



GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT038-3-2-ODJ

OBJECT ORIENTED DEVELOPMENT WITH JAVA

APU2F2106CS(IS) / APD2F2106CS(IS)

HAND OUT DATE: 29 JUNE 2021

HAND IN DATE: 3 SEPTEMBER 2021

WEIGHTAGE: 50%

Name	TP Number
Sia De Long	TP060810
Emerson Kong Zee Xin	TP061400

Table of Contents

1.0 Introduction	3
1.1 Assumptions	3
2.0 Use Case Diagram	4
2.1 Use Case Specifications	9
3.0 Class Diagram.....	15
4.0 Class Concepts.....	16
4.1 Inheritance	16
4.2 Polymorphism.....	18
4.3 Abstraction.....	21
4.4 Encapsulation.....	22
5.0 Main Functions Showcase	24
5.1 Admin Functions	24
5.1.1 Products	25
5.1.1.1 Add Product	26
5.1.1.2 Delete Product	27
5.1.1.3 Edit Product	28
5.1.1.4 View Product	29
5.1.1.5 Search Product	30
5.1.2 Customers	31
5.1.2.1 Add Customer	32
5.1.2.2 Delete Customer	33
5.1.2.3 Edit Customer	34
5.1.2.4 View Customer	35
5.1.2.5 Search Customer	36
5.1.3 Orders	37
5.1.3.1 View and Search All Users Order History	38
5.2 Customer Functions	39
5.2.1 Orders	40
5.2.1.1 Add Order Item	40
5.2.1.2 Update Order Cart.....	41
5.2.1.3 Checkout Order.....	42
5.2.1.4 View and Search Order History	43
6.0 Extra Features	44
6.1 Registration.....	44
6.2 Log File Generation	45
7.0 Conclusion	46
References	47

1.0 Introduction

As the rapid improvement of technology has a very big effect on our society and even daily life, mechanisms and machines tend to also get more advanced. This project is mainly focusing on developing a management system which is related to business owners and the customer. This management system is to help the customers to summarise what is the product that they are buying, the quantity, and also the brand of the product. Not only that, the manager or admin of the business will also be able to monitor and manage the order placed by its various customers. By sorting out which customer is buying which product and the customer's address to be delivered after they have made the purchase.

The main goal of this management system is to help the customer and the business owner to sort out the list of customers, products and orders so that it will be convenient for the shop to deliver and easy for the customer to choose and place order on the product they need. Customers should be able to add, delete, edit, view and search the products they have chosen. Where the admin could do more such as editing the amount of the product, changing or deleting the order of the customers due to specified reasons and more. There will be a graphic user interface where the admin or the customer needs to log in. This is to ensure the privacy and also the security of each customer and the admin of the business.

1.1 Assumptions

- There is one and only admin account but will be used by many staffs.
- Every details input is correct and accurate.

2.0 Use Case Diagram

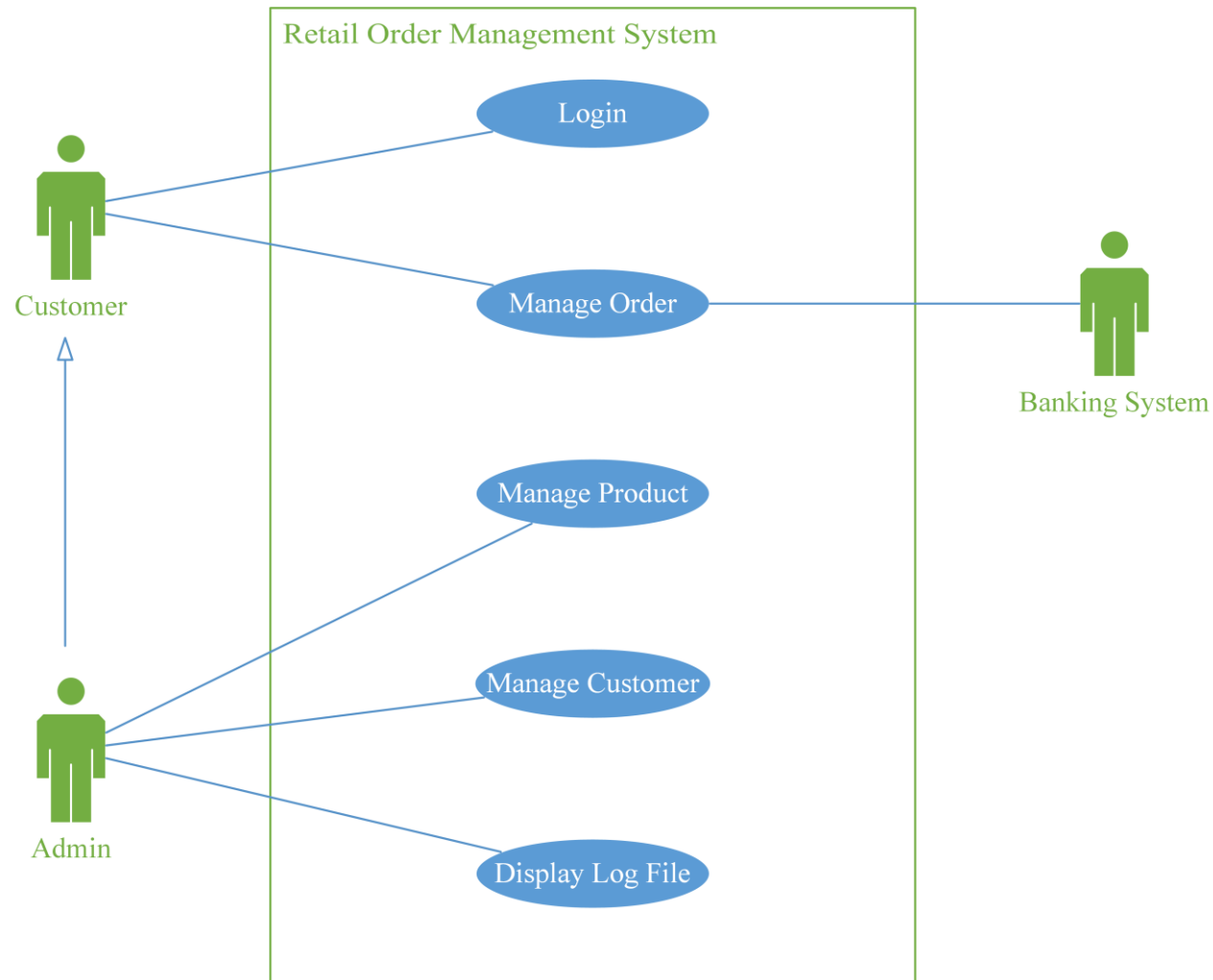


Figure 1: Overview Use Case

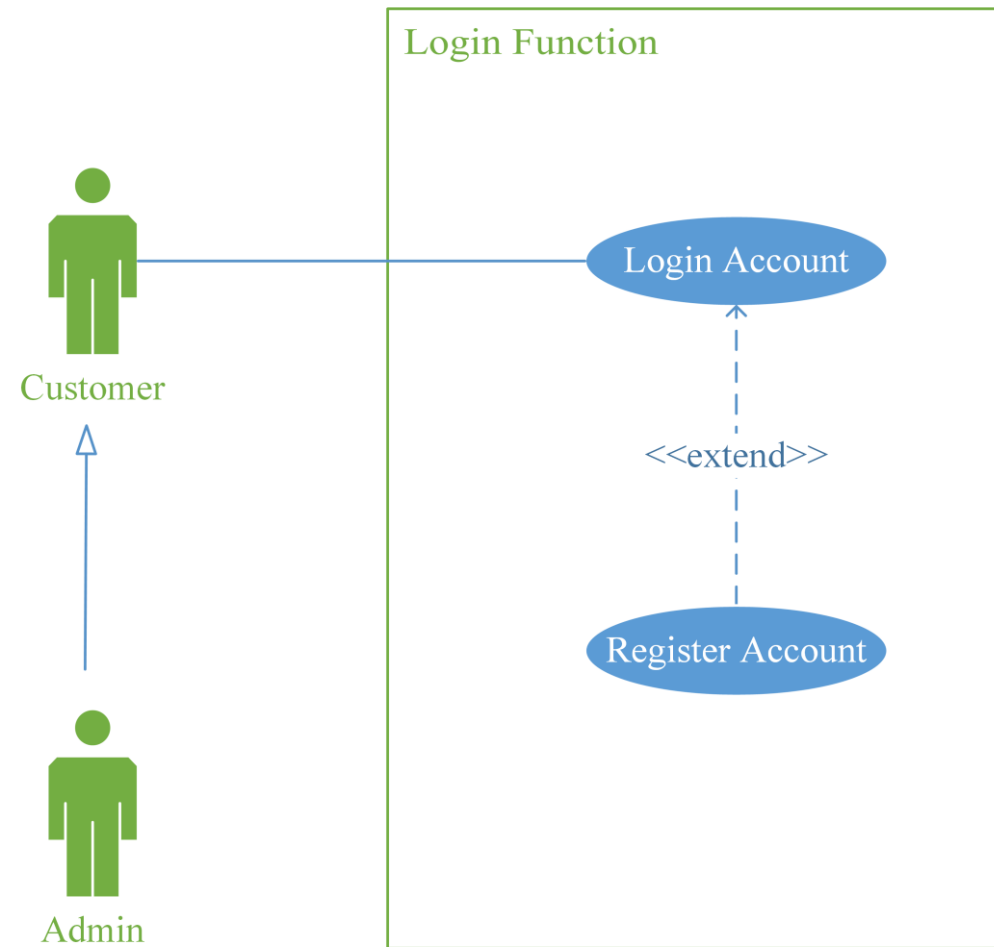


Figure 2: Login Use Case

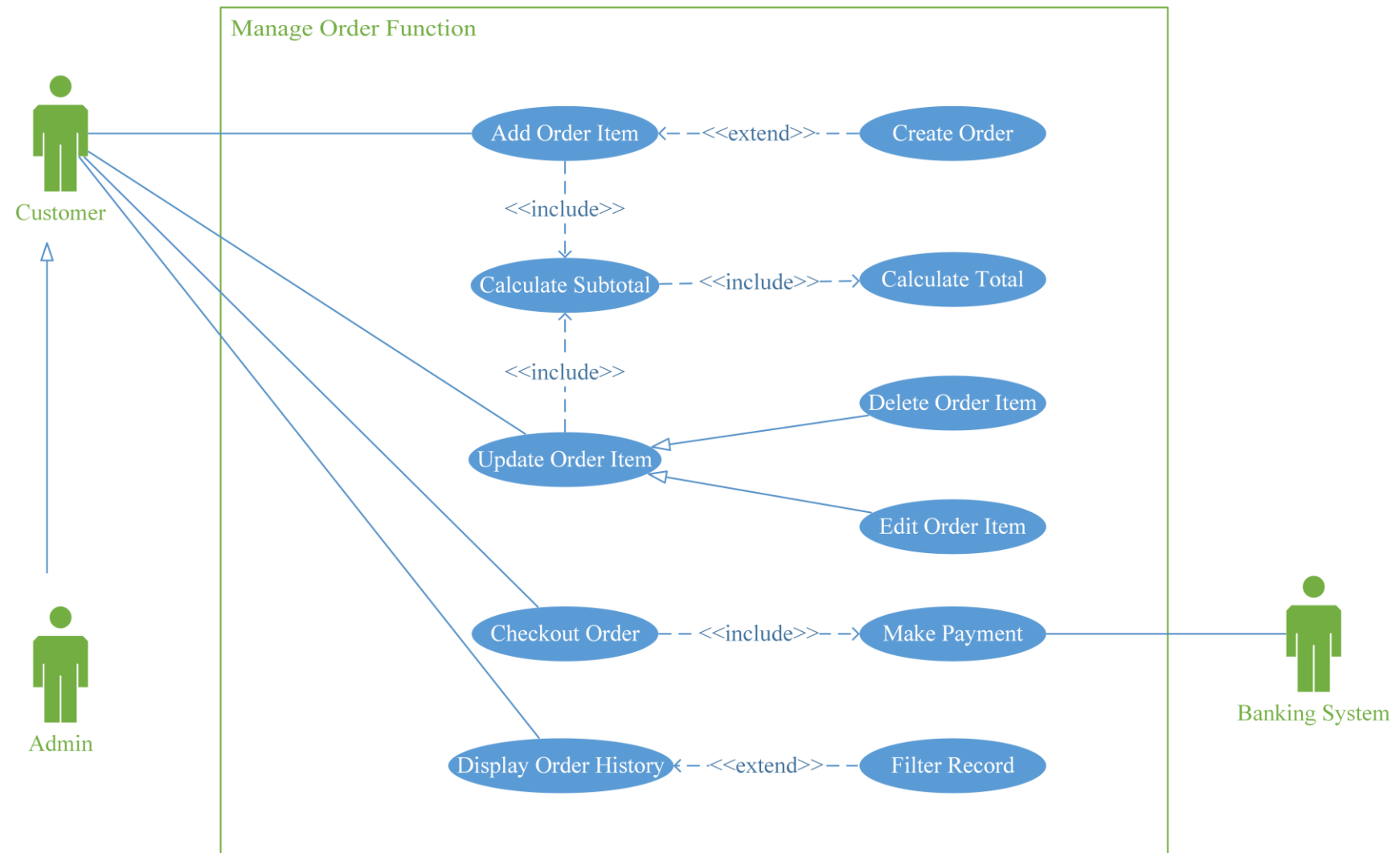


Figure 3: Order Use Case

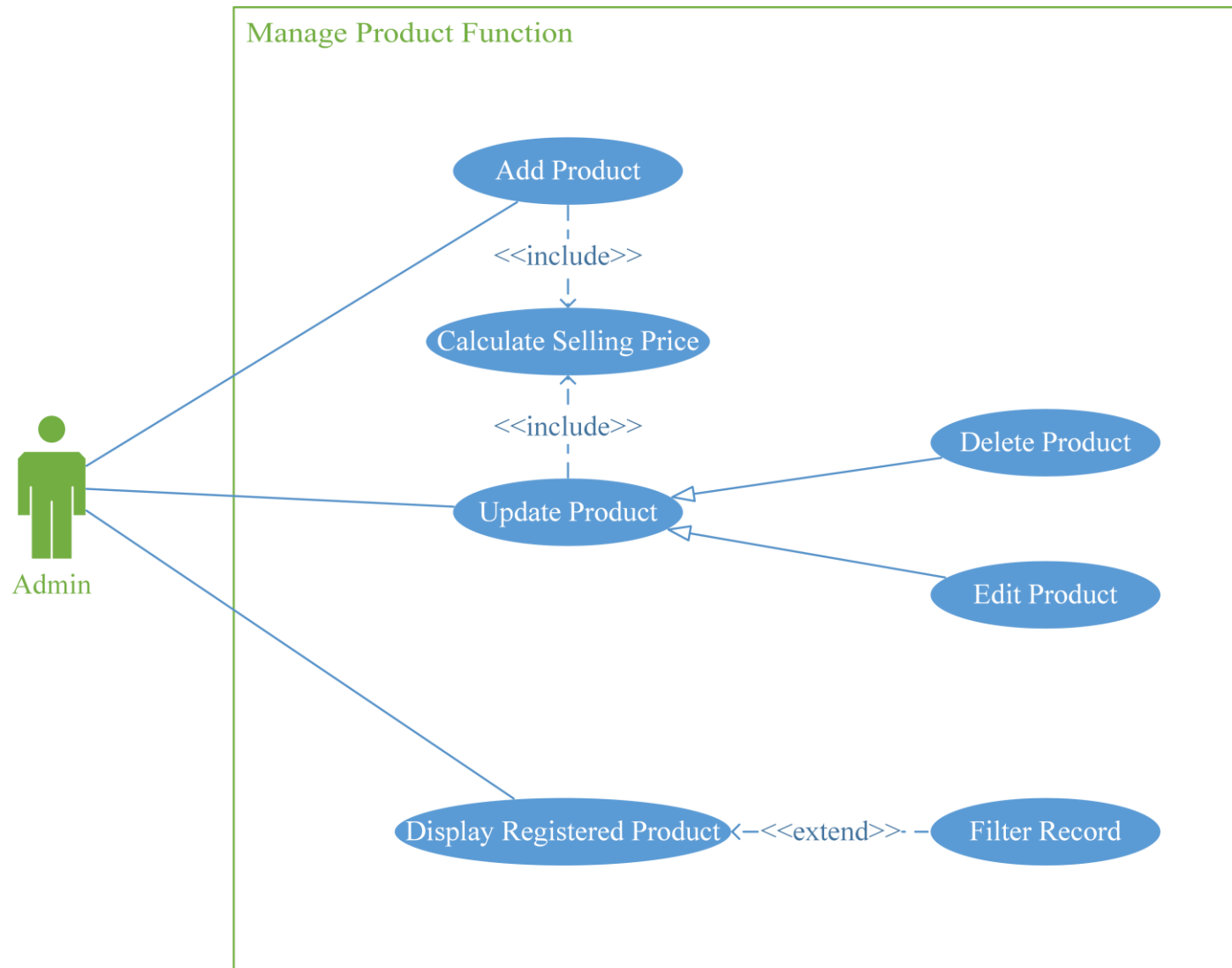


Figure 4: Product Use Case

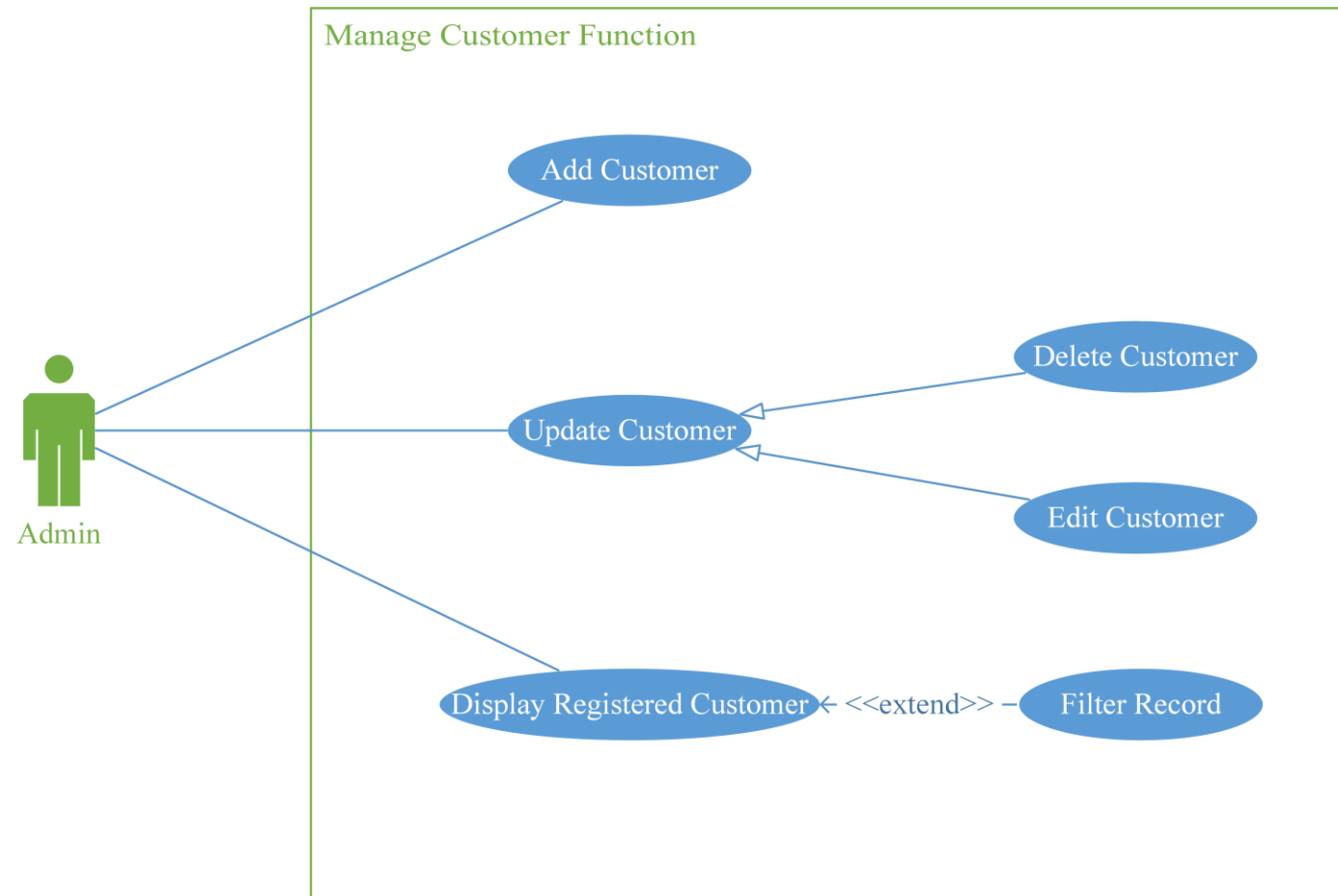


Figure 5: Customer Use Case

2.1 Use Case Specifications

Use Case	Login Account
Brief Description	This use case allows a user to login to their personal account.
Actors	Customer and Admin
Pre-Condition	User open the application and wanting to interact with the system.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when user boot up the application. b) The system waits for user to key in username and password. c) The validate the account and bring the user to main menu.
Alternative Flows	<ul style="list-style-type: none"> b) i) Customer forgot their username and password and contact to admin for help. c) i) Some of the fields are empty. ii) The system search through the text file and did not find a matching username and password.

Use Case	Register Account
Brief Description	This use case allows a user to register a customer role account.
Actors	Customer and Admin
Pre-Condition	User does not have an account to access to the system.
Post-Condition	The system saves the user details in the text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when user choose to register an account in login window. b) The system requests the user to fill in all the require personal details. c) The system check for account duplication before a new account is created.
Alternative Flows	<ul style="list-style-type: none"> b) i) Some of the fields are empty. c) i) The user already has one account under his/her name. ii) Another user already taken the username

Use Case	Add Order Item
Brief Description	This use case allows a user to add order item into their order.
Actors	Customer and Admin
Pre-Condition	User want to add a new item to the order.
Post-Condition	The system saves the item and quantity for the order.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when user click the button to add new order item. b) The system requests the user to choose which item to add. c) The system requests the user to decide the quantity of the chosen item. d) The system adds the item to the order.
Alternative Flows	<ul style="list-style-type: none"> b) i) Every product already added to the order. c) i) The user key in the quantity with a wrong format

Use Case	Create Order
Brief Description	This use case allows a user to create an empty order.
Actors	Customer and Admin
Pre-Condition	<ul style="list-style-type: none"> • The previous order is confirmed • No order has been created
Post-Condition	Order is created in the system for the user.
Main Flow	a) The use case begins when user wants to add order item to the order but do not have an order yet. b) The system will check whether the user is having an existing order or not. If not, the system will create an empty order for the user.
Alternative Flows	b) i) The user is having an order already.

Use Case	Calculate Subtotal
Brief Description	This use case allows a user to calculate subtotal for an item in the order.
Actors	Customer and Admin
Pre-Condition	The user wants to know how much subtotal for a specified amount of quantity.
Post-Condition	Subtotal will be recorded in text file if the item is added.
Main Flow	a) The use case begins when user added item to the order or before added to the order. b) The system will request the user to give a quantity. c) The system will display the subtotal after calculated it.
Alternative Flows	b) i) The user gives invalid quantity format.

Use Case	Calculate Total
Brief Description	This use case allows a user to calculate total of an order.
Actors	Customer and Admin
Pre-Condition	The user wants to know how much total for this order.
Main Flow	a) The use case begins whenever the item in the order updated. b) The system sums all the subtotal for every item added to the order. c) The system will display the order after calculated it.

Use Case	Delete Order Item
Brief Description	This use case allows a user to delete order item in an order.
Actors	Customer and Admin
Pre-Condition	The user added a wrong item into the order.
Post-Condition	Order Item is deleted in the text file for the specified order.
Main Flow	a) The use case begins when user is in update order item window. b) The user gives a 0 quantity to the order item. c) The system will remove the order item from the order. d) A new total is calculated for the order.

Use Case	Edit Order Item
Brief Description	This use case allows a user to edit order item quantity in an order.
Actors	Customer and Admin
Pre-Condition	The user typed in a wrong quantity for the item in an order.
Post-Condition	Everything is also updated followed by what changes on the window.
Main Flow	a) The use case begins when user is in update order item window. b) The user gives a new quantity to the order item. c) The system will update the subtotal and total.
Alternative Flows	b) i) The user gives invalid quantity format.

Use Case	Checkout Order
Brief Description	This use case allows a user to confirm the order.
Actors	Customer and Admin
Pre-Condition	The user has an order with item and confirmed everything.
Main Flow	a) The use case begins when user try to checkout the order in cart. b) The system will display total for the user to double confirm.

Use Case	Make Payment
Brief Description	This use case allows a user to make payment for the order
Actors	Customer, Admin and Banking System
Pre-Condition	The user is in checkout order window
Post-Condition	All the order details will be recorded in text file.
Main Flow	a) The use case begins when user confirm the order details. b) The system will request to choose a payment method. c) The user fills in the required information for a particular payment method.
Alternative Flows	c) i) Some of the fields is empty. ii) Customer forgot their card details and tried to contact bank for help.

Use Case	Display Order History
Brief Description	This use case allows a user to view order history.
Actors	Customer and Admin
Pre-Condition	<ul style="list-style-type: none"> Customer having at least one confirmed order. For Admin, At least one user has at least one confirmed order.
Main Flow	a) The user chooses to view order history in order menu. b) The system generates a simple list on a table.
Alternative Flows	b) i) Less than one confirmed order exist in the system. ii) Prompt user back to order menu.

Use Case	Add Product
Brief Description	This use case allows admin to add product to the system.
Actors	Admin
Pre-Condition	A new product come in stock for sale.
Post-Condition	All the new product details will be recorded in text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin enter the add product form. b) The system request admin to fill in all the product details.
Alternative Flows	<ul style="list-style-type: none"> b) i) Some of the fields are empty. ii) The original price format is wrong.

Use Case	Calculate Selling Price
Brief Description	This use case allows admin to calculate selling price for the product.
Actors	Admin
Pre-Condition	A new product is registering to the system.
Post-Condition	Selling price will be recorded following with the product details in text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin click the add button in add product form. b) The original price will be chuck in to the formula and calculate the selling price of the product.
Alternative Flows	<ul style="list-style-type: none"> b) i) The original price format is wrong.

Use Case	Delete Product
Brief Description	This use case allows admin to delete a registered product in the system.
Actors	Admin
Pre-Condition	A product is no longer going to sell in the shop.
Post-Condition	The product will also be removed from the text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin choose the option to delete a product. b) A product id is chosen by the admin. c) The admin clicks the delete button and the system remove the product from the system.

Use Case	Edit Product
Brief Description	This use case allows admin to edit the product details.
Actors	Admin
Pre-Condition	Some of the product details changed or typing error when registering it.
Post-Condition	Every detail is also updated on text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin choose the option to edit a product. b) A product id is chosen by the admin. c) The system request admin to give a new information for the product. d) The product details changed and update the selling price if needed.
Alternative Flows	<ul style="list-style-type: none"> c) i) Some of the fields are empty. ii) The original price format is wrong.

Use Case	Display Registered Product
Brief Description	This use case allows admin to view the registered product details.
Actors	Admin
Pre-Condition	At least one product registered in the system.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin choose the option to view product in product menu. b) All the registered product will be display on a table.
Alternative Flows	b) i) Less than one product registered in the system, so it will prompt admin back to product menu.

Use Case	Add Customer
Brief Description	This use case allows admin to add a customer to the system.
Actors	Admin
Pre-Condition	A new customer wants to order from the system and is having trouble to register an account.
Post-Condition	All the new customer details will be recorded in text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin enter the add customer form b) The system request admin to fill in all the customer details. c) The system adds the customer to the system.
Alternative Flows	b) i) Some of the fields are empty.

Use Case	Delete Customer
Brief Description	This use case allows admin to delete a registered customer in the system.
Actors	Admin
Pre-Condition	A customer having some mistyping for unchangeable field when registering.
Post-Condition	The customer will also be removed from the text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin choose the option to delete a customer. b) A customer username is chosen by the admin. c) The admin clicks the delete button and the system remove the customer from the system.

Use Case	Edit Customer
Brief Description	This use case allows admin to edit a customer detail.
Actors	Admin
Pre-Condition	Some of the customer details changed or typing error when registering it.
Post-Condition	Every detail is also updated on text file.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin choose the option to edit a customer. b) A customer username is chosen by the admin. c) The system request admin to give a new information for the customer. d) The customer details changed.
Alternative Flows	c) i) Some of the fields are empty.

Use Case	Display Registered Customer
Brief Description	This use case allows admin to view registered customer details.
Actors	Admin
Pre-Condition	At least one customer registered in the system.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when admin choose the option to view customer in customer menu. b) All the registered customer will be display on a table.
Alternative Flows	b) i) Less than one customer registered in the system, so it will prompt admin back to customer menu.

Use Case	Filter Record
Brief Description	This use case allows user to search specified information by filtering the records.
Actors	Customer and Admin
Pre-Condition	There is at least one row of record in the system.
Main Flow	<ul style="list-style-type: none"> a) The use case begins when a table is generated by the user. b) The system request user to give some filter to the records. c) The filtered records will be display again on the table.

3.0 Class Diagram

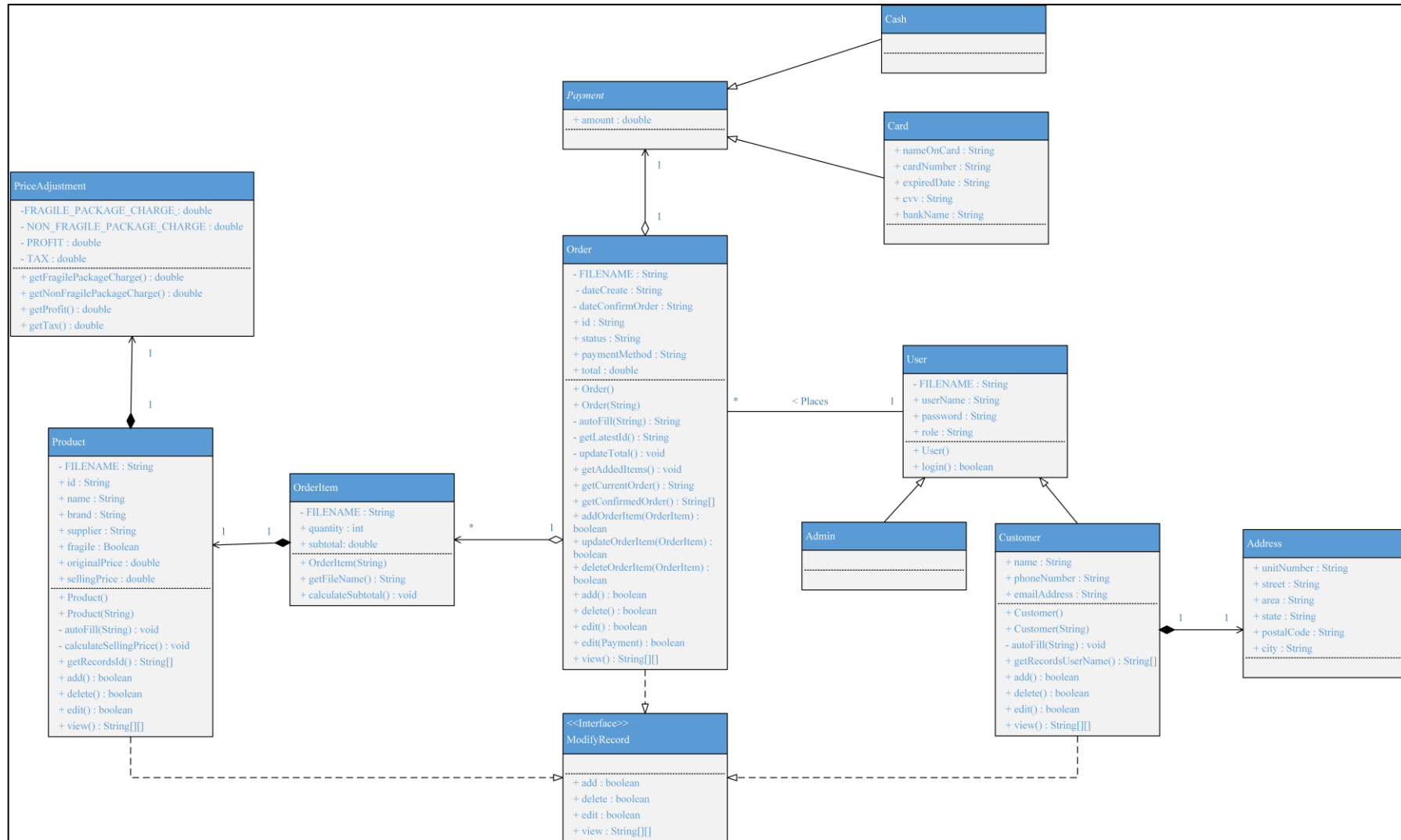


Figure 6: Class Diagram

4.0 Class Concepts

4.1 Inheritance

Inheritance is a mechanism in which one object inherits all the behaviour and properties of the parent object (JavaTpoint, 2021). Or in other words, it means that a class is able to inherit the fields and methods of another class. There are 3 types of inheritance. First type is single inheritance, it means 1 parent class with 1 child class. Second type is multilevel inheritance, multiple inheritance is where a class can inherit from a child class. Hence, the child class will become the base class for the new class. Thirdly is hierarchical inheritance, it means one class that has many sub classes (Guru99, 2021). The common terms used in inheritance are class, sub class, super or parent class. Parent or super class is the base class. It is a class where a subclass or child class inherits the methods or features. While subclass or child class is the inheritance of parent class (Neha, 2021).

There are two parts in the implementation where used inheritance concept which is user and payment act as a parent class. The reason why they are having child classes is because some of the classes will be having duplicated or repeated attributes and methods, hence those classes will be child classes extend from a parent class instead. In details, the generalisation relationship is shown as figure below which also known as inheritance.

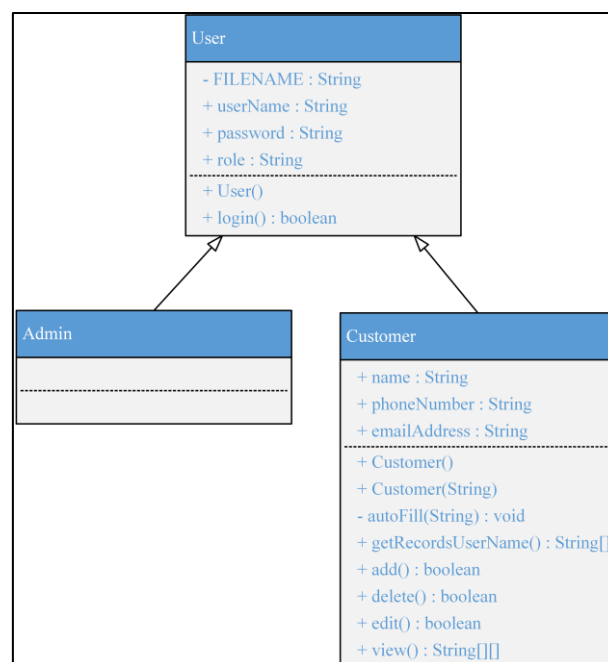


Figure 7: User Class Diagram

From the figure, notice that there are some attributes and method from User class which will also be passing to child classes because Admin and Customer both must have user name, password and role, while they all saved in the same text file. Besides that, they both also must have a login method. Therefore, inheritance will be useful to related all of them together and reduce the code duplication. As for the java code implementation, extends keyword is used by both Admin and Customer class to be able to receive the attributes and method from User as shown below.

```
public class Customer extends User
```

Figure 8: Customer Extends

```
public class Admin extends User
```

Figure 9: Admin Extends

The same case goes for Payment class but Payment is an abstract class and that part will be covering in later section. In details, the generalisation relationship of Payment is shown as figure below which also known as inheritance.

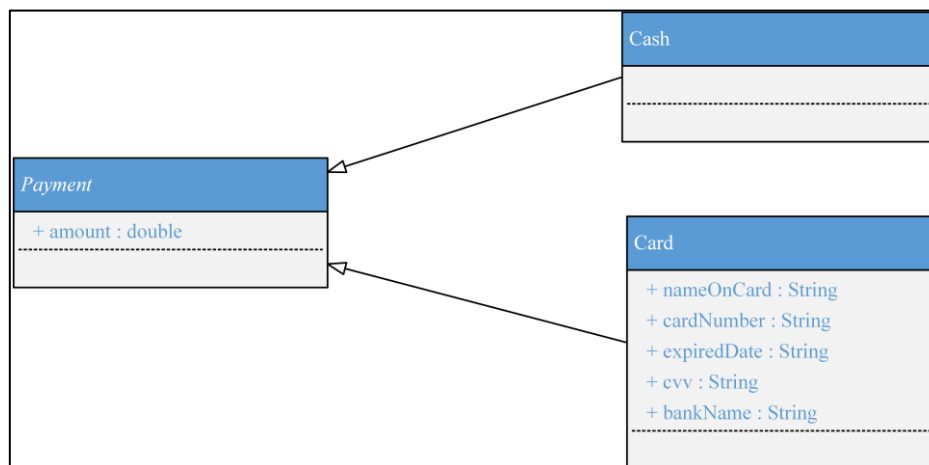


Figure 10: Payment Class Diagram

The parent class for this is much simpler due to the depth of this project because the system is not trying to connect to a bank server, so most of the method is not implemented yet but having this structure for the payment of order increase the reusability of the program and also any future features added can be easily implemented into the system. As for the java code implementation, extends keyword is used by both Cash and Card class to be able to receive the amount attribute from Payment class.

4.2 Polymorphism

Polymorphism occurs when there are one or more classes or objects that are related to each other by inheritance. It is the ability to allow the object to take many forms. In other words, polymorphism is having the same result but with different procedures (Guru99, 2021). Polymorphism allows one interface but with different implementations. Such as having the phone to make a sound when there are notifications. But making a different sound when the notification is from 2 different sources (SINGH, 2014). So, in conclusion, polymorphism is where a subclass can define their own unique function while sharing the functionality of the parent class (Great Learning Team, 2021).

There are a few parts of system used polymorphism, first of all, method overloading is used on some class constructor such as Product class have two constructors with different parameter as shown below.

```
// Constructors
public Product() {
    priceAdjustment = new PriceAdjustment();
}

public Product(String id) {
    priceAdjustment = new PriceAdjustment();
    autoFill(id);
}
```

Figure 11: Product Constructor

From figure above, notice that one of the constructors is having a String parameter. The point of it is there are two way to instantiate the Product class and the compiler will identify which constructor to use depends on what parameter is given when initialising it. Therefore, we can determine whether the execution of autofill() function is necessary of that instantiate or not by constructing it in different way which emphasising polymorphism concept of a same thing but different implementations. As for the java code implementation, constructor with String parameter is used in the situation of display product details which do not required to call again autofill() function. While when adding a new product to the system, obviously the product does not have details yet. Hence the constructor without parameter is used as shown below.

```
// Display Product details in text box
Product product = new Product(id);
```

Figure 12: Instantiate Product

```
// Instantiate product
Product product = new Product();
```

Figure 13: Instantiate Product

Besides that, object casting is also be used in this system for Payment and its sub classes. This technique is being used because there is only Payment class as an attribute in Order class as shown below.

```
public class Order implements ModifyRecord{

    // private attributes
    private final String fileName = "Order.txt";
    private String dateCreate, dateConfirmOrder;

    // Attributes
    public String id, status, paymentMethod;
    public double total;
    public User orderBy;
    public OrderItem[] orderItems;
    public Payment payment;
```

Figure 14: Order Attributes

From the figure, notice that Cash and Card class is not include in them, it is because they might not be used and also might cause code duplication. Instead, polymorphism will be come in handy to object casting between them. For example, when checking out an order, a Cash or Card object will be instantiated depends on chosen payment method. After all the required details are filled, then the object will be upcasted to be a Payment class by calling edit() function as shown below. When upcasting, the data stored in the object will not be deleted but the accessible function will be restricted.

```

if(rdoCash.isSelected()) {
    order.paymentMethod = "Cash";
    Cash cash = new Cash();
    cash.amount = order.total;
    edit = order.edit(cash);
}
else {
    order.paymentMethod = "Card";
    Card card = new Card();
    card.amount = order.total;

    if(!txtNameOnCard.getText().equals("") & !txtCardNumber.getText().equals("") & !txtExpiredDate.getText().equals("") &
        !txtCvv.getText().equals("") & !txtBankName.getText().equals("")){
        card.nameOnCard = txtNameOnCard.getText();
        card.cardNumber = txtCardNumber.getText();
        card.expiredDate = txtExpiredDate.getText();
        card.cvv = txtCvv.getText();
        card.bankName = txtBankName.getText();
        edit = order.edit(card);
    }
    else {
        JOptionPane.showMessageDialog(this, "Empty fields existed!");
        return;
    }
}
}

```

Figure 15: Object Casting

From this part, we still cannot see the benefit from doing this but not when doing it in the opposite way where order details need to be display when in view order history window. As mentioned, the data stored will not deleted, so with that data we can actually check if the Payment is an instance of one of the class then we can object casting it and do a different procedure as shown below. Therefore, we success to use Cash and Card class when Order class has only Payment class by applying polymorphism concept.

```

if (Card.class.isInstance(order.payment)) {
    txtPaymentMethod.setText("Card");
    Card card = (Card) order.payment;
    txtAmount.setText(Double.toString(card.amount));
    nameOnCard = card.nameOnCard;
    cardNumber = card.cardNumber;
    expiredDate = card.expiredDate;
    cvv = card.cvv;
    bankName = card.bankName;
} else if (Cash.class.isInstance(order.payment)) {
    txtPaymentMethod.setText("Cash");
    Cash cash = (Cash) order.payment;
    txtAmount.setText(Double.toString(cash.amount));
}
}

```

Figure 16: Object Casting

4.3 Abstraction

Abstraction is one of the concepts of object-oriented programming that only displays the useful and needed attributes and hides the needless or irrelevant information (Nick, 2021). Abstraction is more like an interface where the user only needs to know what to input and has no idea on how the procedures, coding and back-end work is going on. It is a method of object where users are able to call and then activate the input parameters (JANSSEN, 2017). Abstraction is used to define a boundary for the viewers. So that the users only see and trigger the required input on the interface to allow the back-end mechanism works.

In java, there are two technique including interface and abstract class which can shows the abstraction concept, both of them are implemented in this system. A class can be abstract when given a abstract keyword but there are still difference between them such as some method implementation may exists in abstract class which is not allowed in interface. However, they both cannot be instantiated themselves only, they must have a child class or a class to realise the methods and create an instance. For ModifyRecord, interface is used because the method is clearly straight forward and does not require any method implementation within, different classes such as Product, Customer and Order will have different implementation for the abstract methods within by overriding it as shown below.

```
public interface ModifyRecord {

    // Abstract methods
    boolean add();
    boolean delete();
    boolean edit();
    String[][] view();

}
```

Figure 17: Interface Methods

However, that is not the case for Payment class because there might have some method implementation applied within Payment class in the future if a balance and banking system are addon to this system. Therefore, Payment is created as an abstract class which have child class of Cash and Card.

```
public abstract class Payment {

    // Attributes
    public double amount;

}
```

Figure 18: Abstract class Methods

4.4 Encapsulation

Encapsulation is the mechanism of concluding the data and code acting on the methods together as a single unit. It also hides the values of the structured data to prevent the unauthorized or unwanted person from sabotaging the coding (Braunschweig, nd). With encapsulation, only the methods in the class are exposed, and the programmer does not need to worry about the application but just has to deal with the operation (Nick, 2021).

Therefore, A complete encapsulated class is created named PriceAdjustment by making all the attributes private. All of the attribute only can access through getters methods with public modifier, so that user cannot directly see what is the calculation of selling price is done at the backside of the system and they are not changeable because this encapsulated class is designed for read only. Hence the PriceAdjustment class is created as shown below.

```
public class PriceAdjustment {  
  
    // Encapsulated Class (Read Only)  
    // private fixed attribute  
    private final double fragilePackageCharge = 10;  
    private final double nonFragilePackageCharge = 5;  
    private final double profit = 0.5;  
    private final double tax = 0.06;  
  
    // Getters  
    public double getFragilePackageCharge() {  
        return fragilePackageCharge;  
    }  
  
    public double getNonFragilePackageCharge() {  
        return nonFragilePackageCharge;  
    }  
  
    public double getProfit() {  
        return profit;  
    }  
  
    public double getTax() {  
        return tax;  
    }  
}
```

Figure 19: Price Adjustment Class

Besides that, some of the class also used encapsulation concept but it is not completely encapsulated such OrderItem class have filename attribute which is private but since Order class will sometimes uses the file therefore a getter method is created to make sure the filename will not be changed while all the function is still met. The OrderItem class is created as shown below.

```
public class OrderItem {  
  
    // private fixed attributes  
    private String fileName = "OrderItem.txt";  
  
    // attributes  
    public Product item;  
    public int quantity;  
    public double subtotal;  
  
    // Constructor  
    public OrderItem(String id) {  
        item = new Product(id);  
    }  
  
    // Getter  
    public String getFileName() {  
        return fileName;  
    }  
  
    // Methods  
    public void calculateSubtotal() {  
        subtotal = item.sellingPrice * quantity;  
        subtotal = (double) Math.round(subtotal*10) / 10;  
    }  
}
```

Figure 20: Order Item Class

5.0 Main Functions Showcase

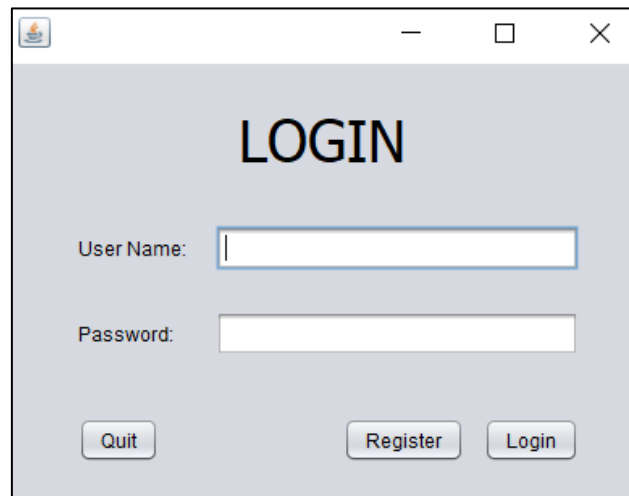
A screenshot of a web application window titled "LOGIN". The window has a light gray background and a white title bar with standard minimize, maximize, and close buttons. The main content area features the word "LOGIN" in large, bold, black capital letters. Below the title, there are two input fields: "User Name:" followed by a white text box with a blue border, and "Password:" followed by a white text box with a blue border. At the bottom of the window, there are three buttons: "Quit", "Register", and "Login", all with a light gray background and a thin blue border.

Figure 21: Login

Since there will be two different user access to the system with different accessibility of functionalities, there is a login function created to identify the role of the user and also keep the personal details private from each customer.

5.1 Admin Functions

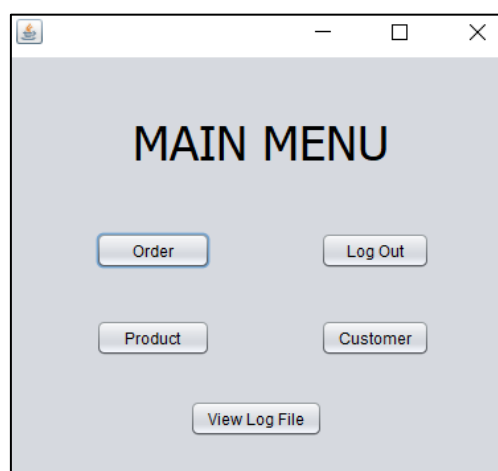
A screenshot of a web application window titled "MAIN MENU". The window has a light gray background and a white title bar with standard minimize, maximize, and close buttons. The main content area features the words "MAIN MENU" in large, bold, black capital letters. Below the title, there are five buttons arranged in a grid: "Order" and "Log Out" in the top row, "Product" and "Customer" in the middle row, and "View Log File" centered at the bottom. All buttons have a light gray background and a thin blue border.

Figure 22: Main Menu

First of all, admin user will be able to access all the functionalities and also some admin function, therefore, those admin only function button will be visible in the main menu if an admin account is login.

5.1.1 Products

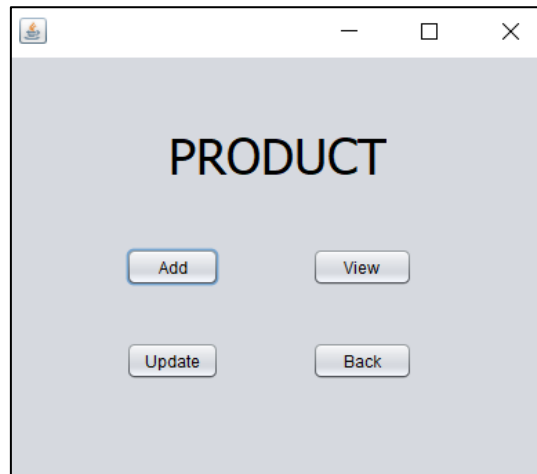
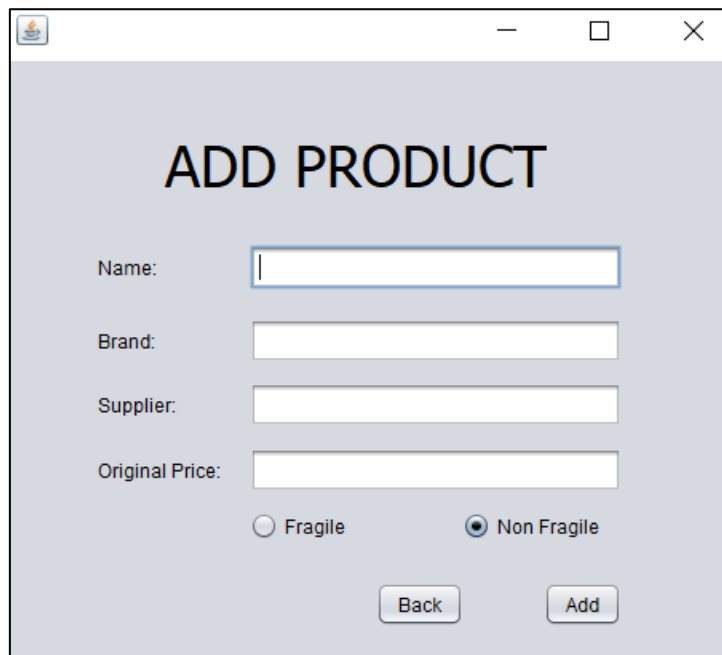


Figure 23: Product Menu

This is one of the admins only functionalities. After clicking the product button on the main menu, it will bring admin to the product menu which showing all the function available related to products within the system.

5.1.1.1 Add Product



ADD PRODUCT

Name:

Brand:

Supplier:

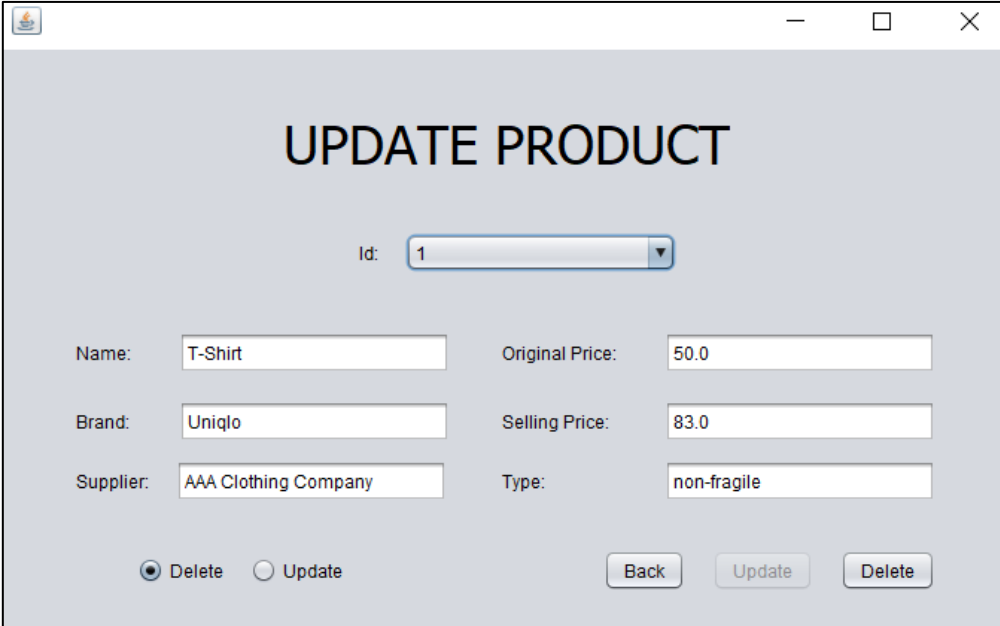
Original Price:

☐ Fragile ☒ Non Fragile

Figure 24: Add Product

For add product window, the main purpose of it is to register a new product to the system by writing the details into a text file without duplication of product. In term of duplication, only product with same name and brand will be considered as duplication because a same product will be produced from different brand. When pressing the add button, the execution will only be done when all fields are filled and there is a validation for original price. If the process is success a message window will prompt out, while when fail, a reason will also be given in the message window.

5.1.1.2 Delete Product



The screenshot shows a window titled "UPDATE PRODUCT" with a standard Windows-style title bar (minimize, maximize, close buttons). The window has a light gray background. At the top center, the title "UPDATE PRODUCT" is displayed in a large, bold, black font. Below the title, there is a label "Id:" followed by a dropdown menu showing the value "1". Below this, there are six text input fields arranged in two columns. The left column contains "Name:" (T-Shirt), "Brand:" (Uniqlo), and "Supplier:" (AAA Clothing Company). The right column contains "Original Price:" (50.0), "Selling Price:" (83.0), and "Type:" (non-fragile). At the bottom left, there are two radio buttons: "Delete" (which is selected) and "Update". At the bottom right, there are three buttons: "Back", "Update", and "Delete".

Field	Value
Id	1
Name	T-Shirt
Brand	Uniqlo
Supplier	AAA Clothing Company
Original Price	50.0
Selling Price	83.0
Type	non-fragile

Mode: ☒ Delete ☐ Update

Buttons: Back, Update, Delete

Figure 25: Delete Product

For delete product function, it will be using the same window with edit product by changing the mode with radio button at the bottom left of the window, switching between mode will change the button enabled properties. Only existing product id will be appeared in the combo box and it allow user to choose id from it for further operation. The main purpose of it is to remove product which already stop selling in the retail shop or if the name or brand typed wrongly when register. If the process is success a message window will prompt out, while when fail, a notice will also be given in the message window.

5.1.1.3 Edit Product

UPDATE PRODUCT

Id: 1

Name: T-Shirt

Brand: Uniqlo

*Supplier: AAA Clothing Company

*Original Price: 50.0

Selling Price: 83.0

Type: non-fragile

☐ Delete ☒ Update

Back Update Delete

Figure 26: Edit Product

This is edit product function. As mentioned, delete and edit are sharing the same window but after the update radio button is checked, delete button will be disabled while supplier and original price text box will be editable for the user. When pressing the update button, the execution will only be done when all fields are filled and there is a validation for original price. If the process is successful a message window will prompt out, while when fail, a reason will also be given in the message window.

5.1.1.4 View Product

View Product

Filter

Id: Supplier:

Name: Original Price:

Brand: Selling Price:

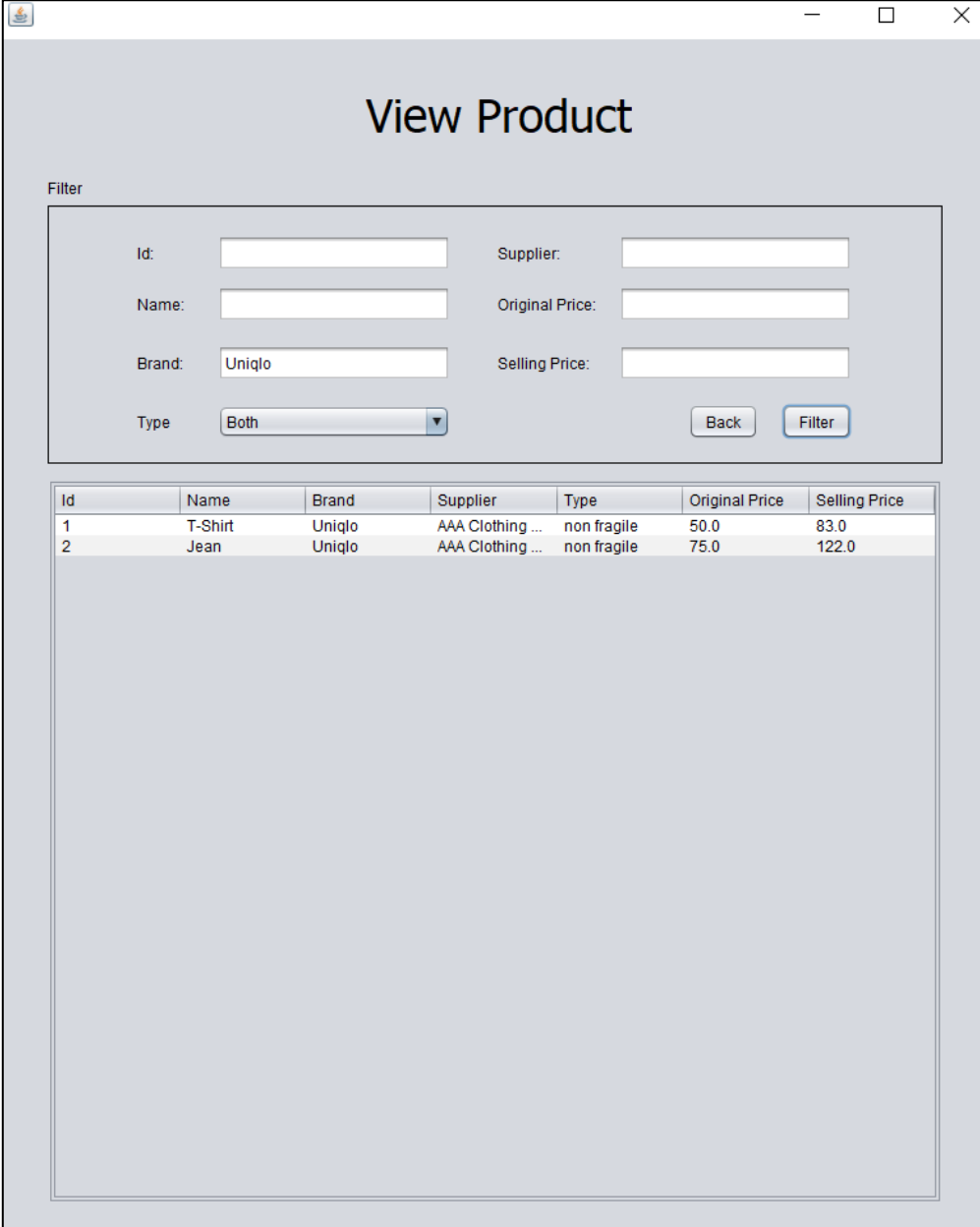
Type:

Id	Name	Brand	Supplier	Type	Original Price	Selling Price
1	T-Shirt	Uniqlo	AAA Clothing ...	non fragile	50.0	83.0
2	Jean	Uniqlo	AAA Clothing ...	non fragile	75.0	122.0
3	Vase	Mr DIY	Clay Company	fragile	15.0	33.4
4	Plate	Ikea	Dining Set Co...	fragile	20.0	41.2
5	Potato Chip	Pringles	Snack Compa...	non fragile	5.0	12.8

Figure 27: View Product

For view product function, it will be using the same window with search product. All the product and details will be showing in the table when opening this window automatically without pressing other button, if the product is too many, there will automatically form a scrollbar for the table, while every column can be resizing and reposition for better information emphasising. The main purpose of it is to check overall product registered in the system.

5.1.1.5 Search Product



The screenshot shows a window titled "View Product". Inside, there is a "Filter" section with the following fields:

- Id:** [Empty text box]
- Supplier:** [Empty text box]
- Name:** [Empty text box]
- Original Price:** [Empty text box]
- Brand:** [Text box containing "Uniqlo"]
- Selling Price:** [Empty text box]
- Type:** [Dropdown menu showing "Both"]

Below the filter section are two buttons: "Back" and "Filter". Below the buttons is a table with the following data:

Id	Name	Brand	Supplier	Type	Original Price	Selling Price
1	T-Shirt	Uniqlo	AAA Clothing ...	non fragile	50.0	83.0
2	Jean	Uniqlo	AAA Clothing ...	non fragile	75.0	122.0

Figure 28: Search Product

For search product function, it can be achieved by giving the filters to the empty textbox while using the same window as view product. Any number of filters can be applied to get the search result to make more precise searching. All the product and details will be showing in the table when opening this window after pressing filter button, if the product is too many, there will automatically form a scrollbar for the table, while every column can be resizing and reposition for better information emphasising. The main purpose of it is to retrieve a particular product detail.

5.1.2 Customers

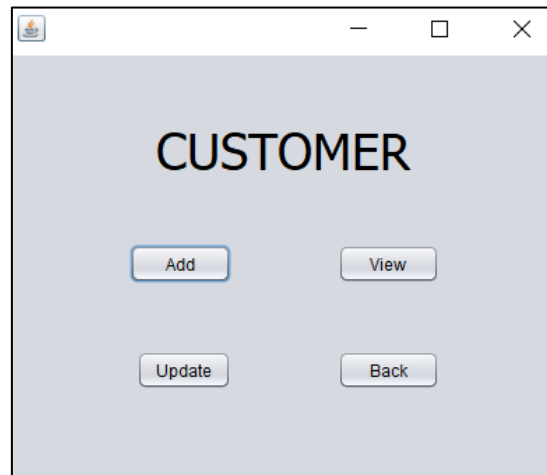
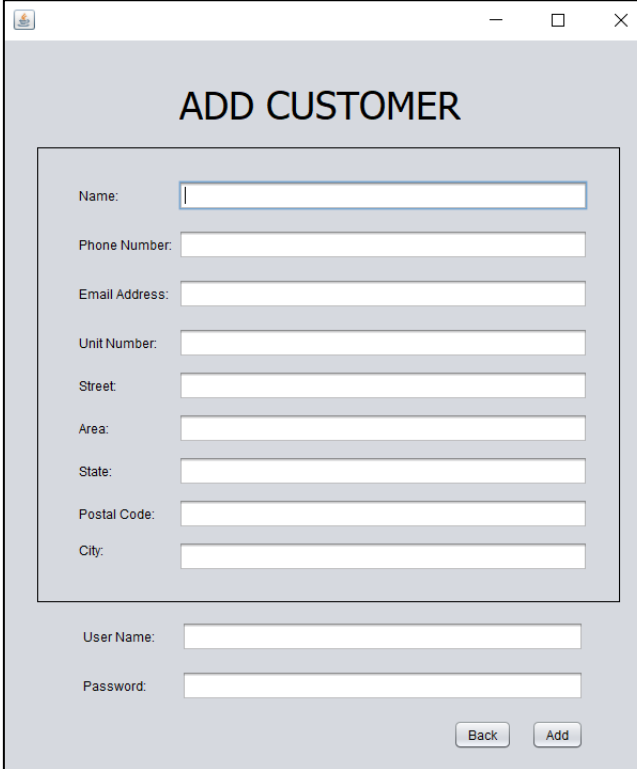


Figure 29: Customer Menu

This is one of the admins only functionalities. After clicking the customer button on the main menu, it will bring admin to the customer menu which showing all the function available related to products within the system.

5.1.2.1 Add Customer



The screenshot shows a window titled "ADD CUSTOMER". Inside the window, there is a form with the following fields:

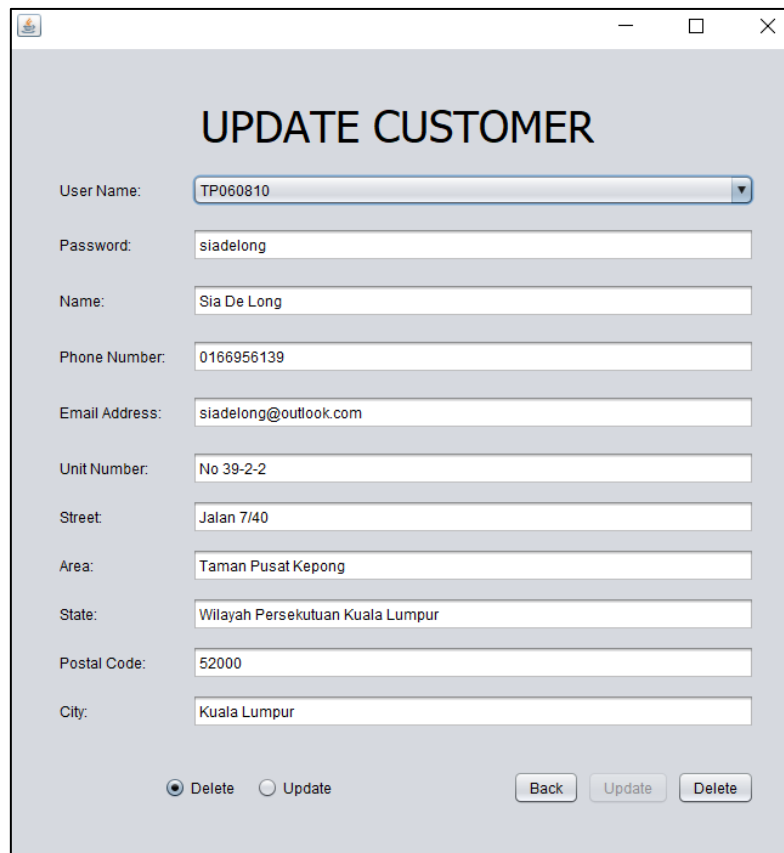
- Name:
- Phone Number:
- Email Address:
- Unit Number:
- Street:
- Area:
- State:
- Postal Code:
- City:
- User Name:
- Password:

At the bottom right of the form, there are two buttons: "Back" and "Add".

Figure 30: Add Customer

For add customer window, the main purpose of it is to register a new product to the system by writing the details into a text file without duplication of customer. In term of duplication, one customer only can register one account in this system else everything other than that will be treat as duplication. When pressing the add button, the execution will only be done when all fields are fille. If the process is success a message window will prompt out, while when fail, an empty field existed message will also be given in the message window.

5.1.2.2 Delete Customer



The screenshot shows a window titled "UPDATE CUSTOMER" with a light gray background. The window contains a form with the following fields and values:

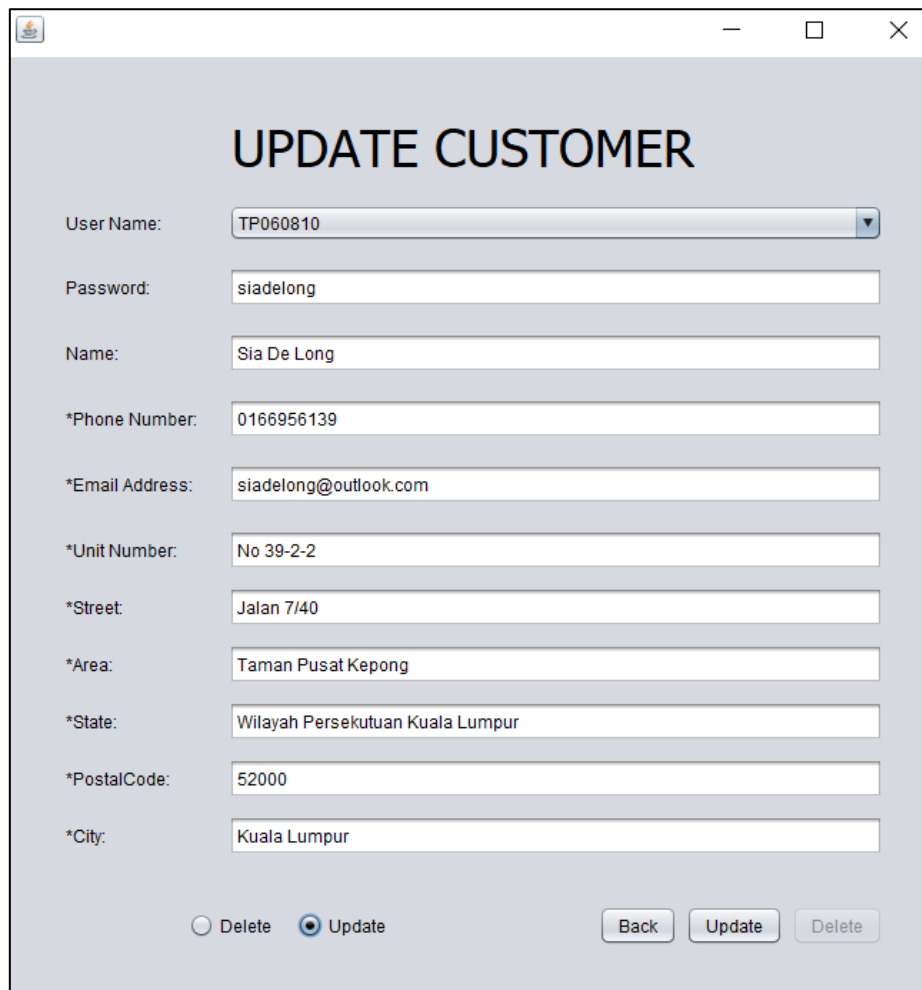
Field	Value
User Name:	TP060810
Password:	siadelong
Name:	Sia De Long
Phone Number:	0166956139
Email Address:	siadelong@outlook.com
Unit Number:	No 39-2-2
Street:	Jalan 7/40
Area:	Taman Pusat Kepong
State:	Wilayah Persekutuan Kuala Lumpur
Postal Code:	52000
City:	Kuala Lumpur

At the bottom left, there are two radio buttons: "Delete" (selected) and "Update". At the bottom right, there are three buttons: "Back", "Update", and "Delete".

Figure 31: Delete Customer

For delete customer function, it will be using the same window with edit customer by changing the mode with radio button at the bottom left of the window, switching between mode will change the button enabled properties. Only existing customer user name will be appeared in the combo box and it allow user to choose id from it for further operation. The main purpose of it is to remove customer for any possible reason. If the process is success a message window will prompt out, while when fail, a notice will also be given in the message window.

5.1.2.3 Edit Customer



The screenshot shows a software window titled "UPDATE CUSTOMER". It contains a form with the following fields and values:

Field Label	Value
User Name:	TP060810
Password:	siadelong
Name:	Sia De Long
*Phone Number:	0166956139
*Email Address:	siadelong@outlook.com
*Unit Number:	No 39-2-2
*Street:	Jalan 7/40
*Area:	Taman Pusat Kepong
*State:	Wilayah Persekutuan Kuala Lumpur
*PostalCode:	52000
*City:	Kuala Lumpur

At the bottom of the form, there are two radio buttons: "Delete" (unchecked) and "Update" (checked). To the right of these are three buttons: "Back", "Update", and "Delete".

Figure 32: Edit Customer

This is edit customer function. As mentioned, delete and edit are sharing the same window but after the update radio button is checked, delete button will be disabled while phone number, email address, unit number, street, area, state, postal code and city text box will be editable for the user. When pressing the update button, the execution will only be done when all fields are filled. If the process is successful a message window will prompt out, while when fail, a reason will also be given in the message window.

5.1.2.4 View Customer

View Customer

Filter

User Name:

Password:

Name:

Phone Number:

Email Address:

Unit Number:

Street:

Area:

State:

Postal Code:

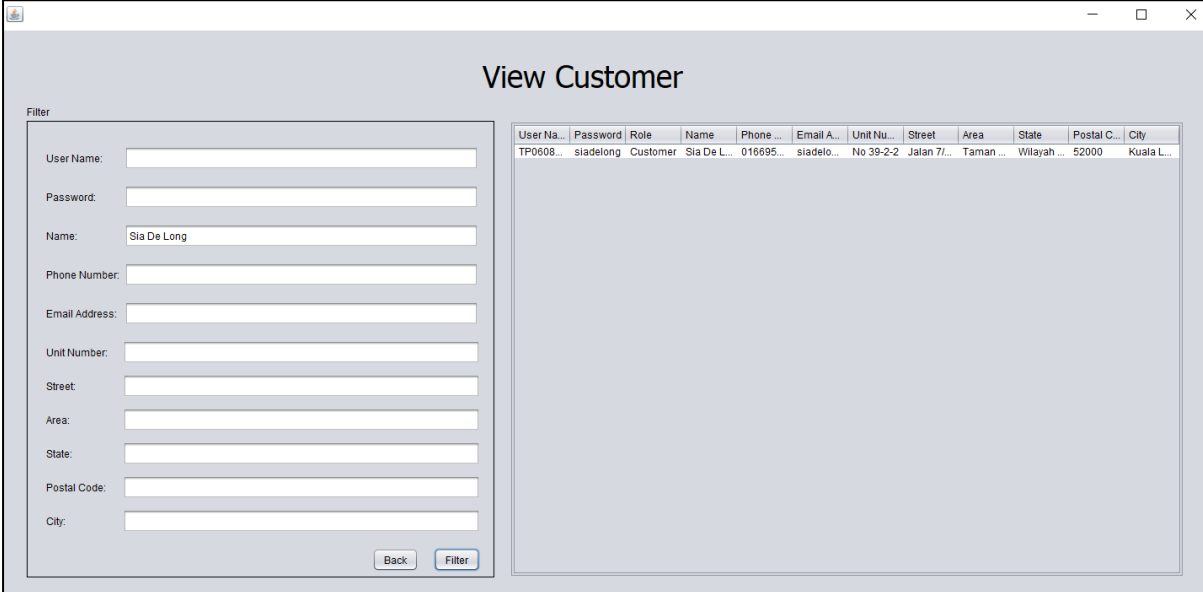
City:

User Na...	Password	Role	Name	Phone ...	Email A...	Unit Nu...	Street	Area	State	Postal C...	City
TP0608...	siadelong	Customer	Sia De L...	016695...	siadelo...	No 39-2-2	Jalan 71...	Taman ...	Wilayah ...	52000	Kuala L...
TP0614...	emerson	Customer	Emerso...	016358...	emerso...	No 8	Jalan 39	Taman ...	Wilayah ...	51200	Kuala L...

Figure 33: View Customer

For view customer function, it will be using the same window with search product. All the customer and details will be showing in the table when opening this window automatically without pressing other button, if the customer is too many, there will automatically form a scrollbar for the table, while every column can be resizing and reposition for better information emphasising. The main purpose of it is to check overall customer registered in the system.

5.1.2.5 Search Customer



The screenshot shows a window titled "View Customer". On the left, there is a "Filter" section with various input fields for searching customers. On the right, there is a table displaying customer information. The "Filter" section includes fields for User Name, Password, Name (pre-filled with "Sia De Long"), Phone Number, Email Address, Unit Number, Street, Area, State, Postal Code, and City. Below these fields are "Back" and "Filter" buttons. The table on the right has columns for User Na..., Password, Role, Name, Phone..., Email A..., Unit Nu..., Street, Area, State, Postal C..., and City. The first row of data shows a customer with User ID TP0608, Password siadelong, Role Customer, Name Sia De L., Phone 016695..., Email siadelo..., Unit No 39-2-2, Street Jalan 7/, Area Taman..., State Wilayah..., Postal Code 52000, and City Kuala L...

User Na...	Password	Role	Name	Phone ...	Email A...	Unit Nu...	Street	Area	State	Postal C...	City
TP0608...	siadelong	Customer	Sia De L...	016695...	siadelo...	No 39-2-2	Jalan 7/...	Taman ...	Wilayah ...	52000	Kuala L...

Figure 34: Search Customer

For search customer function, it can be achieved by giving the filters to the empty textbox while using the same window as view customer. Any number of filters can be applied to get the search result to make more precise searching. All the customer and details will be showing in the table when opening this window after pressing filter button, if the customer is too many, there will automatically form a scrollbar for the table, while every column can be resizing and reposition for better information emphasising. The main purpose of it is to retrieve a particular product detail.

5.1.3 Orders

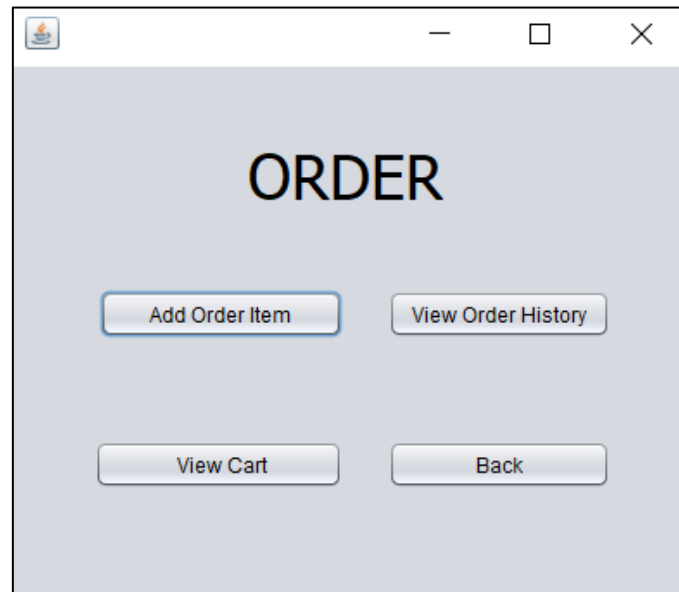
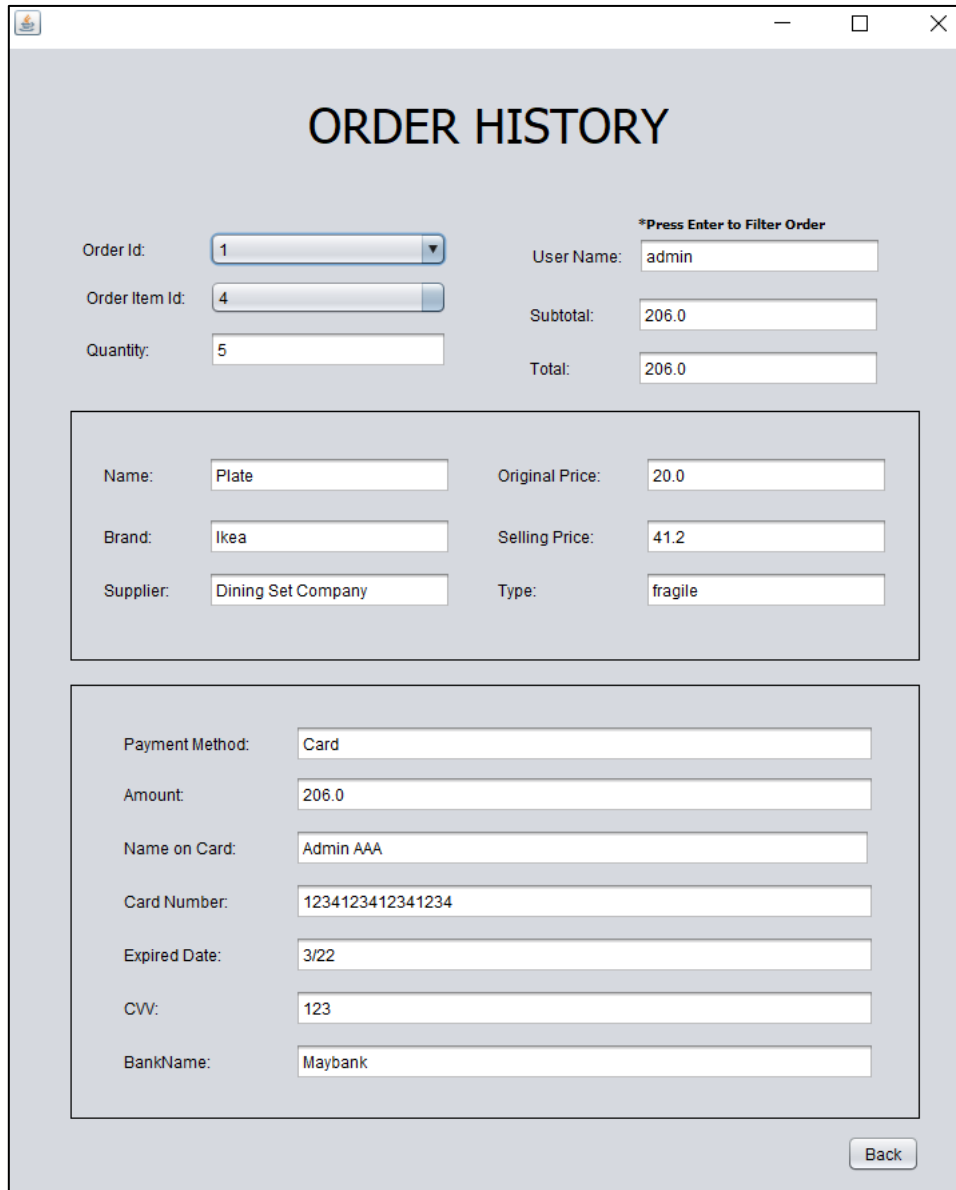


Figure 35: Order Menu

After clicking the order button on the main menu, it will bring admin to the customer menu which showing all the function available related to orders within the system. This section interfaces and functions is same as customer user which will be mentioned in later section, the only difference is when viewing order history as admin user.

5.1.3.1 View and Search All Users Order History



The screenshot shows a web application window titled "ORDER HISTORY". The window contains several input fields and a "Back" button. The fields are organized into three main sections:

- Order Identification:**
 - Order Id: 1 (dropdown menu)
 - Order Item Id: 4 (dropdown menu)
 - Quantity: 5 (text input)
- User and Pricing:**
 - User Name: admin (text input)
 - Subtotal: 206.0 (text input)
 - Total: 206.0 (text input)
- Item Details:**
 - Name: Plate (text input)
 - Brand: Ikea (text input)
 - Supplier: Dining Set Company (text input)
 - Original Price: 20.0 (text input)
 - Selling Price: 41.2 (text input)
 - Type: fragile (text input)
- Payment Information:**
 - Payment Method: Card (text input)
 - Amount: 206.0 (text input)
 - Name on Card: Admin AAA (text input)
 - Card Number: 1234123412341234 (text input)
 - Expired Date: 3/22 (text input)
 - CVV: 123 (text input)
 - BankName: Maybank (text input)

A "Back" button is located at the bottom right of the window.

Figure 36: View Order History

For view and search order function as admin user, all the order and details from all user in the system will be showing in the combo box when opening this window automatically without pressing other button. The user's name of the people who place this order will be display on the textbox while the text box can also use as search function by giving the user's name filter to the textbox. The main purpose of it is to view all user order history and also focus on one user order in the past.

5.2 Customer Functions

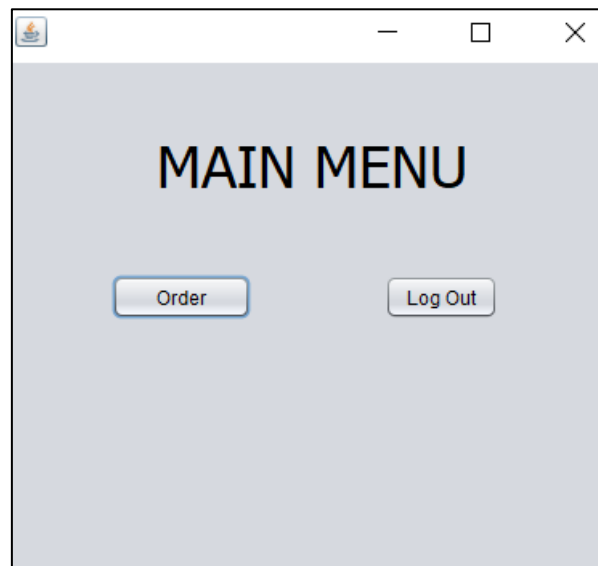


Figure 37: Main Menu

For customer user, this is the main menu they see after login.

5.2.1 Orders

5.2.1.1 Add Order Item

ADD ORDER ITEM

Current Order Id: 2 Order Item Id: 1

Quantity: Subtotal:

*Press Enter to Calculate Subtotal

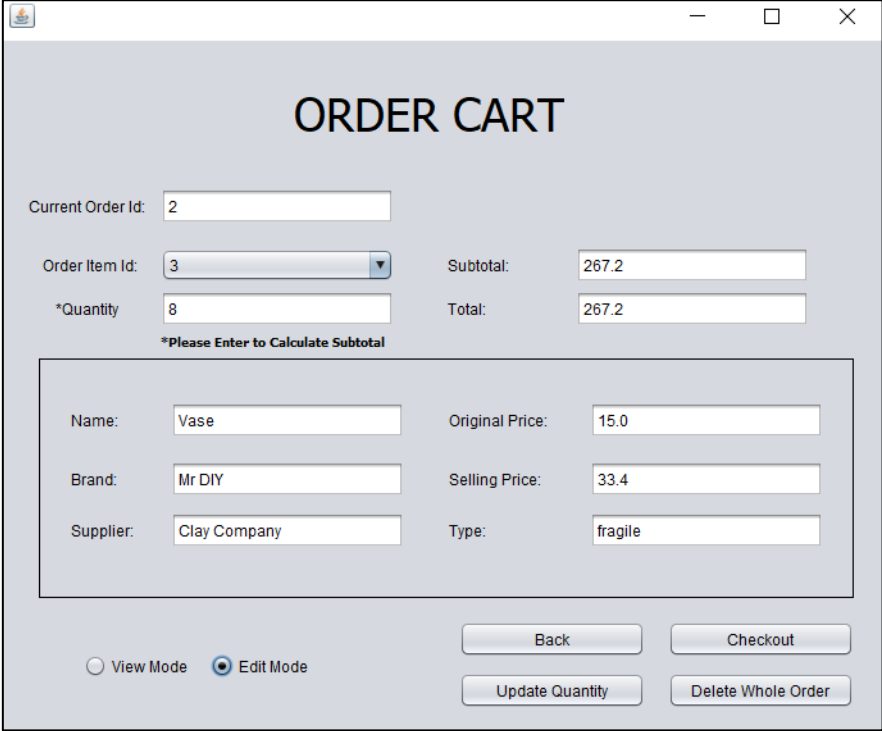
Name:	T-Shirt	Original Price:	50.0
Brand:	Uniqlo	Selling Price:	83.0
Supplier:	AAA Clothing Company	Type:	non-fragile

Back View Cart Add

Figure 38: Add Order Item

For add order item window, the main purpose of it is to add item to the current order, while every user only can have one order until the is checkout and confirmed then another order can be created. When pressing the add button, the execution will only be done when the validation for quantity is pass, user can check the subtotal by pressing enter button or add button. If the process is success a message window will prompt out, while when fail, a reason will also be given in the message window.

5.2.1.2 Update Order Cart

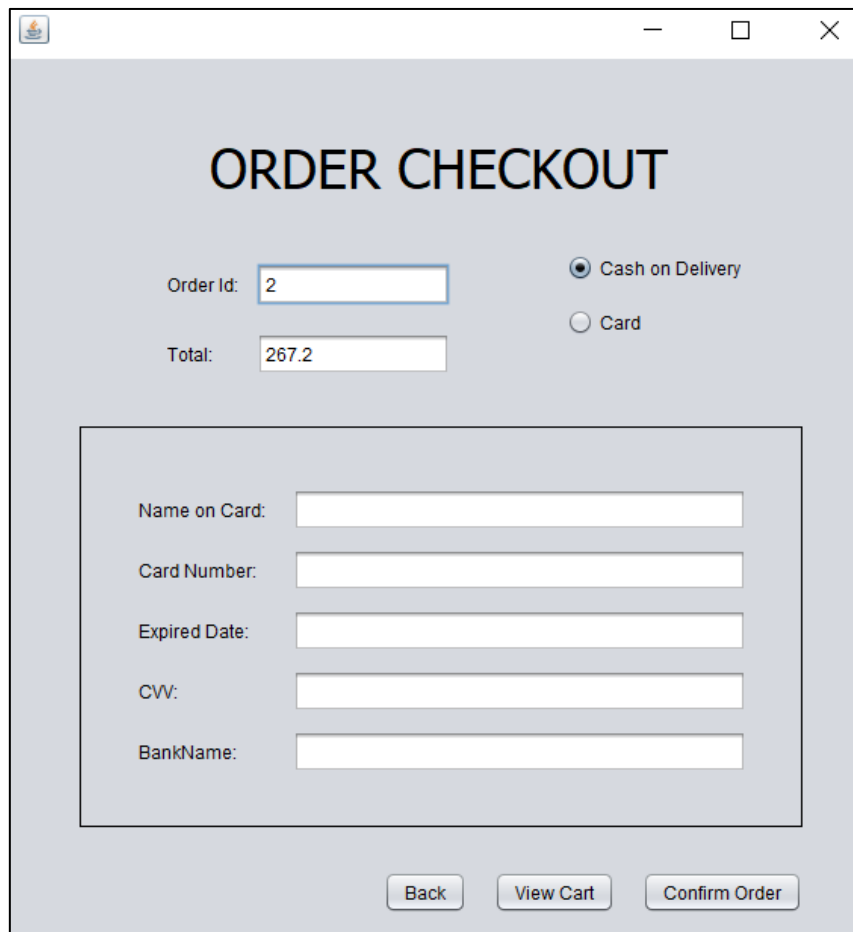


The screenshot shows a window titled "ORDER CART". At the top, it displays "Current Order Id: 2" and "Order Item Id: 3" (with a dropdown arrow). Below these are input fields for "*Quantity" (value 8) and "Subtotal: 267.2". A "Total: 267.2" field is also present. A note below the quantity field says "*Please Enter to Calculate Subtotal". A section below contains item details: "Name: Vase", "Original Price: 15.0", "Brand: Mr DIY", "Selling Price: 33.4", "Supplier: Clay Company", and "Type: fragile". At the bottom, there are radio buttons for "View Mode" and "Edit Mode" (selected). Four buttons are at the bottom right: "Back", "Checkout", "Update Quantity", and "Delete Whole Order".

Figure 39: Order Cart

When in edit mode on order cart window, delete order item, change order item quantity and delete entire order can be executed. By giving 0 quantity to the item, the item will automatically remove from the order and calculate a new total for the order that goes the same to change order item quantity. To delete entire order, simply by pressing the button will delete all the details of this order. The main purpose for this function is to make sure everything before checkout the order.

5.2.1.3 Checkout Order



The screenshot shows a web application window titled "ORDER CHECKOUT". The form contains the following elements:

- Order Id:** A text input field containing the value "2".
- Total:** A text input field containing the value "267.2".
- Payment Method:** Two radio buttons. The first is labeled "Cash on Delivery" and is selected. The second is labeled "Card".
- Card Details Section:** A container with five text input fields:
 - Name on Card:**
 - Card Number:**
 - Expired Date:**
 - CW:**
 - BankName:**
- Buttons:** Three buttons at the bottom: "Back", "View Cart", and "Confirm Order".

Figure 40: Checkout Order

After everything is checked, user can confirm the order by choosing a payment method and checkout. If the user chooses cash on delivery then the details of card do not need to be filled while if card is chosen the fields will be checked if empty before confirming the order. When pressing the update button, the execution will only be done when all fields are filled. If the process is successful a message window will prompt out, while when fail, a reason will also be given in the message window.

5.2.1.4 View and Search Order History

The screenshot shows a web application window titled "ORDER HISTORY". The interface is divided into several sections:

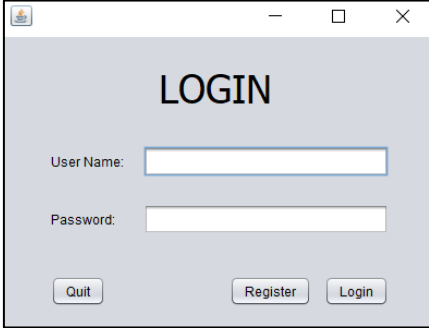
- Search and Summary Section:** Contains input fields for "Order Id:" (a dropdown menu with "2" selected), "Order Item Id:" (a dropdown menu with "3" selected), and "Quantity:" (a text input with "8"). To the right, there are two summary boxes: "Subtotal: 267.2" and "Total: 267.2".
- Item Details Section:** A box containing fields for "Name:" (Vase), "Original Price:" (15.0), "Brand:" (Mr DIY), "Selling Price:" (33.4), "Supplier:" (Clay Company), and "Type:" (fragile).
- Payment Information Section:** A box containing fields for "Payment Method:" (Cash), "Amount:" (267.2), "Name on Card:", "Card Number:", "Expired Date:", "CVV:", and "BankName:".
- Navigation:** A "Back" button is located at the bottom right of the window.

Figure 41: Order History

As mentioned, this function is same as admin but the order id will only contain this current user order history to make sure customer do not get each other personal information from this. Customer can search through desired order in the combo box to display the detail of it.

6.0 Extra Features

6.1 Registration

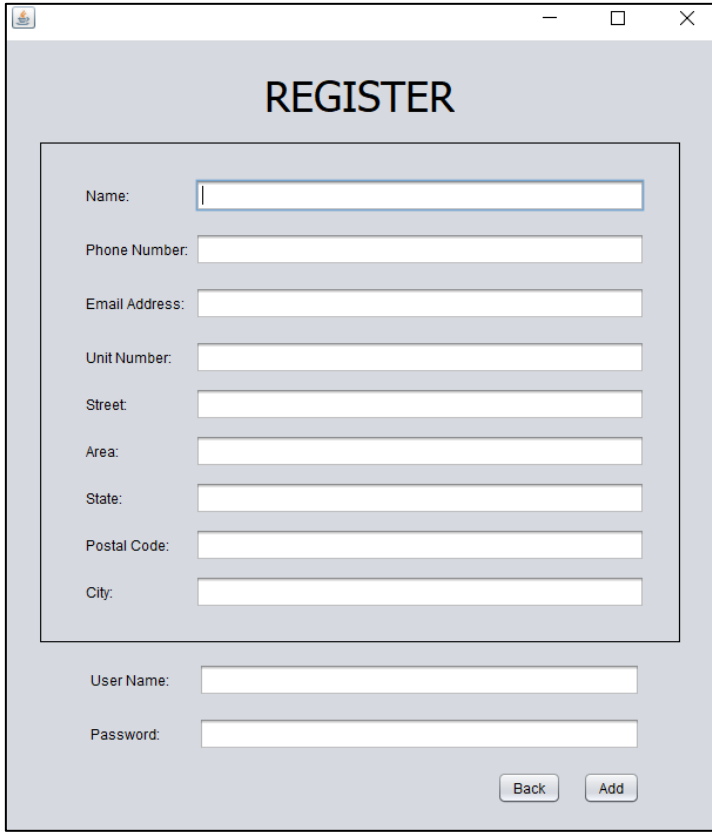
A screenshot of a login window titled "LOGIN". It features two input fields: "User Name:" and "Password:". Below the fields are three buttons: "Quit", "Register", and "Login". The window has a standard title bar with minimize, maximize, and close buttons.

LOGIN

User Name:

Password:

Figure 42: Login

A screenshot of a register window titled "REGISTER". It contains a large form with multiple input fields: "Name:", "Phone Number:", "Email Address:", "Unit Number:", "Street:", "Area:", "State:", "Postal Code:", and "City:". Below this form are "User Name:" and "Password:" fields. At the bottom right are "Back" and "Add" buttons. The window has a standard title bar with minimize, maximize, and close buttons.

REGISTER

Name:

Phone Number:

Email Address:

Unit Number:

Street:

Area:

State:

Postal Code:

City:

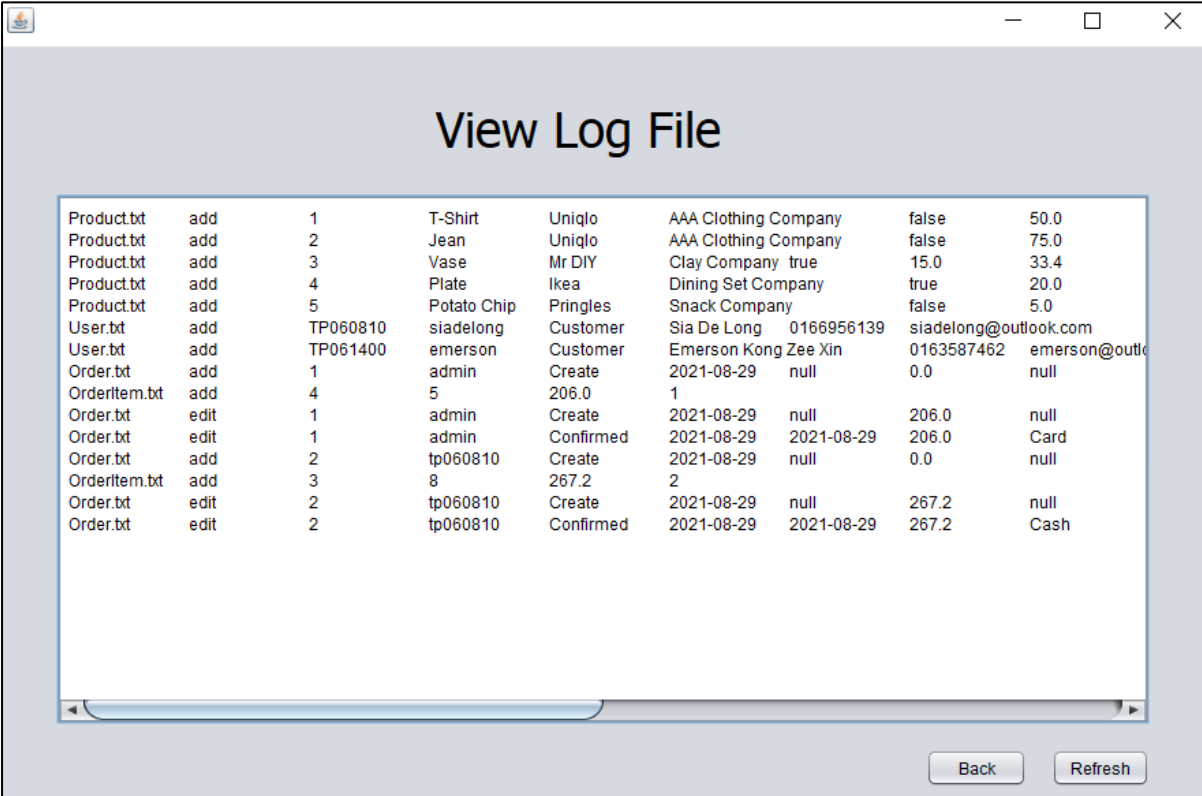
User Name:

Password:

Figure 43: Register

In the requirement only stated admin can add a customer to the system but this is not a efficient in customer perspective, so a registration function is added for customer user to create their own account in the login window. The account created will be in customer role then they will be able to place order.

6.2 Log File Generation



Product.txt	add	1	T-Shirt	Uniqlo	AAA Clothing Company	false	50.0	
Product.txt	add	2	Jean	Uniqlo	AAA Clothing Company	false	75.0	
Product.txt	add	3	Vase	Mr DIY	Clay Company	true	15.0	33.4
Product.txt	add	4	Plate	Ikea	Dining Set Company	true	20.0	
Product.txt	add	5	Potato Chip	Pringles	Snack Company	false	5.0	
User.txt	add	TP060810	siadelong	Customer	Sia De Long	0166956139	siadelong@outlook.com	
User.txt	add	TP061400	emerson	Customer	Emerson Kong Zee Xin	0163587462	emerson@outlook.com	
Order.txt	add	1	admin	Create	2021-08-29	null	0.0	null
OrderItem.txt	add	4	5	206.0	1			
Order.txt	edit	1	admin	Create	2021-08-29	null	206.0	null
Order.txt	edit	1	admin	Confirmed	2021-08-29	2021-08-29	206.0	Card
Order.txt	add	2	tp060810	Create	2021-08-29	null	0.0	null
OrderItem.txt	add	3	8	267.2	2			
Order.txt	edit	2	tp060810	Create	2021-08-29	null	267.2	null
Order.txt	edit	2	tp060810	Confirmed	2021-08-29	2021-08-29	267.2	Cash

Figure 44: View Log File

This is a function for admin user only. When doing any modification to the text file, it will be recorded in the log file. Aside from viewing it from text file directly, admin also can view and refresh it on the system application. Therefore, if any mistake or things that need to be retrieve can be view from this generated log file.

7.0 Conclusion

In conclusion, the management system is able to help the admin or the manager of the business admin to manage the product and the order from the customer so that it will be easier and also clearer for them to perform the deliveries. Not only that, customers are also able to place orders on the specific product by using this management system. Since the graphic user interface of this management system is user friendly, customers and the admin or the manager of the business only need to use a small amount of time to get used to the system.

There are two extra features added into the system. The first one is the customer's registration account feature. Customers that do not have an account for this system could now just register a new account from the login page. It will be convenient for the customer to create an account without going to the store or submitting e-form for a new account. Furthermore, the admin or the manager of the business is able to view all the history of every single change that has been performed in the system either by the customer or by the admin of the business. This is to keep a record of the history for future use.

References

- Braunschweig, D. (nd). *Rebus Community*. Retrieved from Encapsulation:
<https://press.rebus.community/programmingfundamentals/chapter/encapsulation/>
- Great Learning Team. (2021, March 25th). *Great Learning Team*. Retrieved from Polymorphism in Java – An Introduction:
<https://www.mygreatlearning.com/blog/polymorphism-in-java/>
- Guru99. (2021). *Guru99*. Retrieved from Inheritance in Java OOPs: Learn All Types with Example: <https://www.guru99.com/java-class-inheritance.html>
- Guru99. (2021). *Guru99*. Retrieved from Polymorphism in Java OOPs with Example: What is, Dynamic: <https://www.guru99.com/java-inheritance-polymorphism.html>
- JANSSEN, T. (2017, November 23rd). *Stackify*. Retrieved from OOP Concept for Beginners: What is Abstraction?: <https://stackify.com/oop-concept-abstraction/>
- JavaTpoint. (2021). *Java T Point*. Retrieved from Inheritance in Java:
<https://www.javatpoint.com/inheritance-in-java>
- Neha, V. (2021, June 17th). *edureka!* Retrieved from Inheritance in Java – Mastering OOP Concepts: <https://www.edureka.co/blog/inheritance-in-java/>
- Nick. (2021, January 6th). *Nerd Vision*. Retrieved from Polymorphism, Encapsulation, Data Abstraction and Inheritance in Object-Oriented Programming:
<https://www.nerd.vision/post/polymorphism-encapsulation-data-abstraction-and-inheritance-in-object-oriented-programming>
- SINGH, C. (2014). *Beginner Book*. Retrieved from Polymorphism in Java with example:
<https://beginnersbook.com/2013/03/polymorphism-in-java/>