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 honored 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 (CE2002 or CZ2002)	Lab Group	Signature/Date
LEONG HONG YI	SC2002	SS8	[illegible] 06/11/2022
LEONG KAI FONG	SC2002	SS8	[illegible] 06/11/2022
LIM JUN HERN	SC2002	SS8	[illegible] 06/11/2022
TAN JING HAN	SC2002	SS8	[illegible] 06/11/2022
TAN KOK ANN JEFF	SC2002	SS8	[illegible] 06/11/2022

Design Consideration

Our application design aims to ensure that it is easy to maintain, editable and extendable. This can be done by properly using classes' dependencies and minimizing the impact of change in our system.

<u>SOLID Design Principles:</u>

Single Responsibility Principle:

The Single Responsibility Principle (SRP) promotes high cohesion within a class. Here, we satisfy SRP by having each class perform only one highly specific task. Below are examples of how we implemented SRP in our design:

The `MovieTicCal` class is responsible for calculating movie ticket prices based on factors such as `AgeGroup`, `CinemaType`, `MovieType`, `DateType` etc. This is done so that other classes, such as `BookingSession` and `MovieTicket`, do not violate the SRP by recalculating the ticket price, which is not their responsibility.

Furthermore, the `DisplaySeats` class only displays the cinema layouts-in terms of booked and available seats. Although this task can be done by the `Cinema` class, that would be a violation of SRP since it gives `Cinema` class more than one responsibility. Furthermore, by making the method in `DisplaySeat` class public, other classes who wish to display the layout can use this method without repeating the code.

For the `ReviewMaker` class, instead of placing this responsibility on `Movie` class, the `ReviewMaker` class has one task, which is to update the review for a particular movie.

The `DataManager` class is in charge of communication between any data-dependent Classes and the `DataStore` class where all the data is stored. The `DataManager` Class achieves this by being responsible for loading and updating operations from and into the data store.

Open-Closed Principle:

This principle highlights the importance of open for extension but closed for modification. This is visible in some of the flexibility in the design of classes such as `MovieTicket` class, `DisplaySeat` class and so on for future extension..

The `MovieTicket` class is designed in such a way that it can be extended for other types of `MovieTicket`. For example, in the near future if the movie tickets are refundable, `RefundableMovieTicket` class can be created which extends from the `MovieTicket` class. The only difference would be the overriding of the price from positive to negative where negative indicates a refund.

As for the `DisplaySeat` class, it could be extended to a `DisplaySeatFestive` class which displays the cinema layout with some extra details such as Christmas banners when overriding the `displaySeat()` method.

For the `Review` class, it can be extended to an `OfficialReview` class if the cinema decides that they want to provide their own reviews as well. The `printRreview()` method can be overwritten to show that it is an official review from the cinema.

Liskov Substitution Principle:

This principle refers to how an object's subtypes must be substitutable for their base type. For example, we have used the `Account` interface as the base class and the `CinemaStaff class` as the subclass. Here, the implementation of the `login()` method in the `Account` interface was done by `CinemaStaff` class. As the post-conditions are no weaker than the base class (`Account` interface) method and the pre-conditions are no stronger than the base class method, `CinemaStaff` is substitutable for `Account`. The subclass does everything the base class does and even more. Therefore, the Liskov Substitution Principle is not violated.

Object Oriented Concepts

Abstraction

Abstraction involves selecting essential details of an object which distinguish it from other kinds of objects while hiding the insignificant ones. In our projects, real-world objects such as `Cinema`, `Movie` and `Seat` are abstracted and represented as Java classes. For example, the cineplexes are represented by the `Cineplex` objects, which contain their `location` and corresponding `cinemas`.

Encapsulation and Information Hiding

Encapsulation refers to building barriers to protect objects' private data. To achieve encapsulation, attributes in our classes are set to private, which can only be accessed through their respective getter or setter methods. For example, the `ticketSales` of `Movie` objects could only be accessed through the `getTicketSales` method.

Information hiding refers to hiding the implementation detail, such as the inner workings of methods of the class, from users. One of the applications of information hiding in our project is the `DisplaySeat` class, which implements the `displaySeat` method and allows client classes to use it without needing to know the implementation detail.

Inheritance

Inheritance is a mechanism which defines a new class while inheriting the attributes and methods from its parent class. In our project, the `Account` interface is created with a `login` method. All classes that will implement this method, such as the `CinemaStaff` class in our project, have to implement this `login` method in their classes, respectively.

Polymorphism

Polymorphism, known as the most powerful feature in object-oriented programming, allows us to perform a single action in different ways using the same base class. For instance, the constructor of the `ShowTime` class uses `dateOfShow` as one of its parameters with `Calendar` type. Regardless of what object is being passed in, as long as it is the subclass of the `Calendar` class, such as `GregorianCalendar` , the correct constructor will be called, helping us to simplify the code by removing the need to create different methods for each subclass. Other methods in the `ShowTime` class that uses the `dateOfShow's` methods, such as `getDateOfShowFormat`, will call the correct method depending on the actual type of the object as well due to the dynamic binding.

Proposal for new features

Password Hashing

The current method of directly storing the password of `CinemaStaff` as plain text is extremely insecure. To protect our application from the malicious attacks from hackers, one possible new feature will be storing the password as hashed value, such as SHA-256, instead.

Considering encapsulation as well as the Single Responsibility Principle, the `Menu` class in our application is only in charge of displaying the menu and calling the respective function, while the validation of the password is handled by the `CinemaStaff` class. Thus, changes could easily be made within our `CinemaStaff` class without affecting other classes. The user could input their passwords in the `Menu` as before, while the hashing process will be done in the `CinemaStaff` class.

Promotions

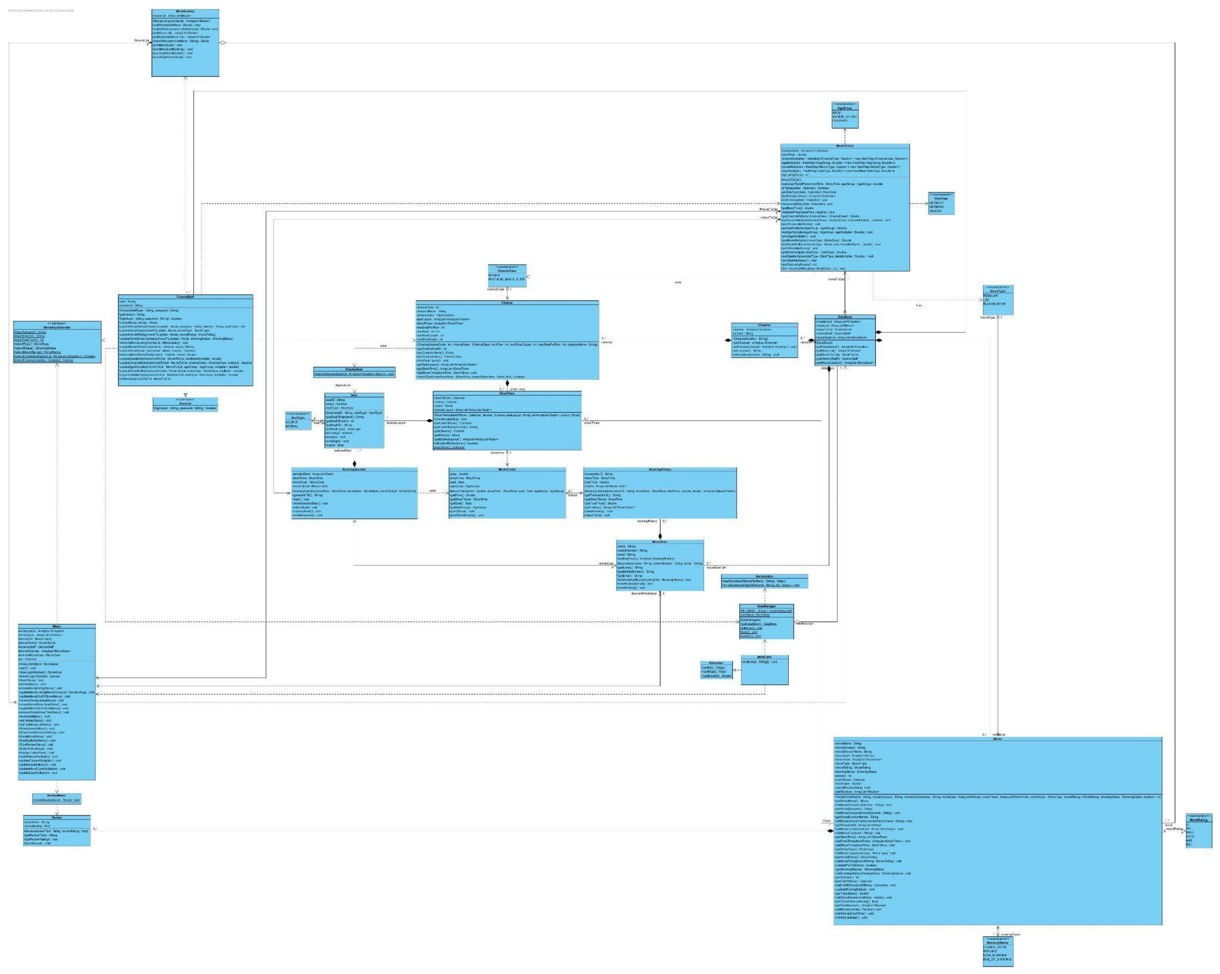
Another new feature that we can add to our system would be allowing admins to implement promotions for different movies to attract more viewers. By setting those movies' type as `PROMOTION`, moviegoers could check which movies are under promotion in the menu, as well as buy the tickets at a lower price.

Based on the Single Responsibility Principle, the calculation of the ticket price has been isolated from the `BookingSession` class and has been completely moved to the `MovieTicCal` class. The `MovieTicCal` class will solely handle the logic of calculating ticket prices.

To achieve high reusability and extensibility in our application, instead of hard coding the ticket price based on different input types, `HashMaps` storing the multiplier for different input types has been used in the `MovieTicCal` class. Thus, to cater for this new feature, we can simply add this new `MovieType` to the `HashMap` without requiring a cascade of changes. In the event we want to update the discount rate for those movies under promotion, the according `HashMap` could be updated without affecting the source code, which improves the maintainability of this class.

Most importantly, due to the low coupling and high cohesion properties of the `MovieTicCal` class, the impact on other classes will be minimal. The `MovieType` class will be updated to cater for the new enumeration, while the `Menu` class will be updated to display the menu for setting and viewing promotions.

UML Class Diagram



Test Cases (All test cases are passed)

S/No.	Description of Test Step	Expected Result
1	**Portal Selection**	
1.1	1. Enter a non-integer 2. Enter an invalid integer	1. “Please enter an integer” Error message. 2. “Invalid integer! Please enter number 1-3!” Error message.
2	**Movie-Goers Menu (Option 2)**	
2.1	**Registering** 1. Enter an invalid email address 2. Enter an invalid name 3. Enter an invalid number	1. "Enter a valid email address format, e.g. abc@123.com" error message 2. “only enter alphabets” error message. 3. “Enter an 8-digit valid phone number, e.g.12345678” error message
2.2	**Login** 1. Enter the email address that is in the database	1. Movie-goer menu will be displayed after logged in
2.3	**List Movie** 1. Show movies 2. Show top 5 movies (sales) 3. Show top 5 movies (ratings)	1. Default movie List 2. Top 5 movie by descending sales ($, excluding GST) 3. Top 5 movie sorted by descending ratings
2.4	**Search Movie** 1. Enter the movie name you want to search	1. If the movie exists, movie title, synopsis, director, duration, showing status, release rating, movie type, cast, and overall reviewer rating is shown 2. Else, it should say “Movie not found”

2.5	**View Movie Details** 1. Select any of the movies by their corresponding number in the movie list	1. Movie title, synopsis, director, duration, showing status, release rating, movie type, cast, and overall reviewer rating is displayed 2. If the number of ratings <= 1, overall reviewer rating will be shown as NA
2.6	**Book Movie** 1. Select a movie 2. Select a showtime	1. Movie List (Only “Preview” or “Now Showing” is listed, notice the difference between the listing shown in 6:55 and 7:06 in the video) 2. Showtime List
2.6.1	**Select Seat** ***(Couple seats are implemented)***	1. Users are only allowed to select unoccupied seat 2. Users are not allowed to leave an empty seat between selected seat
2.6.2	**Unselect Seat**	1. The previously selected seat will be unselected 2. Otherwise, “Seat ID not found!” will be printed
2.6.3	**Make Payment**	1. If the user hasn’t selected any seats, “You have not selected any seat” error will be printed 2. Otherwise, the user will be prompted to select corresponding age group for the seat, payment will be made instantly a. The price of tickets (with 7% GST) will be printed
2.7	**View Booking History** 1. Print all booking details 2. View ticket details	1. Transaction ID, Movie Title, Number of tickets, and Total Price are shown. 2. Select which booking to view ticket details 3. You should see

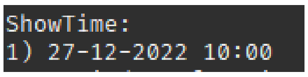
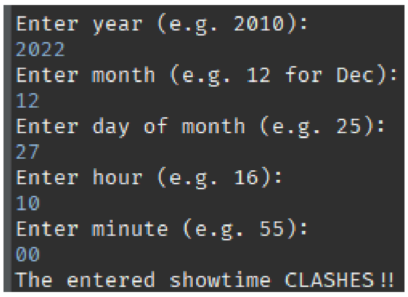
		a. Movie Title, Cinema Code, Date and Time, Seat Number, Price, Age Group
2.8	**Give Review** 1. Select a movie to review from the movie list 2. Enter your review text, followed by a rating score of 1-5.	1. Review will be made successfully 2. If the user gives a rating value less than 1, it will be set to 1 3. If the user gives a rating value more than 5, it will be set to 5
3	**Admin Menu (Option 1)**	
3.1	**Login** 1. Enter the correct admin login details in the database 2. Else, return to the Portal Selection menu again	Admin Menu will be displayed after successfully logged in
3.2	**Create Movie Listing** 1. Enter a movie name, synopsis and the movie director's name. 2. Enter the number of casts 3. Enter the cast names, movie duration, movie type, movie rating and showing status. 4. Choose cineplex and cinema 5. Enter showtime of the movie.	1. You should be able to see the error message "Please enter an integer" 2. You will be required to re-input the number of casts if it is less than 2 3. You should be able to select the options for the types, rating, and showing status 4. You will return to the menu after entering the details for the new movie 5. The new movie will be added to the movie list
3.3	**Update Movie Listing** 1. Upon choosing update movie listing, a menu to select which component to update will be brought up: 2. Choose movie to update	After being updated by the admin, the changes will be reflected when users trying to see the movie details

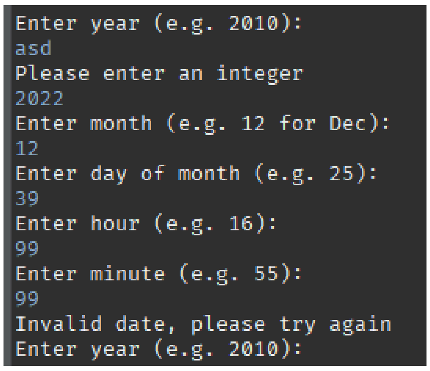
3.3.1	**Update Movie Details** 1. Asked to update movie synopsis 2. Asked to update director name 3. Asked to update number of cast 4. Asked to update each cast name	
3.3.2	**Update Movie Type** 1. Admin will be asked to update movie type	
3.3.3	**Update Movie Showing Status** 1. Admin will be asked to update movie showing status	
3.3.4	**Update Movie Rating** 1. Admin will be asked to update movie rating	
3.4	**Remove Movie Listing** 1. Enter the movie to be deleted	The selected movie will be removed from the movie list as well as the bookable movie list
3.5	**Create Movie ShowTime** 1. Choose the cineplex, cinema and movie	The corresponding showtime will be created
3.6	**Update Movie ShowTime** 1. Enter the movie, cineplex and the cinema 2. Choose the showtime to update 3. Enter the new showtime	The corresponding showtime will be updated if it is not booked by any user
3.7	**Remove Movie ShowTime** 1. Choose a movie to update 2. Choose cineplex and cinema 3. Choose showtime to remove	The corresponding showtime will be removed if it is not booked by any user

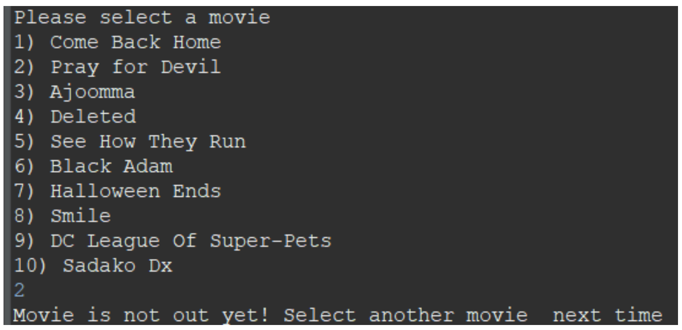
3.8	**Change Ticket Price** 1. Choose which multiplier to update (Base, Cinema, Age, MovieType, Date)	1. The corresponding multiplier will be changed successfully 2. The users should see the new ticket price when they are making payment for their booking
3.9	**Set Holiday** 1. Add/remove the holidays in the set holiday menu.	1. The corresponding holiday date will be added / removed 2. Ticket price is shown correctly when booking is done on that date
3.10	**Set top movie listing** 1. Choose how the movie-goers can rank the top 5 movies by	The corresponding changes will be saved and be reflected in the movie-goer menu
3.11	**Update movie end-of-show date** 1. Select a movie to update 2. Enter the new end-of-show date for the movie	The corresponding movie will be removed from movie list as well as the bookable movie list after the end-of-show date

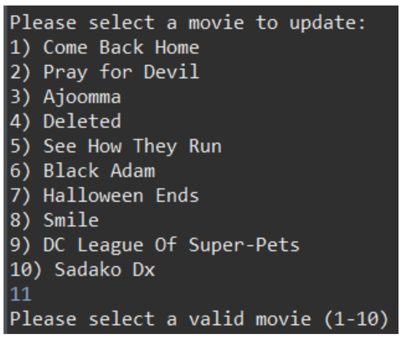
Extra Test Cases not covered in Demo

No	Features	Expected Result

1	Time Clash Check
	● Creating the same showtime in the same cinema, or creating the showtime before the previous movie has ended is not allowed
		
		`ShowTime:`
		`1) 27-12-2022 10:00`
		
		`Enter year (e.g. 2010):`
		`2022`
		`Enter month (e.g. 12 for Dec):`
		`12`
		`Enter day of month (e.g. 25):`
		`27`
		`Enter hour (e.g. 16):`
		`10`
		`Enter minute (e.g. 55):`
		`00`
		`The entered showtime CLASHES!!`

2	Date Validation
	● Error handling for invalid date input
		
		`Enter year (e.g. 2010):`
		`asd`
		`Please enter an integer`
		`2022`
		`Enter month (e.g. 12 for Dec):`
		`12`
		`Enter day of month (e.g. 25):`
		`39`
		`Enter hour (e.g. 16):`
		`99`
		`Enter minute (e.g. 55):`
		`99`
		`Invalid date, please try again`
		`Enter year (e.g. 2010):`

3	Movie Review
	● Review making is not allowed for movies which status is coming soon
		
		`Please select a movie`
		`1) Come Back Home`
		`2) Pray for Devil`
		`3) Ajoomma`
		`4) Deleted`
		`5) See How They Run`
		`6) Black Adam`
		`7) Halloween Ends`
		`8) Smile`
		`9) DC League Of Super-Pets`
		`10) Sadako Dx`
		`2`
		`Movie is not out yet! Select another movie  next time`

4	IndexOutOfBounds Handling
	● The users/admin will be prompted to select a valid index or be returned to the previous menu
		
		`Please select a movie to update:`
		`1) Come Back Home`
		`2) Pray for Devil`
		`3) Ajoomma`
		`4) Deleted`
		`5) See How They Run`
		`6) Black Adam`
		`7) Halloween Ends`
		`8) Smile`
		`9) DC League Of Super-Pets`
		`10) Sadako Dx`
		`11`
		`Please select a valid movie (1-10)`