Able to identify correctly matching control structure and class behavior extraordinarily in the sequence diagram	
Project Functionalities: *Indicate user interaction and the system behaviour through use case diagram illustration*	Understanding C2	Able to apply. But much incorrect user interaction/ system features / relationship/ notations	Able to indicate correct user interaction and system features but most relationship/notations are incorrect	Able to indicate correct user interaction and system features but some relationship/notations are incorrect	Able to indicate correct user interaction and system features but the analysis and design can be improved to be more efficient	Able to indicate correct user interaction and system features extraordinarily	
OO Behavioural Design: *Translate detailed logic of project business rule from use case to activity diagram.*	Understanding C2	Able to understand detailed logic of the system behaviour, but missing most use case translation	Able to understand detailed logic of the system behaviour, but missing some use case translation	Able to understand complete detailed logic of the system behaviour, but in general	Able to understand complete detailed logic of the system behaviour, but lack minor detail	Able to understand complete detailed logic of the system behaviour comprehensively	

OO Structural Analysis: *Construct class state and behaviour*	Applying C3	Unable to construct at any state and behaviour all.	Able to construct. But, much incorrect / missing data type/ visibility / relationship	Able to apply construct but not clear or some incorrect / missing data type/ visibility / relationship	Able to apply construct but can be improved to be more efficient with restructured class / additional state/ behaviour	Able to construct extraordinarily	
OOP Construct: *Explain why a construct (ex: array, selection, loop, data type) is used over other alternatives for the project*	Applying C3	Unable to explain main construct used.	Able to briefly explain main construct used. But, much incorrect	Able to briefly explain main construct used correctly but not clear or some incorrect	Able to sufficiently explain main construct used correctly and clearly	Able to explain main construct used in detail	
OOP Concepts: *Implement encapsulation and inheritance wherever possible in the project design*	Applying C3	Unable to implement both concepts at all.	Able to implement both concepts. But, much incorrect	Able to implement both concepts correctly but not clear or some incorrect	Able to implement both concepts correctly but can be improved to be more efficient	Able to implement both concepts extraordinarily	
						Total	/30

LOD 9, PLO 6, CLO 2: DIGITAL SKILLS *Follow the rules of chosen object-oriented programming language to produce computer program using suitable compiler or IDE.* **(P4)** **PROJECT CODING EVALUATION (10%)**							
Criterion	**Cognitive Level**	**Poor 1**	**Weak 2**	**Satisfactory 3**	**Good 4**	**Excellent 5**	**Score**
Identifiers: *Follow standard convention for identifier naming and data type selection*	P1 Imitation	Unable to comply to standard convention in any way.	Able to accomplish for some identifiers, but with many incorrect or missing access modifier, variables, and naming convention	Able to accomplish for all identifiers, but with improper modifier, data type or naming convention	Able to follow standard naming convention and correct data type for all identifiers. But the identifiers could be improved to be meaningful and efficient	Able to follow standard naming convention meaningfully and efficient data type for all identifier	
OOAD Implementation: *Trace OOAD into implementation coding* *Translate the design into coding*	P3 Precision	Unable to translate the UML diagram into program codes	Able to translate few of the UML diagram into program codes correctly	Able to translate most of the UML diagram into program codes correctly	Able to translate all UML diagram into program codes correctly	Able to translate all UML diagram into program codes label, structure and behaviour correctly and efficiently	
Synthesis: *Apply encapsulation / inheritance / polymorphism in project code*	P4 Articulation	Unable to apply OOP concept in program codes	Able to apply, but with many violations	Able to apply, but with little violation	Able to apply using correct keyword and purpose, but can be improved to be more efficient	Able to apply using correct keyword and purpose efficiently	
Selection: *Formulate selection condition using*	P4 Articulation	Unable to formulate selection	Able to formulate selection	Able to formulate selection condition and action but with	Able to formulate correct selection condition and action,	Able to formulate efficient selection condition with no	

appropriate control structure and control program flow to possible alternatives		condition and action	condition and action, but with many errors	minimal errors	but can be improved to be more efficient	error and comprehensive possible alternative	
Creation: *<u>Develop</u> workable program with organised interface layout and proper exception handling*	P4 Articulation	The program fails to run.	The program run, but with improper exception handling and many warning errors.	The program run correctly but with unorganized interface, but proper exception handling.	The program run correctly with proper interface layout and exception handling, but can be improved to be more efficient	The program run correctly and efficient with proper interface layout and exception handling.	
Modularity: *Build program as a set of functioning units that can be composed into a larger application*	P2 Manipulation	Unable to build modular program	Able to build modular program, but with many violations	Able to build modular program, but with little violation	Able to build modular program, but can be improved to be more efficient	Able to build modular program efficiently	
						Total	/30

LOD 13, PLO 9, CLO 3: PERSONAL SKILLS

Write efficient object-oriented computer programs to solve small-to-moderate scale problems. **(A3)**

PRESENTATION EVALUATION (5%)

Criterion	Cognitive Level	Poor 1	Weak 2	Satisfactory 3	Good 4	Excellent 5	Score
Interest: *Focus* on detail and specific project functionalities	A1 Receiving	The project only focuses on basic function	The project focuses on some specific function without basic function, could have been improved with more effort	The project focuses on some specific function with all basic function, but could have been improved with more effort	The project excellently focuses on common and specific function	The project focuses goes well beyond requirement	
Learning Engagement: *Practice autonomous learning to increase learning process*	A2 Responding	Least attempt to engage in autonomous learning	Minimally engage in autonomous learning with major assistance from lecturer	Putting effort to engage in autonomous learning with some assistance from lecturer	Consistently engage in autonomous learning without assistance	Highly engage in autonomous learning beyond expectation, with an excellent effort	
Initiative: *Commit to successful project completion*	A3 Valuing	No commitment to complete task	Minimal commitment to complete task	Moderate commitment to complete task	Good commitment to complete task	Excellent commitment to complete task	
Effort: Complete the project efficiently *by ability in expanding programming knowledge*	A3 Valuing	No effort to complete task	Complete some part of the project successfully using knowledge delivered	Complete most part of the project successfully using knowledge delivered	Complete project efficiently using knowledge delivered.	Complete project efficiently using additional knowledge.	
						Total	/20