NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

22S1 SC2002 Object Oriented Design and Programming
MOBLIMA Report

SS3 Assignment Group 5

Declaration of Original Work for SC/CE/CZ2002 Assignment

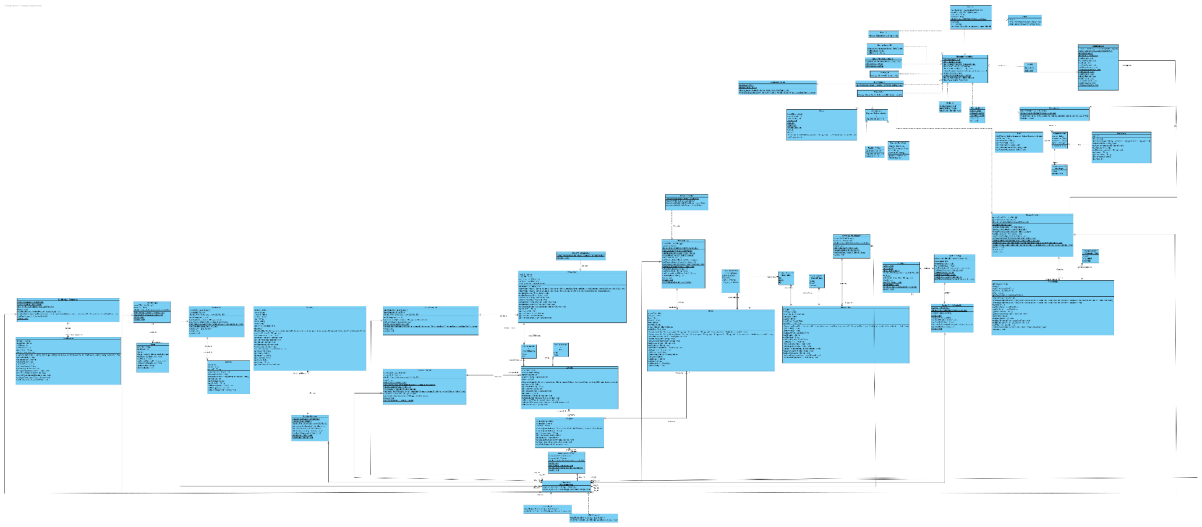
We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below. We have honoured the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work. 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	Lab Group	Signature/ Date
Lee Kar Jun Lester	SC2002	SS3	Lester / 11-11-2022
Lee Yu Xuan	SC2002	SS3	Yu Xuan / 11-11-2022
Lim Boon Hian	SC2002	SS3	Boon Hian / 11-11-2022
Lim Yao Xian	SC2002	SS3	Yao Xian / 11-11-2022
Loke Yong Jian	SC2002	SS3	Yong Jian / 11-11-2022

Important links :

The link of our video: Youtube || Our repo on Github: Github

UML Class Diagram:



A clear diagram is included in the zip file.

Approach Taken:

During the planning of the MOBLIMA application, we wanted the system to be user friendly and simple to upgrade. As our application aims to be flexible and with intuitive inputs. Unlike the existing real life movie booking system which already has an existing database of customers and movies, we had to allow admin users to create movies, movie time slots as well as user creation for the system.

The system also allows registered movie-goers to see the various movie showtimes and see what movies are currently available. If the user has a specific movie in mind, they can search for it. If not, they can view the rankings of the movie ranked by either ticket sales or movie-goer ratings. When the users want to book a movie ticket for a certain movie at a certain time slot, they will be brought to the cineplex layout. From the cineplex layout, users will be able to view the seating plan and see which seats are vacant. Users will also receive a 2 dollar discount (except on public holidays and weekends) when they buy seats from the first 2 rows. Once they want to purchase the ticket, users will be prompted if they want to use a promo code. Users then can choose to pay

via credit/debit card or through PayLah. Once payment has been confirmed, movie-goers can check booking history to confirm that the purchase has been made.

As for users who do not want to be registered with the application, they will still be able to access the same features as registered movie-goers. However, they would need to key in their email address, so that the confirmation of payment can be sent to their email instead.

Admin users of the system have access to more features. Admin users are able to add, remove and update movies in the cinema's movie database. They should include the movie title, director, cast, rating (PG13, NC16 etc) as well as the synopsis into the system . They are able to also view the top 5 movies, either in terms of ticket sales or movie-goer rating. Admin users can also add or remove showtimes to specific movies at specific cineplexes. Additionally, they are able to add in promo-codes. They are able to set the name of the promo codes, the start/end date, quantity of the promo code as well as the discount offered. Admin users are also allowed to edit the system configuration. Firstly, they are able to input which dates of the year are public holidays or special events. This is important as it will affect the pricing of the ticket (Public Holidays dates will not provide any discount). They are also able to view a list of holidays as well as remove any if they wanted to. Secondly, they are able to modify the base price of the tickets as well as the different price modifiers for the different cinema classes. They can also view the current multipliers for each class of cinemas. Lastly, Admin users are able to add in more cinemas into a specific cineplex or add more cineplexes For the cinemas, they are able to input which branch it belongs to, cinemaID, class of cinema, number of seats and which layout type they want. As for the cineplex, admin users can key in the location of where the cineplex is located.

The MOBLIMA application aims to provide a pleasant user experience to both admin staff and movie goers. Should any features need to be updated or implemented, MOBLIMA will be able to do so seamlessly.

Design Consideration:

Design principles have been considered in every possible aspect in the project. Developing the project with the SOLID principles in mind help us to achieve loose coupling and high cohesion,

increasing the reusability and extensibility of the whole project. We will explain how we use them in the following paragraph.

Single Responsibility Principle:

SRP states that a single class should only have a single responsibility and a single reason to change. In our project, for the case of 'Movie' class, instead of having all the functions (adding of movie, updating of movie etc) inside the 'Movie.java' file, we create separate controllers to handle those functions. Hence classes that link to 'Movie.java' file does not have to be edited and recompiled when changes are made to the 'Movie.java'

Open-Closed Principle:

OCP advocates that a module should be closed for modification but open for extension. In our project, we have various Controller Classes responsible for different functions (i.e addMovie(), removeMovie()). But what if in the future there's a class that requires more interactions with the Movie Class but we do not want to alter existing codes as it might affect current users of the controller (i.e Staff Class uses Movie Controller to add movies)? We can use the Open-Closed Principle and let the new Controller Class inherit the functions of the current Movie Controller Class and give it new functions such as selecting some movies to be featured. By doing this, we not only maintain the functions of the Staff Class but we also introduced a new Controller Class that can select movies to be featured

Liskov Substitution Principle:

LSP states that if arguments meant for a base class is instead passed to a subclass of the bass class, it should be able to perform normally or even better. In our project we have a Staff Class. If we need to implement a new class such as a Supervisor Class, this can be easily done by inheriting from the Staff Class as the Supervisor Class should have the same features as the Staff Class or even more features.

Interface Segregation Principle:

ISP states that having multiple specific interfaces is better than having one general interface that does all the work. In our project, for our PaymentMethodInterface. We have two abstract

methods: pay() and validation(). Hence we can implement various payment methods that do payment differently and validate the transaction differently.

Dependency Injection Principle:

DIP states that high level modules should not depend upon low level modules. Rather, both should depend upon abstractions. In our project, to access different file types, we have File_IOInterface which all other file mutators and readers would inherit from. Such as JSONFile_IO and CSVFile_IO. Hence, allowing different file types input and output to be used based upon abstraction and not fixate on a particular file type allowing for dependency injection.

Implementation Details

Additional features implemented:

- : Basic Features

→ : Additional Features

→ Exception handling, error input will be prompted for correct input. (Pg. 11)

Admin Module:

- Login
  - → Hashed Password (Demonstrated in video)
  - → Automatically differentiate Staff from MovieGoer (Demonstrated in video)
- Create/Update/Remove movie listing (Demonstrated in video)
  - → Movie status is automatically updated (Pg. 6)
- Configure system settings (Demonstrated in Video)
  - → Update Public Holiday dates(Demonstrated in Video)
  - → Change Base Price of Tickets (Demonstrated in Video)
  - → Change Price difference between cinema class (Demonstrated in Video)
  - → Supports promotion code/ discount coupons with automatic updates

Movie-goer Module:

- Book and purchase ticket (Demonstrated in video)
  - → Supports multiple payment systems (Demonstrated in video)
  - → Automatic price calculation based on multiple factors (Pg. 9)
  - → Price deduction by inputting promotion code (Demonstrated in video)

- View booking history (Demonstrated in video)
  → Individual booking history for every user (Pg. 10)
  → Able to categorised the booking history into ‘Upcoming’ or ‘Past’ based on current dateTime and the showTime stated on the movieTicket (Demonstrated in video)
- Supports guest booking (Demonstrated in video)

How we handle the data from CSV

With the OOP concept in mind, we decided to stay away from reading and editing the CSV directly, as it will defeat the purpose of this project - that is to practise the concept of OOP. Hence our choice of implementation in this project is by extracting the data from the CSV and using them to create instances for the classes that we have made. In this way, we are able to practise inheritance, abstraction, polymorphism etc.

To do this, we have ‘loadAllClass’ and ‘saveAllClass’ functions.

- ‘loadAllClass’ will be called when starting the application and is responsible for creating the instances from the data in respective CSV
- ‘saveAllClass’ will be called when exiting the application and is responsible for saving all the instances edited back to the respective CSV
  (*we have included ‘class’.save() in some parts of the projects to reflect the real time changes for the purpose of the demo video)

Exception Handling

To have a tidy way of dealing with exception handling, we have developed a java file called ‘ExceptionHandling’ which is responsible for handling all the data inputs. Whenever we want to get inputs from the user, we will do ‘String input = ExceptionHandling.StringScanner()’.

Apart from having exception handling for input such as integer, string and DateTime we have also done exception handling for each of our enumeration classes.

Movie Status Auto Update

The movie is automatically sets to NOW_SHOWING through the following sequences:

- Check through the entire show time list
- If there exist an entries where current time is later than the **start** time of the showing,
- The movie status is automatically updated to NOW_SHOWING

The movie is automatically sets to END_OF_SHOWING through the following sequences:

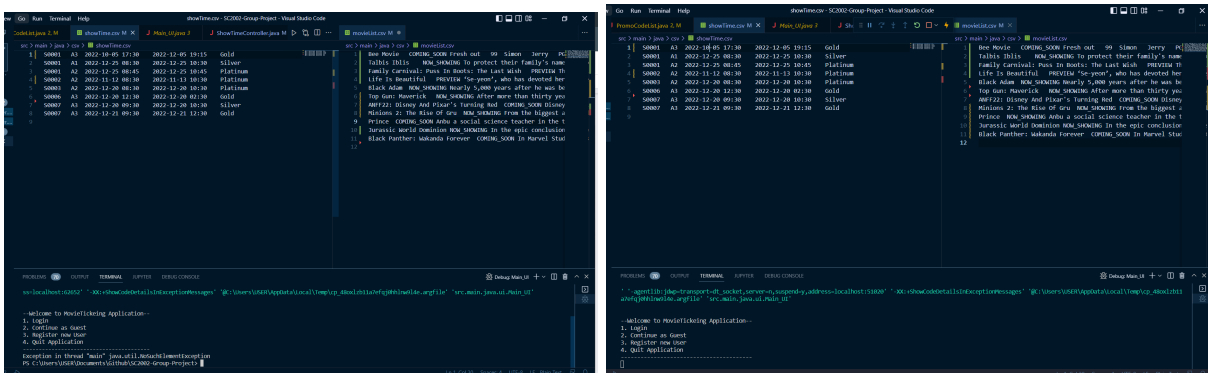
- Check through the entire show time list
- If there exist an entries where current time is later than the **end** time of the showing,
- The showtime is removed
- After removing the showtime, if there exist no showtime for the corresponding movie
- The movie status is automatically updated to END_OF_SHOWING

The movie status is updated when the program starts.

<u>Screenshots:</u>

Notice the following entries:

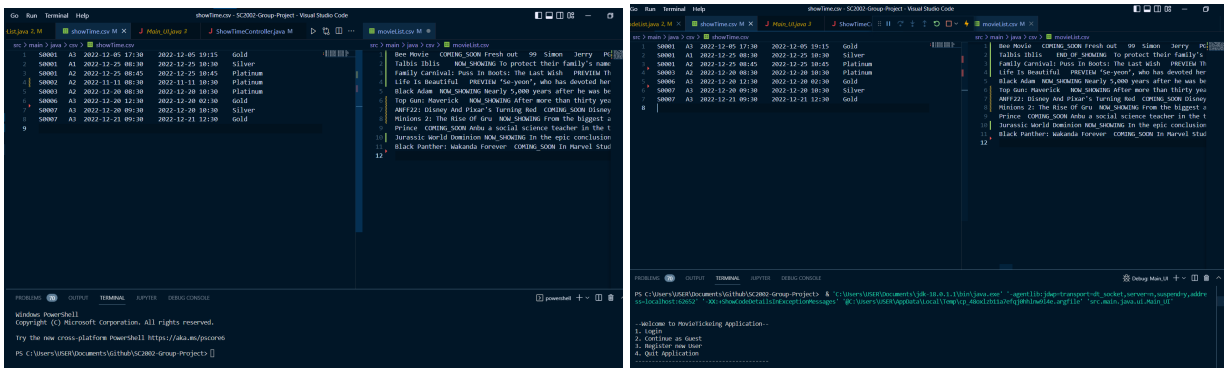
- movieList.csv - MovieTitle: Prince (Line 9) - ShowingStatus: COMING_SOON
- showTime.csv - MovieID : S0001(Line 1)



As there exist entrie(s) where the current time is later than the **start** time of the showing, the movie status of Prince will be set to NOW_SHOWING.

Notice the following entries:

- movieList.csv - MovieTitle: Talbis Iblis (Line 2)
- showTime.csv - MovieID : S0002 (Line 4)



As there exists an entry where the current time is later than the **end** time of the showing, the entry (Line 4 in showTime.csv) is removed.

As there exist no entries for after deleting the showtime, the movie status of Talbis Iblis will be set to END_OF_SHOWING.

Promotion Code Auto Refresh

The promo code is automatically set to available when the two conditions are reached:

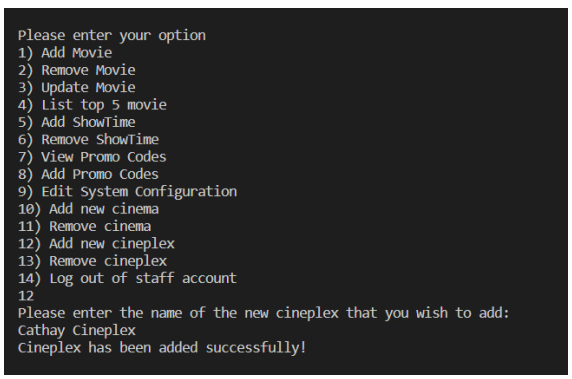
- PromoCodeStatus is not blocked (Blocked: The promo code will not be available until admin changes its status to ready/ available)
- Current time is later than promo code start time.

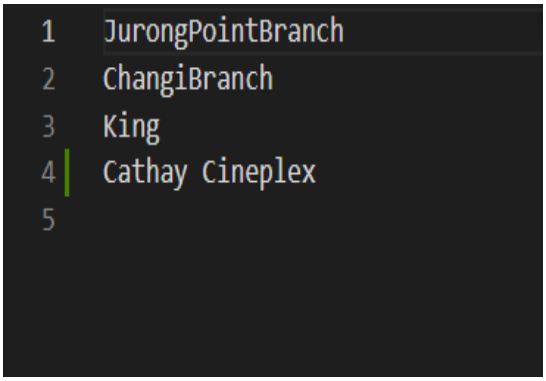
The invalid promo code is automatically deleted when current time is later than promo code end time.

The promo code status is updated when the program starts or when other classes try to get the info of the promocode by invoking functions(e.g. getAvailableCodeList, checkPromoCode).

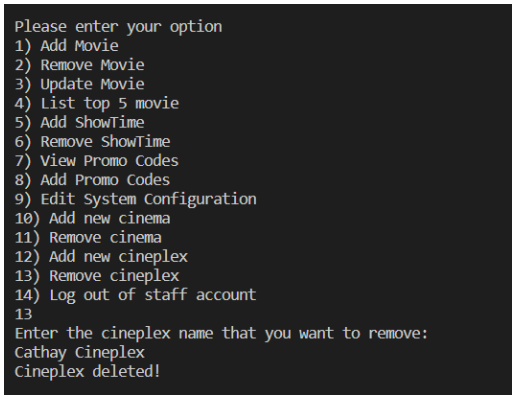
Create/Remove cineplex

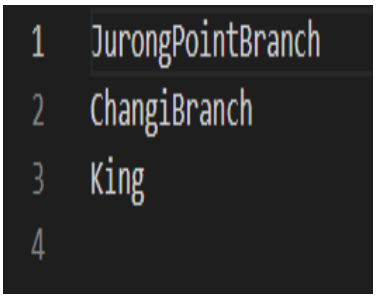
Create a new cineplex





Remove cineplex

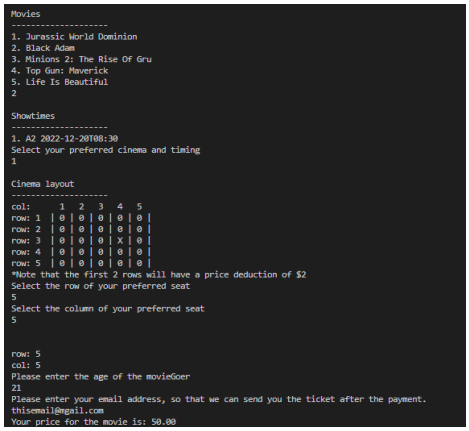




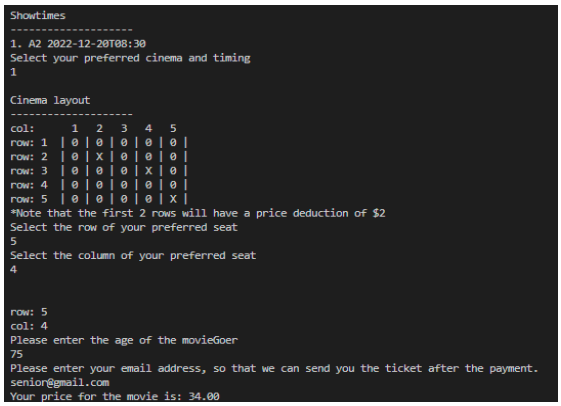
→ Administrator will be able to add or remove any cineplex.

<u>Automatic Price Calculation</u>

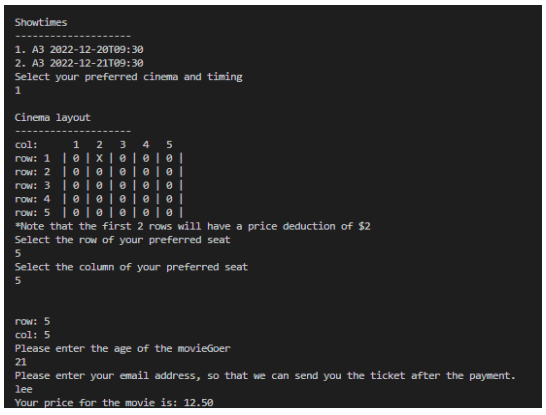
Platinum (20 Dec 22), Adult: $50.



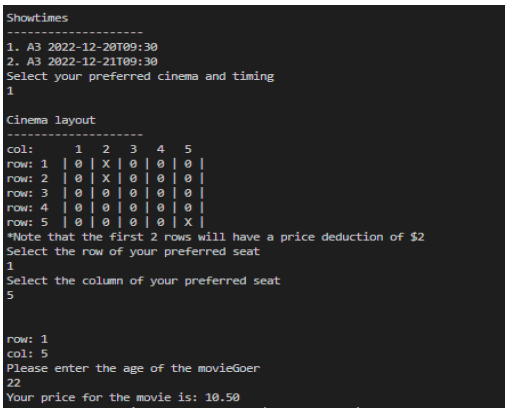
Platinum (20 Dec 22), Senior: $ 34.00.



Silver(20 Dec 22), Adult: $12.50.



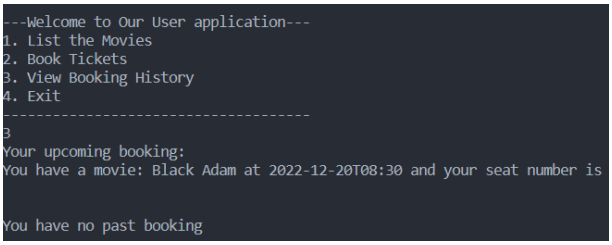
Silver (20 Dec 22), Adult: $10.50 (second row, less premium seats)



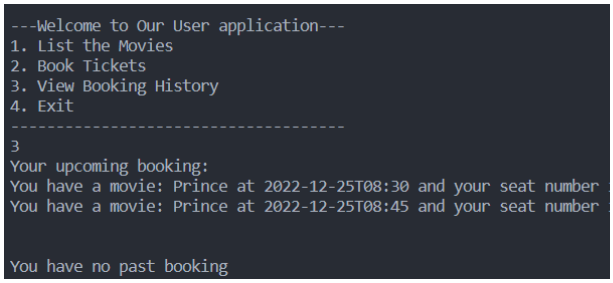
→ The Ticket price takes into consideration, **Age** of MovieGoer, Premium of **Seat**, **Time** of the day, **Class** of Cinema, **Type** of movie, **Day** of the week and eve/**public holiday** to calculate the price of a particular movie screening.

Individual Booking History

Booking History of Yu Xuan



Booking History of Robin



→ **Every user** has their own **dedicated booking history.** They will only be able to view their own booking history and no one else can view it. We have also **separated the bookings** into **upcoming bookings and past bookings**. Allowing the user to differentiate bookings that are already older and upcoming bookings.

Propose of New Features:

New feature 1:

To enhance functionality of MOBLIMA, the application should be able to cater for purchases other than movie tickets, such as food and beverages sold in the cinema. Hence, users can purchase food and beverage as a set with movie tickets which is often seen in real world scenarios.

→ Single Responsibility Principle:

Every class is singly responsible, such as File_IO is only responsible for reading and writing of the files. Hence, to load and save data into a particular file, File_IO is called instead of having a read-write method for every class.

→ Open-Closed Principle:

Items sold have individual classes. To expand the number of items sold, our design can easily be expanded to include more items to be sold such as having a popcorn class to be inherited by an item superclass.

→ Reusability:

To process the payment, Payment_UI can be called and reused to process the payment.

New feature 2:

With the hastening of the finTech industry, it is inevitable that more payment functions will be developed beyond conventional visa, nets or cash payment. One such example is the use of cryptocurrency. Hence expandability to accept newer payment methods should be allowed with minimal changes.

→ Liskov's Substitution Principle:

With implementation of a new payment method, procedure of payment remains largely unchanged as pre-conditions are no stronger, while post-conditions are no weaker than base class.

→ Dependency Injection Principle:

Our current payment method is invoked using a payment interface, such that a new payment method can be easily supported with a new payment class inheriting from PaymentMethodInterface which is both dependent upon abstraction.

Test Case

For all detailed test cases, refer to TestCase.doc attached. Basic test cases are shown here.

Admin Module

1) Login	2) Create Movies	3) Remove Movie
Input: 1, staff1, staffpass Output: Login Successful …	Input: Bee Movie, Fresh out of college, Barry … Cartoon … Output: Movie added successfully!	Input: Bee Movie Output: Movie is deleted!
4) Update Movie	5) Add Promotion Code	6) List Promo Code
Input: Bee Movie, Transformer Movie	Input: 1, staff1, staffpass, 8, S0001,5,2022-11-08 10:30,	Input: 1, staff1, staffpass, 7

Output: Movie is updated!	2022-11-08 12:00, READY, 0.50 Output: Promo Code added successfully!	Output: 1. PromoNow : 0.5 (AVAILABLE)
7) Add Showtime	8) Remove showtime	9) Edit System Config
Input: 1, staff1, staffpass, 5, S0001, A1, 2022-11-08 10:30, 2022-11-08 12:00, Platinum Output: *showtime has been added to csv*	Input: 1, S0001, 1 Output: *showtime has been deleted from csv*	Input: 9, 4, 10.50, 9 , 5 Output:Please enter the new base price: Base price: 10.5

Movie-goer Module

10) Login	11) Purchasing of movie tickets	12) View Booking
Input: 1, yj, 1234 Output: Login Successful …	Input: 1, yj, 1234,2,1,2,1 ,1,2,21,1,1 Output: Dear yj, Please view your details in the 'My Booking History' tab and present it at the ticket counter. Booking successful	Input: 1, yj, 1234,3 Output: Your upcoming booking: You have a movie:Prince …54 Your past booking: You have a movie: Top Gun: Maverick … 12
13) Show Review	14) Add Review	15) Check Movie Details
Input: 1, yj, 1234,1, 1, 3, 3 Output: Review 0: This is okay Rating: 4 Review 1: WOW BEST MOVIE… Rating: 5	1, yj, 1234,1, 1, 3, 2, The movie was good,5,1,1,3,3 Output: (You can see your review is updated)	Input: 1, yj, 1234,1, 1, 1, 1 Output: Title: Titanic Synopsis: James Cameron's 'Titanic' is an epic, … Restriction: PG13