SC2002 OBJECT ORIENTED DESIGN & PROGRAMMING

Fast-food Ordering and Management System (FOMS)

Report of Project Structure Design & Functionality

AY23/24 Semester 2 | FDAD, Group 1

NAME	MATRICULATION NUMBER
Devlin Nathan Waluja	U2320897G
Lim Jun Long	U2321544A
Teo Wei Yew	U2322326G
Nguyen Viet Dung	U2323122H

GitHub Main Page: https://github.com/SC2002FDADG1/SC2002-Fastfood-ordering-and-management-System-FOMS-/tree/main

Declaration of Original Work for SC/CE/CZ2002 Assignment

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.
2.) We have honoured the principles of academic integrity and have upheld the Student Code of Academic Conduct in the completion of this work.
3.) 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
Devlin Nathan Waluja	SC2002	FDAD	[illegible]
Lim Jun Long	SC2002	FDAD	junlong
Teo Wei Yew	SC2002	FDAD	[illegible]
Nguyen Viet Dung	SC2002	FDAD	[illegible]

1 INTRODUCTION

Fast Food Ordering Management System (FOMS) is an integrated system comprising 4 distinct portals for 4 distinct actors - Customer, Staff, Manager, Admin. Customers can browse the menu, place orders, and check their order status. Staff can process orders, Managers manage menu items, and Admin oversees company and staff management. This document outlines the design considerations for FOMS, focusing on object-oriented principles, especially the S.O.L.I.D. design principles, to ensure a modular and maintainable codebase.

2 DESIGN CONSIDERATIONS

The FOMS was built with OOP design considerations in mind to maintain reusability and ease of developer collaboration through modularity. It also takes into consideration codebase maintainability making it sustainable for expansion and updates in the future. We have applied this through the employment of S.O.L.I.D design principles, as learnt in the module.

2.1 SOLID DESIGN PRINCIPLES

2.1.1 SINGLE RESPONSIBILITY PRINCIPLE (SRP)

SRP states that each class should only have one responsibility for a portion of the software's function. This reduces dependency which ensures that changes in one class would not have adverse effects on other classes. In our code, we decided to place the method to calculate the total price for a ***FoodOrder*** (in ***FoodSystem*** package) within the ***Order*** class (in ***OrderSystem*** package). It encapsulates the functionality within the domain where it is most relevant, keeping the ***FoodOrder*** class simple and focused on representing a single order item. ***FoodOrder*** represents a single food item in an order, while ***Order*** manages the collection of ***FoodOrder*** items.

2.1.2 OPEN-CLOSED PRINCIPLE (OCP)

OCP allows for base classes with well-defined methods to be extended through derived classes without requiring modifications to the base class. In FOMS, the ***Food*** class represents generic menu items. By following OCP, we can create derived classes to add new types of food items without altering the original Food class. This approach preserves the stability and integrity of existing code while enabling extensibility through inheritance. It allows developers to expand the system's capabilities by introducing new behaviours and attributes in derived classes, without changing the base class's structure or functionality. We hence reduce the risk of breaking existing functionality when adding new features or extending behaviour.

2.1.3 LISKOV SUBSTITUTION PRINCIPLE (LSP)

LSP guided our approach to designing the class hierarchy within FOMS. We started with an ***Employee*** base class, from which ***Staff***, ***Manager***, and ***Admin*** were derived. By following LSP, we ensured that these derived classes could substitute the base class without altering the system's expected behaviour. This allowed us to use instances of ***Staff***, ***Manager***, or ***Admin*** wherever an ***Employee*** was required, without encountering unexpected issues or system errors. This design approach simplified code maintenance and enhanced flexibility, enabling us to expand the system without the risk of breaking existing functionality. By adhering to LSP, we created a robust and adaptable codebase that supports polymorphism and ensures that derived classes uphold the principles and behaviours of their base class.

2.1.4 INTERFACE SEGREGATION PRINCIPLE (ISP)

ISP in our project was implemented by defining interfaces for distinct operations, allowing classes to focus on specific functionalities without unnecessary clutter. We created interfaces like ***ReadFile***, ***Write***, ***Save***, and ***ReadObject***, each designed to address a particular task within the system. This separation of concerns ensures that classes only need to implement the interfaces they use, promoting modularity. Additionally, our classes implement only the interfaces that align with their intended purpose. When a class is responsible for file operations, it implements ***ReadFile*** and ***Write***, but not ***Save*** or ***ReadObject*** if those functionalities are irrelevant. This approach reduces code bloat, streamlines class responsibilities, and contributes to a more organised codebase. Overall, our use of ISP keeps the system flexible and easier to maintain.

2.1.5 DEPENDENCY INJECTION PRINCIPLE (DIP)

DIP in the FOMS is exemplified by injecting dependencies into classes via constructors or setter methods, rather than instantiating them directly within the class. In FOMS, we pass instances of ***MenuData***, ***OrderData***, or ***PaymentData, StaffData*** through constructors or setters into various portals like ***CustomerPortal*** and ***StaffPortal***. This approach allows high-level classes to interact with abstractions instead of concrete implementations. By relying on dependency injection, the system becomes more adaptable, making it straightforward to replace specific implementations without impacting the overall codebase.

2.2 Reflection

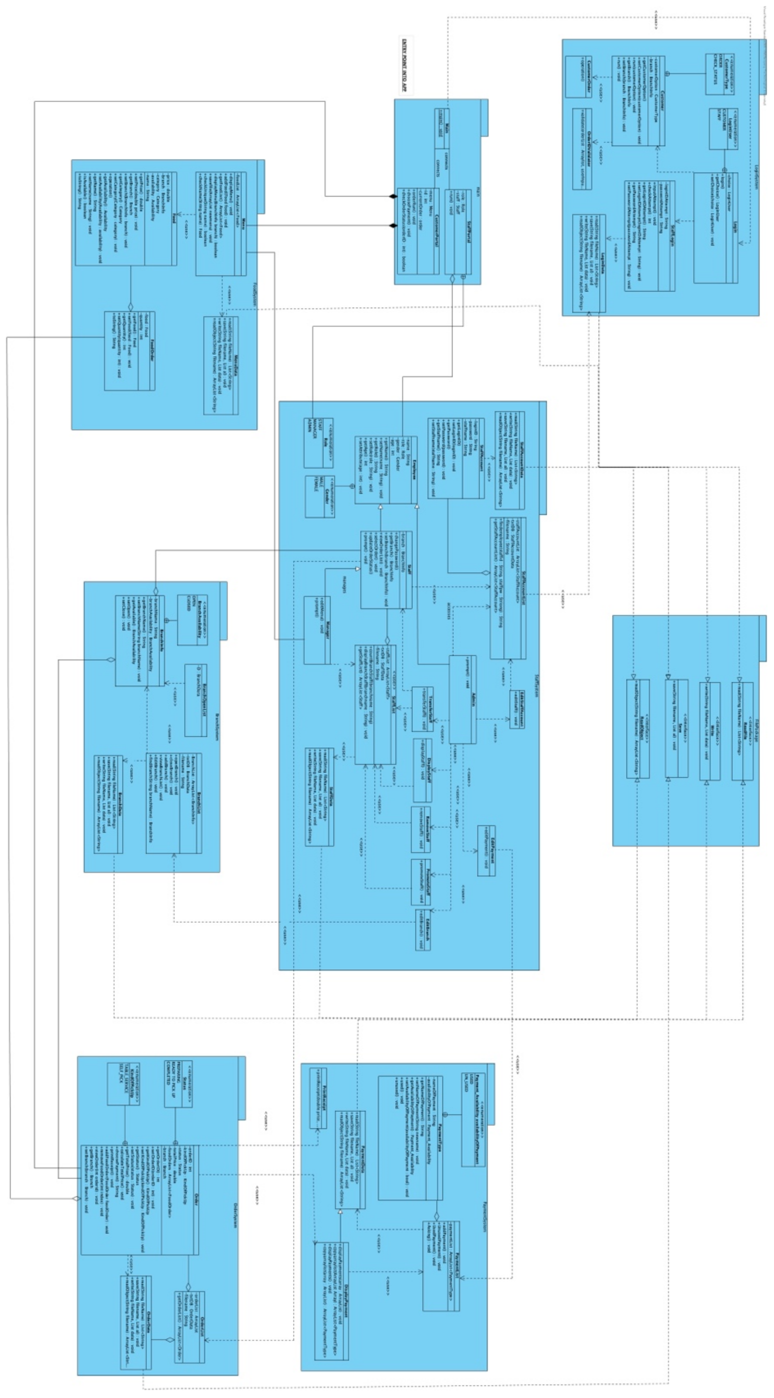
Through the design and execution of this FOMS project, we have had hands-on experience in implementing OOP concepts, allowing for us to apply theory learnt in class to practical work. When we first started this project, we discovered that practically every other section of our code needed to be altered because even the smallest modification would have a domino effect, due to our lack of adherence to S.O.L.I.D principles.

Hence, we tried to make extensive use of the design principles throughout our task to achieve high cohesion and loose coupling. This would then allow for high flexibility, ease of maintenance, and scalability.

In addition, we had to consider all user roles, including admins, customers, and staff, to prevent conflicts between them. For example, admins have access to staff information, while customers do not. Managers are granted the ability to edit the menu, distinguishing their privileges from regular staff, despite both being part of the company's staff.

Upon reflection, the incorporation of S.O.L.I.D. concepts into our project not only strengthened its technical foundation but also facilitated more efficient teamwork. We were able to create more seamless development iterations and lessen the possibility of conflicts during concurrent work by cultivating a modular and well-organised codebase. Furthermore, by granting customised access to functionalities, the explicit division of responsibilities and rights among users enhanced user experience while also guaranteeing data security and integrity. Overall, the project's goals were successfully and efficiently attained thanks to the careful use of design principles.

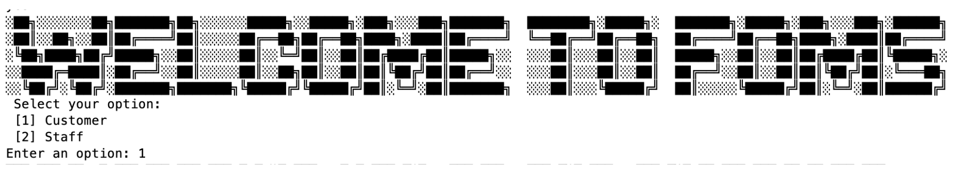
3 UML DIAGRAM



4 Testing

4.1 Login Page

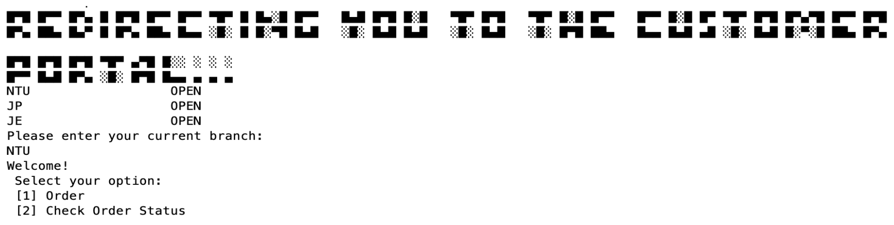
Upon entering the system, the user can choose from either ***Customer*** or ***Staff***.



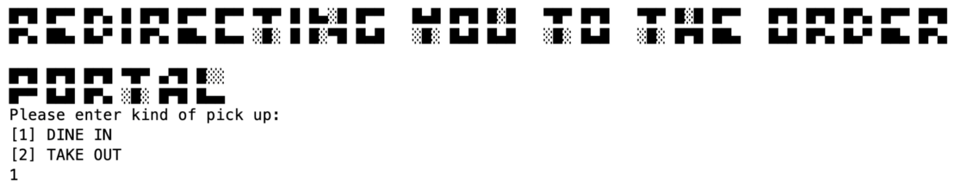
```
 Select your option:
 [1] Customer
 [2] Staff
Enter an option: 1
```

4.2 Customer Interfaces (4.2.1 + 4.2.2)

Redirected to customer portal. Customer can now choose to either ***Order*** or ***Check Order Status***.

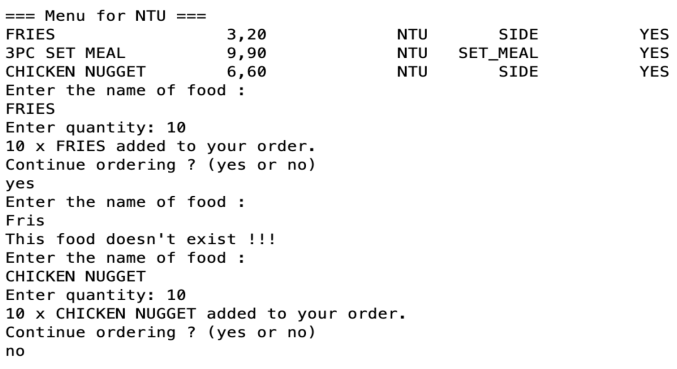


```
NTU                OPEN
JP                 OPEN
JE                 OPEN
Please enter your current branch:
NTU
Welcome!
 Select your option:
 [1] Order
 [2] Check Order Status
```



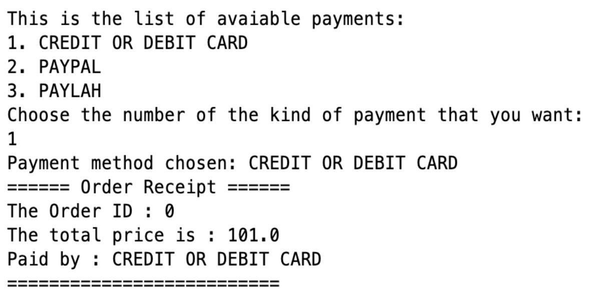
```
Please enter kind of pick up:
[1] DINE IN
[2] TAKE OUT
1
```

Redirected to Order Portal. Customer can choose to either ***Dine In*** or ***Take Out***.



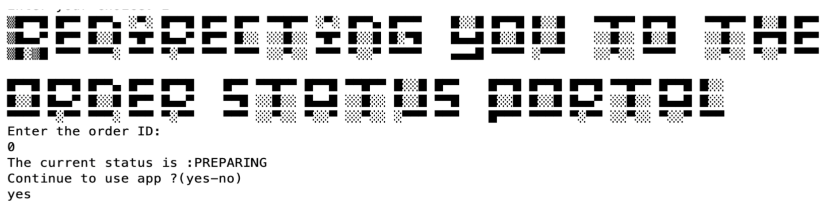
```
=== Menu for NTU ===
FRIES                3,20            NTU       SIDE          YES
3PC SET MEAL         9,90            NTU   SET_MEAL          YES
CHICKEN NUGGET       6,60            NTU       SIDE          YES
Enter the name of food :
FRIES
Enter quantity: 10
10 x FRIES added to your order.
Continue ordering ? (yes or no)
yes
Enter the name of food :
Fris
This food doesn't exist !!!
Enter the name of food :
CHICKEN NUGGET
Enter quantity: 10
10 x CHICKEN NUGGET added to your order.
Continue ordering ? (yes or no)
no
```

Menu shown. Customer enters ***name of food*** and ***quantity.***



```
This is the list of avaiable payments:
1. CREDIT OR DEBIT CARD
2. PAYPAL
3. PAYLAH
Choose the number of the kind of payment that you want:
1
Payment method chosen: CREDIT OR DEBIT CARD
====== Order Receipt ======
The Order ID : 0
The total price is : 101.0
Paid by : CREDIT OR DEBIT CARD
=========================
```

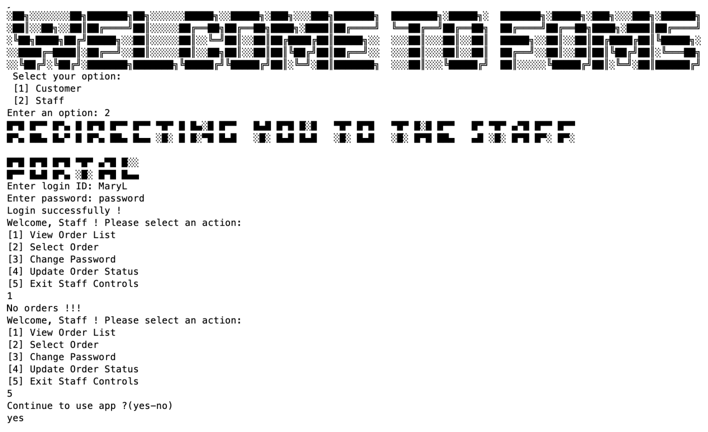
When customer decides to stop ordering. Customer will choose from a ***list of available payments***. Receipt will then be displayed.



```
REDIRECTING YOU TO THE
ORDER STATUS PORTAL
Enter the order ID:
0
The current status is :PREPARING
Continue to use app ?(yes-no)
yes
```

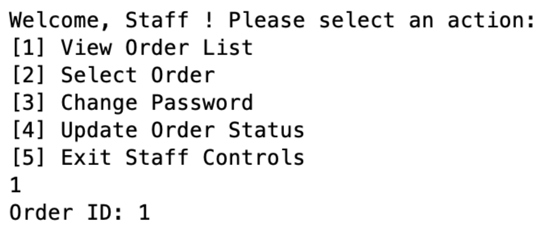
Redirected to Order Status Portal. Customer can enter order ID to check current status.

4.3.1 Staff Interface



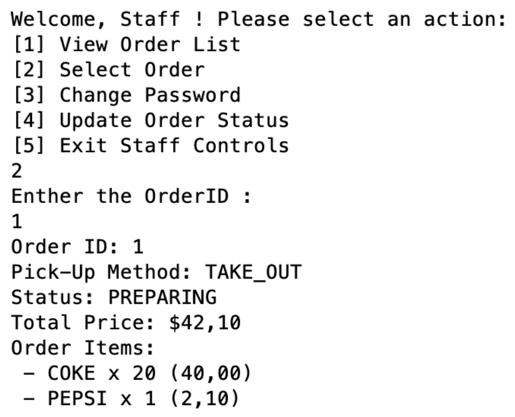
```
WELCOME TO FOMS
 Select your option:
 [1] Customer
 [2] Staff
Enter an option: 2
REDIRECTING YOU TO THE STAFF
PORTAL
Enter login ID: MaryL
Enter password: password
Login successfully !
Welcome, Staff ! Please select an action:
[1] View Order List
[2] Select Order
[3] Change Password
[4] Update Order Status
[5] Exit Staff Controls
1
No orders !!!
Welcome, Staff ! Please select an action:
[1] View Order List
[2] Select Order
[3] Change Password
[4] Update Order Status
[5] Exit Staff Controls
5
Continue to use app ?(yes-no)
yes
```

Redirected to Staff Portal. Normal Staff enters his/her ***Login Id*** and ***Password***. A list of actions that the normal Staff can perform will be displayed.



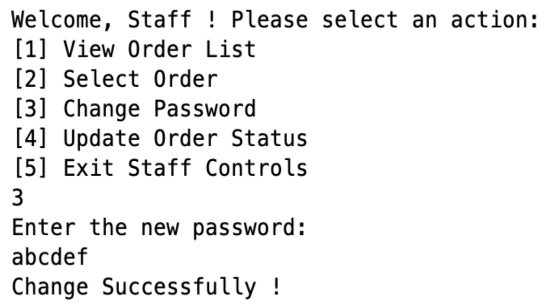
```
Welcome, Staff ! Please select an action:
[1] View Order List
[2] Select Order
[3] Change Password
[4] Update Order Status
[5] Exit Staff Controls
1
Order ID: 1
```

Staff views list of orders.



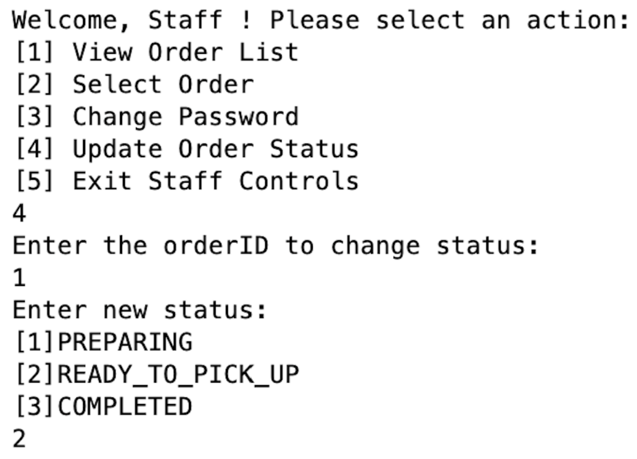
```
Welcome, Staff ! Please select an action:
[1] View Order List
[2] Select Order
[3] Change Password
[4] Update Order Status
[5] Exit Staff Controls
2
Enther the OrderID :
1
Order ID: 1
Pick-Up Method: TAKE_OUT
Status: PREPARING
Total Price: $42,10
Order Items:
 - COKE x 20 (40,00)
 - PEPSI x 1 (2,10)
```

Staff selects order and order information will be displayed.



```
Welcome, Staff ! Please select an action:
[1] View Order List
[2] Select Order
[3] Change Password
[4] Update Order Status
[5] Exit Staff Controls
3
Enter the new password:
abcdef
Change Successfully !
```

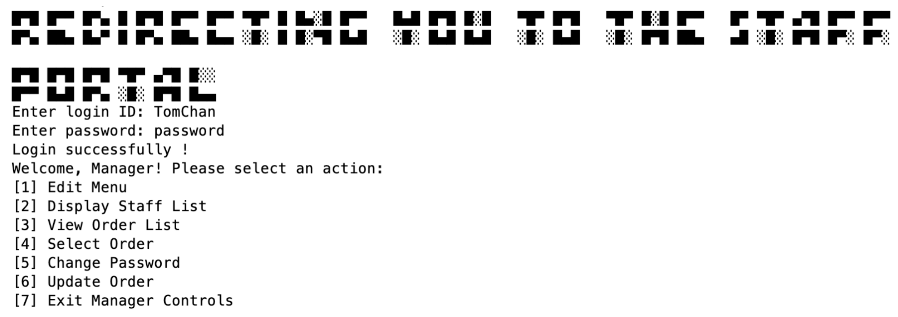
Staff changes login password.



```
Welcome, Staff ! Please select an action:
[1] View Order List
[2] Select Order
[3] Change Password
[4] Update Order Status
[5] Exit Staff Controls
4
Enter the orderID to change status:
1
Enter new status:
[1]PREPARING
[2]READY_TO_PICK_UP
[3]COMPLETED
2
```

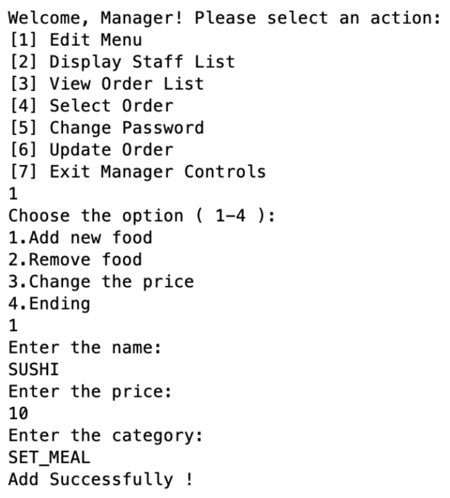
Staff updates order status. For example, order status *preparing* is changed to *ready_to_pick_up*.

4.3.2 Manager Interface



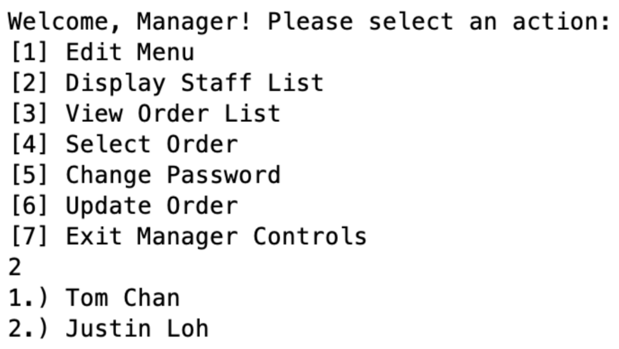
```
REDIRECTING YOU TO THE STAFF
PORTAL
Enter login ID: TomChan
Enter password: password
Login successfully !
Welcome, Manager! Please select an action:
[1] Edit Menu
[2] Display Staff List
[3] View Order List
[4] Select Order
[5] Change Password
[6] Update Order
[7] Exit Manager Controls
```

Redirected to Staff Portal. Manager enters his/her ***Login Id*** and ***Password***. A list of actions that the manager can perform will be displayed.



```
Welcome, Manager! Please select an action:
[1] Edit Menu
[2] Display Staff List
[3] View Order List
[4] Select Order
[5] Change Password
[6] Update Order
[7] Exit Manager Controls
1
Choose the option ( 1-4 ):
1.Add new food
2.Remove food
3.Change the price
4.Ending
1
Enter the name:
SUSHI
Enter the price:
10
Enter the category:
SET_MEAL
Add Successfully !
```

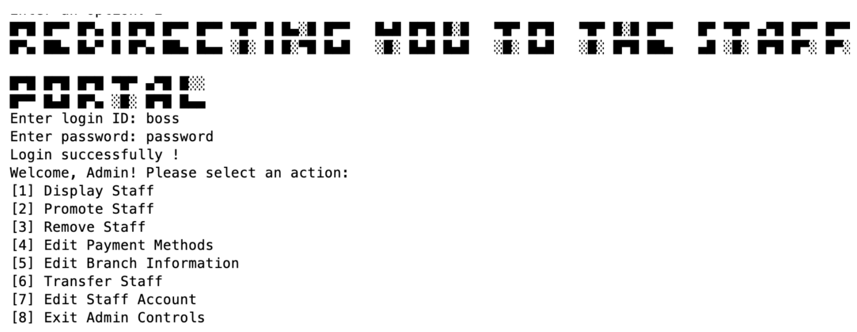
Manager edits menu by adding new food item.



```
Welcome, Manager! Please select an action:
[1] Edit Menu
[2] Display Staff List
[3] View Order List
[4] Select Order
[5] Change Password
[6] Update Order
[7] Exit Manager Controls
2
1.) Tom Chan
2.) Justin Loh
```

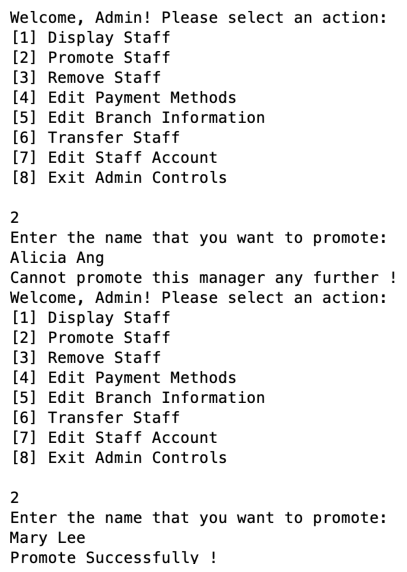
Manager displays staff list.

4.3.3 Admin Interface



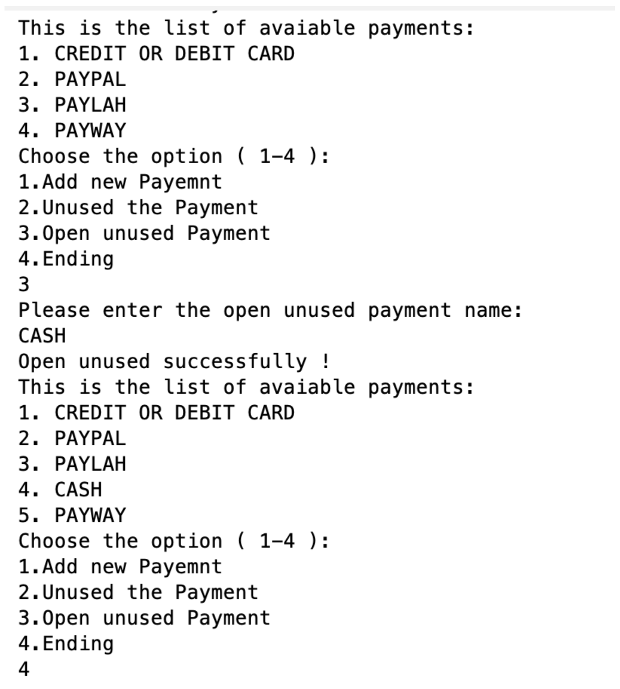
```
REDIRECTING YOU TO THE STAFF
PORTAL
Enter login ID: boss
Enter password: password
Login successfully !
Welcome, Admin! Please select an action:
[1] Display Staff
[2] Promote Staff
[3] Remove Staff
[4] Edit Payment Methods
[5] Edit Branch Information
[6] Transfer Staff
[7] Edit Staff Account
[8] Exit Admin Controls
```

Redirected to Staff Portal. Admin enters his/her ***Login Id*** and ***Password***. A list of actions that the Admin can perform will be displayed.



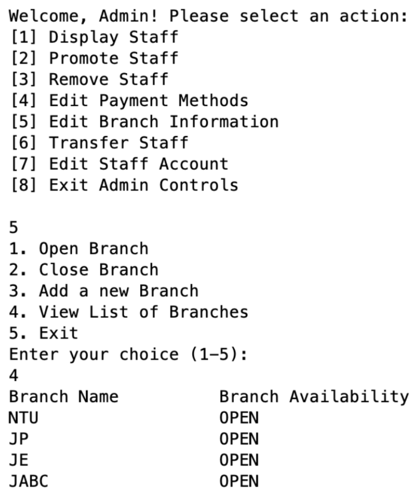
```
Welcome, Admin! Please select an action:
[1] Display Staff
[2] Promote Staff
[3] Remove Staff
[4] Edit Payment Methods
[5] Edit Branch Information
[6] Transfer Staff
[7] Edit Staff Account
[8] Exit Admin Controls

2
Enter the name that you want to promote:
Alicia Ang
Cannot promote this manager any further !
Welcome, Admin! Please select an action:
[1] Display Staff
[2] Promote Staff
[3] Remove Staff
[4] Edit Payment Methods
[5] Edit Branch Information
[6] Transfer Staff
[7] Edit Staff Account
[8] Exit Admin Controls

2
Enter the name that you want to promote:
Mary Lee
Promote Successfully !
```

Admin promotes a staff. If staff is already manager, staff can’t be promoted.



```
This is the list of avaiable payments:
1. CREDIT OR DEBIT CARD
2. PAYPAL
3. PAYLAH
4. PAYWAY
Choose the option ( 1-4 ):
1.Add new Payemnt
2.Unused the Payment
3.Open unused Payment
4.Ending
3
Please enter the open unused payment name:
CASH
Open unused successfully !
This is the list of avaiable payments:
1. CREDIT OR DEBIT CARD
2. PAYPAL
3. PAYLAH
4. CASH
5. PAYWAY
Choose the option ( 1-4 ):
1.Add new Payemnt
2.Unused the Payment
3.Open unused Payment
4.Ending
4
```

Admin edits payments methods.



```
Welcome, Admin! Please select an action:
[1] Display Staff
[2] Promote Staff
[3] Remove Staff
[4] Edit Payment Methods
[5] Edit Branch Information
[6] Transfer Staff
[7] Edit Staff Account
[8] Exit Admin Controls

5
1. Open Branch
2. Close Branch
3. Add a new Branch
4. View List of Branches
5. Exit
Enter your choice (1-5):
4
Branch Name          Branch Availability
NTU                  OPEN
JP                   OPEN
JE                   OPEN
JABC                 OPEN
```

Admin edits branch information.