Declaration of Original Work

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
Gwendalene Ionna (U2321501C)	SC2002	FDAB	26 April 2024
Priya Rekah D/O M Ravi Kumar (U2223260H)	SC2002	FDAB	26 April 2024
Rachel Tan Min Zhi (U2320358K)	SC2002	FDAB	26 April 2024
Rafiabdul Subuhan Afreen (U2320269B)	SC2002	FDAB	26 April 2024
Sanjana Shanmugasundaram (U2321261L)	SC2002	FDAB	26 April 2024

1. Executive Summary

The Food Ordering and Management System (FOMS) is an intuitive interface to streamline the ordering process for customers and the management of business operations and transactions. With error handling and input validation, it enhances system reliability and the user's experience.

2. Design Considerations

We organised the packages by their features to hide implementation details with package visibility. Within the packages, we used the entity-control-boundary architecture to partition the responsibilities further and follow the principle of separation of concerns. As such, this structure ensures high cohesion and reduces coupling between unrelated classes.

2.1 Use of Object-Oriented Principles

2.1.1 Abstraction

To achieve data and process abstraction, we privatised all attributes and used protected methods whenever possible. Additionally, we utilised interfaces and abstract classes to hide the implementation details from the client program, FOMSApp. For instance, we created a Payment abstract class to define the abstract methods implemented in the CreditCard and Online classes, which we will elaborate further on in section 2.3.3 Abstract Factory.

2.1.2 Inheritance

Since the manager and administrator are assigned additional responsibilities along with those of a staff, we used inheritance to reuse the Staff's attributes and methods and create specialised methods within the Manager and Admin classes.

2.1.3 Encapsulation

Through accessor and mutator methods, we ensured secure access and modification of our private attributes. Since our system handles passwords, which are considered sensitive data, we hashed them with the SHA-512 algorithm for data security.

```
MessageDigest md = MessageDigest.getInstance( algorithm: "SHA-512");
```

Figure 1. Usage of SHA-512 for hashing

2.1.4 Polymorphism

Polymorphism was applied by overriding the methods in the classes that extend the interfaces created for our authorisation factory method (ActiveFactory interface), payment abstract factory (PaymentFactory interface), and filter pattern (IFilter interface). Each of these methods had different styles of implementation in their respective subclasses.

2.2 Use of SOLID Design Principles

2.2.1 Single Responsibility Principle (SRP)

We organised staff permissions using separate "Actions" classes, each tailored to specific roles (Staff, Manager, Admin). These classes are further categorised by the object they interact with.

Figure 2: Examples of integrating SRP

2.2.2 Open-Closed Principle (OCP)

Our payment functionality follows this principle. If new payment categories need to be added, they can simply extend the abstract Payment class in the new classes, which ensures that the same abstract methods can be implemented (in a different manner if need be). We used the Abstract Factory pattern to provide a mechanism to generate families of linked or dependent items without re-declaring the Payment classes, allowing for extensibility without modification, hence following OCP.

2.2.3 Liskov Substitution Principle (LSP)

We ensured that within the Manager and Admin classes, the preconditions are not stronger than those in the Staff class, and the postconditions are not weakened.

2.2.4 Interface Segregation Principle (ISP)

For each staff "Actions" class, we created a corresponding interface. Hence, we demonstrate ISP by ensuring that, for instance, the IAdminStaffActions only contains a set of methods specifically for staff management. This ensures that the AdminStaffActions class does not have to implement abstract methods it will not be able to; for example, it does not have to implement

order management actions. This discourages unrelated dependencies and promotes clearer, more maintainable programming.

2.2.5 Dependency Inversion Principle (DIP)

Considering that permissions for staff members are likely to be modified, we ensured that the Staff, Manager, and Admin classes (high-level modules) and their respective "Actions" classes (low-level modules) depend on abstraction. For instance, our Manager class (high-level module) has a dependency on an instance of the ManagerMenuActions class (low-level module), which implements the IManagerMenuActions interface.

```
public class Manager extends Staff {
    3 usages
    private final IManagerMenuActions managerActions = new ManagerMenuActions();
```

Figure 3. DIP to access ManagerMenuActions class with an interface in Manager class

2.3 Use of Design Patterns

2.3.1 Singleton pattern

The Singleton pattern ensures that the class has only one instance throughout the application and provides global access to that instance. We used it to get the user's input with Java's Scanner class (InputScanner). Additionally, we implemented it for the classes that we only require a single instance of, such as the classes that serve to contain a list of the respective objects (BranchDirectory, StaffDirectory, MenuDirectory, PaymentDirectory, OrderQueue) and the XlsxHelper classes, which contain methods for serialising records into excel files. Doing so ensured that we did not have to pass these classes as method parameters into other methods to use them, simplifying data access and maintaining data integrity.

2.3.2 Factory Method

The Factory Method is a creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to determine which objects to create. We used this method to handle our authorization functionalities, which require us to create the corresponding "Active" user object based on the role of the staff logging in. For instance, we will instantiate an "ActiveStaff" object if the staff logging in has the "Staff" role. Before logging in, the system will not know which "Active" user object to create since it does not know the user's role.

However, with the factory method, we can dynamically determine the correct "Active" user object to instantiate. This method improves code readability and allows for future updates because the object generation mechanism is isolated within the factory classes, making it simple to extend or modify without affecting existing code.

```
// Login
staff = login(staffDirectory);
// Set active staff
if (staff.getRole() == StaffRoles.STAFF) {
    activeStaff = staffFactory.initActive(staff);
} else if (staff.getRole() == StaffRoles.MANAGER) {
    activeManager = managerFactory.initActive(staff);
} else if (staff.getRole() == StaffRoles.ADMIN) {
    activeAdmin = adminFactory.initActive(staff);
}
```

Figure 4. Factory Method Implementation in StaffActions class

2.3.3 Abstract Factory

Abstract Factory is a creational design pattern to produce families of related objects without specifying their concrete classes. This was used to allow for extensibility when adding new payment methods. We assume that there are currently two payment categories that the payment methods will fall under, Credit/Debit Card and Online payment. Hence, we created two factories. Based on the chosen category, the respective factory creates the new object with the required functionality. This prevents tight coupling between concrete products and the client code.

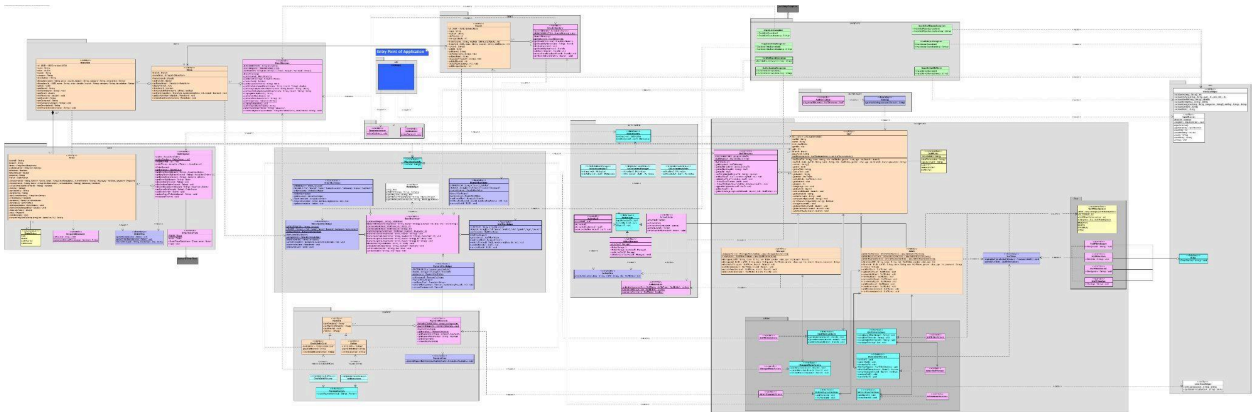
```
switch (category) {
    case "Credit/Debit Card":
        payment = new CreditDebitFactory().createPayment(method);
        break;
    case "Online":
        payment = new OnlineFactory().createPayment(method);
        break;
}
```

Figure 5. Creation of Payment methods through factories

2.3.4 Filter Pattern

Filter Pattern enables developers to filter a set of objects using different criteria. We used this to filter the staff records by their age, role, gender, and branch. This was done by implementing the interface IFilter in the filter classes (StaffFilterAge, StaffFilterBranch, StaffFilterGender, and StaffFilterRole). We used this pattern considering future modifications that require us to filter based on multiple conditions as it allows us to chain filter classes together by creating classes that serve as logical operations. Hence, encouraging loose coupling.

3. UML Class Diagram



For a Clearer UML Diagram:

https://drive.google.com/file/d/1LiODB0zQNm4i42jA4Lph1YNx_9esJWqn/view?usp=drive_link or navigate to the UML folder and select the SC2002projuml.jpg file.

4. Testing

4.1 Manager's action: Menu Management

Test Case 1: Add Unique Menu Item

Test Case 2: Update Price and Description of Existing Menu Item

```
-----------------------------
StaffID: Alexei
Role: MANAGER
1. Add Menu Item
2. Update Menu Item
3. Remove Menu Item
-----------------------------
Enter option: 1
Enter name: Burger
Enter price ($): 7.50
Categories:
1. Side
2. Burger
3. Set meal
4. Drink
Select Category: 2
Enter description: Delicious Burger
Item successfully added in menu!
-----------------------------
```

```
Enter option: 2
| No. | Category   | Name                | Price ($)  | Description                                                                                                |
==========================================================================================================================
| 1   | side       | Fries               | 3.7        | Hot piping fries                                                                                           |
| 2   | side       | Chicken Nugget      | 6.6        | Fresh chicken                                                                                              |
| 3   | side       | Chicken Tenders     | 6.9        | Fresh chicken                                                                                              |
| 4   | drink      | Tea                 | 5.3        | Hot piping tea                                                                                             |
| 5   | Set meal   | Pasta               | 5.0        | Slurp                                                                                                      |
| 6   | Burger     | Burger              | 7.5        | Delicious Burger                                                                                           |
Enter the index of the item you want to update: 2
1. Update name
2. Update price
3. Update category
4. Update description
Select option (5 to quit): 2
Enter price ($): 4.0
Item successfully updated in menu!
1. Update name
2. Update price
3. Update category
4. Update description
Select option (5 to quit): 4
Enter new description: Fresh small chicken
Item successfully updated in menu!
1. Update name
2. Update price
3. Update category
4. Update description
Select option (5 to quit): 5
Item successfully updated in menu!
```

(Left: Add Menu Item, Right: Update Menu Item)

Test Case 3: Remove Existing Menu Item

```
Enter option: 3
| No. | Category   | Name                | Price ($)  | Description                                                                                                |
==========================================================================================================================
| 1   | side       | Fries               | 3.7        | Hot piping fries                                                                                           |
| 2   | side       | Chicken Nugget      | 4.0        | Fresh small chicken                                                                                        |
| 3   | side       | Chicken Tenders     | 6.9        | Fresh chicken                                                                                              |
| 4   | drink      | Tea                 | 5.3        | Hot piping tea                                                                                             |
| 5   | Set meal   | Pasta               | 5.0        | Slurp                                                                                                      |
| 6   | Burger     | Burger              | 7.5        | Delicious Burger                                                                                           |
Enter the number of the item you want to remove: 5
Item successfully removed from menu!
```

4.2 Order Processing

Test Case 4: New Order With Customisations and Takeaway Option

Test Case 5: New Order With Dine-in Option

```
Menu items in branch NTU:
| No. | Category   | Name                | Price ($)  | Description
==============================================================================
| 1   | side       | Chicken Nugget      | 6.6        | Fresh chicken
| 2   | side       | Chicken Tenders     | 6.9        | Fresh chicken
| 3   | drink      | Tea                 | 5.3        | Hot piping tea
| 4   | Burger     | Burger              | 6.7        | Hot burger with sauce
Select item (5 to quit): 1
Select item (5 to quit): 2
Select item (5 to quit): 3
Select item (5 to quit): 5

Selected Items:
1. Chicken Nugget
2. Chicken Tenders
3. Tea
Which item would you like to customise? (enter 4 to quit): 3
What customisation would you like? less sugar

Selected Items:
1. Chicken Nugget
2. Chicken Tenders
3. Tea
Which item would you like to customise? (enter 4 to quit): 4

Pickup Options:
1. Takeaway
2. Dine-In
Please select pickup option: 1
```

```
Menu items in branch NTU:
| No. | Category   | Name                | Price ($)  | Description
==============================================================================
| 1   | side       | Chicken Nugget      | 6.6        | Fresh chicken
| 2   | side       | Chicken Tenders     | 6.9        | Fresh chicken
| 3   | drink      | Tea                 | 5.3        | Hot piping tea
| 4   | Burger     | Burger              | 6.7        | Hot burger with sauce
Select item (5 to quit): 1
Select item (5 to quit): 5

Selected Items:
1. Chicken Nugget
Which item would you like to customise? (enter 2 to quit): 2

Pickup Options:
1. Takeaway
2. Dine-In
Please select pickup option: 2
```

(Left: Customisations and Takeaway, Right: Dine-in)

4.3 Payment Integration & 4.4 Order Tracking

Test Case 6: Simulate Payment With Credit/Debit Card

Test Case 7: Simulate Payment With Online Payment Platform

Test Case 8: Track Status of Existing Order with Order ID

```
Available payment methods:
1. Credit/Debit Card
2. PayNow
3. PayPal
How would you like to pay?: 1
Enter the card number: 123456789012345 6
Enter the CVV: 127
Payment by Credit/Debit Card is successful
Order placed successfully.

========================================
|            Order Receipt             |
========================================
| Order ID: T-101                      |
|--------------------------------------|
| Items:                               |
|   - Chicken Nugget                   |
|   - Chicken Tenders                  |
|   - Tea                              |
```

```
Available payment methods:
1. Credit/Debit Card
2. PayNow
3. PayPal
How would you like to pay?: 2
Enter mobile number: 12345678
Payment by PayNow is successful.
Order placed successfully.

========================================
|            Order Receipt             |
========================================
| Order ID: D-103                      |
|--------------------------------------|
| Items:                               |
|   - Chicken Nugget                   |
|
```

```
Please select option:
1. Browse Menu
2. Check Order Status
3. Collect Order
4. Return back
Your choice: 2

Enter orderID (enter quit to exit): D-103
Status of Order ID: D-103 is NEW
```

(Left: Credit/Debit, Middle: Online Payment, Right: Track Order Status)

4.5 Staff Actions

Test Case 9: Login as Staff, Display New Orders and Verify All New Orders in Branch

Test Case 10: Process New Order and Update Status to READY

```
StaffID: kumarB
Role: STAFF
1. Display new orders
2. View details of an order
3. Set Order as Ready
4. Change Password
5. Logout
-----------------------------
Enter option: 1
New orders in the queue:
----------------------------------------

Order ID: D-101
Items:
- Chicken Nugget, customisations: test
- Chicken Nugget, customisations: test
Pickup option: Dine In
Status: NEW
```

```
Order ID: D-103
Items:
- Chicken Nugget, customisations: test
- Chicken Nugget, customisations: test
Pickup option: Dine In
Status: NEW

Enter orderID (enter quit to exit): D-103
Order status updated from NEW to READY for order ID: D-103
```

(Left: Display New Orders, Right: Update Status to READY)

4.6 Manager Actions

Test Case 11: Login as Manager and Display Staff List in Branch

Test Case 12: Manager Display New Orders as Seen in Test Case 9

```
StaffID: Alexei
Role: MANAGER
1. Display Staff
2. Menu Management
3. Order Management
4. My Account
-----------------------------
Enter option: 1
Staff Details of NTU branch
| StaffID         | Name                | Branch     | Role       | Gender   | Age   |
==================================================================================
| Alexei          | Alexei              | NTU        | MANAGER    | M        | 25    |
| kumarB          | kumar Blackmore     | NTU        | STAFF      | M        | 33    |
-----------------------------
```

```
StaffID: Alexei
Role: MANAGER
1. Get New Orders
2. Get Details of an Order
3. Set Order as Ready
-----------------------------
Enter option: 1
New orders in the queue:
----------------------------------------

Order ID: D-102
Items:
- Chicken Nugget
- Tea, customisations: less sugar
Pickup option: Dine In
Status: NEW
```

(Left: Display Staff List, Right: Display New Orders)

4.7 Admin Actions

Test Case 13: Close Branch and Ensure Branch Does Not Display For Customer

```
Branches:
1. Jurong East
2. North Spine Plaza
3. Jurong Point
Select Branch:
1
Branch successfully closed and staff records for this branch have been deleted.
```

```
==================================
|              Menu              |
|        Welcome Customer!       |
==================================
Branches:
1. North Spine Plaza
2. Jurong Point
Select Branch:
```

(Left: Branch Closed, Right: Customer Interface)

Test Case 14: Login as Admin and Display Staff List with Filters

```
Branches:
1. Jurong Point
2. Jurong East
Select Branch:
1
Staff Details of JP branch
| StaffID    | Name                | Branch     | Role       | Gender   | Age   |
==========================================================================
| MaryL      | Mary lee            | JP         | STAFF      | F        | 44    |
| jake45     | Jake                | JP         | MANAGER    | M        | 28    |
-----------------------------
```

(Filtered by Branch)

```
Enter role (S/M):
S: Staff
M: Manager
S
Staff Details
| StaffID    | Name                | Branch     | Role       | Gender   | Age   |
==========================================================================
| MaryL      | Mary lee            | JP         | STAFF      | F        | 44    |
| JustinL    | Justin Loh          | JE         | STAFF      | M        | 49    |
| johndoe    | John Doe            | JE         | STAFF      | M        | 32    |
-----------------------------
```

(Filtered by Role)

```
Enter gender (M/F): F
Female Staff Details
| StaffID         | Name                | Branch     | Role       | Gender   | Age   |
==================================================================================
| MaryL           | Mary lee            | JP         | STAFF      | F        | 44    |
```

(Filtered by Gender)

```
Enter lower bound for age: 40
Staff Details of staff whose age is equals to 40 and above
| StaffID         | Name                | Branch     | Role       | Gender   | Age   |
==================================================================================
| MaryL           | Mary lee            | JP         | STAFF      | F        | 44    |
| JustinL         | Justin Loh          | JE         | STAFF      | M        | 49    |
```

(Filtered by Age)

Test Case 15: Assign Managers to Branches with the Quota/Ratio Constraint

```
Enter staff ID: JustinL
Select a branch to transfer to
Branches:
1. North Spine Plaza
2. Jurong Point
3. Jurong East
Select Branch: 2
The manager quota for Jurong Point is already met.
```

```
Staff Details of JP branch
| StaffID         | Name                | Branch     | Role       | Gender   | Age   |
==================================================================================
| jake45          | Jake                | JP         | MANAGER    | M        | 28    |
| MaryL           | Mary lee            | JP         | MANAGER    | F        | 44    |
| kumarB          | kumar Blackmore     | JP         | MANAGER    | M        | 33    |
```

(Left: Staff Not Promoted to Manager due to Quota, Right: Verification)

Test Case 16: Promote a Staff to a Manager

```
| StaffID        | Name                | Branch     | Role       | Gender   | Age   |
==================================================================================
| MaryL          | Mary lee            | JP         | STAFF      | F        | 44    |
| JustinL        | Justin Loh          | JE         | STAFF      | M        | 49    |
| johndoe        | John Doe            | JE         | STAFF      | M        | 32    |
| kumarB         | kumar Blackmore     | NTU        | STAFF      | M        | 33    |
Enter staff ID: johndoe
The manager quota for Jurong East has not been met.
Staff member with ID johndoe has been promoted to Manager.
```

```
Manager Details
| StaffID        | Name                | Branch     | Role       | Gender   | Age   |
==================================================================================
| Alexei         | Alexei              | NTU        | MANAGER    | M        | 25    |
| jake45         | Jake                | JP         | MANAGER    | M        | 28    |
| johndoe        | John Doe            | JE         | MANAGER    | M        | 32    |
```

(Left: Staff Promoted to Manager, Right: Verification)

Test Case 17: Transfer a staff/manager and verify that the transfer is reflected.

```
Enter staff ID: MaryL
Select a branch to transfer to
Branches:
1. North Spine Plaza
2. Jurong Point
3. Jurong East
Select Branch: 3
The manager quota for Jurong East has not been met.
Staff member with ID MaryL has been transferred to Jurong East successfully
```

```
Staff Details of JE branch
| StaffID        | Name                | Branch     | Role       | Gender   | Age   |
==================================================================================
| JustinL        | Justin Loh          | JE         | MANAGER    | M        | 49    |
| MaryL          | Mary lee            | JE         | MANAGER    | F        | 44    |
```

(Left: Manager Transferred, Right: Verification)

4.8 Customer Interface

Test Case 18: Check Status by OrderID and Verify that it changed to COMPLETED

```
Please select option:
1. Browse Menu
2. Check Order Status
3. Collect Order
4. Return back
Your choice: 2

Enter orderID (enter quit to exit): T-101
Status of Order ID: T-101 is READY
```

```
Please select which order to pickup: 1
Order completed. Thank you for dining with us!
Please select option:
1. Browse Menu
2. Check Order Status
3. Collect Order
4. Return back
Your choice: 2

Enter orderID (enter quit to exit): T-101
Status of Order ID: T-101 is COMPLETED
```

(Left: Order Shows READY for Customer, Right: Order Shows COMPLETED When Customer Collects)

4.9 Error Handling

Test Case 19: Attempt to Add Duplicate Menu Item and Display Error Message

Test Case 20: Attempt to Place Empty Order and Display Error Message

```
1. Add Menu Item
2. Update Menu Item
3. Remove Menu Item
-----------------------------
Enter option: 1
Enter name: Burger
Enter price ($): 7.50
Categories:
1. Side
2. Burger
3. Set meal
4. Drink
Select Category: 2
Enter description: yummy burger
Menu item not inserted, there is a duplicate item.
```

```
Menu items in branch NTU:
| No. | Category   | Name                | Price ($)  | Description
=========================================================================
| 1   | side       | Chicken Nugget      | 6.6        | Fresh chicken
| 2   | side       | Chicken Tenders     | 6.9        | Fresh chicken
| 3   | Burger     | Burger              | 7.5        | Yummy burger
Select item (4 to quit): 4
Please select at least one item.
```

(Left: Error Message for Duplicate Item, Right: Error Message for Empty Order)

4.10 Extensibility

Test Case 21: Add New Payment Method

Test Case 22: Open New Branch Without Affecting Existing Functionalities

```
Please enter name of new payment method: PayWave
Updated list of payment methods are:
Available payment methods:
1. Credit/Debit Card
2. PayNow
3. PayPal
4. PayLah
5. PayWave
```

```
StaffID: boss
Role: ADMIN
1. Add branch
2. Close Branch
-----------------------------
Enter option: 1
Enter branch name: AMK
Enter branch location: Ang Mo Kio
Enter staff quota: 8
Branch successfully created!
```

(Left: New Payment Method Added, Right: New Branch Added)

4.11 Order Cancellation

Test Case 23: Uncollected Order is Cancelled Beyond a Specified Timeframe (5 minutes)

```
Order ID: T-101
Items:
- Chicken Nugget
- Chicken Tenders, customisations: less salt
Pickup option: Takeaway
Status: CANCELLED
```

(Order Cancelled)

4.12 Login System

Test Case 24: Login as Staff with Incorrect Credentials and Display Error Message

Test Case 25: Login as Staff to Change Password and Login Again with New Password

```
================================
|          Login System        |
|        Welcome Staff!        |
================================
Please select option:
1. Login
2. Return back
Your choice: 1

Enter staff ID: afreen
Staff does not exist. Please enter a valid staffID.
Enter staff ID: Alexei
Enter password: alexei
Incorrect password, please try again.
```

```
StaffID: Alexei
Role: MANAGER
1. Change Password
2. Logout
-----------------------------
Enter option: 1

Please enter new password: alexei
Password successfully changed!
```

```
================================
|          Login System        |
|        Welcome Staff!        |
================================
Please select option:
1. Login
2. Return back
Your choice: 1

Enter staff ID: Alexei
Enter password: alexei
Logged in!
```

(Left: Incorrect Staff ID and Password (Test Case 24),
Middle: Password Changed Successfully (Test Case 25), Right: Logged in with New Password (Test Case 25))

4.13 Staff List Initialisation

Test Case 26: Upload Staff List During System Initialisation

	A	B	C	D	E	F	G	
1	id	name	staffID	role	gender	age	branch	password
2	67136f7c-fcd0-45f1-8859-9e3d183faeb3	Alexei	Alexei	M	M	25	NTU	b109f3bbbc244eb82441917ed06d618b9008dd09b3be
3	9234b60f-23e1-4b28-9732-c987de0605a8	Mary lee	MaryL	S	F	44	JP	b109f3bbbc244eb82441917ed06d618b9008dd09b3be
4	22148748-4c7e-4331-8686-d2c6d3c27e22	Justin Loh	JustinL	S	M	49	JE	b109f3bbbc244eb82441917ed06d618b9008dd09b3be
5	ec084e54-155c-4a11-b8e8-04df6cfe3c87	Boss	boss	A	F	62	NULL	3f462bdfa4ef03329b86779baf5d7601c3899dcb47d25
6	90b56832-f283-491f-86b0-ee90b6ac2aee	Jake	jake45	M	M	28	JP	b109f3bbbc244eb82441917ed06d618b9008dd09b3be
7	2198db1f-d555-4c6e-be37-5ee944f2ebc8	kumar Blackmore	kumarB	S	M	33	NTU	b109f3bbbc244eb82441917ed06d618b9008dd09b3be
8	1b08ee2e-932b-44e8-8f1a-81a8f5487389	John Doe	johndoe	M	M	32	NTU	b109f3bbbc244eb82441917ed06d618b9008dd09b3be

(Staff List)

```
Enter role (S/M):
S: Staff
M: Manager
S
Staff Details
| StaffID    | Name                | Branch     | Role       | Gender   | Age   |
==============================================================================
| MaryL      | Mary lee            | JP         | STAFF      | F        | 44    |
| JustinL    | Justin Loh          | JE         | STAFF      | M        | 49    |
| johndoe    | John Doe            | JE         | STAFF      | M        | 32    |
-----------------------------
```

(Staff List Filtered by Role)

4.14 Data Persistence

Test Case 27: Perform Multiple Sessions of Application, Adding, Updating and Removing Menu Items and Verify the Changes Made in One Session Persist

name	price	branch	category	description
Fries	4.3	JP	side	Hot piping fries
Coke	3.2	JE	drink	Drink
Cole Slaw	2.7	JE	side	Fresh
3pc Set Meal	5.2	JE	set meal	50% off
Pepsi	2.1	JE	Drink	Cold drink
Chicken Nugget	6.9	JP	side	Fresh chicken
Chicken Nugget	6.6	NTU	side	Fresh chicken
Chicken Tenders	6.9	NTU	side	Fresh chicken
Milkshake	4.0	NTU	Drink	Macha flavoured

name	price	branch	category	description
Fries	4.3	JP	side	Hot piping fries
Coke	3.2	JE	drink	Drink
Cole Slaw	2.7	JE	side	Fresh
3pc Set Meal	5.2	JE	set meal	50% off
Pepsi	2.1	JE	Drink	Cold drink
Chicken Nugget	6.9	JP	side	Fresh chicken
Chicken Nugget	6.6	NTU	side	Fresh chicken
Chicken Tenders	6.9	NTU	side	Fresh chicken

(Left: Before Change to Menu, Right: After Change to Menu)

5. Reflection

At the start of our project, we dedicated time to thoroughly analyse the project guidelines. Each team member crafted an individual UML diagram based on our understanding, which led to different interpretations of the project. Through constructive discussion, we came to a consensus on the UML's preliminary design. This foundational phase also included establishing key packages such as Branch, Order, and Payment, and allocating tasks based on our strengths.

We held weekly meetings for code review and to strategise our subsequent steps to handle errors and enhance our code based on the design principles. For example, after discussing, we decided to split the Admin class into AdminBranchActions and AdminStaffActions, incorporating SOLID principles into our design.

As our project progressed, some of the biggest challenges that we faced were overlooking errors in our code, such as the timer function where our orders were cancelled before the timer ended, implementing payment extensibility, serialising our interfaces, and creating our UML, where there were constant changes to be made since we kept expanding our classes. A big hurdle was harmonising our distinct coding styles and integrating diverse perspectives into a cohesive implementation.

To address these challenges, we prioritised clear and open communication. Our in-person meetings were crucial, allowing us to collectively review code, share improvements, and integrate OOP concepts. Regular updates and support requests in our group chat facilitated teamwork and alignment. Furthermore, we tackled the technical challenges by conducting additional research to determine the best approach to tackle our issues. This led us to learn new concepts that were not covered in the curriculum, such as singletons, factory method, abstract factory, filter pattern, Java Timer object, and serialisation with Excel files.

Overall, this project was a meaningful learning opportunity that not only taught us how to integrate design principles into our code but also enhanced our soft skills, especially teamwork and communication.