ASSIGNMENT CE/CZ2002: OBJECT-ORIENTED DESIGN & PROGRAMMING

BUILDING AN OO APPLICATION (MOBLIMA)

LAB: SS5

GROUP: 3

MEMBERS:

NG WOON YEE

NG ZHENG KAI

OH DING ANG

PALANISWAMY TARUN KUMAR

PHUN WEI CHENG RUSSELL

Contents

Report Overview

MOvie Booking and LIsting Management Application (MOBLIMA) is an application to computerize the processes of making online booking and purchase of movie tickets. This report details the design process of building the application. The application is developed using Java language.

Design Considerations

Programs (Boundaries)

The program package contains all the applications that manage the information between user and Managers. MoblimaMainApp is the main user interface for our application.

GuestApp - Accessed by guest users without an account. This limits the privileges a user can access. Users can only view movie information and seating availability. To book tickets, users will have to create a new member account or log in to an existing one.

MemberApp - Used by customers. Members are able to view movie listings, movie and session information, book tickets and view their booking history. They are also able to amend their bookings, input reviews and ratings for movies they have watched, view the sales report for the top few movies that are currently showing and the ticket pricing for various categories (GOLD class, student ticket, 3D movie, etc.)

AdminApp - Used by admin users. AdminApp is able to do everything that MemberApp can do, including amending bookings for members, as well as accessing admin privileges. Admin privileges include updating or creating new movies and sessions and configuring system settings such as holidays and prices.

UI Pages - Called on by Guest, Member and/or AdminApp. Users will interact with these user interface pages for various functions. For example, UIListingAndBooking.java under programs are called by all 3 user apps. This page will display all the available functions (what they are able to do and view) to the users in the console, take in their choice as input and call on the Manager functions to handle the logic for the user’s selected function (for example, View Movie Listings).

Managers (Control)

Managers act as controllers, handling all the main logic of the application. The MainApp will call on the different manager files to perform various functions. For example, for a member to view prices, the user would have to login in and select the option to view prices. On the system, MoblimaMainApp will create an instance of the MemberApp in programs (at log-in time) for the user, where he will choose to view prices. Selecting the option to view prices will create an instance of the PriceManager from Managers. PriceManager will handle the logic to display the necessary ticketing details.

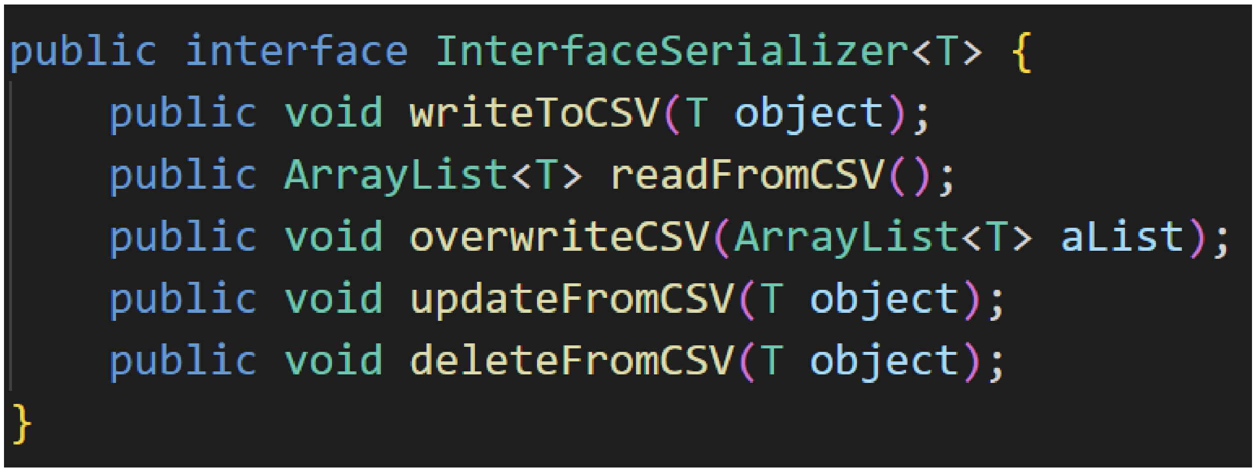
Models (Entity)

Models are entities which store various information. Models are being created as objects and accessed by managers. Models will retrieve the relevant data from Serializers, which handle the permanent data storage from our CSV.

Serializers

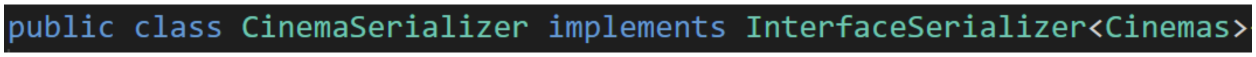
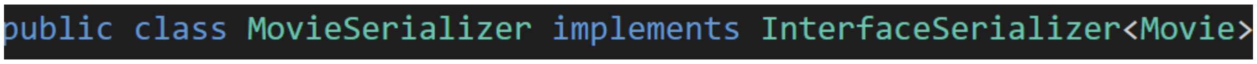
Serializers manage the databases. This allows for efficient and secure storage as information will be permanent and not be erased upon termination of the application.

- Contains one main InterfaceSerializer.



```java
public interface InterfaceSerializer<T> {
    public void writeToCSV(T object);
    public ArrayList<T> readFromCSV();
    public void overwriteCSV(ArrayList<T> aList);
    public void updateFromCSV(T object);
    public void deleteFromCSV(T object);
}
```

- All the child serializers will implement the above Interface.
- For example,


```
public class CinemaSerializer implements InterfaceSerializer<Cinemas>
```

```
public class MovieSerializer implements InterfaceSerializer<Movie>
```

Design Principles Used

Single Responsibility Principle - Only one reason for a class to change

Each class has its own responsibility and purpose. We implemented a model-view-controller architectural pattern to program logic into three interconnected elements. For example, we have a UIAmendBooking.java file under the program to use as a user interface to record user's choices, while the function for amending booking falls under AmendBookingManager.java, and the classes are created in our model folder which involves Transaction.java, Session.java and etc. At a time, only one reason for a class to change and hence a 'God' class is avoided.

Open-Close Principle - Open for extension, close for modification

We implemented abstraction on the type of cinemas such as PlatinumCinema, Gold Cinema and Regular Cinema with Cinemas as base class. This allowed us for an easy extension if the Cineplex decided to build a new cinema, we can then extend the Cinemas class for a new subclass without modifying the original base class, hence reducing the ripple effect.

Liskov- Substitution Principle - Subtypes must be substitutable for base types

Our subclasses such as PlatinumCinema, GoldCinema and Regular Cinema could be a complete substitute for superclass - Cinemas. This is because our subclasses can execute all the functions that a superclass can do. It overrides and enhances the functionality of the superclass but does not reduce it. Hence, the subclasses are substitutable for their base class.

Interface Segregation Principle - Client-specific interfaces are better than one general purpose interface

In our code, we have implemented two different interfaces with specific purposes. Firstly, an interfaceSerializer was created and allows all entity-specific serializers to implement it. These entity-specific serializers were created to handle the passing of information between models and

CSV databases. Besides, we have a login interface which is implemented by adminManager and loginManager to verify users' information while logging in.

<u>Dependency Injection Principle</u> - High-level and Low-level modules should depend on abstractions rather than on each other; Abstractions should not depend upon details. Details should depend upon abstractions.

In our solution, we have implemented the MVC framework. The Manager Package handles the information flow between Program package and Models package. Hence, programs and models need not be aware of each other and only talk to the manager, preventing direct access between them. This enables Programs and Models to be independently designed.

<u>OODP Concepts</u>

Abstraction

Abstraction is the concept of object-oriented programming that shows only essential attributes and hides unnecessary information. InterfaceSerializer is created as an interface class and is implemented in each of the specific serializer classes. This allows simplified information presented to the developer to reduce the complexity of the code and makes it easier to understand.

Encapsulation

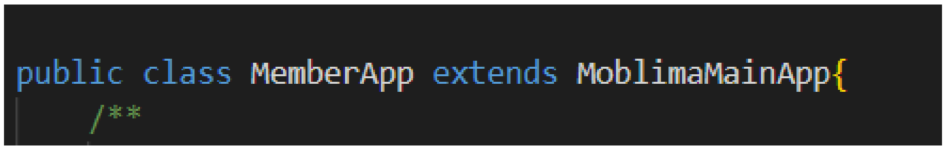
Encapsulation ensures the protection of certain field data within a class. Public methods such as get and set can only be used to access these private data. In the application, all attributes under "Models" are private. This ensures data is safe within each class.

Inheritance

Inheritance allows new class methods to be inherited from the superclass. This made greater reusability and efficiency in the code. As shown in all the serializers, it inherits from the InterfaceSerializer, which has all the necessary methods to read the CSV. This allows functions to be added in if necessary for specific serializers, reducing duplicates in the code.

Polymorphism

Polymorphism refers to the ability of a message to be displayed in more than one form.



```
public class MemberApp extends MoblimaMainApp{
    /**
```

In this case, MemberApp extends the MoblimaMainApp. This is because both of them are user interfaces, where the system will take in user's choices to call on the different managers accordingly. Hence, both perform the same function of displaying available options for users and taking in their input (same methods, but different outcomes as they will call on different functions based on user choices).

<u>**Assumption**</u>

1) Senior citizen and Student tickets can be purchased online without validation of identity or age. The validation will be done upon entering the cinema.
2) SessionsID are all unique
3) Only customers with MOBLIMA accounts can book the tickets.
4) All transactionID is unique, however, ID will be the same if the booking is made at the same time
5) Payments are carried out by an external system and are always successful, including refunds when amending bookings.
6) All admins have the same level of authority in the app.
7) Admins' identities are already verified before the creation of new admin users.

<u>**Future Features**</u>

1. Sending SMS to Users before Movie starts

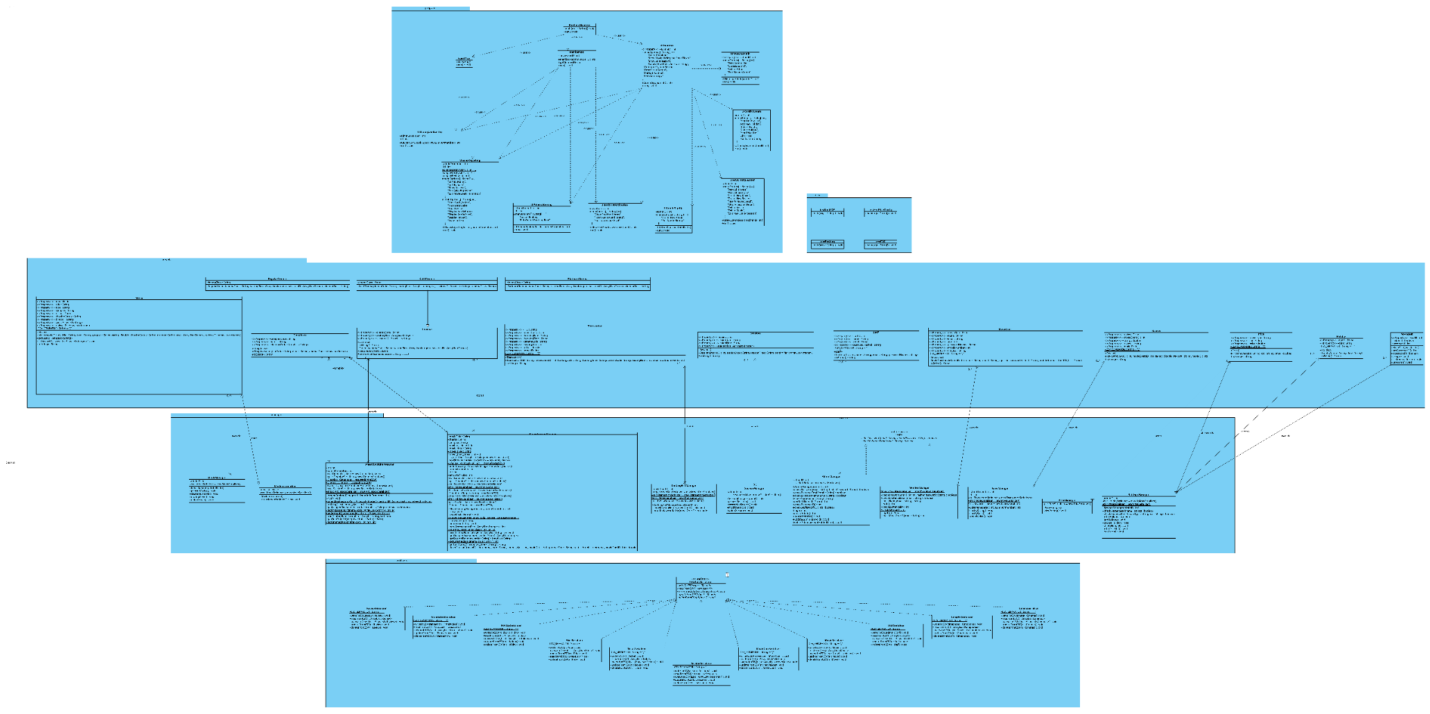
For this extension, we can create a new SMS manager class and SMS entity class. We can use the Observer Design Pattern here. In the SMS manager class, we can add a new method to get the contact number and the booking date. There will be a method to get the movieGoer transactionID and compare it with the transactionData both using the serializer to get the booking date. Then there will be another method to send the SMS to the user. A message will be sent out to the respective movieGoer before the movie. We can see that we do not need to change any existing code inside our system to accommodate this new function because our code aligns with both Open-Close and Single Responsibility Principles, and hence impact of changes is minimized.

2. **Adding a new attribute to MovieGoer**
   Adding loyalty points as a new attribute in MovieGoer Model class for discounts or privileges based on a loyalty points system. For example, MovieGoer will be rewarded 20 points per transaction and the points can be used to discount future ticket prices and/or claim snacks like popcorn. This new feature is easily achievable due to our design considerations - it could be done by only editing the MemberApp.java (View), MovieGoer.java (Model) and MemberManager.java (Controller). This is because MemberManager.java handles the logic and function upon user request while the MovieGoer.java model will then interact with the MemberManager.java without interacting with the MemberApp.java (View). Hence, Dependency Injection, Open-Closed and Single Responsibility Principles are achieved.

UML Class Diagram

(Refer to the attached link for a clearer picture.)

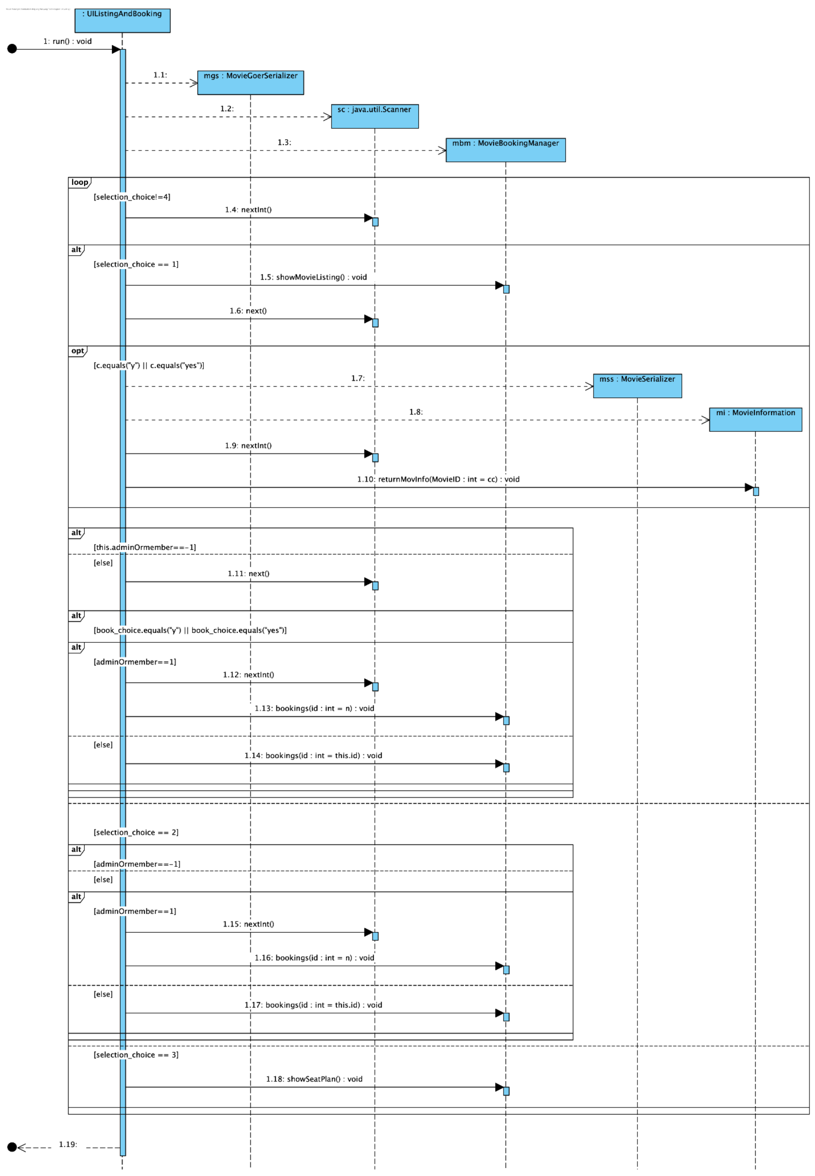
https://github.com/woonyee28/sc2002-project/blob/main/Class%20Diagram%20v5.png



UML Sequence Diagram

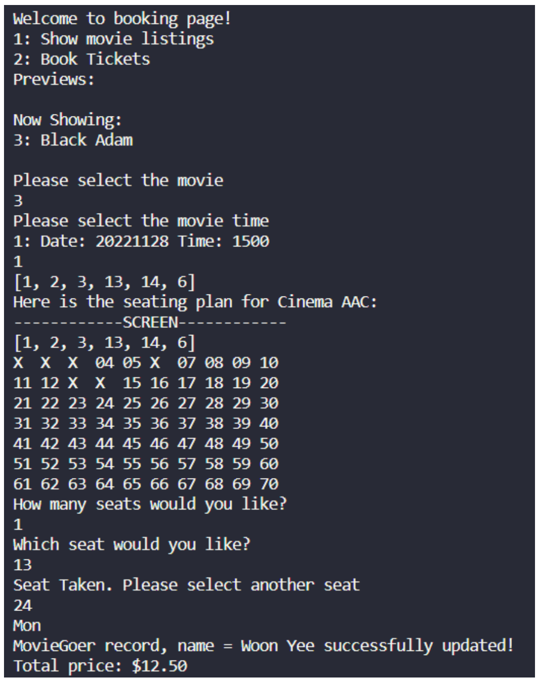
(Refer to the attached link for a clearer picture.)

https://github.com/woonyee28/sc2002-project/blob/main/Sequence%20Diagram%20For%20Booking.png



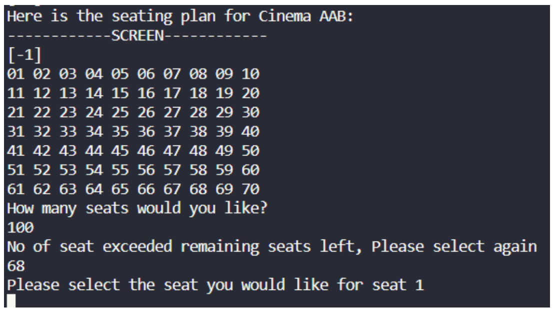
Test cases (Not shown in demo video)

Booking the occupied:



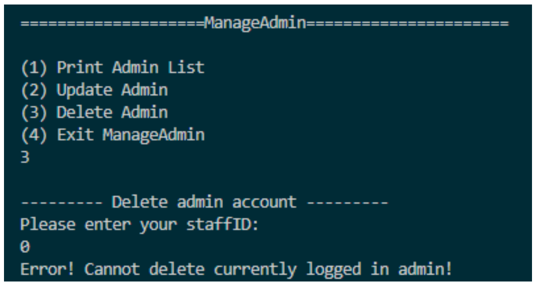
```
Welcome to booking page!
1: Show movie listings
2: Book Tickets
Previews:

Now Showing:
3: Black Adam

Please select the movie
3
Please select the movie time
1: Date: 20221128 Time: 1500
1
[1, 2, 3, 13, 14, 6]
Here is the seating plan for Cinema AAC:
------------SCREEN-----------
[1, 2, 3, 13, 14, 6]
X  X  X  04 05 X  07 08 09 10
11 12 X  X  15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
How many seats would you like?
1
Which seat would you like?
13
Seat Taken. Please select another seat
24
Mon
MovieGoer record, name = Woon Yee successfully updated!
Total price: $12.50
```

Too much seat selected:



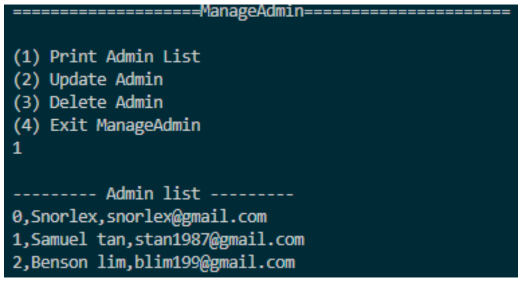
```
Here is the seating plan for Cinema AAB:
------------SCREEN-----------
[-1]
01 02 03 04 05 06 07 08 09 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
How many seats would you like?
100
No of seat exceeded remaining seats left, Please select again
68
Please select the seat you would like for seat 1
```

Deleting currently logged in admin:



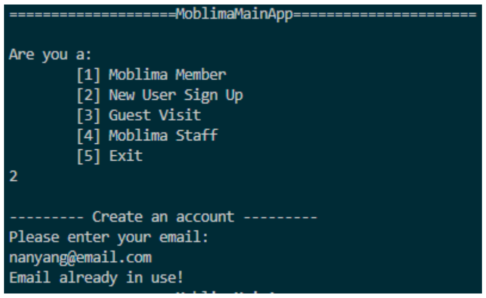
```
===================ManageAdmin======================

(1) Print Admin List
(2) Update Admin
(3) Delete Admin
(4) Exit ManageAdmin
3

--------- Delete admin account ---------
Please enter your staffID:
0
Error! Cannot delete currently logged in admin!
```

Password masked for admin list:



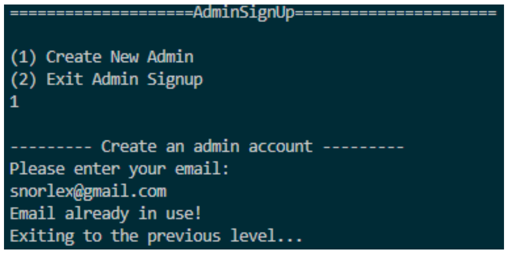
```
===================ManageAdmin======================

(1) Print Admin List
(2) Update Admin
(3) Delete Admin
(4) Exit ManageAdmin
1

--------- Admin list ---------
0,Snorlex,snorlex@gmail.com
1,Samuel tan,stan1987@gmail.com
2,Benson lim,blim199@gmail.com
```

Check if email is already in use when signing up (member):



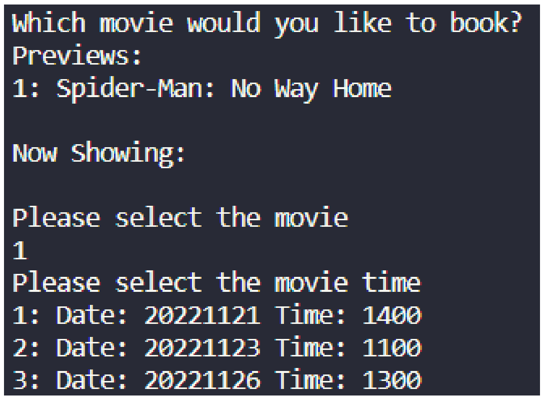
```
==================MoblimaMainApp=====================

Are you a:
        [1] Moblima Member
        [2] New User Sign Up
        [3] Guest Visit
        [4] Moblima Staff
        [5] Exit
2

--------- Create an account ---------
Please enter your email:
nanyang@email.com
Email already in use!
```

Check if email is already in use when signing up (admin):



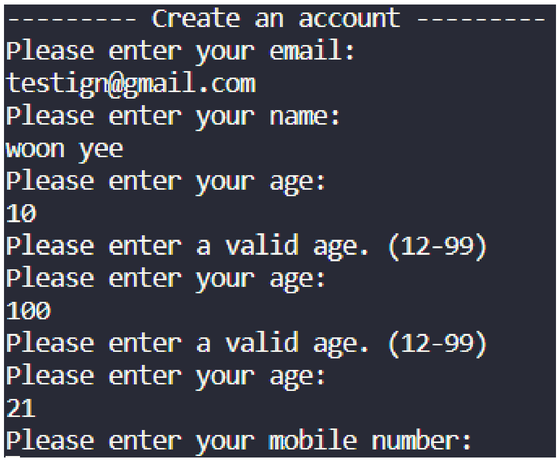
```
===================AdminSignUp=====================

(1) Create New Admin
(2) Exit Admin Signup
1

--------- Create an admin account ---------
Please enter your email:
snorlex@gmail.com
Email already in use!
Exiting to the previous level...
```

Sessions are sorted according to date & time:



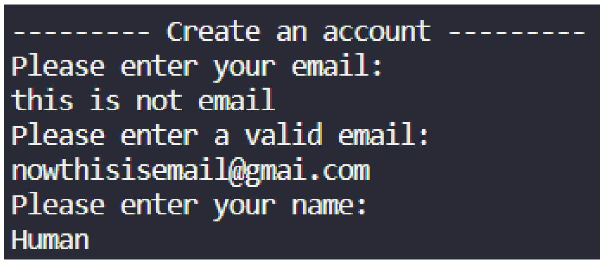
```
Which movie would you like to book?
Previews:
1: Spider-Man: No Way Home

Now Showing:

Please select the movie
1
Please select the movie time
1: Date: 20221121 Time: 1400
2: Date: 20221123 Time: 1100
3: Date: 20221126 Time: 1300
```

Check valid age:



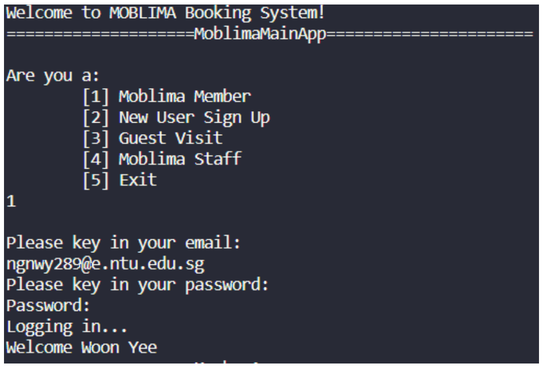
```
--------- Create an account ---------
Please enter your email:
testign@gmail.com
Please enter your name:
woon yee
Please enter your age:
10
Please enter a valid age. (12-99)
Please enter your age:
100
Please enter a valid age. (12-99)
Please enter your age:
21
Please enter your mobile number:
```

Check valid email:



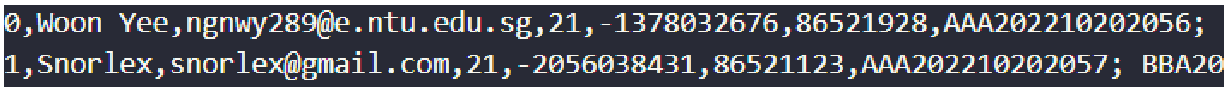
```
--------- Create an account ---------
Please enter your email:
this is not email
Please enter a valid email:
nowthisisemail@gmai.com
Please enter your name:
Human
```

Password Masking:



```
Welcome to MOBLIMA Booking System!
===================MoblimaMainApp=====================

Are you a:
        [1] Moblima Member
        [2] New User Sign Up
        [3] Guest Visit
        [4] Moblima Staff
        [5] Exit
1

Please key in your email:
ngnwy289@e.ntu.edu.sg
Please key in your password:
Password:
Logging in...
Welcome Woon Yee
```

Password in hashed format: (ID,Name,Email,Age,passwordHashed,Mobile,TID_list)



```
0,Woon Yee,ngnwy289@e.ntu.edu.sg,21,-1378032676,86521928,AAA202210202056;
1,Snorlex,snorlex@gmail.com,21,-2056038431,86521123,AAA202210202057; BBA20
```

Declaration of Original Work for SC/CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honoured the principles of academic integrity and have upheld the Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (CE2002 or CZ2002)	Lab Group	Signature /Date
NG WOON YEE	SC2002	SS5	Woon
NG ZHENG KAI	SC2002	SS5	[illegible]
OH DING ANG	SC2002	SS5	[illegible]
PALANISWAMY TARUN KUMAR	SC2002	SS5	[illegible]
PHUN WEI CHENG RUSSELL	SC2002	SS5	NIL (Submitted WBS)