NANYANG TECHNOLOGICAL UNIVERSITY

SC2002 Object Oriented Design & Program Assignment

<u>Declaration of Original Work for CE/CZ2002 Assignment</u>

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
Liaw Rui Xian	SC2002	SDDA	[illegible] 25/4/2024
Meagan Eng Pei Ying	SC2002	SDDA	[illegible] 25/4/2024
Leow Yi Shian	SC2002	SDDA	Shian 25/4/2024
Siew Ting Hooi	SC2002	SDDA	[illegible] 25/4/2024

Content

1. Introduction **3**

2. Approach **3**

3. Design Principle **3**

3.1. Single responsibility Principle (SRP) 3

3.4. Interface Segregation (ISP) 4

3.5. Dependency Inversion Principle 4

4. OODP Concept Implementation **4**

4.1. Abstraction 4

4.2. Encapsulation 5

4.3. Inheritance 5

4.4. Dependency 5

5. Design Overview & Assumption **5**

5.1. FOMS Design Concept: 5

5.2. UML Colour Assumption: 6

6. UML Class Diagram **6**

7. Code Test Result and Analysis **7**

8. Reflection **11**

8.3. What are the weaknesses of our FOMS? 11

8.6. How can we do better in future? 12

9. GitHub Link **12**

1. Introduction

In today's fast-paced society, efficiency and ease of use are critical, especially in the busy atmosphere of a fast food restaurant. In order to adapt to the changing needs of customers and optimise operations, it is critical to create a fast food ordering and management system (FOMS). It was envisioned that FOMS would modernise the traditional fast food process using object-oriented (OO) principles and Java programming. Its main goal is to create a smooth interface that facilitates customers to place orders, personalise orders, complete transactions, and for employees to efficiently process orders and menu items. Therefore, this project will utilise the Java language and related Object Oriented Design and Programming (OODP) principles and tools to develop FOMS.

2. Approach

Our approach involves leveraging resources from GitHub to gather references and insights, allowing us to build upon existing knowledge and avoid principle mistakes. We will enhance the initial design by refining and iterating on the class diagram using Visual Paradigm, integrating lessons learnet. To facilitate seamless collaboration, we have opted for Replit as our online collaborative Java coding environment, enabling real-time teamwork and code sharing. Additionally, ChatGPT plays a role in facilitating progress by providing guidance. This comprehensive approach ensures efficient utilisation of resources, effective collaboration, and continuous improvement of our Fast Food Ordering and Management System (FOMS) project.

3. Design Principle

We will discuss about 5 principles we took into consideration when designing FOMS.

3.1. Single responsibility Principle (SRP)

SRP asserts that there is only one reason for a class to change. Essentially, it suggests that each class has only one responsibility or purpose in the system. By following the SRP principle, classes such as Payment only focus on payment related functions such as addPaymentMethod and removePaymentMethod. Classes such as Order and MenuItem also only focus on setting and editing its own attributes.

3.2. Open-Closed Principle (OCP)

OCP emphasises that software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. By following the OCP Principle, in our code, the PaymentMethod class follows this principle by allowing any new Payment method to inherit the class and extend it without needing to change any code. In the Staff class as well, we can add any new instance of a class without needing to change any part of the code that is related to it.

3.3. Liskov Substitution Principle (LSP)

Objects of a superclass should be replaceable with objects of its subclasses without affecting the correctness of the program. Between the PaymentMethod and CreditCard class, the CreditCard class inherits the methods and attributes but at the same time also has its own attributes and methods that are more than the PaymentMethod class.

3.4. Interface Segregation (ISP)

ISP holds that customers should not be forced to rely on interfaces they do not use. The ISP encourages the creation of smaller, more specific interfaces to meet the needs of individual customers rather than implementing large, monolithic interfaces. This principle promotes loose coupling and high cohesion, allowing for greater flexibility and easier maintenance of the codebase because clients will not rely on the functionality they do not need.

3.5. Dependency Inversion Principle

DIP asserts that high-level modules do not depend on low-level modules, but that both depend on abstract modules. By adhering to the DIP principle, developers can achieve decoupling between components, resulting in a more modular, flexible, and easy-to-maintain system. This principle also helps simplify testing and promotes the reuse of components in different environments, resulting in a more robust and scalable software architecture.

4. OODP Concept Implementation

4.1. Abstraction

Abstraction is the process of hiding the complex implementation details and showing only the essential features of the object. In our FOMS, we used abstraction in all our user-viewing classes such as the AdminInterface ManagerInterface, StaffInterface and CustomerInterface. By doing this, we used abstraction to only show the information needed.

4.2. Encapsulation

Encapsulation refers to the bundling of data (attributes) and methods (functions) to hide the internal state of an object and only exposing the necessary functionalities to the outside world. In classes such as Staff, Branch, Order and MenuItem, we only allowed private access to our attributes in the class. If the data of the attributes are needed, we use getters and setters to access them.

4.3. Inheritance

Inheritance is a mechanism in which a new class (subclass) is derived from an existing class (superclass), inheriting its attributes and methods. In our FOMS for example, the Admin class inherits the Staff class and the CreditCard, Cash and OnlinePayment class inherits the PaymentMethod class. This promotes code reusability and establishes a hierarchical relationship between classes.

4.4. Dependency

Dependency represents a relationship where a change in one class (the dependent class) may affect another class (the independent class). Between our OrderList and Payment class, a change in the OrderList will affect the total amount in the Payment class. The OrderList class provides service to the Payment class.

5. Design Overview & Assumption

5.1. FOMS Design Concept:

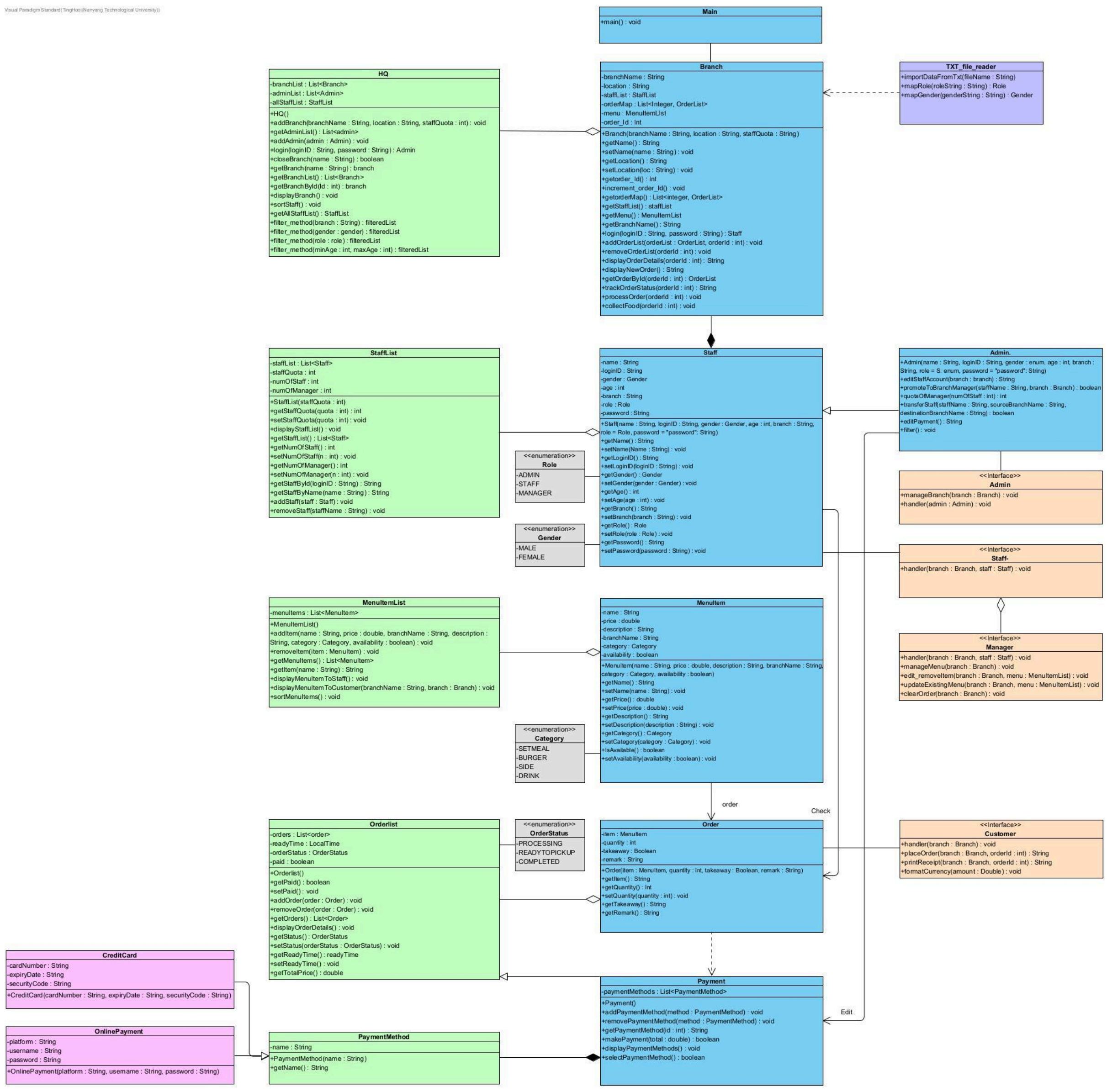
To begin with, our FOMS design centres around distinct branches, each encompassing various elements like staff, menus, and orders. We then categorise users into four main populations: customers, staff, manager, and admin. Each population has its dedicated interface, tailored to their specific roles and functionalities. Within our system, we define several categorical classes, such as branch, staff, menu items, orders, and payment systems. Most of these classes maintain their own lists, serving as repositories for relevant data and operations. Finally, we establish relationships between these classes, employing concepts like inheritance, aggregation, dependence, and association to interconnect them effectively. With this groundwork laid out, we proceed to develop the corresponding class code.

To achieve our objectives, we utilised specific tools to streamline our workflow. We employed Visual Paradigm for creating detailed class diagrams, aiding in visualising the structure and relationships within our system. Additionally, we opted for Replit as our collaborative online platform for coding, enabling seamless collaboration and real-time editing among team members regardless of their geographical locations.

<u>5.2. UML Colour Assumption:</u>

Blue - Categorical Class, Lilac - File Reader, Green - List, Orange - Interface
Grey - Enumeration, Pink - Object

6. UML Class Diagram

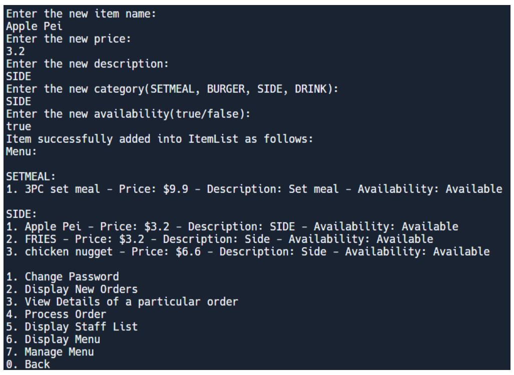

Visual Paradigm Standard(Tinghao(Nanyang Technological University))
Main
+main() : void
Branch
-branchName : String
-location : String
-staffList : StaffList
-orderMap : List<Integer, OrderList>
-menu : MenuItemLIst
-order_Id : Int
+Branch(branchName : String, location : String, staffQuota : String)
+getName() : String
+setName(name : String) : void
+getLocation() : String
+setLocation(loc : String) : void
+getorder_Id() : Int
+increment_order_Id() : void
+getorderMap() : List<integer, OrderList>
+getStaffList() : staffList
+getMenu() : MenuItemList
+getBranchName() : String
+login(loginID : String, password : String) : Staff
+addOrderList(orderList : OrderList, orderId : int) : void
+removeOrderList(orderId : int) : void
+displayOrderDetails(orderId : int) : String
+displayNewOrder() : String
+getOrderById(orderId : int) : OrderList
+trackOrderStatus(orderId : int) : String
+processOrder(orderId : int) : void
+collectFood(orderId : int) : void
TXT_file_reader
+importDataFromTxt(fileName : String)
+mapRole(roleString : String) : Role
+mapGender(genderString : String) : Gender
HQ
-branchList : List<Branch>
-adminList : List<Admin>
-allStaffList : StaffList
+HQ()
+addBranch(branchName : String, location : String, staffQuota : int) : void
+getAdminList() : List<admin>
+addAdmin(admin : Admin) : void
+login(loginID : String, password : String) : Admin
+closeBranch(name : String) : boolean
+getBranch(name : String) : branch
+getBranchList() : List<Branch>
+getBranchById(Id : int) : branch
+displayBranch() : void
+sortStaff() : void
+getAllStaffList() : StaffList
+filter_method(branch : String) : filteredList
+filter_method(gender : gender) : filteredList
+filter_method(role : role) : filteredList
+filter_method(minAge : int, maxAge : int) : filteredList
StaffList
-staffList : List<Staff>
-staffQuota : int
-numOfStaff : int
-numOfManager : int
+StaffList(staffQuota : int)
+getStaffQuota(quota : int) : int
+setStaffQuota(quota : int) : void
+displayStaffList() : void
+getStaffList() : List<Staff>
+getNumOfStaff() : int
+setNumOfStaff(n : int) : void
+getNumOfManager() : int
+setNumOfManager(n : int) : void
+getStaffById(loginID : String) : String
+getStaffByName(name : String) : String
+addStaff(staff : Staff) : void
+removeStaff(staffName : String) : void
<<enumeration>>
Role
-ADMIN
-STAFF
-MANAGER
<<enumeration>>
Gender
-MALE
-FEMALE
Staff
-name : String
-loginID : String
-gender : Gender
-age : int
-branch : String
-role : Role
-password : String
+Staff(name : String, loginID : String, gender : Gender, age : int, branch : String, role = Role, password = "password": String)
+getName() : String
+setName(Name : String) : void
+getLoginID() : String
+setLoginID(loginID : String) : void
+getGender() : Gender
+setGender(gender : Gender) : void
+getAge() : int
+setAge(age : int) : void
+getBranch() : String
+setBranch(branch : String) : void
+getRole() : Role
+setRole(role : Role) : void
+getPassword() : String
+setPassword(password : String) : void
Admin.
+Admin(name : String, loginID : String, gender : enum, age : int, branch : String, role = S: enum, password = "password": String)
+editStaffAccount(branch : branch) : String
+promoteToBranchManager(staffName : String, branch : Branch) : boolean
+quotaOfManager(numOfStaff : int) : int
+transferStaff(staffName : String, sourceBranchName : String, destinationBranchName : String) : boolean
+editPayment() : String
+filter() : void
<<Interface>>
Admin
+manageBranch(branch : Branch) : void
+handler(admin : Admin) : void
<<Interface>>
Staff-
+handler(branch : Branch, staff : Staff) : void
<<Interface>>
Manager
+handler(branch : Branch, staff : Staff) : void
+manageMenu(branch : Branch) : void
+edit_removeItem(branch : Branch, menu : MenuItemList) : void
+updateExistingMenu(branch : Branch, menu : MenuItemList) : void
+clearOrder(branch : Branch) : void
MenuItemList
-menuItems : List<MenuItem>
+MenuItemList()
+addItem(name : String, price : double, branchName : String, description : String, category : Category, availability : boolean) : void
+removeItem(item : MenuItem) : void
+getMenuItems() : List<MenuItem>
+getItem(name : String) : String
+displayMenuItemToStaff() : void
+displayMenuItemToCustomer(branchName : String, branch : Branch) : void
+sortMenuItems() : void
MenuItem
-name : String
-price : double
-description : String
-branchName : String
-category : Category
-availability : boolean
+MenuItem(name : String, price : double, description : String, branchName : String, category : Category, availability : boolean)
+getName() : String
+setName(name : String) : void
+getPrice() : double
+setPrice(price : double) : void
+getDescription() : String
+setDescription(description : String) : void
+getCategory() : Category
+setCategory(category : Category) : void
+IsAvailable() : boolean
+setAvailability(availability : boolean) : void
<<enumeration>>
Category
-SETMEAL
-BURGER
-SIDE
-DRINK
order
Check
Orderlist
-orders : List<order>
-readyTime : LocalTime
-orderStatus : OrderStatus
-paid : boolean
+Orderlist()
+getPaid() : boolean
+setPaid() : void
+addOrder(order : Order) : void
+removeOrder(order : Order) : void
+getOrders() : List<Order>
+displayOrderDetails() : void
+getStatus() : OrderStatus
+setStatus(orderStatus : OrderStatus) : void
+getReadyTime() : readyTime
+setReadyTime() : void
+getTotalPrice() : double
<<enumeration>>
OrderStatus
-PROCESSING
-READYTOPICKUP
-COMPLETED
Order
-item : MenuItem
-quantity : int
-takeaway : Boolean
-remark : String
+Order(item : MenuItem, quantity : int, takeaway : Boolean, remark : String)
+getItem() : String
+getQuantity() : Int
+setQuantity(quantity : int) : void
+getTakeaway() : String
+getRemark() : String
<<Interface>>
Customer
+handler(branch : Branch) : void
+placeOrder(branch : Branch, orderId : int) : String
+printReceipt(branch : Branch, orderId : int) : String
+formatCurrency(amount : Double) : void
CreditCard
-cardNumber : String
-expiryDate : String
-securityCode : String
+CreditCard(cardNumber : String, expiryDate : String, securityCode : String)
OnlinePayment
-platform : String
-username : String
-password : String
+OnlinePayment(platform : String, username : String, password : String)
PaymentMethod
-name : String
+PaymentMethod(name : String)
+getName() : String
Payment
-paymentMethods : List<PaymentMethod>
+Payment()
+addPaymentMethod(method : PaymentMethod) : void
+removePaymentMethod(method : PaymentMethod) : void
+getPaymentMethod(id : int) : String
+makePayment(total : double) : boolean
+displayPaymentMethods() : void
+selectPaymentMethod() : boolean
Edit

7. Code Test Result and Analysis

We have edited the code after the presentation such that we can now pass all the test cases.

Test Cases

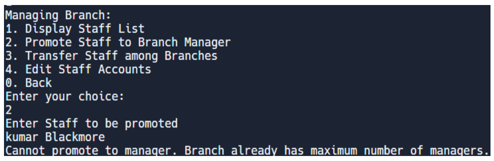
1: Manager - Add new menu item



```
Enter the new item name:
Apple Pei
Enter the new price:
3.2
Enter the new description:
SIDE
Enter the new category(SETMEAL, BURGER, SIDE, DRINK):
SIDE
Enter the new availability(true/false):
true
Item successfully added into ItemList as follows:
Menu:

SETMEAL:
1. 3PC set meal - Price: $9.9 - Description: Set meal - Availability: Available

SIDE:
1. Apple Pei - Price: $3.2 - Description: SIDE - Availability: Available
2. FRIES - Price: $3.2 - Description: Side - Availability: Available
3. chicken nugget - Price: $6.6 - Description: Side - Availability: Available

1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
5. Display Staff List
6. Display Menu
7. Manage Menu
0. Back
```

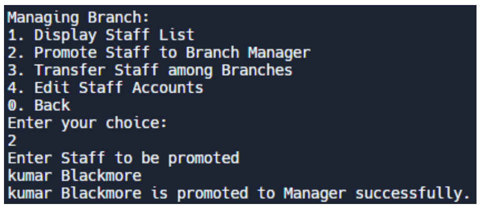
15: Admin - Assign Managers to branches according to quota constraint

When (numOfStaffs >= 1 && numOfStaffs <= 4), only 1 manager is allowed



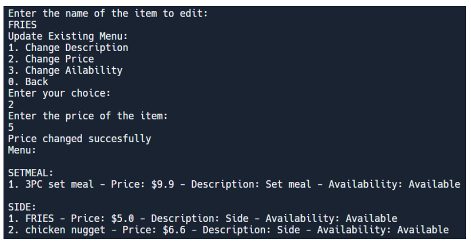
```
Managing Branch:
1. Display Staff List
2. Promote Staff to Branch Manager
3. Transfer Staff among Branches
4. Edit Staff Accounts
0. Back
Enter your choice:
2
Enter Staff to be promoted
kumar Blackmore
Cannot promote to manager. Branch already has maximum number of managers.
```

When (numOfStaffs >= 5 && numOfStaffs <= 8), can assign 2 managers

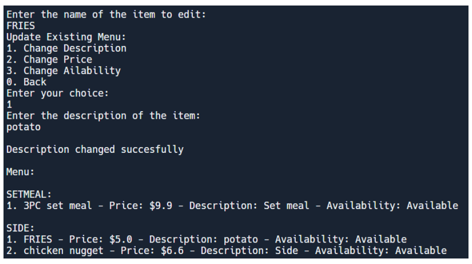


```
Managing Branch:
1. Display Staff List
2. Promote Staff to Branch Manager
3. Transfer Staff among Branches
4. Edit Staff Accounts
0. Back
Enter your choice:
2
Enter Staff to be promoted
kumar Blackmore
kumar Blackmore is promoted to Manager successfully.
```

2: Manager - Update price and description of menu item

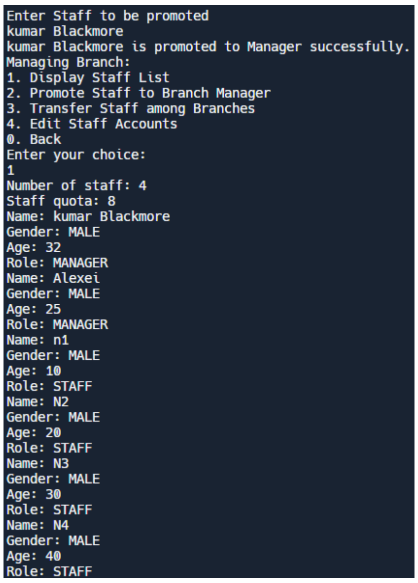


```
Enter the name of the item to edit:
FRIES
Update Existing Menu:
1. Change Description
2. Change Price
3. Change Ailability
0. Back
Enter your choice:
2
Enter the price of the item:
5
Price changed succesfully
Menu:

SETMEAL:
1. 3PC set meal - Price: $9.9 - Description: Set meal - Availability: Available

SIDE:
1. FRIES - Price: $5.0 - Description: Side - Availability: Available
2. chicken nugget - Price: $6.6 - Description: Side - Availability: Available
```



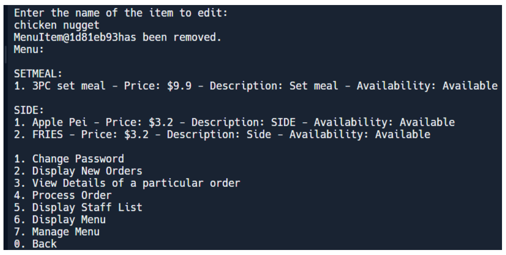
```
Enter the name of the item to edit:
FRIES
Update Existing Menu:
1. Change Description
2. Change Price
3. Change Ailability
0. Back
Enter your choice:
1
Enter the description of the item:
potato

Description changed succesfully

Menu:

SETMEAL:
1. 3PC set meal - Price: $9.9 - Description: Set meal - Availability: Available

SIDE:
1. FRIES - Price: $5.0 - Description: potato - Availability: Available
2. chicken nugget - Price: $6.6 - Description: Side - Availability: Available
```

16: Admin - Promote Staff to Manager



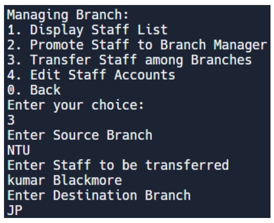
```
Enter Staff to be promoted
kumar Blackmore
kumar Blackmore is promoted to Manager successfully.
Managing Branch:
1. Display Staff List
2. Promote Staff to Branch Manager
3. Transfer Staff among Branches
4. Edit Staff Accounts
0. Back
Enter your choice:
1
Number of staff: 4
Staff quota: 8
Name: kumar Blackmore
Gender: MALE
Age: 32
Role: MANAGER
Name: Alexei
Gender: MALE
Age: 25
Role: MANAGER
Name: n1
Gender: MALE
Age: 10
Role: STAFF
Name: N2
Gender: MALE
Age: 20
Role: STAFF
Name: N3
Gender: MALE
Age: 30
Role: STAFF
Name: N4
Gender: MALE
Age: 40
Role: STAFF
```

3: Manager - Remove an existing menu item



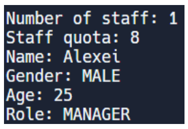
```
Enter the name of the item to edit:
chicken nugget
MenuItem@1d81eb93has been removed.
Menu:

SETMEAL:
1. 3PC set meal - Price: $9.9 - Description: Set meal - Availability: Available

SIDE:
1. Apple Pei - Price: $3.2 - Description: SIDE - Availability: Available
2. FRIES - Price: $3.2 - Description: Side - Availability: Available

1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
5. Display Staff List
6. Display Menu
7. Manage Menu
0. Back
```

17: Transfer Staffs among branches



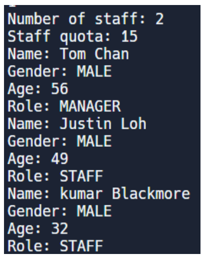
```
Managing Branch:
1. Display Staff List
2. Promote Staff to Branch Manager
3. Transfer Staff among Branches
4. Edit Staff Accounts
0. Back
Enter your choice:
3
Enter Source Branch
NTU
Enter Staff to be transferred
kumar Blackmore
Enter Destination Branch
JP
```

NTU:



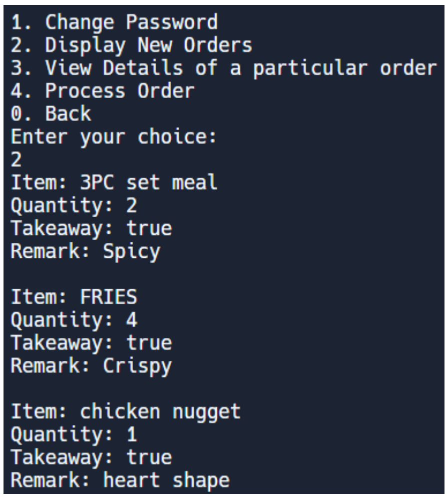
```
Number of staff: 1
Staff quota: 8
Name: Alexei
Gender: MALE
Age: 25
Role: MANAGER
```

JP:



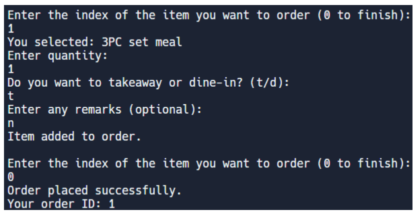
```
Number of staff: 2
Staff quota: 15
Name: Tom Chan
Gender: MALE
Age: 56
Role: MANAGER
Name: Justin Loh
Gender: MALE
Age: 49
Role: STAFF
Name: kumar Blackmore
Gender: MALE
Age: 32
Role: STAFF
```

4: Place new order with takeaway

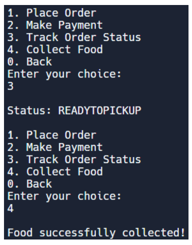


```
1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
0. Back
Enter your choice:
2
Item: 3PC set meal
Quantity: 2
Takeaway: true
Remark: Spicy

Item: FRIES
Quantity: 4
Takeaway: true
Remark: Crispy

Item: chicken nugget
Quantity: 1
Takeaway: true
Remark: heart shape
```

18: Customer - Place new order & Collect Food

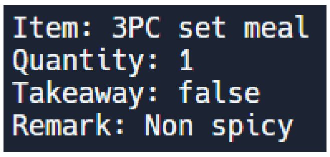


```
Enter the index of the item you want to order (0 to finish):
1
You selected: 3PC set meal
Enter quantity:
1
Do you want to takeaway or dine-in? (t/d):
t
Enter any remarks (optional):
n
Item added to order.

Enter the index of the item you want to order (0 to finish):
0
Order placed successfully.
Your order ID: 1
```



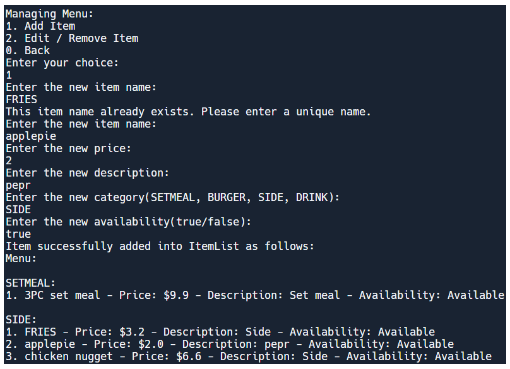
```
1. Place Order
2. Make Payment
3. Track Order Status
4. Collect Food
0. Back
Enter your choice:
3

Status: READYTOPICKUP

1. Place Order
2. Make Payment
3. Track Order Status
4. Collect Food
0. Back
Enter your choice:
4

Food successfully collected!
```

5: Place new order with dine-in



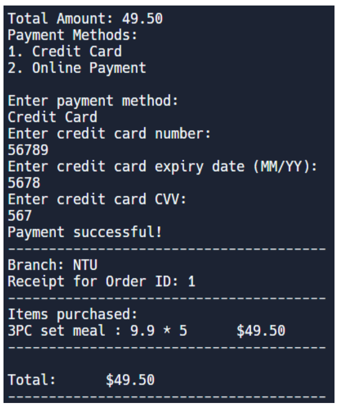
```
Item: 3PC set meal
Quantity: 1
Takeaway: false
Remark: Non spicy
```

19: Error Handling: Duplicate names



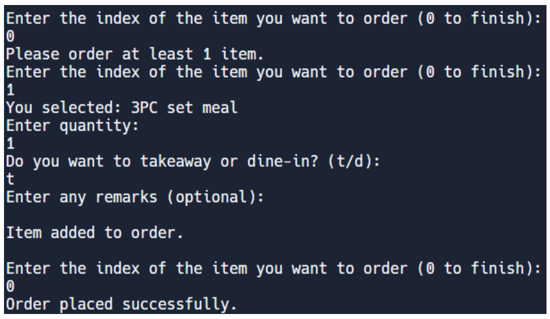
```
Managing Menu:
1. Add Item
2. Edit / Remove Item
0. Back
Enter your choice:
1
Enter the new item name:
FRIES
This item name already exists. Please enter a unique name.
Enter the new item name:
applepie
Enter the new price:
2
Enter the new description:
pepr
Enter the new category(SETMEAL, BURGER, SIDE, DRINK):
SIDE
Enter the new availability(true/false):
true
Item successfully added into ItemList as follows:
Menu:

SETMEAL:
1. 3PC set meal - Price: $9.9 - Description: Set meal - Availability: Available

SIDE:
1. FRIES - Price: $3.2 - Description: Side - Availability: Available
2. applepie - Price: $2.0 - Description: pepr - Availability: Available
3. chicken nugget - Price: $6.6 - Description: Side - Availability: Available
```

6: Credit Card Payment



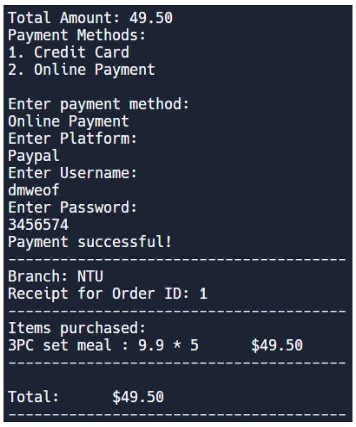
```
Total Amount: 49.50
Payment Methods:
1. Credit Card
2. Online Payment

Enter payment method:
Credit Card
Enter credit card number:
56789
Enter credit card expiry date (MM/YY):
5678
Enter credit card CVV:
567
Payment successful!
----------------------------------------
Branch: NTU
Receipt for Order ID: 1
----------------------------------------
Items purchased:
3PC set meal : 9.9 * 5      $49.50
----------------------------------------

Total:      $49.50
----------------------------------------
```

20: Error Handling: Do not process order without items



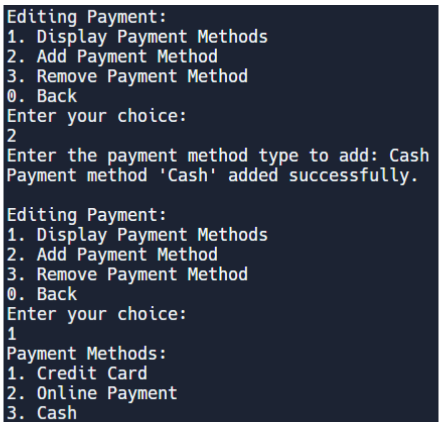
```
Enter the index of the item you want to order (0 to finish):
0
Please order at least 1 item.
Enter the index of the item you want to order (0 to finish):
1
You selected: 3PC set meal
Enter quantity:
1
Do you want to takeaway or dine-in? (t/d):
t
Enter any remarks (optional):

Item added to order.

Enter the index of the item you want to order (0 to finish):
0
Order placed successfully.
```

7: Online Payment



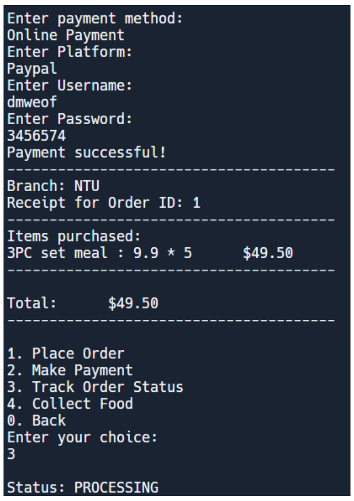
```
Total Amount: 49.50
Payment Methods:
1. Credit Card
2. Online Payment

Enter payment method:
Online Payment
Enter Platform:
Paypal
Enter Username:
dmweof
Enter Password:
3456574
Payment successful!
----------------------------------------
Branch: NTU
Receipt for Order ID: 1
----------------------------------------
Items purchased:
3PC set meal : 9.9 * 5      $49.50
----------------------------------------

Total:      $49.50
----------------------------------------
```

21: Admin - Add new payment method



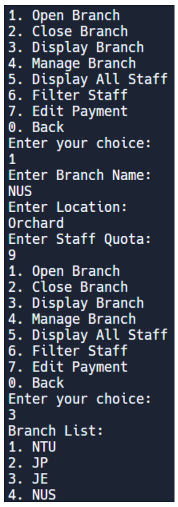
```
Editing Payment:
1. Display Payment Methods
2. Add Payment Method
3. Remove Payment Method
0. Back
Enter your choice:
2
Enter the payment method type to add: Cash
Payment method 'Cash' added successfully.

Editing Payment:
1. Display Payment Methods
2. Add Payment Method
3. Remove Payment Method
0. Back
Enter your choice:
1
Payment Methods:
1. Credit Card
2. Online Payment
3. Cash
```

8: Track order status



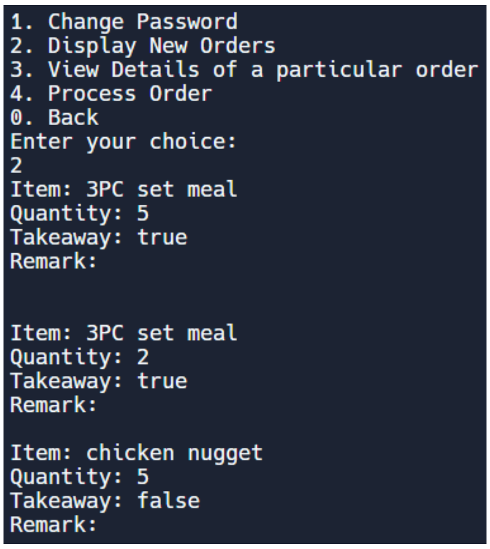
```
Enter payment method:
Online Payment
Enter Platform:
Paypal
Enter Username:
dmweof
Enter Password:
3456574
Payment successful!
----------------------------------------
Branch: NTU
Receipt for Order ID: 1
----------------------------------------
Items purchased:
3PC set meal : 9.9 * 5      $49.50
----------------------------------------

Total:      $49.50
----------------------------------------

1. Place Order
2. Make Payment
3. Track Order Status
4. Collect Food
0. Back
Enter your choice:
3

Status: PROCESSING
```

22: Admin - Open New Branch



```
1. Open Branch
2. Close Branch
3. Display Branch
4. Manage Branch
5. Display All Staff
6. Filter Staff
7. Edit Payment
0. Back
Enter your choice:
1
Enter Branch Name:
NUS
Enter Location:
Orchard
Enter Staff Quota:
9
1. Open Branch
2. Close Branch
3. Display Branch
4. Manage Branch
5. Display All Staff
6. Filter Staff
7. Edit Payment
0. Back
Enter your choice:
3
Branch List:
1. NTU
2. JP
3. JE
4. NUS
```

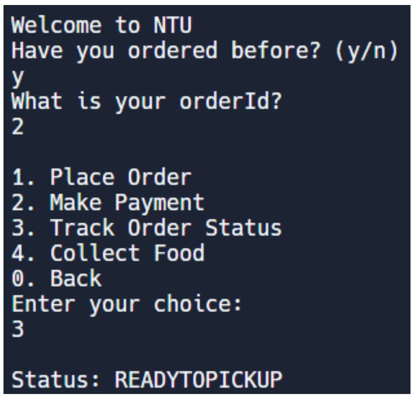
9: Staff - Display new order



```
1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
0. Back
Enter your choice:
2
Item: 3PC set meal
Quantity: 5
Takeaway: true
Remark:


Item: 3PC set meal
Quantity: 2
Takeaway: true
Remark:

Item: chicken nugget
Quantity: 5
Takeaway: false
Remark:
```

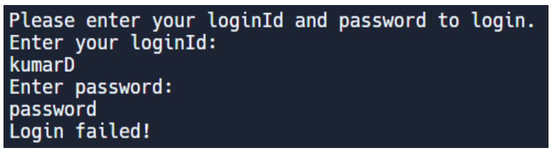
23: Automatic Order Cancellation

10: Staff - Process order, change order status to "Ready to pickup"



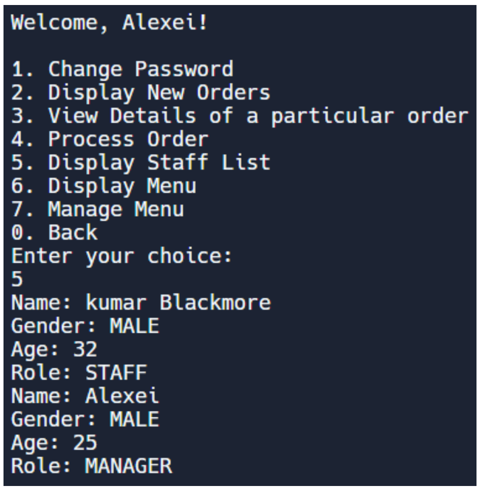
```
Welcome to NTU
Have you ordered before? (y/n)
y
What is your orderId?
2

1. Place Order
2. Make Payment
3. Track Order Status
4. Collect Food
0. Back
Enter your choice:
3

Status: READYTOPICKUP
```

24: Login with incorrect credentials



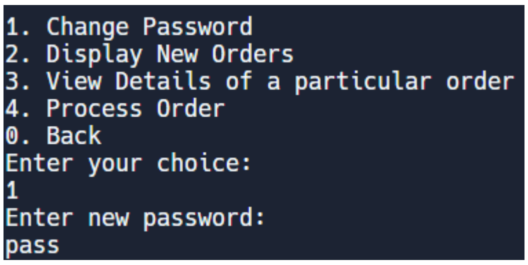
```
Please enter your loginId and password to login.
Enter your loginId:
kumarD
Enter password:
password
Login failed!
```

11: Manager - Display staff list

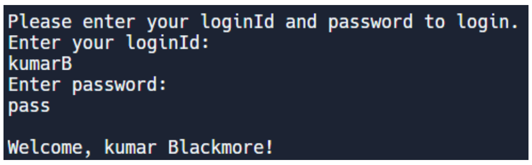


```
Welcome, Alexei!

1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
5. Display Staff List
6. Display Menu
7. Manage Menu
0. Back
Enter your choice:
5
Name: kumar Blackmore
Gender: MALE
Age: 32
Role: STAFF
Name: Alexei
Gender: MALE
Age: 25
Role: MANAGER
```

25: Change password

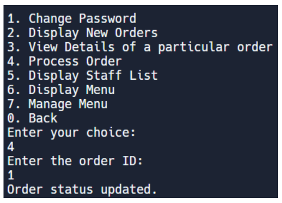


```
1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
0. Back
Enter your choice:
1
Enter new password:
pass
```

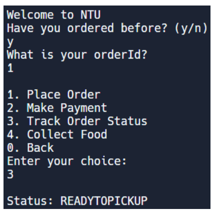


```
Please enter your loginId and password to login.
Enter your loginId:
kumarB
Enter password:
pass

Welcome, kumar Blackmore!
```

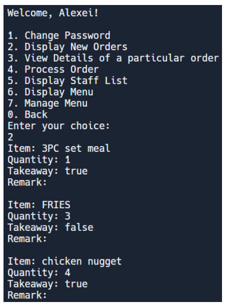
12: Manager - Process order



```
1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
5. Display Staff List
6. Display Menu
7. Manage Menu
0. Back
Enter your choice:
4
Enter the order ID:
1
Order status updated.
```

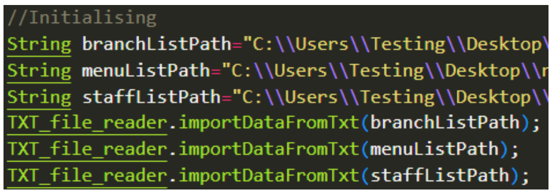


```
Welcome to NTU
Have you ordered before? (y/n)
y
What is your orderId?
1

1. Place Order
2. Make Payment
3. Track Order Status
4. Collect Food
0. Back
Enter your choice:
3

Status: READYTOPICKUP
```

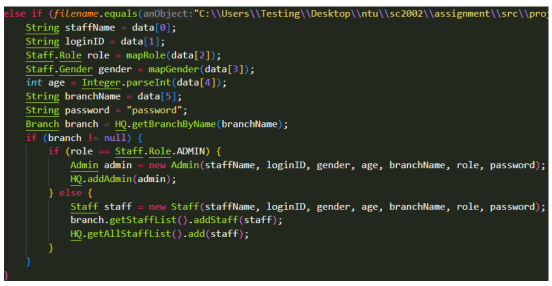


```
Welcome, Alexei!

1. Change Password
2. Display New Orders
3. View Details of a particular order
4. Process Order
5. Display Staff List
6. Display Menu
7. Manage Menu
0. Back
Enter your choice:
2
Item: 3PC set meal
Quantity: 1
Takeaway: true
Remark:

Item: FRIES
Quantity: 3
Takeaway: false
Remark:

Item: chicken nugget
Quantity: 4
Takeaway: true
Remark:
```

26: Staff List Initialization by uploaded file

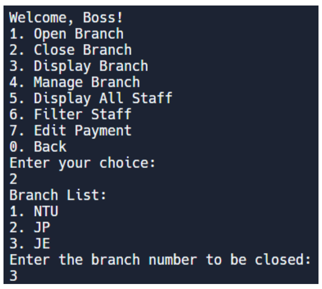


```
//Initialising
String branchListPath="C:\\Users\\Testing\\Desktop\
String menuListPath="C:\\Users\\Testing\\Desktop\\r
String staffListPath="C:\\Users\\Testing\\Desktop\\
TXT_file_reader.importDataFromTxt(branchListPath);
TXT_file_reader.importDataFromTxt(menuListPath);
TXT_file_reader.importDataFromTxt(staffListPath);
```

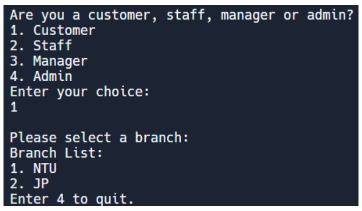


```
else if (filename.equals(anObject:"C:\\Users\\Testing\\Desktop\\ntu\\sc2002\\assignment\\src\\proj
    String staffName = data[0];
    String loginID = data[1];
    Staff.Role role = mapRole(data[2]);
    Staff.Gender gender = mapGender(data[3]);
    int age = Integer.parseInt(data[4]);
    String branchName = data[5];
    String password = "password";
    Branch branch = HQ.getBranchByName(branchName);
    if (branch != null) {
        if (role == Staff.Role.ADMIN) {
            Admin admin = new Admin(staffName, loginID, gender, age, branchName, role, password);
            HQ.addAdmin(admin);
        } else {
            Staff staff = new Staff(staffName, loginID, gender, age, branchName, role, password);
            branch.getStaffList().addStaff(staff);
            HQ.getAllStaffList().add(staff);
        }
    }
}
```

13: Admin - Close branch

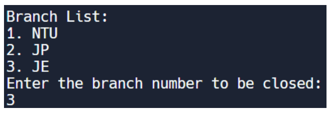


```
Welcome, Boss!
1. Open Branch
2. Close Branch
3. Display Branch
4. Manage Branch
5. Display All Staff
6. Filter Staff
7. Edit Payment
0. Back
Enter your choice:
2
Branch List:
1. NTU
2. JP
3. JE
Enter the branch number to be closed:
3
```

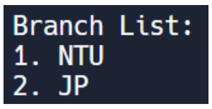


```
Are you a customer, staff, manager or admin?
1. Customer
2. Staff
3. Manager
4. Admin
Enter your choice:
1

Please select a branch:
Branch List:
1. NTU
2. JP
Enter 4 to quit.
```

27: Data Persistence

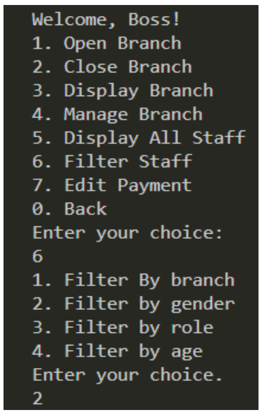


```
Branch List:
1. NTU
2. JP
3. JE
Enter the branch number to be closed:
3
```

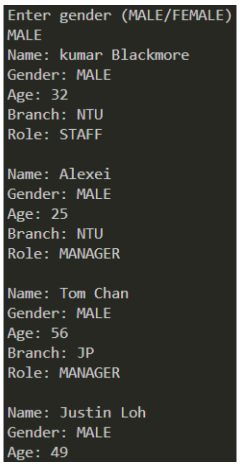


```
Branch List:
1. NTU
2. JP
```

14: Admin - Filter staff



```
Welcome, Boss!
1. Open Branch
2. Close Branch
3. Display Branch
4. Manage Branch
5. Display All Staff
6. Filter Staff
7. Edit Payment
0. Back
Enter your choice:
6
1. Filter By branch
2. Filter by gender
3. Filter by role
4. Filter by age
Enter your choice.
2
```



```
Enter gender (MALE/FEMALE)
MALE
Name: kumar Blackmore
Gender: MALE
Age: 32
Branch: NTU
Role: STAFF

Name: Alexei
Gender: MALE
Age: 25
Branch: NTU
Role: MANAGER

Name: Tom Chan
Gender: MALE
Age: 56
Branch: JP
Role: MANAGER

Name: Justin Loh
Gender: MALE
Age: 49
```

8. Reflection

8.3. What are the weaknesses of our FOMS?

Some classes stray from the single responsibility principle, leading to code repetition and decreased readability. Although we've implemented polymorphism in classes like 'filter,' broader use can enhance modularity and flexibility. Leveraging polymorphism further streamlines code, encourages reuse, and improves maintainability, aligning with our goal of best practices in object-oriented design.

8.4. Which is the most difficult problem when creating FOMS?

We found it hard to determine which methods should go under the same class at first. We also found it difficult creating many classes as suggested by the SRP and ISP because we did not know how to link the classes together to use the functions interchangeably properly. We also did not know how to properly use the access modifiers which led us to use public in most cases. Lastly, we also found job distribution difficult as every part is interconnected.

8.5. How do we overcome this problem?

To Strive for balance in class and function organisation. We define clear responsibilities for each function and group related ones to form cohesive classes. To avoid overloading classes, we aim for high cohesion instead. Besides, to maintain a balance between complexity and abstraction for manageability, we review regularly and refactor for adaptability, fostering clarity and efficiency in the system.

8.6. How can we do better in future?

Before diving into this project, we should have made it a priority to seek out existing examples for reference. This will not only save us valuable time compared to starting from scratch but also provide a solid foundation to build upon. We should also fully grasp the concept of Object-Oriented Design Principles (OODP) before starting, such as abstraction, encapsulation, inheritance, and dependency so that we can have a correct direction in the beginning stages of planning.

9. GitHub Link

https://github.com/LeowYiShian/SC2002-Assignment.git