Declaration of Original Work for SC2002 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	Lab Group	Signature/Date
Angelina Tang un Gei	SC2002	FDAC	 26/4/2024
Lu Shanshan	SC2002	FDAC	 26/04/2024
Ng Jia Xin	SC2002	FDAC	
Low Jo Yi, Nicole (Liu Zuyi)	SC2002	FDAC	 26/04/2024
Kumaresan Gayathri Devi	SC2002	FDAC	 26/4/2024

DESIGN CONSIDERATIONS

SOLID PRINCIPLE:

To avoid a rotting design characterised by rigidity, fragility and immobility, we anchored the design on our system on the SOLID principles. By doing so, we are able to improve reusability, extensibility and mobility of the system, enabling us to achieve loose coupling and high cohesion.

1. **Single responsibility principle (SRP)**

   Using SRP, which states that a class should have only one reason to change, we created classes that only have a single responsibility. For instance, the 'NewOrder' class is only responsible for storing the order information as well as updating the order statuses. The only reason to change it is if new status types are introduced. The segregation of responsibilities among classes reduces coupling which increases flexibility and modularity of our system. As such, we are able to make changes to a class in the system with minimal impacts on the other classes.

2. **Liskov substitution principle (LSP)**

   According to the LSP, objects of a superclass should be replaceable with objects of a subclass without affecting the application which means that subclasses should not require their methods to be used in a more restrictive way than the superclass. We created a 'PaymentMethod' abstract class which defines a contract for payment operations that all payment types within our system adhere to. The subclasses, 'Cash()', 'OnlineBanking()' and 'CreditCard()', will provide their own implementation of the method declared in 'PaymentMethod'. As the method remains the same for the subclasses, they are able to replace 'PaymentMethod' without altering the correct functioning of the system.

3. **Interface segregation principle (ISP)**

   As the ISP states that no client should be forced to depend on methods it does not use and many client specific interfaces are better than one general-purpose interface, we strategically created specific interfaces. This promotes reusability and extensibility of our system. For instance, the 'BranchStaffList' interface defines methods only necessary for 'ConcreteBranchStaffList'.

4. **Open/closed principle (OCP)**

   Considering OCP, we have designed our system such that the classes and functions are open for extension but closed for modification. One such abstract class, 'PaymentMethod', was created to facilitate addition of new payment options without modifying existing code. We are able to extend the functionality of payments without the need to change the classes that depend on the 'PaymentMethod' class.

5. **Dependency inversion principle (DIP)**

   In line with the DIP, we ensured that the classes do not rely directly on concrete implementations but rather on interfaces and abstractions. We created interfaces such as branch staff list and concrete branch staff list class which implements the methods. Classes such as staff list are reliant on branch staff list instead of concrete branch staff list which ensures flexibility in introducing new functionalities. Another example is the payment method with the classes - online banking and credit card extensions. This inversion enables us to more easily add a new method of payment.

APPROACH TAKEN:

Abstraction:

Abstraction is achieved by the separation of implementation details from the abstract concepts and behaviours by using abstract classes and interfaces. For the reusability, extensibility, and modularity, as new branch types, payment methods, or staff management implementations can be added without modifications. Usage of Abstraction can be seen in:

1. Abstract Class 'PaymentMethod': It has an abstract method called processPayment(). Concrete classes like OnlineBanking(), CreditCard(), and Cash() extend this abstract class. These 3 classes have to implement processPayment().
2. Interface 'BranchStaffList': It declares methods like addStaffToBranch(Account Staff), promoteToManager(String username), etc. The ConcreteBranchStaffList class implements this interface and provides the actual implementation for these methods.

Inheritance:

Inheritance is a mechanism that allows a class to inherit properties and methods from another class, known as the superclass or parent class. The class that inherits is called the subclass or child class. The code promotes code reuse and avoids duplication of common properties and methods across related classes by the usage of inheritance. Subclasses can inherit and extend the functions of their superclasses, while also adding their own specific behaviour. The usage of inheritance can be seen in:

1. **Account Class Hierarchy:**
   a. 'Account' – 'Name', 'age', 'gender', 'LoginUsername', and 'role'
   b. 'ManagerStaff" and 'NormalStaff" extends 'Account'
2. **Branch Class Hierarchy:**
   a. 'Branch' – 'location', 'staffQuota' and 'addOrderToBranchList(NewOrder order)'

b. ‘NTUBranch’, ‘JPBranch’, ‘JEBranch’ extend ‘Branch’

Polymorphism:

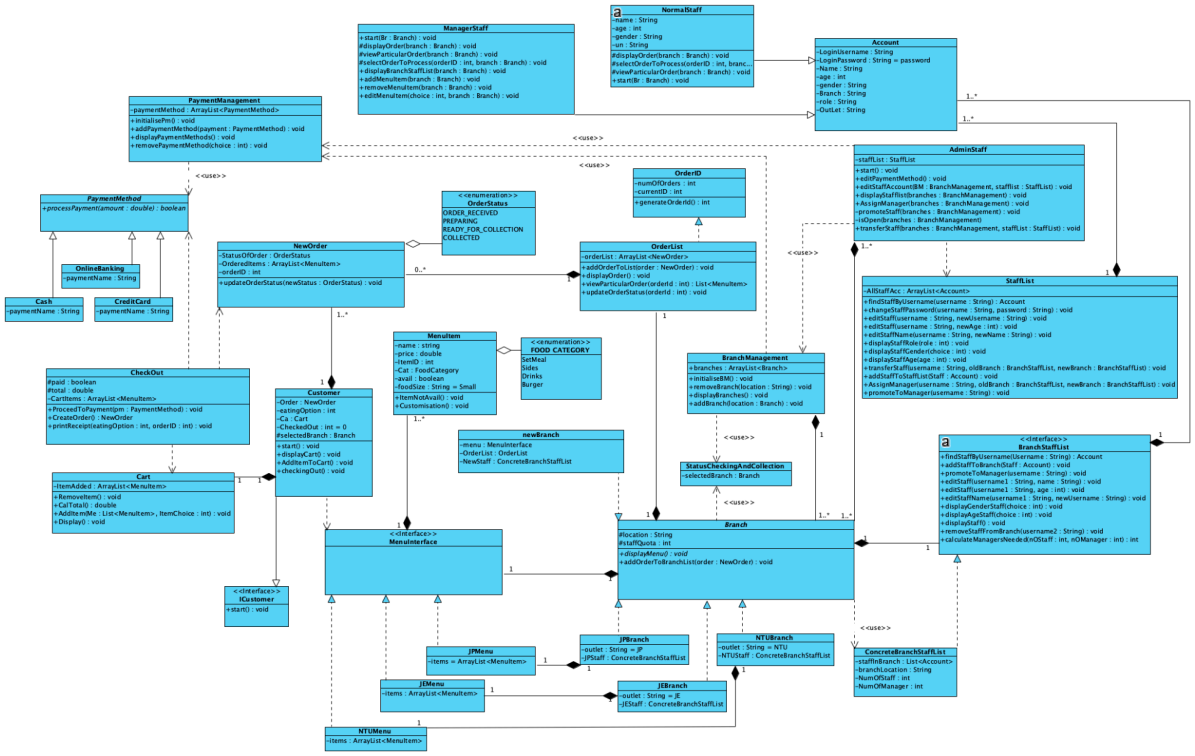
Polymorphism is the ability of an object to take on many forms. It allows objects of different classes to be treated as objects of a common superclass. The code achieves flexibility and extensibility by the leverage of polymorphism. Objects of different classes can be treated as objects of a common superclass of interface, which allows code to reuse and the ability to write more generic and maintainable code. The usage of polymorphism can be seen in:

1. **Method Overloading:**
   There are 2 editStaff methods: editStaff(String username, String newUsername) and editStaff(String username, int newAge). While both methods have the same name, different parameters are passed into them.
2. **Method Overriding:**
   The PaymentMethod abstract class defines a contract (processPayment() method) that any payment implementation must follow. The ‘OnlineBanking’, ‘CreditCard’ and ‘Cash’ classes provide their own implementation of the ‘processPayment()’. Objects of these concrete classes can be treated as objects of the ‘PaymentMethod’ abstract class, allowing the polymorphic behaviour in the payment process.

Encapsulation:

Encapsulation was achieved by making class members private and providing public methods to access and modify them, ensuring data integrity and hiding the internal implementation.

UML DIAGRAM



Object oriented and design:

- The code uses object oriented learning with classes representing different entities in the restaurant management system.
- Inheritance is used to create hierarchies of related classes.
- Interfaces are used to define contracts and enable polymorphism.

Modular and Extensible Architecture:

- Designed in a modular way, with separate classes handling different responsibilities, such as account management, and payment processing.
- This design allows for easier extension and maintenance of the system, as new features or changes can be implemented by adding or modifying specific components.

User Interfaces and interactions:

- The code includes interface-related classes which provide entry points for customer and staff interactions with the system.

- These user interface classes handle tasks like placing orders, collecting orders and managing staff accounts.

Administrative Functionality:

- The 'AdminStaff' class provides administrative functionalities such as managing staff accounts, managing payment methods and opening and closing branches.
- This centralised administrative functionality allows for the management of the overall restaurant operations.

Branch and Menu Management:

- 'Branch' class and its subclasses handle the management of different branch locations including their staff quota, order list and menus.
- 'NTUMenu', 'JEMenu' and 'JPMenu' manage the menu items for each branch allowing for branch-specific customization.

TEST CASES AND RESULT

Test Case 1: We first chose the 'Add menu item' option and entered 'chicken', '5.50' and 'Sides' for the item name, price and category respectively. Test case passed and a new item 'chicken' was added to the menu list.

```
1. View all orders
2. View particular orders
3. Process order
4. Display branch staff list
5. Add menu item
6. Remove menu item
7. Edit menu item
8. Exit
Enter your choice: 5
Enter item name
chicken
Enter item price
5.50
Enter item category
Sides
1    FRIES     3.2
2    3PC set meal     9.9
3    chicken nugget     6.6
4    chicken     5.5
```

Test Case 2: We first chose the 'Edit menu item' option and updated the price of 'FRIES' from 3.20 to 5.50. Test case passed and the price was successfully updated and reflected in the menu.

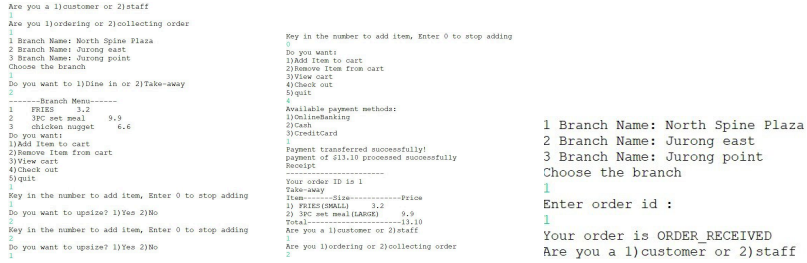
```
1. View all orders
2. View particular orders
3. Process order
4. Display branch staff list
5. Add menu item
6. Remove menu item
7. Edit menu item
8. Exit
Enter your choice: 7
1.Edit Menu Item name
2.Edit Menu Item price
3.Edit Menu Item availability
2
1    FRIES     3.2
2    3PC set meal     9.9
3    chicken nugget     6.6
Which item are you editing
1
Key in the new price :
5.0
1    FRIES     5.0
2    3PC set meal     9.9
3    chicken nugget     6.6
```

```
Are you a 1)customer or 2)staff
1
Are you 1)ordering or 2)collecting order
1
1 Branch Name: North Spine Plaza
2 Branch Name: Jurong east
3 Branch Name: Jurong point
Choose the branch
1
Do you want to 1)Dine in or 2)Take-away
1
-------Branch Menu------
1    FRIES     5.0
2    3PC set meal     9.9
3    chicken nugget     6.6
```

Test Case 3: We chose the 'Remove menu item' option and removed '3PC set meal' from the menu list. Test case passed, item was successfully removed and not displayed in the menu anymore.

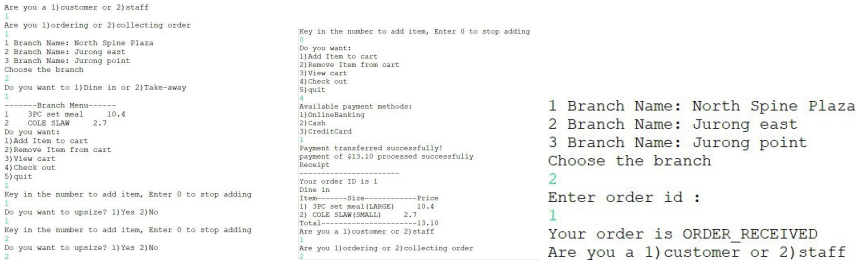
```
Enter your choice: 6
1    FRIES     3.2
2    3PC set meal     9.9
3    chicken nugget     6.6
Which item you want to remove?
2
Item removed!
1    FRIES     3.2
2    chicken nugget     6.6
```

Test case 4: We chose the option to takeaway and then chose the ‘Add item to cart’ option. We added 2 items to our cart, with the option to upsize being our customisation. The second item was upsized while the first was not. After paying successfully, the customer can check the order status which indicated ORDER_RECEIVED. Test case passed.



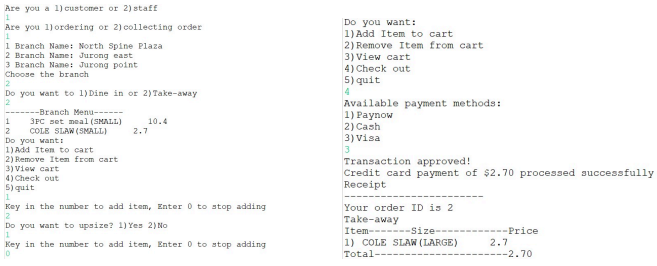
```
Are you a 1)customer or 2)staff
1
Are you 1)ordering or 2)collecting order
1
1 Branch Name: North Spine Plaza
2 Branch Name: Jurong east
3 Branch Name: Jurong point
Choose the branch
1
Do you want to 1)Dine in or 2)Take-away
2
-------Branch Menu-------
1    FRIES     3.2
2    3PC set meal     9.9
3    chicken nugget     6.6
Do you want:
1)Add Item to cart
2)Remove Item from cart
3)View cart
4)Check out
5)quit
1
Key in the number to add item, Enter 0 to stop adding
1
Do you want to upsize? 1)Yes 2)No
2
Key in the number to add item, Enter 0 to stop adding
2
Do you want to upsize? 1)Yes 2)No
1
Key in the number to add item, Enter 0 to stop adding
0
Do you want:
1)Add Item to cart
2)Remove Item from cart
3)View cart
4)Check out
5)quit
4
Available payment methods:
1)OnlineBanking
2)Cash
3)CreditCard
1
Payment transferred successfully!
payment of $13.10 processed successfully
Receipt
----------------------
Your order ID is 1
Take-away
Item-------Size-----------Price
1) FRIES(SMALL)     3.2
2) 3PC set meal(LARGE)     9.9
Total---------------------13.10
Are you a 1)customer or 2)staff
1
Are you 1)ordering or 2)collecting order
2
1 Branch Name: North Spine Plaza
2 Branch Name: Jurong east
3 Branch Name: Jurong point
Choose the branch
1
Enter order id :
1
Your order is ORDER_RECEIVED
Are you a 1)customer or 2)staff
```

Test case 5: We chose the option to dine in and then chose the ‘Add item to cart’ option. We added 2 items to our cart, with the option to upsize being our customisation. The first item was upsized while the second was not. Test case passed since the receipt prints out the ordered items with their customisations.



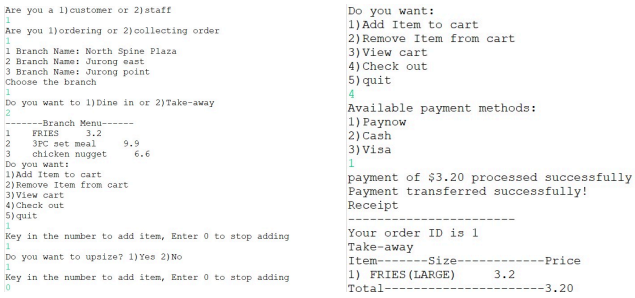
```
Are you a 1)customer or 2)staff
1
Are you 1)ordering or 2)collecting order
1
1 Branch Name: North Spine Plaza
2 Branch Name: Jurong east
3 Branch Name: Jurong point
Choose the branch
2
Do you want to 1)Dine in or 2)Take-away
1
-------Branch Menu-------
1    3PC set meal     10.4
2    COLE SLAW     2.7
Do you want:
1)Add Item to cart
2)Remove Item from cart
3)View cart
4)Check out
5)quit
1
Key in the number to add item, Enter 0 to stop adding
1
Do you want to upsize? 1)Yes 2)No
1
Key in the number to add item, Enter 0 to stop adding
2
Do you want to upsize? 1)Yes 2)No
2
Key in the number to add item, Enter 0 to stop adding
0
Do you want:
1)Add Item to cart
2)Remove Item from cart
3)View cart
4)Check out
5)quit
4
Available payment methods:
1)OnlineBanking
2)Cash
3)CreditCard
1
Payment transferred successfully!
payment of $13.10 processed successfully
Receipt
----------------------
Your order ID is 1
Dine in
Item-------Size-----------Price
1) 3PC set meal(LARGE)     10.4
2) COLE SLAW(SMALL)     2.7
Total---------------------13.10
Are you a 1)customer or 2)staff
1
Are you 1)ordering or 2)collecting order
2
1 Branch Name: North Spine Plaza
2 Branch Name: Jurong east
3 Branch Name: Jurong point
Choose the branch
2
Enter order id :
1
Your order is ORDER_RECEIVED
Are you a 1)customer or 2)staff
```

Test case 6 : Test case passed since Visa (credit card) payment was successful and it is printed that the payment is processed successfully.

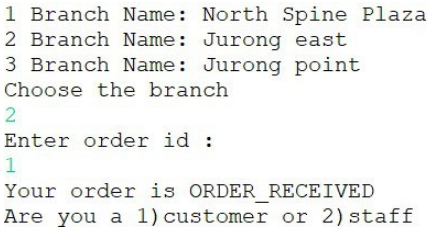


```
Are you a 1)customer or 2)staff
1
Are you 1)ordering or 2)collecting order
1
1 Branch Name: North Spine Plaza
2 Branch Name: Jurong east
3 Branch Name: Jurong point
Choose the branch
2
Do you want to 1)Dine in or 2)Take-away
2
-------Branch Menu-------
1    3PC set meal(SMALL)     10.4
2    COLE SLAW(SMALL)     2.7
Do you want:
1)Add Item to cart
2)Remove Item from cart
3)View cart
4)Check out
5)quit
1
Key in the number to add item, Enter 0 to stop adding
2
Do you want to upsize? 1)Yes 2)No
1
Key in the number to add item, Enter 0 to stop adding
0
Do you want:
1)Add Item to cart
2)Remove Item from cart
3)View cart
4)Check out
5)quit
4
Available payment methods:
1)Paynow
2)Cash
3)Visa
3
Transaction approved!
Credit card payment of $2.70 processed successfully
Receipt
----------------------
Your order ID is 2
Take-away
Item-------Size-----------Price
1) COLE SLAW(LARGE)     2.7
Total---------------------2.70
```

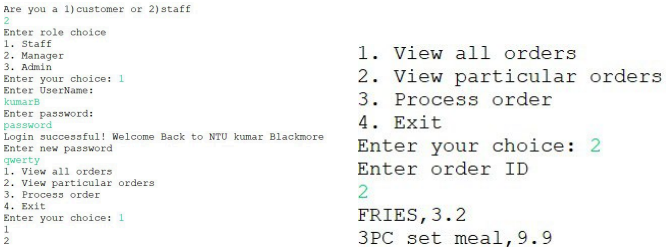
Test case 7: Test case passed since Paynow (online payment platform) payment was successful and it is printed that the payment is processed successfully.



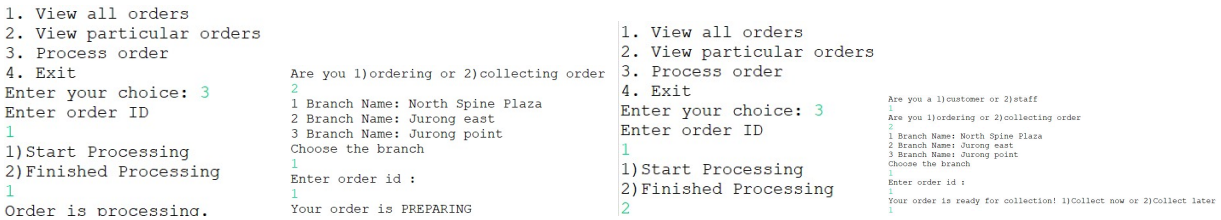
Test case 8: Test case passed since ‘ORDER_RECEIVED’ was displayed using the order ID of 1.



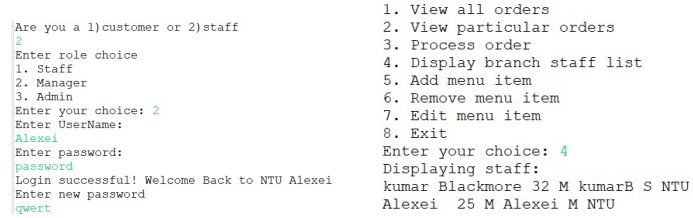
Test case 9: Test case passed. Logged in as a staff member and we can see that there are new orders of order IDs 1 and 2. To view any particular order, we will choose the ‘View particular orders’ option.



Test case 10: Staff processed the new order and the order status will be shown as ‘PREPARING’ for the customer. When the staff has finished processing the order, the customer will see that their order is ready for collection. Test case passed.



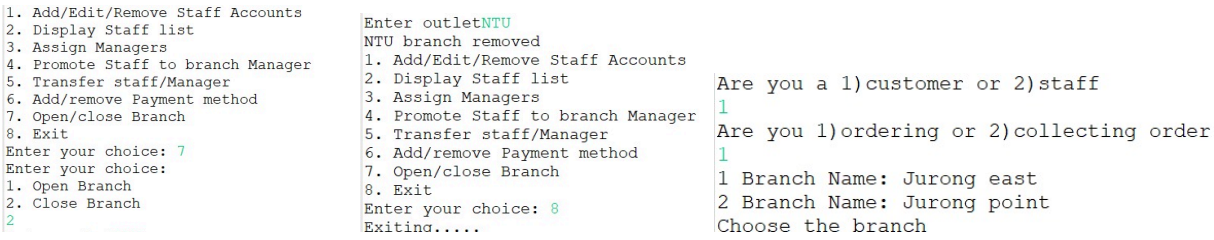
Test case 11: Logged in as ‘Alexei’, the manager in the NTU branch and successfully displayed the staff list in the NTU branch which includes Alexei and kumar Blackmore.



Test case 12: Test case passed since the manager was able to process an order successfully.

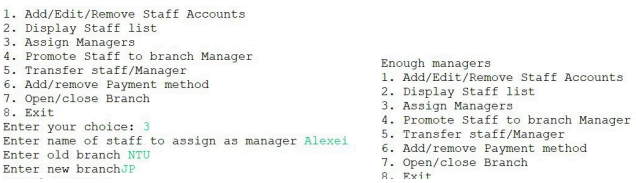
```
Login successful! Welcome Back to NTU Alexei
Enter new password
qwerty
1. View all orders
2. View particular orders
3. Process order
4. Display branch staff list
5. Add menu item
6. Remove menu item
7. Edit menu item
8. Exit
Enter your choice: 3
Enter order ID
1
Order processed successfully.
```

Test case 13: NTU branch was closed. Test case passed since the NTU branch does not appear in the Customer interface anymore.

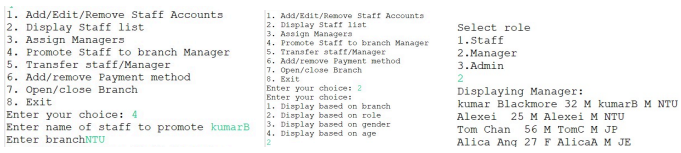


Test case 14: Test case passed. Staff list was successfully displayed according to the different filters.

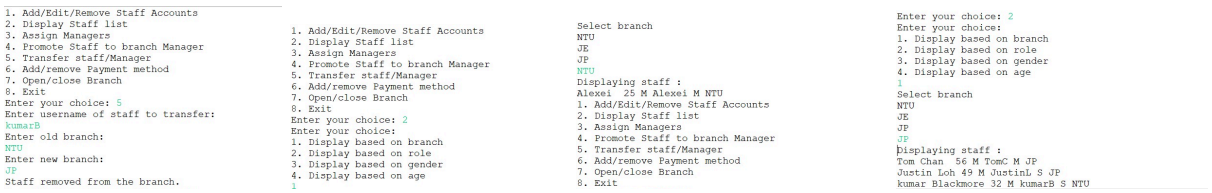
Test case 15: Alexei was originally a manager in the NTU branch but she was assigned manager in the JP branch. However, since the JP branch only has 2 staff in total, only 1 manager is needed and therefore Alexei cannot be assigned to the JP branch. Test case passed.



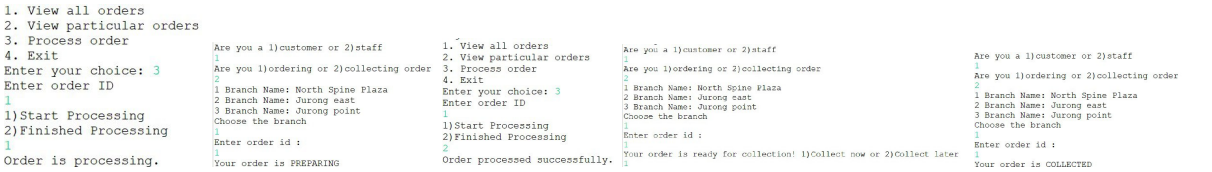
Test case 16: Test case passed since kumar Blackmore was promoted to Manager in the NTU branch and it is displayed in the staff list.



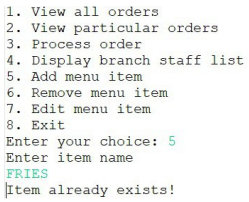
Test case 17: Test case passed since kumar Blackmore was transferred from the NTU to JP branch and his name is not displayed in the NTU staff list anymore, instead it is displayed in the JP staff list.



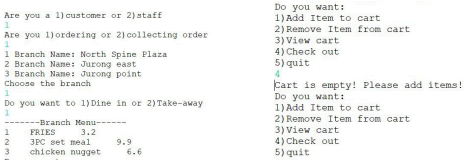
Test case 18: Customer places a new order and is given an order ID of 1. After staff processes the order, the customer uses the ID to check the status, which is that it is ready for collection. After the customer collects the order, the order status shows that it is collected.



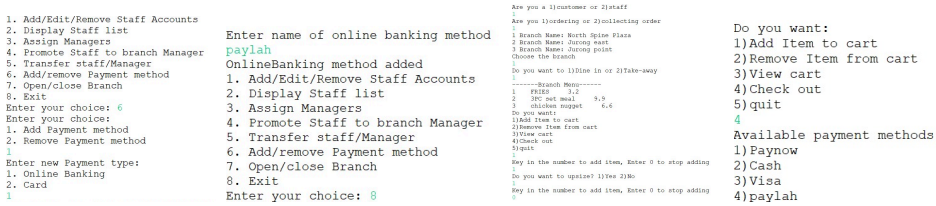
Test case 19: Test case passed since an error message was displayed when ‘FRIES’ was added but it already exists in the NTU menu and cannot be added into the menu.



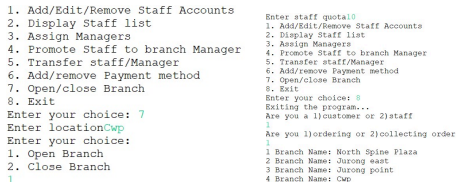
Test case 20: Test case passed since an error message was printed and customer was prompted to add an item again when the customer tried to cart out when there was nothing in the cart.



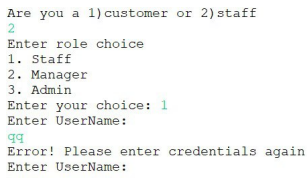
Test case 21: New payment method ‘paylah’ was added. Test case passed since it is displayed when the customer checks out their cart.



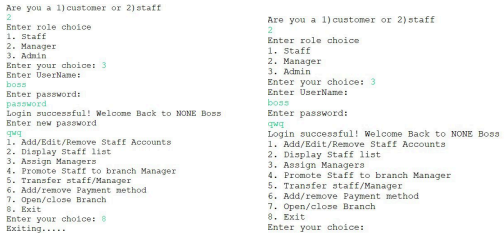
Test case 22: Opened a new branch called ‘Cwp’. Test case passed since the new branch ‘Cwp’ was successfully displayed in the Customer interface along with the original 3 branches.



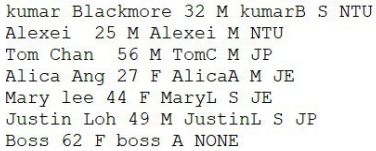
Test case 24: Test case passed since an error message was displayed when an incorrect staff username was entered.



Test case 25: Logged in as admin, changed her default password and logged in again. Test case passed since she was able to log in successfully with the new password.

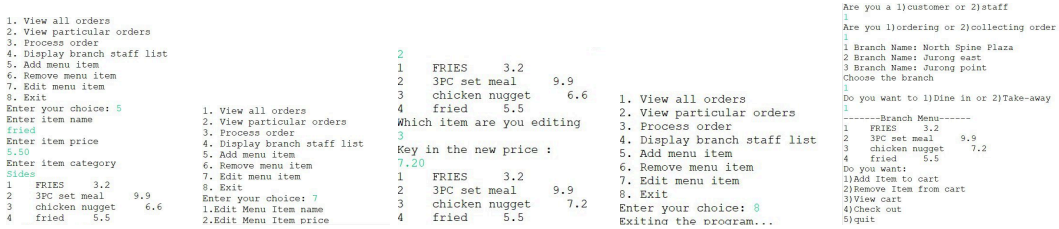


Test case 26: Test case passed since it correctly prints out the information of all staff.



kumar Bla	kumarB	S	M	32 NTU
Alexei	Alexei	M	M	25 NTU
Tom Chan	TomC	M	M	56 JP
Alica Ang	AlicaA	M	F	27 JE
Mary lee	MaryL	S	F	44 JE
Justin Loh	JustinL	S	M	49 JP
Boss	boss	A	F	62 NONE

Test case 27: Test case passed since new menu item ‘fried’ was added, price of chicken nugget was changed from 6.60 to 7.20 and these changes are still visible in subsequent sessions.



REFLECTION

In the early stages of developing the Fast Food Ordering and Management System (FOMS), we encountered several challenges in distilling complex, real-world processes into a clear and concise UML class diagram. The task of breaking down the system into more specific and well-defined classes - categorised as concrete, abstract or interface - also proved to be tedious. Additionally, we struggled with assigning the most suitable attributes and methods for each class which was vital as we wanted our system to achieve loose coupling and high cohesion.

Initially, we did a rough outline of the class diagram before diving into coding. However, it soon proved to be a mistake as we are not sure of the relationships between each class since we did not have a clear blueprint to develop the system. Furthermore, as we separated the classes and did the code individually, we found it difficult to ensure that each class not only performed its function in isolation but also integrated harmoniously with others to form a fully functional system. Recognising our mistake, we adopted a more iterative approach to design by continuously refining it after various discussions. Through debates and communications between team members, we were able to correct the mistakes and enhance our understanding of the relationship between the various classes. We also learnt how to apply the SOLID principles for the design of our system.

Although the system has been continuously refined as we work on it, there is still room for improvements. We wanted to make sure that the system worked before we improved on its efficiency and design; however as we were tight on time, the system exhibits higher coupling than ideal and each class was closely linked to other classes. Moreover, we did not manage to test out more test cases on top of what has been provided to us hence we missed the opportunity of correcting potential errors in our system. Lastly, we could have used more interfaces and abstract class instead of concrete class.

In conclusion, throughout the course of this project, the practical application of Object-Oriented Programming (OOP) concepts have deepened our understanding and we learnt how to adapt the knowledge garnered from SC2002 to tackle this complex project.