

**School of Computer Science and Engineering**

**AY 23/24, SC2002: Object Oriented Programming**

**Seminar Class: SDDA**

**Group: 1**

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
Justin Loh Teng Hao U2220126L	SC2002	SDDA G1	   24-04-24
Lim Kiat Sen, Jaron U2222010K	SC2002	SDDA G1	   24-04-24
Ian Boo Jing Yong U2222374H	SC2002	SDDA G1	   24-04-24
Andy Chan Yan Meng U2221216B	SC2002	SDDA G1	   24-04-24
Koh Jia Hui Rachel U2222747H	SC2002	SDDA G1	   24-04-24

**1. Design Considerations**

**1.1 Design Strategies**

We approached the design of the Fast-food Ordering Management System (FOMS) with the intention to achieve a consistent high level of cohesion, and loose coupling. As such, we adopted a three-layer architecture, with a presentation layer, business logic layer and database layer. Our presentation layer comprises boundary objects that handle user inputs and outputs, effectively acting as our user interface, communicating with the business logic layer to retrieve data and perform actions based on user input. The

business logic layer comprises control classes which contain our application's logic, rules and workflows for each use case, encapsulating the core functionality. This middle layer communicates with the presentation layer for receiving user input and providing output, while communicating with the data access layer for retrieving and persisting data. Lastly, our data access layer comprises data access objects, responsible for interacting with our entity classes through creating, reading, updating and deleting of content in our serialised data to achieve data persistence.

## **1.2 Key Strategies enabling Extensibility and Maintainability**

Throughout our process in developing the system, we focused on ensuring modularity with the system. We were able to break our system down into smaller, manageable and interchangeable use cases for development, while communicating possible shared functions for inheritance. This allowed us to develop, test and debug the system independently, while also ensuring that one module is less likely to impact other modules, thus enhancing maintainability.

At the same time, we placed much emphasis on utilising polymorphism for functionalities we wished to implement. This promotes a reusable architecture that anticipates change and promotes flexibility, its implementation aligned with our focus on modularity and makes it easier to modify and extend the system without altering its core structure.

Lastly, we programmed interfaces by first defining the capabilities of a class without tying it down to a specific implementation. This thus enabled us to be clear on the methods of classes, while enabling continuous refactoring of our code, promoting the maintainability of our system.

## **1.3 Assumptions Made**

We have implemented a 2 hour limit for customers to collect their order, which is refreshed whenever the order DAO is instantiated. This time limit is based on the recommended amount of time food can be left out before becoming hazardous to human health, ensuring that the food collected by customers is fresh and safe. We also assumed that an order is only prepared by a kitchen after it has been paid for by customers.

## **2. Applied Design Principles**

### **2.1 Single Responsibility Principle: "A module should be responsible to one, and only one, actor."**

In the presentation layer, boundary objects have the sole purpose of interfacing with the user, collecting inputs and presenting outputs, without involving any business logic or handling of data. These boundary objects are further segregated into the staff or customer function it is associated with, thus forming its

respective forms that are generated from the user's choice. In doing so, each class has only one reason to change, and each class is tasked with a specific functionality that does not overlap with responsibilities of other classes. This thus minimises the effect of changes by increasing the encapsulation and modularity of our system, increasing our ease in understanding, testing and refining.

## **2.2 Open/Closed Principle (OCP):** “*Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification*”

OCP is applied throughout our system, but is most prominent in our customer payment use case OCP is applied by creating a strategy interface, “PaymentService”, whose methods “simulatePayment()” and “authenticatePayment()” is implemented by the respective payment services, such as “OnlinePaymentService” and “CreditDebitPaymentService”. As such, should one wish to add a new payment method, for instance by cash, new behaviours can now implement the PaymentService interface, allowing for the addition of new behaviours with minimal changes to existing code, demonstrating both inheritance and polymorphism, as the payment method called is bound dynamically.

## **2.3 Liskov Substitution Principle (LSP):** “*Let $\phi(x)$ be a property provable about objects $x$ of type $T$ . Then $\phi(y)$ should be true for objects $y$ of type $S$ where $S$ is a subtype of $T$ .*”

We apply LSP throughout our system for any interface used. For instance, we ensure that objects created by the factory are always instances of a type that is suitable for the base interface “Form”, which the factory promises to return. This ensures that each Form can be treated the same as any other form that conforms to the interface, fulfilling LSP as it guarantees that subclasses are proper substitutes for their Form interface. Moreover, adhering to LSP guarantees the extensibility and modularity of strategies and facades mentioned in 2.2, since its modular nature is contingent on a class being treated the same as any other class that inherits the same interface, creating its “swappable” ability.

## **2.4 Interface Segregation Principle (ISP):** “*No code should be forced to depend on methods it does not use.*”

When designing our system, we abided by the ISP to shape our interfaces to be lean and client-specific, preventing the “fat interface” problem where an interface has more methods than its clients need or use. For instance, despite StaffApp and the respective views having similar methods of calling a boundary class, we separated them to create two interfaces: StaffView and AppDisplay. This ensures that our app and views only implement their necessary methods, without being burdened by irrelevant functionalities that do not apply to its specific processing needs.

**2.5 Dependency Inversion Principle (DIP):** “*A. High-level modules should not import anything from low-level modules. Both should depend on abstractions (e.g., interfaces). B. Abstractions should not depend on details. Details (concrete implementations) should depend on abstractions.*”

The DIP is seamlessly integrated into our system through the use of a facade pattern in “Forms”. By interacting with the “Form” interface, high-level functionalities in “CustomerApp” are unaffected by changes in the specifics of lower-level forms such as “CustomerOrderingForm” for customer ordering functions or “CustomerPostPaymentForm”, for customer payment functions. This decouples the high-level functionalities from the specifics of its underlying implementations, thus adhering to DIP by solely depending on its “Form” abstraction. This thus enables us to add more customer functionalities with the least effort needed in the future, extending the functionalities in our system.

### **3. Current Improvements**

#### **3.1 Concurrent Updates to Database**

For improvements of our system, we considered the situation when there are concurrent customers ordering and staff members preparing orders. Initially, as the system was built for one user’s access, updates to the database only occur when a use case has finished. Thus, we refactored the controllers to call for its respective data access object to save the data upon any confirmed change, directly affecting the serialised object and maintaining data uniformity amongst users. Additionally, we implemented an automatic refresh of the data contained within the data access objects that occurs whenever any option is selected, creating ease in the checking of updates.

#### **3.2 User Experience**

We have improved user experience with our system by displaying all required information to the user whenever a choice is necessary. For instance, when a customer is prompted to enter the branch they are in, we will first display all possible branches. As such, users gain a better experience with our system’s interface, thus increasing its effectiveness.

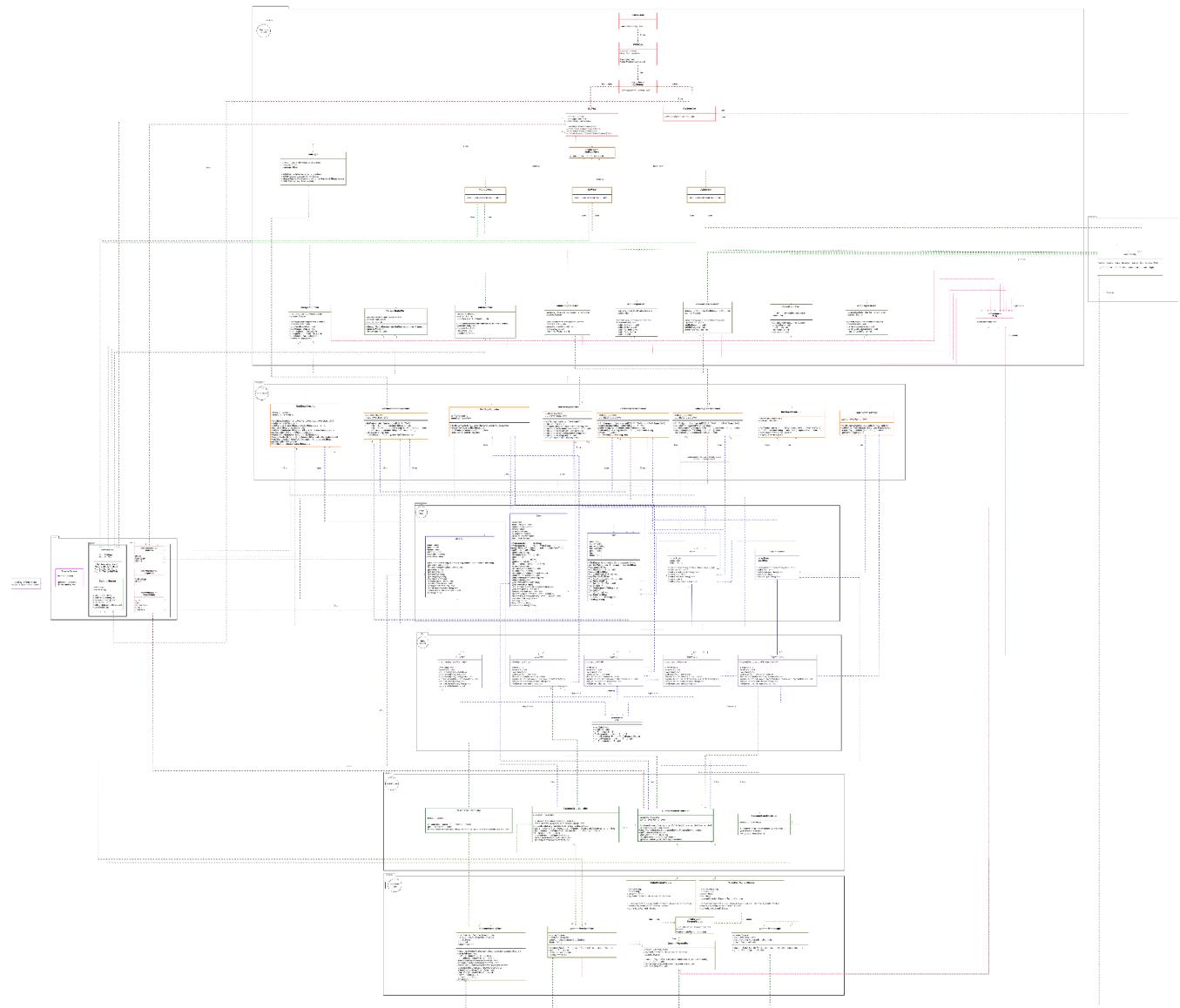
### **4. Future Improvements**

#### **4.1 Change in Data Format**

Due to specified constraints, the storage of data is in the form of a Microsoft Excel Open XML Format Spreadsheet (XLSX) file. Since XLSX is used by Microsoft Excel, a common program used for administrative work, saving it in such a format enables better integration with the Fast-food restaurant’s

administrative processes. However, our current implementation involves rewriting the entire file for each update, which can be computationally costly, and creates the possibility of data inconsistencies should two independent processes write to the same file. In the future, usage of a database application would be preferred to handle the possibility of concurrent updates to the database at the same time.

## 5. Detailed UML Class Diagram (Please refer to Image attached for clearer View)



## **6. Testing (Please watch the Screen Recordings for more detailed Live-Demos + Error Handlings)**

### **6.1 Manager's action: Menu Management**

**Test Case 1:** Add new menu item with unique name, price, description, category and verify the menu item is added successfully.

Enter the branch:						
Enter the name of the new menu item:	NTU					
chicken wings						
Enter the branch of the new menu item:	NTU					
Enter the price of the new menu item:	10.00					
Enter the category of the new menu item:	MENU ITEMS					
	Name	Category	Price (\$)	Branch	Description	Availability
	FRIES	sides	3.20	NTU	Delicious French Fries!	true
	BBQ set meal	sides meal	5.00	NTU	3 pieces of delicious chicken!	true
	chicken wings	sides	6.00	NTU	maggies	true
	chicken wings	sides	5.00	NTU	yummy wings	true
Press enter to return to the Manager Menu Actions...						
Press enter to return to the Manager Menu Actions...						

**Test Case 2:** Update the price and description of an existing menu item, and verify that the changes are reflected in the menu.

```

Enter the name of the menu item to edit: chicken nugget
Enter the branch of the menu item: NTU
Enter the price for the new menu item: 6.66
Enter all to skip: 
Enter the category for the new menu item: sides
Enter the branch for the new menu item: NTU
Enter all to skip: 
Enter the name for the new menu item: chicken wings
Enter the branch for the new menu item: NTU
Enter the price for the new menu item: 18.00
Enter all to skip: 
Enter the category for the new menu item: sides
Enter the branch for the new menu item: NTU
Enter all to skip: 
Press enter to return to the Manager Menu Actions...

```

(Negative price handling shown in the video)

**Test Case 3:** Remove an existing menu item. Verify that the menu item is no longer available.

Enter the branch:					
NTC					
Name	Category	Price (\$)	Branch	Description	Availability
PATES	1.00	9.99	NTC	Delicious French Fries	true
SPC set meal	set meal	9.99	NTC	3 Fries of delicious chicken!	true
chicken nugget	size	6.99	NTC	nuggets	true

Press enter to return to the Manager Menu Actions...

(Unknown branch handling shown in the video)

## 6.2 Order Processing:

**Test Case 4:** Place a new order with multiple food items, customise some items, and choose takeaway option, and verify order is successfully created.

MENU ITEMS				
Name	Category	Price (\$)	Branch	Description
FRIES	side	3.20	NTU	delicious French Fries
Taco (small)	side	3.20	NTU	choice of delicious chicken!
chicken nugget	side	6.60	NTU	nuggets

Enter the name of the item to order (type "done" to finish): 3PC set meal  
 Enter the number of items: 1  
 Enter the name of the item to order (type "done" to finish): FRIES  
 Enter the item's special requests: I  
 Enter the name of the item to order (type "done" to finish): done  
 Enter the name of the item to order (type "done" to finish): done  
 Enter your choice (1 for Take Away, 2 for Dine-in): 1

---

YOUR CART			
Item Name	Price (\$)	Special Requests	
3PC set meal	9.98	I give me ketchup!	
FRIES	3.20	nil	
FRIES	3.20	nil	
Total	16.30		

Dine-in Option Selected: Takeaway

Please enter to return to the Order Menu... ↵

Enter your choice (1-3): 1

View orders in the NTU branch.  
 Please enter your order ID: 5

---

Order ID: 5

---

Order Details:  
 Order Status: NEW  
 Total Price: \$16.30  
 Dine-in: No

---

3PC set meal - \$9.90  
 FRIES - \$3.20  
 FRIES - \$3.20

---

Press enter to return to the Post Order Menu...

**Test Case 5:** Place new order with dine-in option, and verify order is created with correct preferences.

(Invalid input for dine-in/takeaway selection shown in the video)

(User friendliness - Ability to change from dine-in to takeaway shown in Supplementary Video 1, and ability to remove and add items into cart in Supplementary Video 2)

MENU ITEMS				
Name	Category	Price (\$)	Branch	Description
SPC set meal	set meal	9.98	NTU	5 Pieces of delicious French chicken nuggets
chicken nugget		6.68		nuggets

Enter the name of the item to update (type "done" to finish): chicken nugget

Enter the item's special requests: null

Enter the name of the item to order (type "done" to finish): done

Enter the name of the item to order (type "done" to finish): done

Enter your choice (1 for Take Away, 2 for Dine-in): 3

Invalid input. Please enter 1 or 2.

Enter your choice (1 for Take Away, 2 for Dine-in): 3

---

YOUR CART		
Item Name	Price (\$)	Special Requests
chicken nugget	6.68	null
Total	6.68	

Dine-in Option Selected: Dine-in

Press enter to return to the Post Order Menu...

---

Enter your choice (1-3):  
1  
View orders in the NTU branch.  
Please enter your order ID: 6

=====

Order ID: 6

=====

**Order Details:**  
**Order Status:** NEW  
**Total Price:** \$6.60  
**Dine-in:** Yes

=====

chicken nugget - \$6.60



=====

Press enter to return to the Post Order Menu...

### **6.3 Payment Integration:**

**Test Case 6:** Simulate a payment for using a credit/debit card, and verify that the payment is processed successfully.

```
|-----Simulate Payment-----| Receipt for Order ID: 6
|
Credit/Debit Card Payment Selected - Please enter your payment details
Available Credit/Debit Payment Methods:
  VISA, Enter CardProvider:
  VISA
Enter card number:
1234567890123456
Enter card number:
1234567890123456
Enter expiry date:
07/29
Enter cvc:
763
Authenticating payment...
Processing credit/debit card payment for Card Provider: VISA
Payment successful!
|
Order Details:
Order ID: 6
Order Status: PAID
Total Price: $6.66
Dine-in: No
-----
chicken nugget - $6.66
-----
Thank you for your purchase!
=====
Press enter to return to the Main Menu...
```

**Test Case 7:** Simulate a payment using an online payment platform (e.g. PayPal) and verify that the payment is processed successfully.

(Scenario where customer tries to pay for an order that is already paid for is shown in the video)

(Scenario where customer enters an invalid payment type covered in Supplementary Video 3)

Simulate Payment  
Online Payment Selected - Please enter your payment details  
Available Online Payment Methods:  
Paypal, Credit Card, Enter Online Payment Provider:  
Paypal  
Enter email: linksjaron123@gmail.com  
Enter password: asasqwe  
Processing online payment for domain: Paypal with email: linksjaron123@gmail.com  
Authentication in progress...  
Payment successful!

Receipt for Order ID: 7  
Order Details:  
Order ID: 7  
Order Status: PAID  
Total Price: \$3.20  
Dine-in: No  
  
FRIES - \$3.20  
Thank you for your purchase!  
Press enter to return to the Main Menu...

#### 6.4 Order Tracking:

**Test Case 8:** Track the status of an existing order using the order ID, and verify that the correct status is displayed.

(Invalid input for order ID that has not yet been created is shown in the video)

View orders in the NTU branch.  
Please enter your order ID: 5  
Order ID: 5  
Order Details:  
Order Status: NEW  
Total Price: \$16.30  
Dine-in: No  
  
3PC set meal - \$9.90  
FRIES - \$3.20  
FRIES - \$3.20  
Press enter to return to the Post Order Menu...

#### 6.5 Staff Actions:

**Test Case 9:** Login as a staff member and display new orders, and verify that the staff can see all the new orders in the branch he/she is working.

(Explanations in the video to show the verification for all new orders, which in our case is based on our assumption that orders that are placed and paid for by customers. These are orders that are ready to be prepared by the kitchen staff at the restaurant)

Login Portal  
Choose an option:  
1. Login  
2. Change Password  
Order ID: 7  
Order Details:  
Order Status: PAID  
Total Price: \$3.20  
Dine-in: No  
  
FRIES - \$3.20  
Press enter to return to the Staff Order Menu...

Enter role: 1/2/3  
1. Admin, 2. Staff, 3. Manager  
Enter username: kumarB  
Enter password: password

**Test Case 10:** Process a new order, updating its status to “Ready to pickup”, and verify that the order status is updated correctly.

Staff Order Actions  
Choose an options:  
1. Display all orders  
2. Display details of a particular order  
3. Process order  
4. Go to homescreen  
Enter your choice (1-4):  
3  
Order Order ID: 7  
Order Status for Order ID: 7 has been changed to READY  
Press enter to return to the Staff Order Menu...

Enter Order ID: 7  
Order ID: 7  
Order Details:  
Order Status: READY  
Total Price: \$3.20  
Dine-in: No  
  
FRIES - \$3.20  
Press enter to return to the Staff Order Menu...

#### 6.6 Manager Actions:

**Test Case 11:** Login as manager and display the staff list in the manager's branch, and verify that the staff list is correctly displayed.

(Explanations in the video to show that the staff list is correctly displayed)

Login Portal  
Choose an option:  
1. Login  
2. Change Password  
Name: kumar Blackmore  
Login ID: kumarB  
Role: STAFF  
Gender: M  
Age: 32  
Branch: NTU  
  
Staff Details:  
Name: Alexei  
Login ID: Alexei  
Role: MANAGER  
Gender: M  
Age: 25  
Branch: NTU  
  
====End of Staff List=====

1  
Enter role: 1/2/3  
1. Admin, 2. Staff, 3. Manager  
3  
Enter username: Alexei  
Enter password: password

### Test Case 12: Similar function as test case 10 - Process Order

(Invalid input for order ID that has not yet been created is shown in the video)

Enter Order ID: 5	Enter your choice (1-4): 2 Enter Order ID: 5
Order ID: 5	=====
Order Details: Order Status: PAID Total Price: \$6.00 Dine-in: No	Order Details: Order Status: READY Total Price: \$6.40 Dine-in: No
FRIES - \$3.20 FRIES - \$3.20	FRIES - \$3.20 FRIES - \$3.20
Press enter to return to the Staff Order Menu... Press enter to return to the Staff Order Menu...	

### 6.7 Admin Actions:

#### Test Case 13: Close a branch, and verify that branch does not display in Customer's Interface anymore.

Admin Branch Actions	
Choose an option: 1.Open Branch 2.Close Branch 3.View All Branches 4.Go to Homescreen	
0. NTU 1. JP 2. JE 3. JW	0. NTU 1. JP 2. JE Please enter th
Enter your choice (1-4): 1. Enter branch name: Branch closed successfully: JW Press enter to return to the Admin Branch Menu...	

#### Test Case 14: Login as an admin and display the staff list with filters (branch, role, gender, age), and verify that the staff list is correctly filtered.

(Explanations in the video to show that the filter is working correctly)

(Supplementary Video 4 shows the error handling of the wrong inputs as filters)

Staff Details: Name: kumar Blackmore Login ID: kumarB Role: STAFF Gender: M Age: 32 Branch: NTU	Staff Details: Name: Alexei Login ID: Alexei Role: MANAGER Gender: M Age: 25 Branch: NTU
Staff Details: Name: Alicia Ang Login ID: AliciaA Role: MANAGER Gender: F Age: 27 Branch: JE	Staff Details: Name: Tom Chan Login ID: TomC Role: MANAGER Gender: M Age: 56 Branch: JP
Staff Details: Name: Mary lee Login ID: MaryL Role: STAFF Gender: F Age: 44 Branch: JE	Staff Details: Name: Andy Login ID: test2 Role: STAFF Gender: M Age: 23 Branch: JE
Staff Details: Name: Naren Jaron Login ID: test3 Role: STAFF Gender: M Age: 23 Branch: JE	Staff Details: Name: Tom Chan Login ID: TomC Role: MANAGER Gender: M Age: 56 Branch: JP

#### Test Case 15: Assign managers to branches with the quota/ratio constraint, and verify that managers are assigned correctly.

(Explanations in the video on handling of checking of constraints)

Choose an option: 1.Assign Manager to Branch 2.Promote Staff to Manager 3.Transfer Staff 4.Go to Homescreen	Choose an option: 1.Assign Manager to Branch 2.Promote Staff to Manager 3.Transfer Staff 4.Go to Homescreen	Staff Details: Name: Tom Chan Login ID: TomC Role: MANAGER Gender: M Age: 56 Branch: JE
Enter your choice (1-4): 1. Enter Staff ID: Tom Enter branch to assign: JP This branch already has the required number of managers. Please enter to return to the Admin Assignment Menu...	Enter your choice (1-4): 1. Enter Staff ID: Tom Enter branch to assign: JP This branch already has the required number of managers. Please enter to return to the Admin Assignment Menu...	Successful Verification

#### Test Case 16: Promote a staff to branch Manager, and verify that the staff is promoted successfully.

Enter the branch name: JP Displaying Staff List.	Admin Assignment Actions	Enter the branch name: JP Displaying Staff List.
Staff Details: Name: Kumar Blackmore Login ID: kumarB Role: STAFF Gender: M Age: 32 Branch: JP	Choose an option: 1.Assign Manager to Branch 2.Promote Staff to Manager 3.Transfer Staff 4.Go to Homescreen	Staff Details: Name: Kumar Blackmore Login ID: kumarB Role: MANAGER Gender: M Age: 32 Branch: JP
Enter your choice (1-4): 2. Enter Staff ID: kumarB Enter old branch: NTU Enter new branch: JP Staff promoted successfully. Press enter to return to the Admin Assignment Menu...	Enter Staff ID: kumarB Enter old branch: NTU Enter new branch: JP Staff promoted successfully. Press enter to return to the Admin Assignment Menu...	

#### Test Case 17: Transfer a staff/manager among branches, and verify that the transfer is reflected in the system.

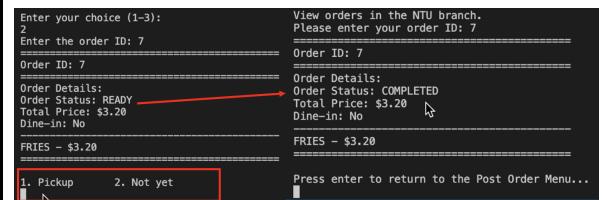
(Invalid branch entered handling is shown in the video)

Choose an option: 1.Assign Manager to Branch 2.Promote Staff to Manager 3.Transfer Staff 4.Go to Homescreen	Displaying Staff List.
Enter your choice (1-4): 3. Enter Staff ID: kumarB Enter old branch: NTU Enter new branch: JP Staff transferred successfully. Press enter to return to the Admin Assignment Menu...	Staff Details: Name: Kumar Blackmore Login ID: kumarB Role: STAFF Gender: M Age: 32 Branch: NTU
	Displaying Staff List. Staff Details: Name: Kumar Blackmore Login ID: kumarB Role: STAFF Gender: M Age: 32 Branch: JP

## 6.8 Customer Interface

**Test Case 18:** Place a new order, check the order status using the order ID, and collect the food. Verify that the order status changes from “Ready to pickup” to “completed”

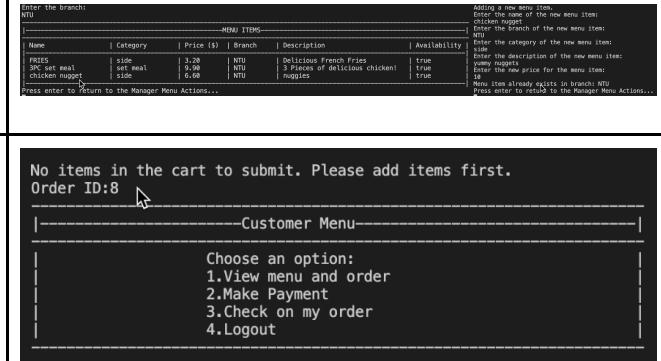
(Explanations in the video of the whole process, including error handling of customer trying to pick up order when it is not ready)



## 6.9 Error Handling:

**Test Case 19:** Attempt to add a menu item with duplicate name, and verify that an appropriate error message is displayed

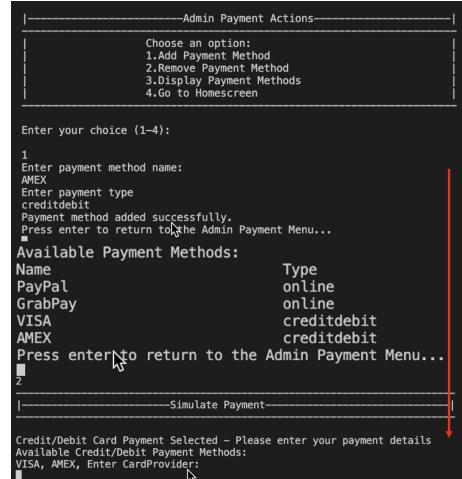
**Test Case 20:** Attempt to process an order without selecting any items, verify that an error message prompts the user to select items.



## 6.10 Extensibility:

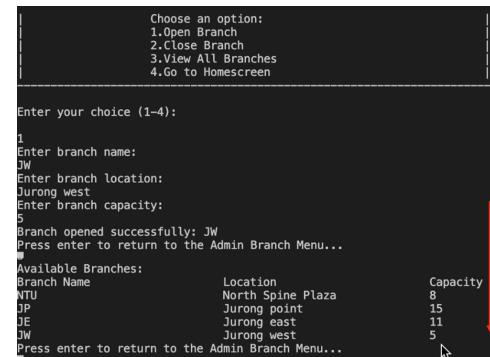
**Test Case 21:** Add a new payment method, verify that the new payment method is successfully added.

(Explanations in the video on extensibility)



**Test Case 22:** Open a new branch, verify that the new branch is added without affecting existing functionalities

(Explanations in the video on extensibility)



## 6.11 Order Cancellation:

**Test Case 23:** Place a new order and let it remain uncollected beyond the specified timeframe, and verify that the order is automatically cancelled and removed from the “Ready to pickup” list.

(Explanation found in the video)

```
Enter your choice (1-3):
1
View orders in the NTU branch.
Please enter your order ID: 5
=====
Order ID: 5
=====
Order Details:
Order Status: CANCELLED
Total Price: $3.20
Dine-in: No
=====
FRIES - $3.20
=====
Press enter to return to the Post Order Menu...
```

## 6.12 Login System:

**Test Case 24:** Attempt to log in with the incorrect credentials as a staff member, verify that an appropriate error message is displayed

(Explanation on error handling found in the video)

```
Enter role: 1/2/3
1. Admin, 2. Staff, 3. Manager
2
Enter username: KUMARB Wrong username
Enter password: - Case sensitive
password
Username not found.
Login failed. Please try again.

Enter role: 1/2/3
1. Admin, 2. Staff, 3. Manager
2
Enter username: kumarB Wrong password
Enter password: - Case sensitive
password
Username and password do not match.
Login failed. Please try again.

Enter role: 1/2/3
1. Admin, 2. Staff, 3. Manager
1
Enter username: kumarB Wrong Role
Enter password: password
User: kumarB does not belong to ADMIN role. Please try again.
```

**Test Case 25:** Login as a staff member, change the default password, and log in again with the new password, and verify that the password change functionality works as expected

(Verification of successful login can be seen in the video)

```
|-----Login Portal-----|
| Choose an option:
  1. Login
  2. Change Password |
|-----|
2
Enter username: kumarB
Enter old password: password
Enter new password: Hello!
Password changed successfully.
Press enter to return to the Login Menu...
```

## 6.13: Staff List Initialization

**Test Case 26:** Upload a staff list file during system initialization.  
Verify that the staff list is correctly initialised based on the uploaded file.

(Explanation found in the video)

The screenshot shows a terminal window with a table of staff data at the top and a log message below it. The table has columns A through G: Name, ID, Password, Role, Gender, Age, and Branch. The log message shows staff details for 'Sofia The Second' and ends with a prompt to press enter to return to the Admin Display Menu.

A	B	C	D	E	F	G
1 Name	ID	Password	Role	Gender	Age	Branch
2 kumar Blackmore	kumarB	password	S	M	32	NTU
3 Alexei	Alexei	password	M	M	25	NTU
4 Tom Chan	TomC	password	M	M	56	JP
5 Alicia Ang	AliciaA	password	M	F	27	JE
6 Mary lee	MaryL	password	S	F	44	JE
7 Justin Loh	JustinL	password	S	M	49	JP
8 Boss	boss	password	A	F	62	NA
9 Ian	test1	password	S	M	23	JE
10 Andy	test2	password	S	M	23	JE
11 Jaron	test3	password	S	M	23	JE
12 Rachel	test4	password	S	F	21	JE
13 Joshua	test5	password	S	M	30	JE
14 Arslan	test6	password	S	M	40	JE
15 Julius	test7	password	S	M	24	JE
16 Sofia The Second	test8	password	S	F	12	JE

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER AZURE
Role: STAFF
Gender: M
Age: 24
Branch: JE
=====
Staff Details:
Name: Sofia The Second
Login ID: test8
Role: STAFF
Gender: F
Age: 12
Branch: JE
=====
=====End of Staff List=====
Press enter to return to the Admin Display Menu...
```

## 6.14: Data Persistence

**Test Case 27:** Perform multiple sessions of the application updating, adding and removing menu items, and verify that changes in 1 session persist and are visible in subsequent sessions.

(Explanations found in the video)

The screenshot shows two terminal sessions. Session 1 shows a menu item table with items like FRIES, 3PC set meal, and chicken nugget. Session 2 shows a modified menu item table where the 3PC set meal item has been updated to include sides. Both sessions show a prompt to press enter to return to the Manager Menu Actions.

MENU ITEMS					
Name	Category	Price (\$)	Branch	Description	Availability
FRIES	side	3.20	NTU	Delicious French Fries!	true
3PC set meal	set meal	9.90	NTU	3 Pieces of delicious chicken!	true
chicken nugget	side	6.60	NTU	nuggies	true

```
Session 1
Enter the branch: NTU
=====
MENU ITEMS
=====
| Name | Category | Price ($) | Branch | Description | Availability |
| FRIES | side | 3.20 | NTU | Delicious French Fries! | true |
| 3PC set meal | set meal | 9.90 | NTU | 3 Pieces of delicious chicken! | true |
| chicken nugget | side | 6.60 | NTU | nuggies | true |
=====
Press enter to return to the Manager Menu Actions...
Session 2
Enter the branch: NTU
=====
MENU ITEMS
=====
| Name | Category | Price ($) | Branch | Description | Availability |
| 3PC set meal | set meal | 9.90 | NTU | 3 Pieces of delicious chicken! | true |
| chicken wings | sides | 5.00 | NTU | yummy wings | true |
=====
Press enter to return to the Manager Menu Actions...
```

## **7. Reflection**

From this assignment, we have seen the importance of design considerations with regards to SOLID principles through real application. Initially, we had built functional code without design considerations, resulting in a monolithic architecture where any change in our code triggered a ripple effect. This resulted in a lot of time spent in maintaining code and updating it. Eventually, realising that our methods were wrong, we decided to redo the project. In our second iteration, we split the system into use cases, and utilised a UML class diagram to have a visual understanding on whether we have achieved the intended loose coupling with high cohesiveness. At the same time, while developing our UML diagram, we challenged each other's suggestions as to whether they abided by the SOLID principles.

We were thus able to develop the system in parallel, maximising our time usage which was key as we were already set back due to our prior failed instance. With our newly developed system, we found that maintenance was much easier, and we were thus able to perform code refactoring frequently during the development process, while creating more functionalities such as payment methods that were easily extendable. Overall, we have realised the importance of planning and designing with SOLID principles in mind, not just for the high cohesion and loose coupling that enables the high flexibility, easy maintenance and easy extension, but also for the modularity it enables which enables parallel development, ensuring effective and efficient teamwork.