

A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Module Code	CT038-3-2-ODJ
Intake Code	APU/APD2F2211IT(MBT)
Lecturer Name	Mr. Kau Guan Kiat
Hand-out Date	15 December 2022
Hand-in Date	3 March 2023

Student ID	Student Name
TP059963	Yip Zi Xian
TP060983	Tan Jia Yi

Table of Contents

1.0	Introduction.....	4
2.0	Assumptions.....	4
3.0	Class Diagram	5
4.0	Use Case Diagram	6
5.0	Object-oriented Code Implementation	14
5.1	Encapsulation	14
5.2	Inheritance	17
5.3	Abstraction.....	19
5.4	Overriding.....	20
5.5	Polymorphism.....	22
5.6	Interface	23
6.0	User Manual	26
6.1	General / Guest.....	26
6.1.1	Login / Sign Up Page.....	26
6.1.2	Market Store Page	30
6.2	Admin	31
6.2.1	Admin Dashboard.....	31
6.2.2	Category Management	32
6.2.3	Item Management	34
6.2.4	Delivery Management.....	36
6.2.5	Feedback Management.....	37
6.2.6	Order Management	38
6.2.7	Payment Management.....	39
6.2.8	User Management	40
6.2.9	Generate Report.....	42

6.3	Delivery Staff	45
6.3.1	Staff Dashboard	45
6.3.2	Delivery Management.....	46
6.3.3	View Feedback	48
6.4	Customer	49
6.4.1	Customer Dashboard.....	49
6.4.2	Online Speed Mall.....	49
6.4.3	Profile Management.....	51
6.4.4	Cart Management	52
6.4.5	Order History	54
6.4.6	Feedback History	56
7.0	Conclusion	60
8.0	References	60
9.0	Appendix.....	60
9.1	JFreeChart.....	60
9.2	Figma Wireframe	61

1.0 Introduction

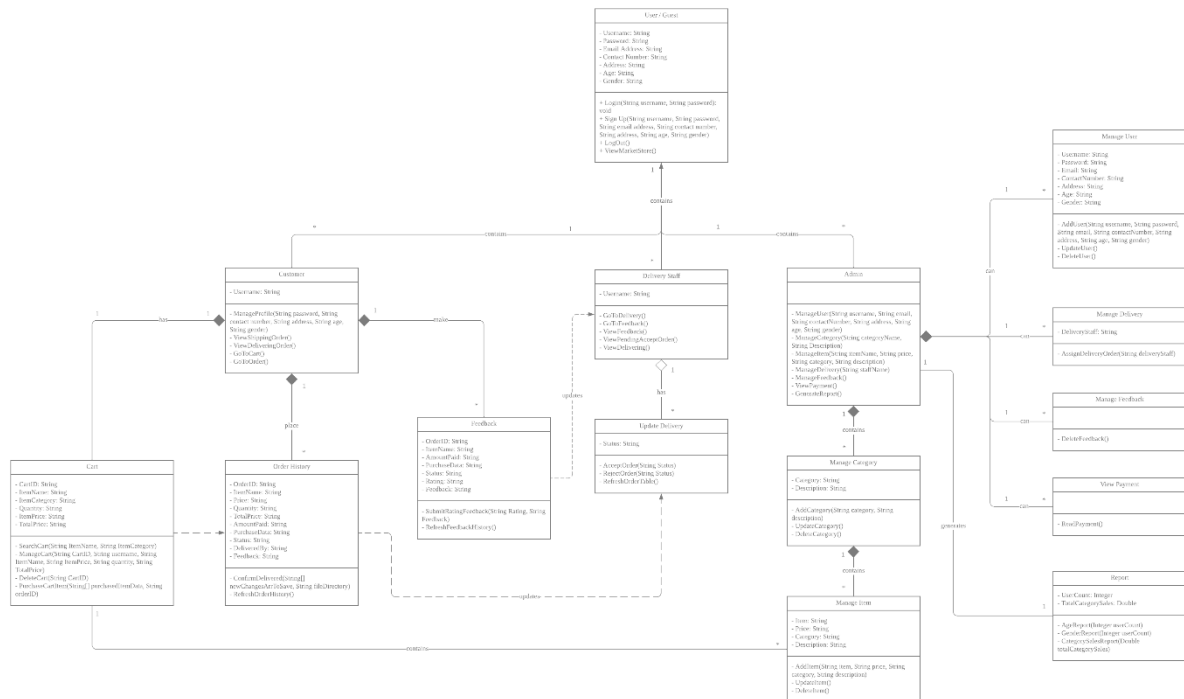
Speed Parcel Online Order and Delivery Management System (SPOODMS) is an order delivery system written in Java using NetBeans IDE version 15 along with JDK version 19. It is a system simulation for order delivery where it consists of four users type, guest, admin, delivery staff, and customer. SPOODMS is just an application simulating a real-world order delivery system like Lazada, Shoppe, Grab etc., thus it will not have a cloud database or actual payment transaction. All databases are saved locally, and assumptions are made in Chapter 2. Use case diagram and class diagram are provided to illustrate the whole system structure and have a brief overview of different functions available in SPOODMS.

2.0 Assumptions

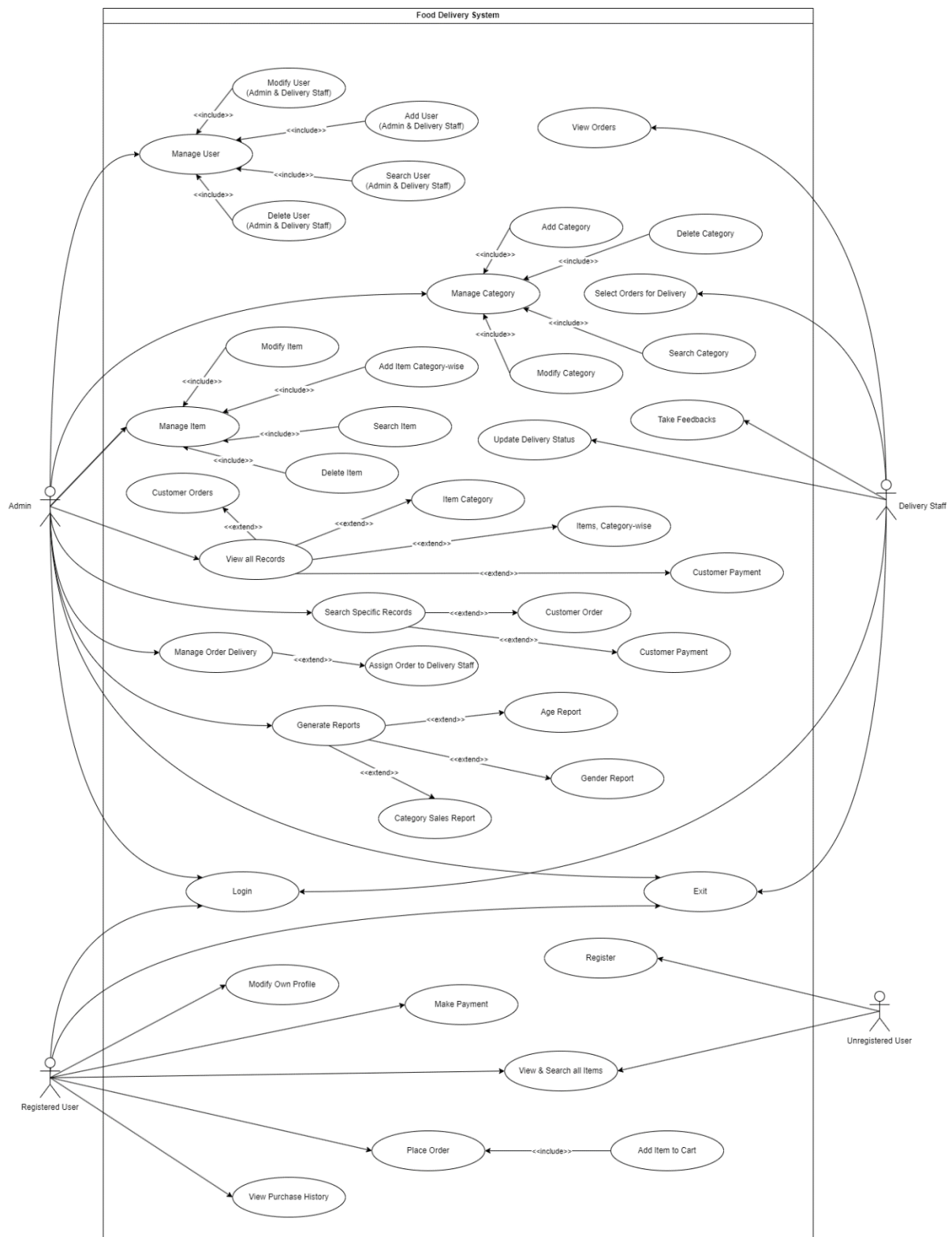
Since SPOODMS is not a full-pledged application, there will be some assumptions made:

- i. All payment transactions are paid to SPOODMS through debit or credit card or online banking method.
- ii. Admins and delivery staffs are well-trained to utilize SPOODMS.
- iii. Customers are not dyslexic and not suffering from color vision deficiency (CVD) because sometimes SPOODMS will have disabled buttons or error messages.
- iv. Delivery staffs are paid monthly based on the numbers of order delivery they accepted and delivered successfully.
- v. The profit of SPOODMS is taken from the total sales of all orders.
- vi. Item added to cart by customer, if has been changed or updated by admin, will be considered as last item in stock, price of the item in cart will not be changed.

3.0 Class Diagram



4.0 Use Case Diagram



Use Case	Register
Brief Description	Unregistered users can register an account
Actors	Unregistered User
Preconditions	Unregistered users must go to the Welcome Page window
Flows of Event	<ul style="list-style-type: none"> • Unregistered users open the Welcome Page window. • Unregistered users click on the “Register” button. • The system displays all the requirements needed to register an account such as username, password, gender, age, phone, email, and home address. • Unregistered users fill in all the details in the columns. • The system changes them to a registered user if all requirements are accepted. • The system displays a pop-up message to inform them register unsuccessful if all requirements are not accepted.

Use Case	Login
Brief Description	Users can login their account using username and password
Actors	Admin, Delivery Staff, Registered User
Preconditions	Users must go to the Welcome Page window
Flows of Event	<ul style="list-style-type: none"> • Users open the Welcome Page window. • Users choose login. • Users enter username and password. • System login to their account if username and password are correct. • System displays a pop-up message to inform users if username or password incorrect.

Use Case	Exit
Brief Description	Users can logout their account
Actors	Admin, Delivery Staff, Registered User
Preconditions	Users must login to the system
Flows of Event	<ul style="list-style-type: none"> • Admin, delivery staff, and registered users click on the “Logout” button. • The system logs them out and back to the Welcome Page window.

Use Case	Modify User
Brief Description	The admin modifies the admin and delivery staff accounts
Actors	Admin
Preconditions	Admin must go to the User Management window
Flows of Event	<ul style="list-style-type: none"> • Admin selects a user from the table. • Admin click on the “Modify” button. • Admin modifies the user account. • Admin save the changes.

Use Case	Add User
Brief Description	The admin adds the admin and delivery staff accounts
Actors	Admin
Preconditions	Admin must go to the User Management window
Flows of Event	<ul style="list-style-type: none">• Admin click on “Add” button.• Admin enters the details for a new user account.• Admin save the new account.

Use Case	Search User
Brief Description	The admin searches the admin and delivery staff accounts
Actors	Admin
Preconditions	Admin must go to the User Management window
Flows of Event	<ul style="list-style-type: none">• Admin enters username in the column.• Admin click on “Search” button.• System shows the searched user on the table.

Use Case	Delete User
Brief Description	The admin deleted the admin and delivery staff accounts
Actors	Admin
Preconditions	Admin must go to the User Management window
Flows of Event	<ul style="list-style-type: none">• Admin selects a user from the table.• Admin click on the “Delete” button.• Admin save the changes.

Use Case	Add Category
Brief Description	The admin adds a new category
Actors	Admin
Preconditions	Admin must go to the Category Management window
Flows of Event	<ul style="list-style-type: none">• Admin click on “Add” button.• Admin enters the details for the new category.• Admin save the changes.

Use Case	Delete Category
Brief Description	The admin deleted a category
Actors	Admin
Preconditions	Admin must go to the Category Management window
Flows of Event	<ul style="list-style-type: none">• Admin selects a category.• Admin click on the “Delete” button.• Admin save the changes.

Use Case	Modify Category
Brief Description	The admin modifies the category
Actors	Admin
Preconditions	Admin must go to the Category Management window
Flows of Event	<ul style="list-style-type: none"> • Admin selects a category. • Admin modifies the category. • Admin save the changes.

Use Case	Search Category
Brief Description	The admin searches a category
Actors	Admin
Preconditions	Admin must go to the Category Management window
Flows of Event	<ul style="list-style-type: none"> • Admin enters the name of the category in column. • Admin click on the “Search “ button. • System displays the category to admin.

Use Case	Modify Item
Brief Description	The admin modifies the item
Actors	Admin
Preconditions	Admin must go to the Item Management window
Flows of Event	<ul style="list-style-type: none"> • Admin selects a category. • Admin selects an item in category. • Admin modifies the item. • Admin save the changes.

Use Case	Add Item Category-wise
Brief Description	The admin adds a new item
Actors	Admin
Preconditions	Admin must go to the Item Management window
Flows of Event	<ul style="list-style-type: none"> • Admin click on “Add” button. • Admin enters the details for the new item. • Admin save the changes.

Use Case	Search Item
Brief Description	The admin searches an item
Actors	Admin
Preconditions	Admin must go to the Item Management window
Flows of Event	<ul style="list-style-type: none"> • Admin enters the name of the item in column. • Admin click on the “Search” button. • System displays the item to admin.

Use Case	Delete Item
Brief Description	The admin deleted an item
Actors	Admin
Preconditions	Admin must go to the Item Management window
Flows of Event	<ul style="list-style-type: none">• Admin selects an item.• Admin click on the “Delete” button.• Admin save the changes.

Use Case	View Customer Order Records
Brief Description	Admin views the records of customer orders
Actors	Admin
Preconditions	Admin must go to the Order Management window
Flows of Event	<ul style="list-style-type: none">• Admin opens the Order Management window.• System displays all records of customer orders on the table.

Use Case	View Item Category Records
Brief Description	Admin views the records of item categories
Actors	Admin
Preconditions	Admin must go to the Category Management window
Flows of Event	<ul style="list-style-type: none">• Admin opens the Category Management window.• System displays all records of item categories on the table.

Use Case	View Item Records
Brief Description	Admin views the records of items follow category-wise
Actors	Admin
Preconditions	Admin must go to the Item Management window
Flows of Event	<ul style="list-style-type: none">• Admin opens the Item Management window.• System displays all records of items follow category-wise on the table.

Use Case	View Customer Payment Records
Brief Description	Admin views the records of customer payment
Actors	Admin
Preconditions	Admin must go to the Payment Management window
Flows of Event	<ul style="list-style-type: none">• Admin opens the Payment Management window.• System displays all records of customer payment on the table.

Use Case	Search Customer Order Records
Brief Description	Admin search the records of customer orders
Actors	Admin
Preconditions	Admin must go to the Order Management window
Flows of Event	<ul style="list-style-type: none">• Admin opens the Order Management window.• Admin enters the username in column.• Admin click on the “Search” button.• System displays the records of customer orders that search by admin on the table.

Use Case	Search Customer Payment Records
Brief Description	Admin search the records of customer payment
Actors	Admin
Preconditions	Admin must go to the Payment Management window
Flows of Event	<ul style="list-style-type: none">• Admin opens the Payment Management window.• Admin enters the username in column.• Admin click on the “Search” button.• System displays the records of customer payment that search by admin on the table.

Use Case	Assign Order to Delivery Staff
Brief Description	Admin can assign the pending order to a delivery staff
Actors	Admin
Preconditions	Admin must go to the Delivery Management window
Flows of Event	<ul style="list-style-type: none">• Admin opens the Delivery Management window.• System displays all delivery lists on the table.• Admin selects an order ID and a delivery staff.• Admin click on the “Assign” button.

Use Case	View Orders
Brief Description	Delivery staff can view all orders waiting for delivery
Actors	Delivery Staff
Preconditions	Delivery staff must go to the Delivery Management window
Flows of Event	<ul style="list-style-type: none">• Delivery staff opens the Delivery Management window.• The system displays all orders waiting for delivery on the table.

Use Case	Select Orders for Delivery
Brief Description	Delivery staff can select orders to delivery
Actors	Delivery Staff
Preconditions	Delivery staff must go to the Delivery Management window
Flows of Event	<ul style="list-style-type: none"> • Delivery staff opens the Delivery Management window. • The system displays all orders waiting for delivery on the table. • Delivery staff selects an order from the table. • Delivery staff click on the “Accept Order” button.

Use Case	Take Feedbacks
Brief Description	Delivery staff can take feedbacks from users
Actors	Delivery Staff
Preconditions	Delivery staff must go to the Feedback Management window
Flows of Event	<ul style="list-style-type: none"> • Delivery staff opens the Feedback Management window. • The system displays all the delivery records with user feedbacks on the table.

Use Case	Update Delivery Status
Brief Description	Delivery staff can update the delivery status after delivered
Actors	Delivery Staff
Preconditions	Delivery staff must go to the Staff Dashboard window
Flows of Event	<ul style="list-style-type: none"> • Delivery staff login to the Staff Dashboard window. • The system displays all the orders accepted by delivery staff on the table. • Delivery staff selects an order from the table. • Delivery staff click on the “Update” button. • The system changes the delivery status to “Out of Delivery”.

Use Case	View & Search all Items
Brief Description	All users can view and search all items in categories
Actors	Registered User, Unregister User
Preconditions	Users must go to the Category window
Flows of Event	<ul style="list-style-type: none"> • Users open the categories. • The system displays all categories and items on the table. • Users enter an item name in column. • Users click on the “Search” button. • The system displays the matched items on the table.

Use Case	Modify Own Profile
Brief Description	Registered users can modify all the details in their own profile
Actors	Registered User
Preconditions	Registered users must login and go to User Profile window
Flows of Event	<ul style="list-style-type: none"> Registered users open the User Profile window. The system displays all the details such as username, password, home address, age, gender, phone, and email. Registered users edit the details in the columns. Registered users click on the “Update” button to save changes.

Use Case	Add Item to Cart
Brief Description	Registered users can select items and add them into the cart
Actors	Registered User
Preconditions	Registered users must login and go to categories
Flows of Event	<ul style="list-style-type: none"> Registered users open the categories. The system displays all items in categories. Registered users select all items and add them into the cart. The system displays all the selected items in the cart.

Use Case	Make Payment
Brief Description	Registered users can make payment to purchase items
Actors	Registered User
Preconditions	Registered users must login and go to the cart
Flows of Event	<ul style="list-style-type: none"> Registered users open their cart. The system displays all the items they added into the cart. Registered users click on the “Purchase” button. Registered users make an online payment for all selected items. The system will inform them after payment successful or failed.

Use Case	View Purchase History
Brief Description	Registered users can view their purchase history after make payment.
Actors	Registered User
Preconditions	Registered users must login and go to the Purchase History window
Flows of Event	<ul style="list-style-type: none"> Registered users open the Purchase History window. The system displays all the purchase histories they have made.

5.0 Object-oriented Code Implementation

According to Half, R. (2023), object-oriented programming is a method of writing code in separate section or different classes. Hence, debugging the code can be easier as the problem are always from one class. This avoids the hassle of making changes to each file that contains the bugged class. Beside that, code can be reused again and again without declaring a new class. For example, there are username and password attributes available in Customer and Admin class, but admin will have distinct functionalities compared to customer. A general User class containing username and password attributes can be created and inherit to Customer and Admin class. Moreover, object-oriented programming also promotes code flexibility where inputs to a method can have different outputs. For instance, customers give a feedback rating where all inputs are taken by one method but the output varies based on the value customers given.

In SPOODMS, there are six of many object-oriented methods, encapsulation, inheritance, abstraction, overriding, polymorphism, and interface. Here are some code snippets along with the explanations:

5.1 Encapsulation

```
/**
 * In WelcomePage.java
 */
package oodms.general;

import oodms.oop.AddNewCustomer;

..//

public class WelcomePage extends javax.swing.JFrame {

..//

    private void
    proceedBtnActionPerformed(java.awt.event.ActionEvent evt) {

..//

        AddNewCustomer customer = new AddNewCustomer(getUsername,
        getPassword, getEmail, getContact, getAddress, getAge, getGender);

    }

..//
}
```

```
/**
 * In AddNewCustomer.java
 */
package oodms.oop;

public class AddNewCustomer {
    private final String username;
    private final String password;
    private final String email;
    private final String contact;
    private final String address;
    private final String age;
    private final String gender;

    public AddNewCustomer(String username, String password, String
email, String contact, String address, String age, String gender) {
        this.username = username;
        this.password = password;
        this.email = email;
        this.contact = contact;
        this.address = address;
        this.age = age;
        this.gender = gender;

        // OOP Method - Get Multidimensional Array
        String[][] credentialsArr = new
Create3DArray().create3D("/oodms/database/credentials.txt");

        // New Customer Array
        String[] newCustomerArr = {username, password, email,
contact, address, age, gender};
```

```
        // Write file
        credentialsArr = addNewCustomer(credentialsArr,
newCustomerArr);

        new FlushAndWrite().flushAndWrite(credentialsArr,
"src/oodms/database/credentials.txt");
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public String getEmail() {
        return email;
    }
}
```

```
public String getContact() {  
    return contact;  
}  
  
public String getAddress() {  
    return address;  
}  
  
public String getAge() {  
    return age;  
}  
  
public String getGender() {  
    return gender;  
}  
  
// Add New Customer Data into Existing Customer Data  
public static String[][] addNewCustomer(String[][] existingArr,  
String[] newArr) {  
    int arrRow = existingArr.length;  
    int arrCol = existingArr[0].length;  
  
    String[][] newExistingArr = new String[arrRow + 1][arrCol];  
  
    for(int i = 0; i < arrRow; i++) {  
        System.arraycopy(existingArr[i], 0, newExistingArr[i],  
0, arrCol);  
    }  
  
    System.arraycopy(newArr, 0, newExistingArr[arrRow], 0,  
arrCol);  
  
    return newExistingArr;  
}
```

Based on the code snippet above, it demonstrates one of the object-oriented programming methods in Java, encapsulation. In **WelcomePage.java**, **AddNewCustomer** class is called by importing the class since it is in another package called **oodms.oop**. The constructor **AddNewCustomer** is then called by initialising a new variable to it and in this case the variable name is customer. To shorten the code, instead of using set functions for each argument to be parsed to **AddNewCustomer**, variables for each argument can be parsed directly into the constructor. All variables which are needed to be in the constructors are obtained from **JTextField** where users will input relevant information based on the **JLabel** guiding them.

After that, the arguments holding users' input will be parsed into the **AddNewCustomer.java**. Here, username, password, email, contact, address, age, and gender are declared as **private final** string variables. **Private** means that all variables can only be accessed only within the class itself whereas **final** will ensure that all variables will remain constant throughout each

code execution of adding a new customer. Each variable will have their corresponding get functions to return a string of the corresponding users' attributes.

Finally, a **static** function, **addNewCustomer**, is created to extract existing data from `credentials.txt` and includes a new array for newly registered customer. It requires two arguments, which is a 2D array for existing users' data and an array of newly registered customer and it will return a 2D array which contains the data for the new customer. To save the data of existing and new customer back into the text file, **FlushAndWrite** class is called to flush or clear the whole text file and rewrite the data according to new 2D array, **credentialsArr** where the **previous** 2D array has been overwritten by the return 2D array of **addNewCustomer** function.

5.2 Inheritance

```
/**
 * In UpdatePurchasedItemInDelivery.java
 */
package oodms.oop;

public class UpdatePurchasedItemInDelivery extends
UpdatePurchasedItem {
    @Override
    public void updatePurchasedItemData(String[] purchasedItemData,
String orderID) {
        // Get customer address
        String[][] getCustomerAddress = new
Create3DArray().create3D("/oodms/database/credentials.txt");
        String theCustomerAddress = null;

        for(String[] getArr : getCustomerAddress) {
            if(getArr[0].equalsIgnoreCase(purchasedItemData[1])) {
                theCustomerAddress = getArr[4];
            }
        }

        String[] purchasedItemDataInDeliveryArr = new String[] {
            "Unknown",
            orderID,
            "Unassigned",
            purchasedItemData[1],
            theCustomerAddress
        };

        String[][] existingArr = new
Create3DArray().create3D("/oodms/database/delivery.txt");
```

```
int arrRow = existingArr.length;
int arrCol = existingArr[0].length;

String[][] newExistingArr = new String[arrRow + 1][arrCol];

for(int i = 0; i < arrRow; i++) {
    System.arraycopy(existingArr[i], 0, newExistingArr[i],
0, arrCol);
}

System.arraycopy(purchasedItemDataInDeliveryArr, 0,
newExistingArr[arrRow], 0, arrCol);

// Flush and Write
new FlushAndWrite().flushAndWrite(newExistingArr,
"src/qodms/database/delivery.txt");
}
```

Based on the code snippet above, it demonstrates one of the object-oriented programming methods in Java, inheritance. Like the **abstraction** explanation, in **UpdatePurchasedItemInDelivery.java**, **UpdatePurchasedItemInDelivery** children class **extends** to the **UpdatePurchasedItem** parent class, allowing access to the abstract method available within the class and provide a body to it. Having inheritance, it allows the children class to inherit everything that is provided within the parent class without the need to repeat in the children class, so that it will reduce the redundancy of the overall code. **@Override** annotation is used to overwrite the empty abstract method declared in **UpdatePurchasedItem** parent class to provide a body to update purchased item data into **delivery.txt** based on the stored data format.

5.3 Abstraction

```
/**
 * In UpdatePurchasedItem.java
 */
package oodms.oop;

public abstract class UpdatePurchasedItem {
    public abstract void updatePurchasedItemData(String[]
purchasedItemData, String orderID);
}

/**
 * In UpdatePurchasedItemInOrder.java
 */
package oodms.oop;

..//

public class UpdatePurchasedItemInOrder extends UpdatePurchasedItem
{
    @Override
    public void updatePurchasedItemData(String[] purchasedItemData,
String orderID) {
..//

    }
}
```

Based on the code snippet above, it demonstrates one of the object-oriented programming methods in Java, abstraction. In **UpdatePurchasedItem.java**, an abstract class is created along with a method called **updatePurchasedItemData** that required two arguments, a string array of the purchased item data and the order ID in string. The method won't return any values as it is declared as **void** and it does not have a body.

In **UpdatePurchasedItemInOrder.java**, the method from which declared as abstract in **UpdatePurchasedItem.java** will have a body assigned to it using the **@Override** annotation. Abstract **UpdatePurchasedItem** class is used to provide a common interface or contract for any class that needs to update purchased item data but saves into a different text file with unique format by using a distinct data writing logic compared to each other. It also defines an abstract **UpdatePurchasedItemData** method as a signature where all subclasses must implement.

By using an abstract class and method, the code will allow developers to further include different implementations of updating purchased item data in the future while maintaining the common interface.

5.4 Overriding

```
/**
 * In CustomerDashboard.java
 */
package oodms.customer;

import javax.swing.JPanel;
import oodms.oop.MouseHover;

..//

public class CustomerDashboard extends javax.swing.JFrame {

    ..//

    /**
     * Creates new form AdminDashboard
     * @param acceptUsername
     */
    public CustomerDashboard(String acceptUsername) {

        ..//

        // Add mouse hover event
        String[] componentTexts = new String[] {
            "Profile Management",
            "Cart Management",
            "Order History",
            "Feedback History"
        };

        JPanel[] allJPanel = new JPanel[] {backgroundPanel1};

        MouseHover mh = new MouseHover();
        mh.mouseHoverEffect(componentTexts, allJPanel);
    }

    ..//
}
```

```

/**
 * In MouseHover.java
 */
package oodms.oop;

import java.awt.Component;
import java.awt.Font;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class MouseHover {
    public void mouseHoverEffect(String[] componentTexts, JPanel[]
allJPanel) {
        Font plainKarla = new Font("Karla", Font.PLAIN, 14);
        Font boldKarla = new Font("Karla", Font.BOLD, 14);

        for(JPanel jPanel : allJPanel) {
            Component[] components = jPanel.getComponents();

            for(Component component : components) {
                if(component instanceof JLabel jLabel) {
                    var componentDisplayText = ((JLabel)
component).getText();

```

```

                for(String componentText : componentTexts) {
                    if(componentDisplayText.equals(componentText)) {
                        JLabel label = jLabel;
                        label.addMouseListener(new
MouseAdapter() {
                            @Override
                            public void mouseEntered(MouseEvent
e) {
                                label.setFont(boldKarla);
                            }

                            @Override
                            public void mouseExited(MouseEvent e)
{
                                label.setFont(plainKarla);
                            }
                        });
                    }
                }
            }
        }
    }
}

```

Based on the code snippet above, it demonstrates one of the object-oriented programming methods in Java, overriding. In **CustomerDashboard.java**, **MouseHover** class will be imported from another package called **oodms.oop**. Within the **MouseHover** class, there will be a method called as **mouseHoverEffect** that will require two arguments, a string array and a

JPanel array. The method will loop through each JPanel and each string to find whether there are JLabel in the current looped JPanel with the same text as the text provided in the string array. If the current looped JPanel contains JLabel that possesses the same text as provided in the string array, it will execute the **mouseHoverEffect** and implements a hover effect to the JLabel.

In **MouseHover.java**, **MouseListener** is added to the **JLabel label** which will have a new **MouseAdapter** class. Within **MouseAdapter** class, there are methods of **mouseEntered** and **mouseExited**, which is required to be overwritten to create a hover effect. **@Override** annotation is used to indicate that these methods are overriding the same-named methods in the **MouseListener** interface.

In this case, **mouseEntered** and **mouseExited** methods will have new implementations of setting the font of the label to **bold** and setting the font of the label back to **plain** respectively. **MouseListener** interface is responsible for handling these mouse events and the overridden methods provided will give users to have a hover effect experience when their cursor is over or away from the **JLabel label**.

5.5 Polymorphism

```
/**
 * In FeedbackHistoryDisplay.java
 */
package oodms.customer;

...//

import oodms.oop.FeedbackRating;

...//

public class FeedbackHistoryDisplay extends javax.swing.JFrame {
    ...//

    private void ratingSliderMouseReleased(java.awt.event.MouseEvent
    evt) {
        int getRatingValue = ratingSlider.getValue();

        FeedbackRating displayRating = new FeedbackRating();
        ratingName.setText(displayRating.getRating(getRatingValue));
    }
}
```

```
/**
 * In FeedbackRating.java
 */
package oodms.oop;

public class FeedbackRating {
    public String getRating(int rating) {
        String theRating = switch (rating) {
            case 1 -> "Extremely Poor";
            case 2 -> "Below Average";
            case 3 -> "Mediocre";
            case 4 -> "Good";
            case 5 -> "Excellent";
            default -> "Extremely Poor";
        };

        return theRating;
    }
}
```

Based on the code snippet above, it demonstrates one of the object-oriented programming methods in Java, polymorphism. In **FeedbackHistoryDisplay.java**, **FeedbackRating** class is called by importing it from another package, **oodms.oop**. In **FeedbackRating.java**, there will be a method named **getRating** that will require an integer value called **rating** where it will accept the users' value from a **JSlider**.

The value will be parsed through the **getRating** method and return a string value which differs based on the rating given by the user. This is where polymorphism takes place, the same method but will give a different output based on the input. For example, if the users drag the **JSlider** to the third position, it will return "**Mediocre**". If the user ignores the **JSlider**, it will return "**Extremely Poor**" as the default value.

5.6 Interface

```
/**
 * In AddNewItemToCart.java
 */
package oodms.oop;

interface toCart {
    public String[][] addToCart(String[][] existingArr, String[]
newArr);
}

public final class AddNewItemToCart implements toCart {
    private final String username;
    private final String item;
    private final String price;
    private final String quantity;
```

```
private final String totalPrice;

public AddNewItemToCart(String username, String item, String
price, String quantity, String totalPrice) {
    this.username = username;
    this.item = item;
    this.price = price;
    this.quantity = quantity;
    this.totalPrice = totalPrice;

    String[][] cartArr = new
Create3DArray().create3D("/odms/database/cart.txt");

    String[] newCartArr = new String[] {username, item, price,
quantity, totalPrice};

    cartArr = addToCart(cartArr, newCartArr);
    new FlushAndWrite().flushAndWrite(cartArr,
"sdc/odms/database/cart.txt");
}

public String getUsername() {
    return username;
}

public String getItem() {
    return item;
}

public String getPrice() {
    return price;
}

public String getQuantity() {
    return quantity;
}
```

```
public String getTotalPrice() {
    return totalPrice;
}

@Override
public String[][] addToCart(String[][] existingArr, String[]
newArr) {
    int arrRow = existingArr.length;
    int arrCol = existingArr[0].length;

    String[][] newExistingArr = new String[arrRow + 1][arrCol];

    for(int i = 0; i < arrRow; i++) {
        System.arraycopy(existingArr[i], 0, newExistingArr[i],
0, arrCol);
    }

    System.arraycopy(newArr, 0, newExistingArr[arrRow], 0,
arrCol);

    return newExistingArr;
}
```


Based on the code snippet above, it demonstrates one of the object-oriented programming methods in Java, interface. In **AddNewItemToCart.java**, the interface **toCart** is used to define a contract for implementing classes that can add items to a shopping cart. This interface requires that any class implementing it **must have** a method called “**addToCart**” that requires a 2D string array representing the existing shopping cart and a string array representing the new item to be added to the shopping cart. If not, the class will throw an error, asking to change it to an abstract class instead. This method will return a 2D string array containing the new item data.

The **AddNewItemToCart** class implements **toCart** interface and provides a specific implementation for the **addToCart** method. Since it is added the item data into a text field, all fields are declared as **private final** and a constructor to take in these fields and adds the new item to existing cart. The use of interfaces and classes in the code above has included other object-oriented programming like **encapsulation**.

6.0 User Manual

6.1 General / Guest

General functions are functions that Speed Parcel Online Order and Delivery Management System (SPOODMS) provides for all users including admin, delivery staff, registered and unregistered customers. These functions are open to public view.

6.1.1 Login / Sign Up Page

Welcome to Speed Parcel

Speed Parcel

[Log In](#) [Sign Up](#)

Username:

Password:

Confirm Password:

Email Address:

Contact Number:

Address:

Age:

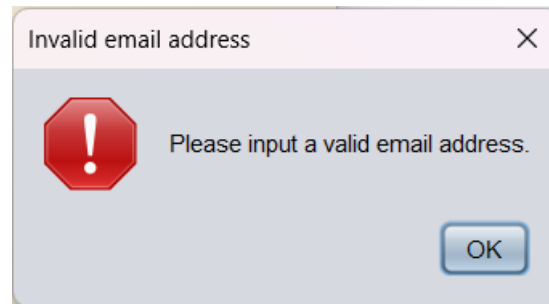
Gender:

[Proceed](#) [Guest](#)

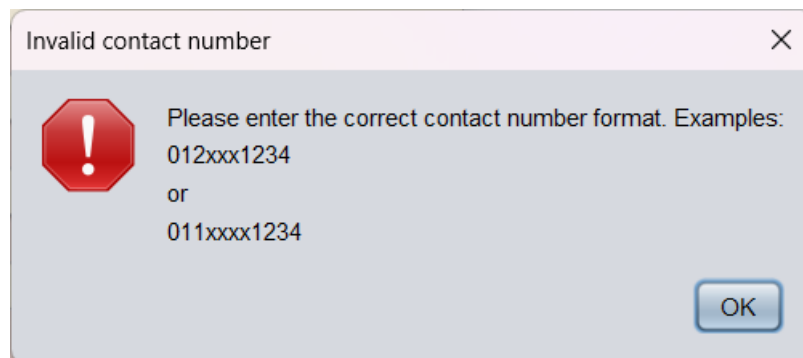
This is a screenshot of the Sign-Up page. New unregistered customer must field up the relevant fields which are mandatory in order to gain access to their own dashboards. There will be three different dashboard, admin dashboard, customer dashboard, and customer dashboard which

will be discussed in their own sub-chapters. User will require to fill up their username, password and confirm password, email, contact number, address, age and gender.

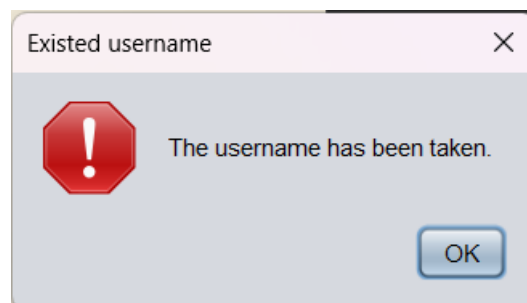
When user inputs the wrong email format, without “@” and “.”, the system will prompt an error message to the user.



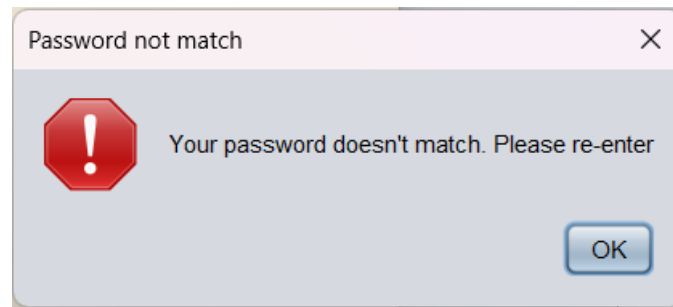
When user inputs the wrong contact number format, which is not within 10 or 11 numbers in length, the system will prompt an error message to the user and provides the correct example format for user.



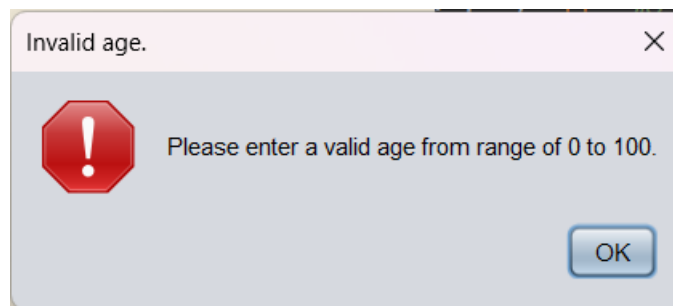
When user inputs an existed username, the system will prompt the user to retype a unique username.



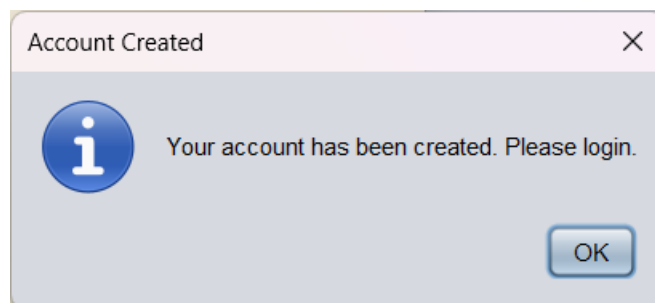
When user inputs different passwords for password field and confirm password field, the system will prompt the user to retype and confirm both passwords are the same.



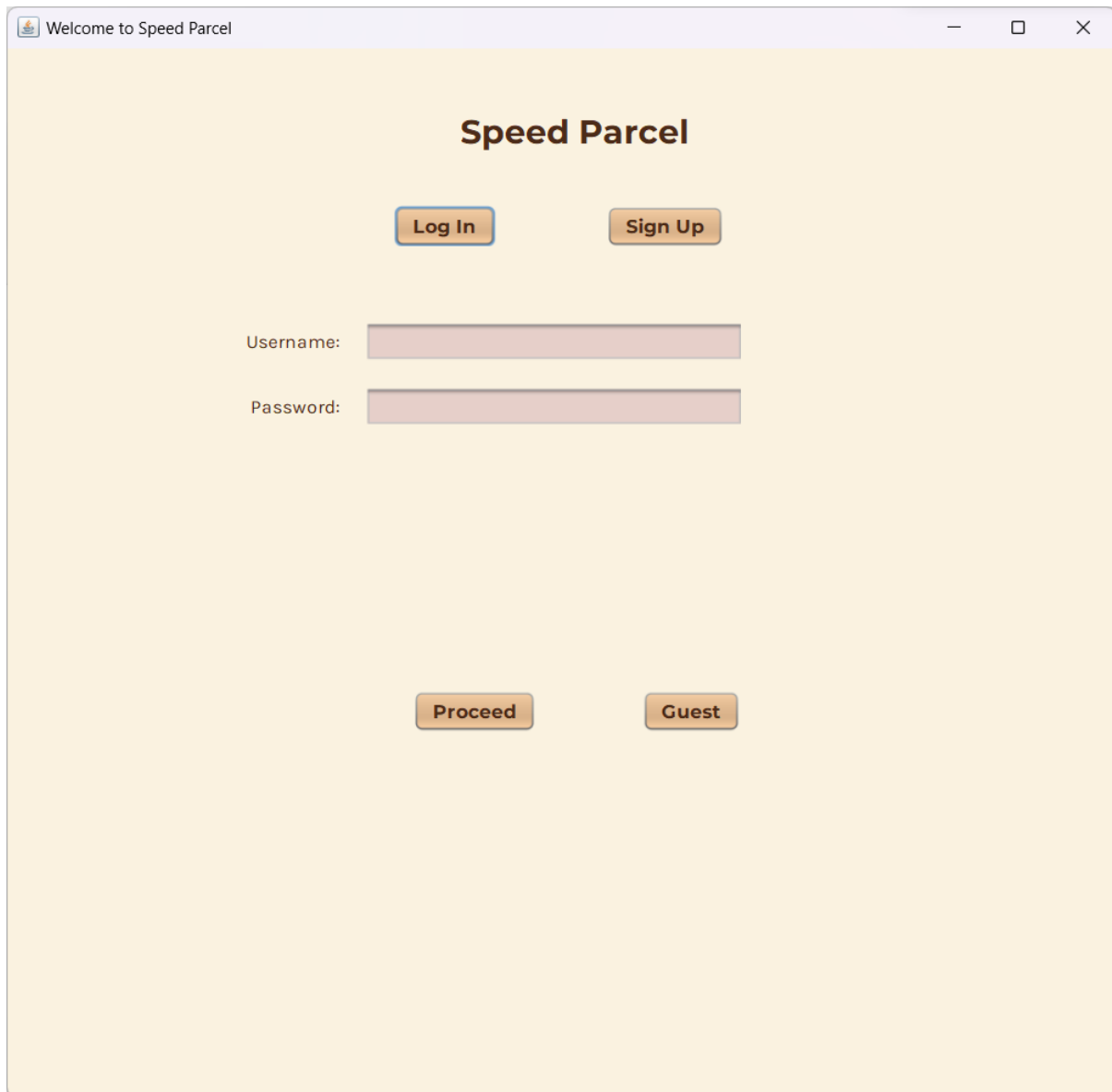
When the user inputs age value of less than 0 or more than 100, the system will prompt the user to re-enter a valid age.



If the user filled up all fields successfully and without any error, the system will prompt the user about their account has been created and can login to their dashboard.

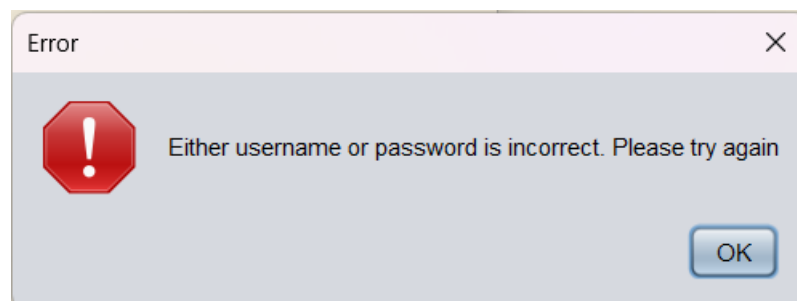


After having the account created, the user can click on the Log In button to change to the login interface and proceed with their login.



A screenshot of a Java Swing window titled "Welcome to Speed Parcel". The window has a light yellow background. At the top center, the text "Speed Parcel" is displayed in a bold, dark font. Below this, there are two buttons: "Log In" and "Sign Up". Underneath the buttons, there are two text input fields. The first is labeled "Username:" and the second is labeled "Password:". At the bottom of the window, there are two more buttons: "Proceed" and "Guest".

If the user typed the username or password wrongly, the system would prompt the user about their invalid inputs and rejects their access.



6.1.2 Market Store Page

When the user presses the guest button beside the proceed button, it will take the user to the Online Speed Mall page. User will be accessing the system as a guest and can search for items, filter through different categories and check for all item details.

Online Speed Mall
Great deals always come with great delivery speed

Search

Enter Name: **Search**

Filter Category:

Details

Item Name:

Price (RM):

Description:

Quantity: 0 **Add To Cart**

Back

List

Item Name	Price (RM)	Category	Description
Cranberry	4.90	Food and Beverages	Organic from Australia
Strawberry	3.90	Food and Beverages	Local from Cameron Highlands
Polo Shirt	89.90	Clothes	Free size Polo shirt for men
Black Skirt	69.90	Clothes	A full body skirt for women
Wooden Phone St...	39.90	Accessories	An ergonomic phone stand suitab...
Blueberry	5.90	Food and Beverages	Organic from New Zealand
Blackberry	7.90	Food and Beverages	Organic from Japan

Online Speed Mall
Great deals always come with great delivery speed

Search

Enter Name: **Search**

Filter Category: Food and Beverages

Details

Item Name: Blackberry

Price (RM): 7.90

Description: Organic from Japan

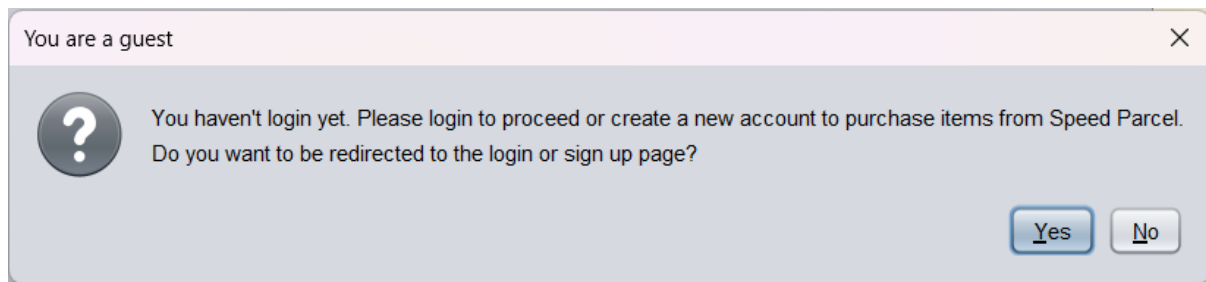
Quantity: 0 **Add To Cart**

Back

List

Item Name	Price (RM)	Category	Description
Cranberry	4.90	Food and Beverages	Organic from Australia
Strawberry	3.90	Food and Beverages	Local from Cameron Highlands
Blueberry	5.90	Food and Beverages	Organic from New Zealand
Blackberry	7.90	Food and Beverages	Organic from Japan

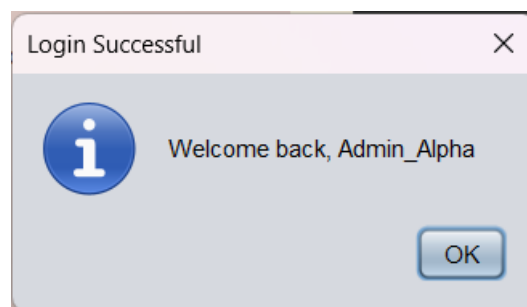
However, when the user tries to click the Add To Cart button, the system will prompt the user to proceed to login / sign up page or to remain in the Online Speed Mall page.



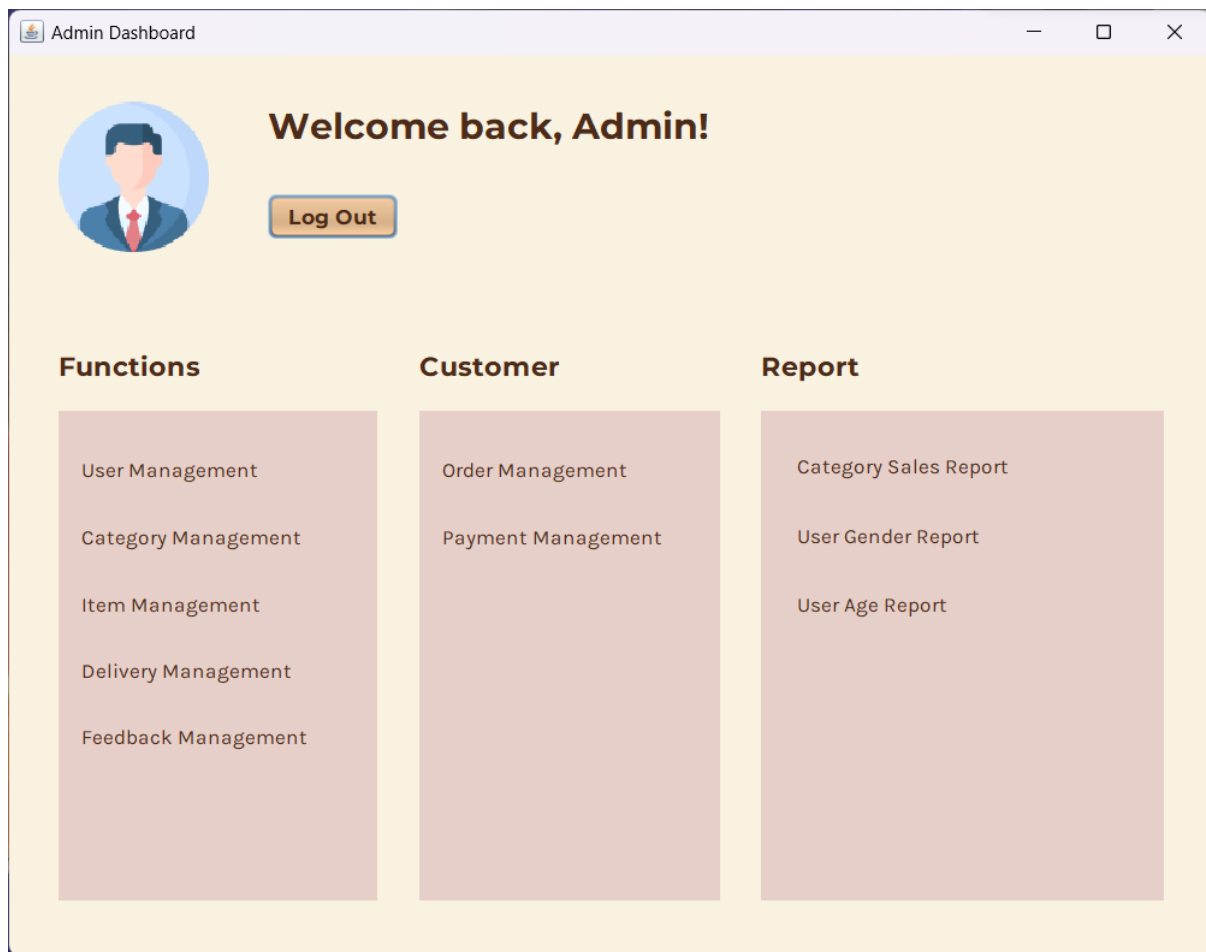
6.2 Admin

6.2.1 Admin Dashboard

When an admin login into the SPOODMS, given that he or she inputted the correct username and password, the system will redirect the admin to the admin dashboard. All admin will have access to the same dashboard.



Admin will have access to three main categories which are general functions, customer management, and generating report. General functions contains user management, category management, item management, delivery management, and feedback management, where admin can manipulate all the data. Customer functions contains order management and payment management, and admin can only read these data as they are important for SPOODMS to make profits from order deliveries. Besides that, admin can generate bar graph and pie charts about category sales, total users' gender, and total users' age.



6.2.2 Category Management

Admin can add a new category for items to have them to be filtered easily through the Category Management Display. When admin pressed Search, it will return all available categories along with their description and how many items are under this category.

The screenshot shows a web application titled "Category Management". It features a search bar with a "Search" button, a details form with "Category Name" and "Description" fields, and buttons for "Add", "Edit", "Save", "Delete", and "Back". A table lists existing categories:

Category Name	Children Items	Description
Food and Beverages	4	All food and beverages available in supermar...
Clothes	2	All clothes available in supermarket retail st...
Accessories	1	All accessories available for your workspace

When the admin wants to add a new category, he or she can press the Add button to type in the category name and the description of the category. If there is an empty field, the system will prompt the admin to fill up the empty field.

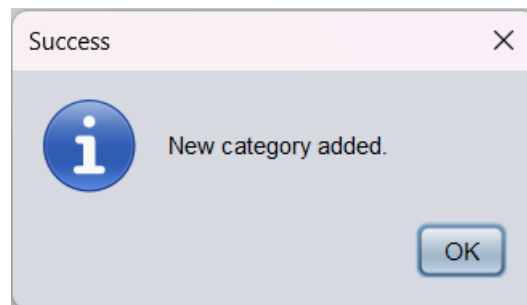


When the admin adds the same category name, regardless of the letter case, the system will prompt the users about an existing category name.

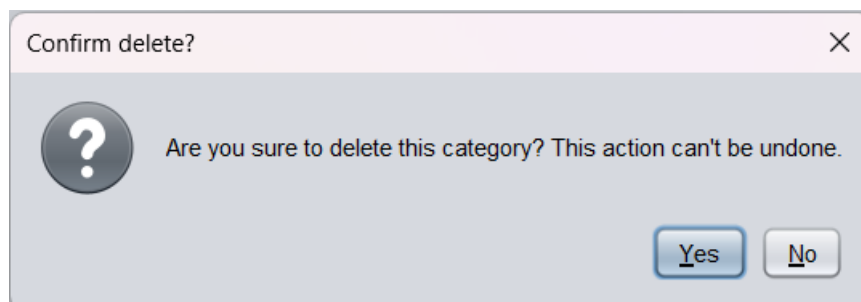


When the admin successfully added a new category, the system will prompt the admin that a new category has been added. However, the admin will need to proceed to Item Management

to view the category because there is no item assigned to the new category, thus it returns a zero value.



When the admin selected a row in the table, the Edit button and Delete button will become Enabled. Admin can edit, save, and delete the existing categories. The Save button will only be enabled when the admin pressed on the Edit button. When the admin saves an edited existing category, the system will also check for similar categories in the database. When the admin deletes an existing category, the system will prompt the admin for confirmation before actually deleting the category. Even though the category is deleted, all items that have been assigned to the category will remain in the database.



6.2.3 Item Management

Admin can add a new item through the Item Management Display. Before adding a new item, admin can filter through the existing items name along with the categories.

The screenshot shows the 'Item Management' application window. It has a title bar with standard window controls. The main content area is divided into three sections: 'Search', 'Details', and 'List'.

Search Section: Contains a text input field labeled 'Enter Item:', a 'Search' button, and a dropdown menu labeled 'Filter Category:'.

Details Section: Contains four input fields: 'Item Name:', 'Price (RM):', 'Category:', and 'Description:'. Below these are four buttons: 'Add', 'Edit', 'Save', and 'Delete'. At the bottom of this section is a 'Back' button.

List Section: Contains a table with the following data:

Item Name	Price	Category	Description
Cranberry	4.90	Food and Beverages	Organic from Australia
Strawberry	3.90	Food and Beverages	Local from Cameron Highlands
Polo Shirt	89.90	Clothes	Free size Polo shirt for men
Black Skirt	69.90	Clothes	A full body skirt for women
Wooden Phone St...	39.90	Accessories	An ergonomic phone stand suitable...
Blueberry	5.90	Food and Beverages	Organic from New Zealand
Blackberry	7.90	Food and Beverages	Organic from Japan
HDMI Port Cable	39.90	Electronics	A popular cable that replaced VGA c...

This screenshot shows the same 'Item Management' application window, but with the 'Filter Category' dropdown menu set to 'Food and Beverages'. The 'List' table now only displays items belonging to this category:

Item Name	Price	Category	Description
Cranberry	4.90	Food and Beverages	Organic from Australia
Strawberry	3.90	Food and Beverages	Local from Cameron Highlands
Blueberry	5.90	Food and Beverages	Organic from New Zealand
Blackberry	7.90	Food and Beverages	Organic from Japan

When the admin presses the Add button, it will allow the admin to add a new item to the database. It will also have the same function as adding a new category for check whether there is a similar existing item in the database.

The Edit and Delete button will be enabled when the admin selected a row of the table similarly to the Category Management Display. The Save button will also only be enabled when the admin pressed on the Edit button. The system will also prompt the admin to confirm before deleting an existing item.

The text field to input the item price can only be typed using numbers. If the admin types using letters, nothing will appear in the text field.

6.2.4 Delivery Management

When the admin wants to assign an order to a delivery staff, the admin can navigate to the Delivery Management Display. Admin can filter through the deliver status of each order and find which order is “Delivered”, “Ongoing” and “Unassigned”.

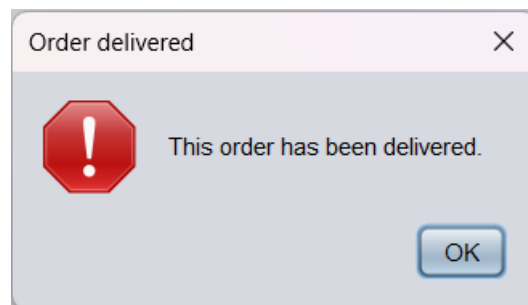
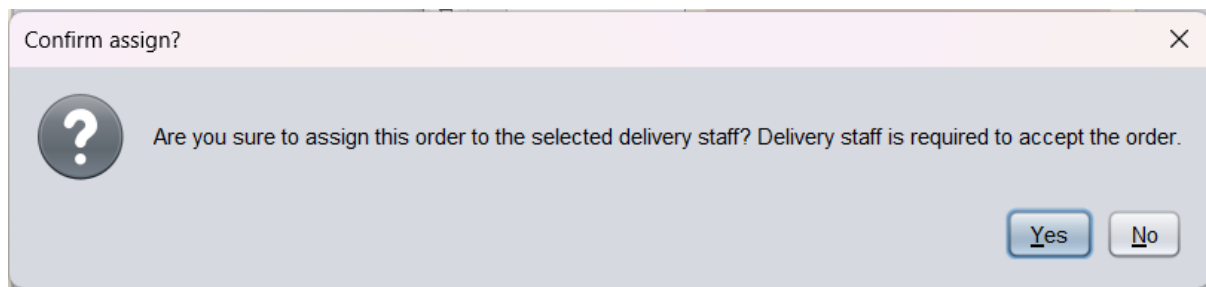
The screenshot shows a web application window titled "Delivery Management". The interface is divided into three main sections: Search, Details, and List.

- Search:** Contains a "Filter Status:" dropdown menu and a "Search" button.
- Details:** Contains form fields for "Delivery Staff:" (a dropdown menu showing "Delivery_Alpha"), "Order ID:" (a text field), "Order Status:" (a dropdown menu showing "Unassigned"), "Customer Name:" (a text field), and "Customer Address:" (a text area). At the bottom of this section are "Edit" and "Save" buttons.
- List:** Contains a table with the following data:

Delivery Staff	Order ID	Order Status	Customer Name	Customer Address
Delivery_Alpha	15000100	Delivered	UserAlpha	24, Jalan Home
Delivery_Alpha	15001500	Ongoing	UserAlpha	24, Jalan Home

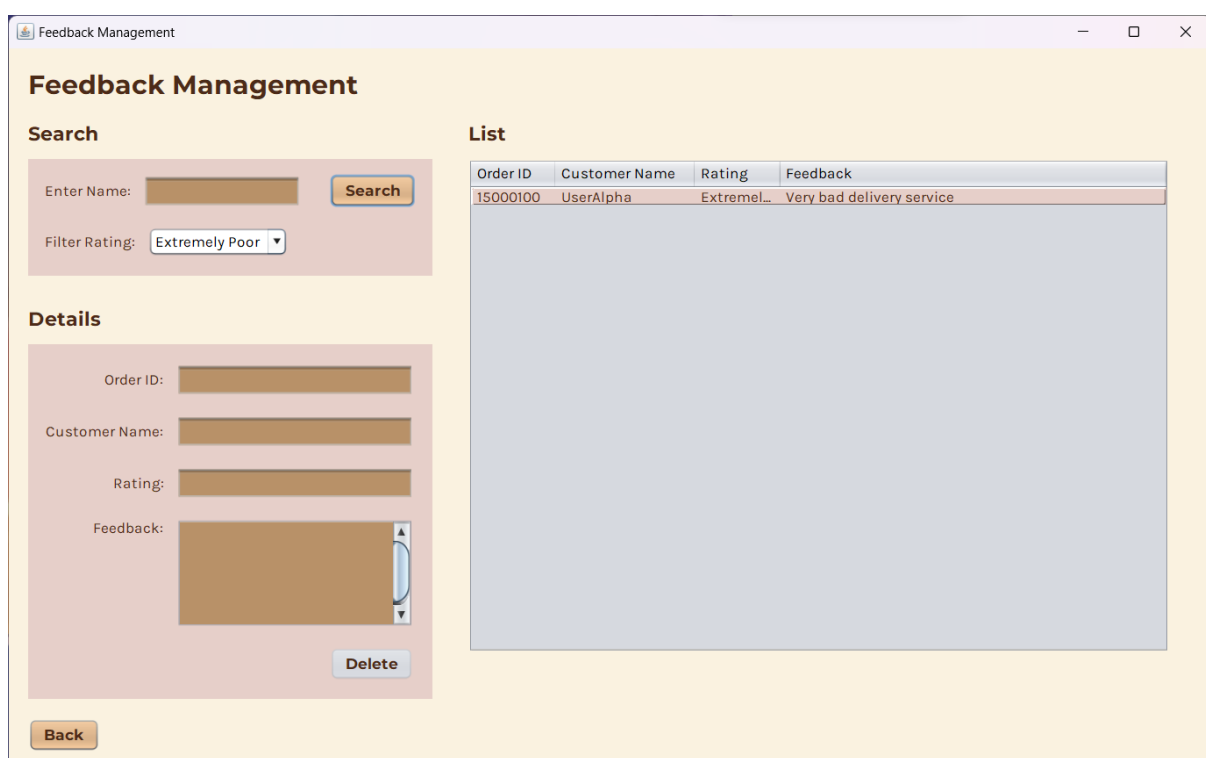
At the bottom left of the window is a "Back" button.

When the admin selects a row from the table, it will display the order details at the bottom left corner of the JPanel. Admin can only edit the delivery staff field and the order status field, any information displayed on text field relevant to customer are disabled. When admin can choose to edit the order details, but if the order status is delivered, the system will prompt the admin that the order has been delivered when admin tries to edit and save the delivered order.

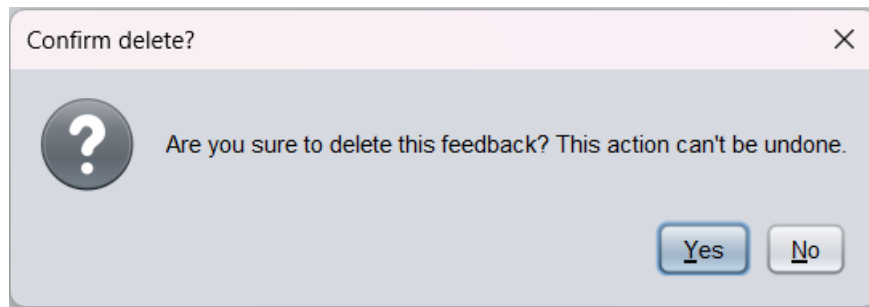


6.2.5 Feedback Management

Admin can review all customer feedbacks in Feedback Management Display. Similar to item management and delivery management, a search function with filter is provided to the admin to ease the navigation.



However, since feedback are from customers, admin has no permission to edit the feedback content and can only read their feedbacks. Although sometimes customers' feedback may include vulgar words, so admin will have the permission to delete the feedback.



6.2.6 Order Management

Admin can view all customers' orders in Order Management Display. A search and filter functions are provided for the ease of searching the correct data.

Order ID	Customer Name	Item Name	Quantity	Item Price (RM)	Total Price (RM)	Order Date	Order Status
----------	---------------	-----------	----------	-----------------	------------------	------------	--------------

Admin can only view these orders because it is exclusive for customers and any data manipulation is prohibited. Assuming customers will contact admin through email or contact number, admin can refer to this display to confirm with customers about issues regarding their order status.

The screenshot shows a web application titled "Order Management". It features a search section on the left with input fields for "Enter Name:" and "Order Status:", a "Search" button, and a "Details" section below it. The details section contains input fields for "Order ID:", "Customer Name:", "Item Name:", "Quantity:", "Price (RM):", "Total Price (RM):", "Order Date:", and "Order Status:". A "Back" button is located at the bottom left. On the right, there is a "List" section containing a table with the following data:

Order ID	Customer Name	Item Name	Quantity	Item Price (RM)	Total Price (RM)	Order Date	Order Status
15000100	UserAlpha	Blackberry	5	7.90	39.50	2023-01-25	Delivered
15001500	UserAlpha	Cranberry	6	4.90	29.40	2023-01-26	Ongoing

6.2.7 Payment Management

In Payment Management Display, admin can review which customers have paid their order by filtering through each order payment status. When the delivery staff made a deliver to a customer, the customer will need to confirm that they receive their order parcels. After the customer confirm their orders, their payment status will be updated to “Paid” if not it will remain as “Pending”.

The screenshot shows a web application titled "Payment Management". It features a search bar with "Enter Name:" and a "Search" button, and a "Filter Paid:" dropdown menu. Below the search bar is a "Details" section with input fields for Payment ID (16002350), Order ID (15000100), Customer Name (User), Filter Paid (Paid), Total Price (RM) (39.50), Shipment Fees (RM5), Process Fee (4%) (41.08), and Amount Payable (46.08). A "Back" button is located at the bottom left. On the right, a "List" section displays a table with the following data:

Payment ID	Order ID	Customer Name	Filter Paid	Total Price (RM)	Amount Payable
16002350	15000100	User	Paid	39.50	46.08
16002230	15001500	User	Pending	29.40	35.58

Customers' order includes total price, shipping fees, process fee, and amount payable. Total price is calculated based on the quantity of item the customer purchased. Amount payable is calculated based on the total price with shipping fees of RM5 and process fee of 4%.

6.2.8 User Management

In User Management Display, admin can add, edit, save and delete users data for customers, delivery staff, and admin. All data validation for email address and contact number are like Login / Sign Up page.

User Management

Search

Enter Name:

Details

Username:

Password:

Email Address:

Contact Number:

Address:

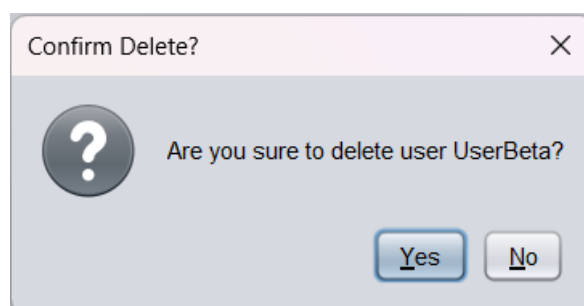
Age:

Gender:

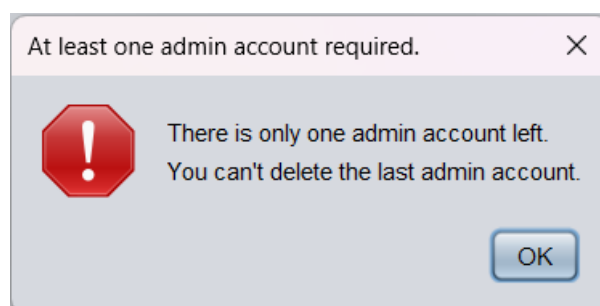
List

Username	Email Address	Contact Number	Address	Age	Gender
Admin_Alpha	adminalpha@gmail.com	0134445555	28, King Lane, Ipoh, Perak	20	Male
Delivery_Alpha	deliveryalpha@gmail.com	0165557777	29, Bishop Street, Germany	30	Male
Delivery_Beta	deliverybeta@gmail.com	0165553377	31, Bishop Street, Germany	45	Male
UserAlpha	useralpha@gmail.com	0146765858	24, Jalan Home	34	Others
UserBeta	userbeta@gmail.com	0123536688	46, Jalan Wall	76	Female

When the admin wants to delete a user, the system will prompt for confirmation to avoid any mistakes. The prompt message will also highlight the selected user's username.

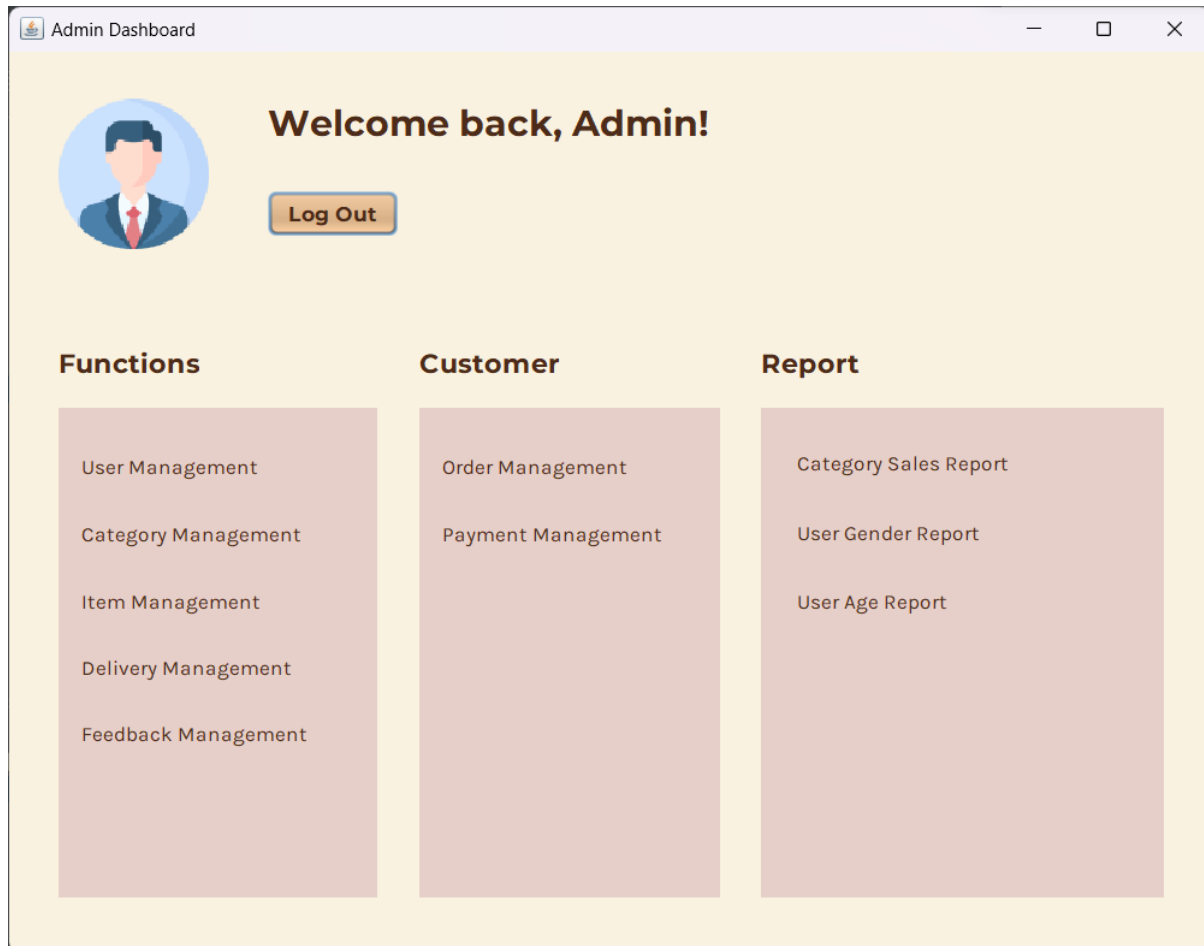


However, when the admin wants to delete the last admin account available in the SPOODMS, the system will prompt a reject message because the system must and always need at least one admin account.

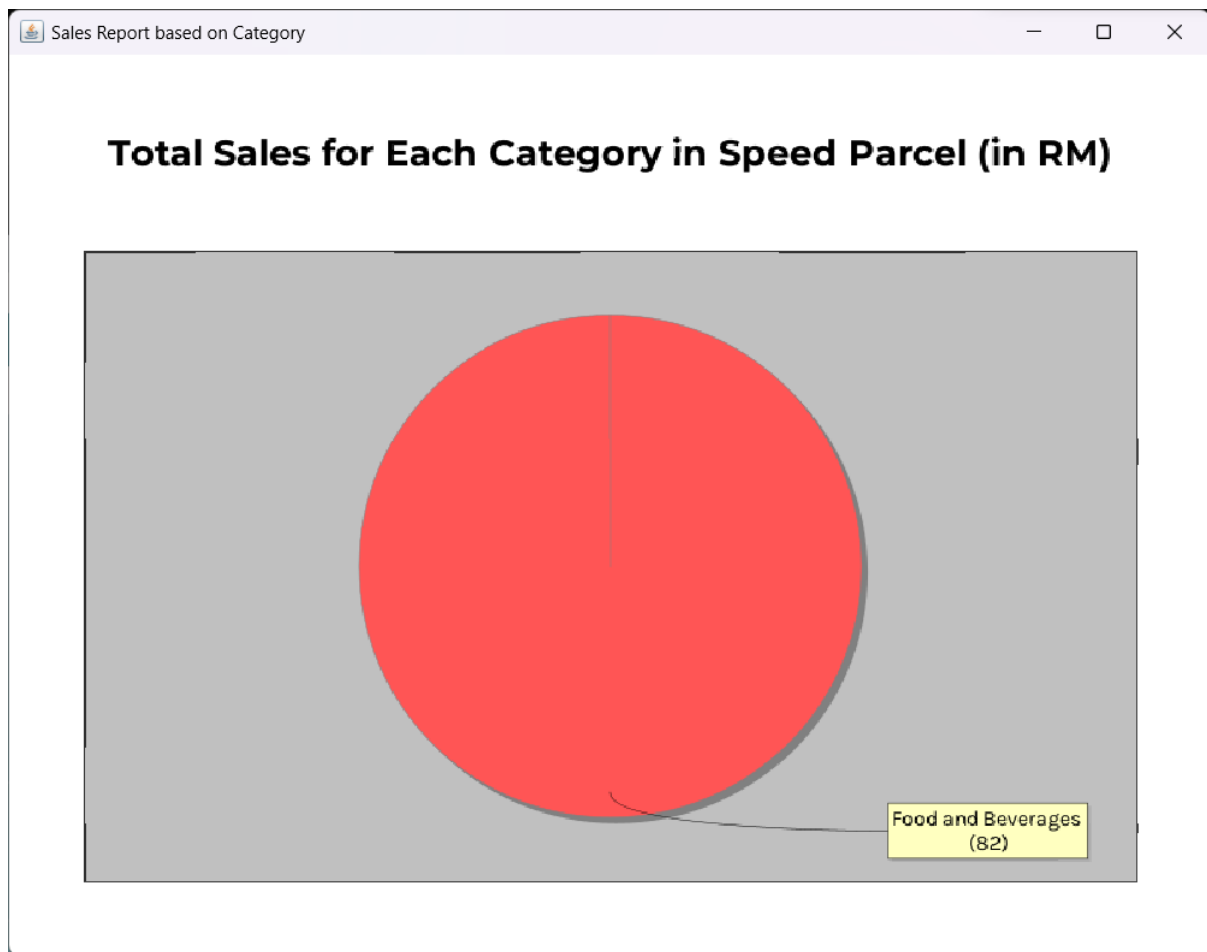


6.2.9 Generate Report

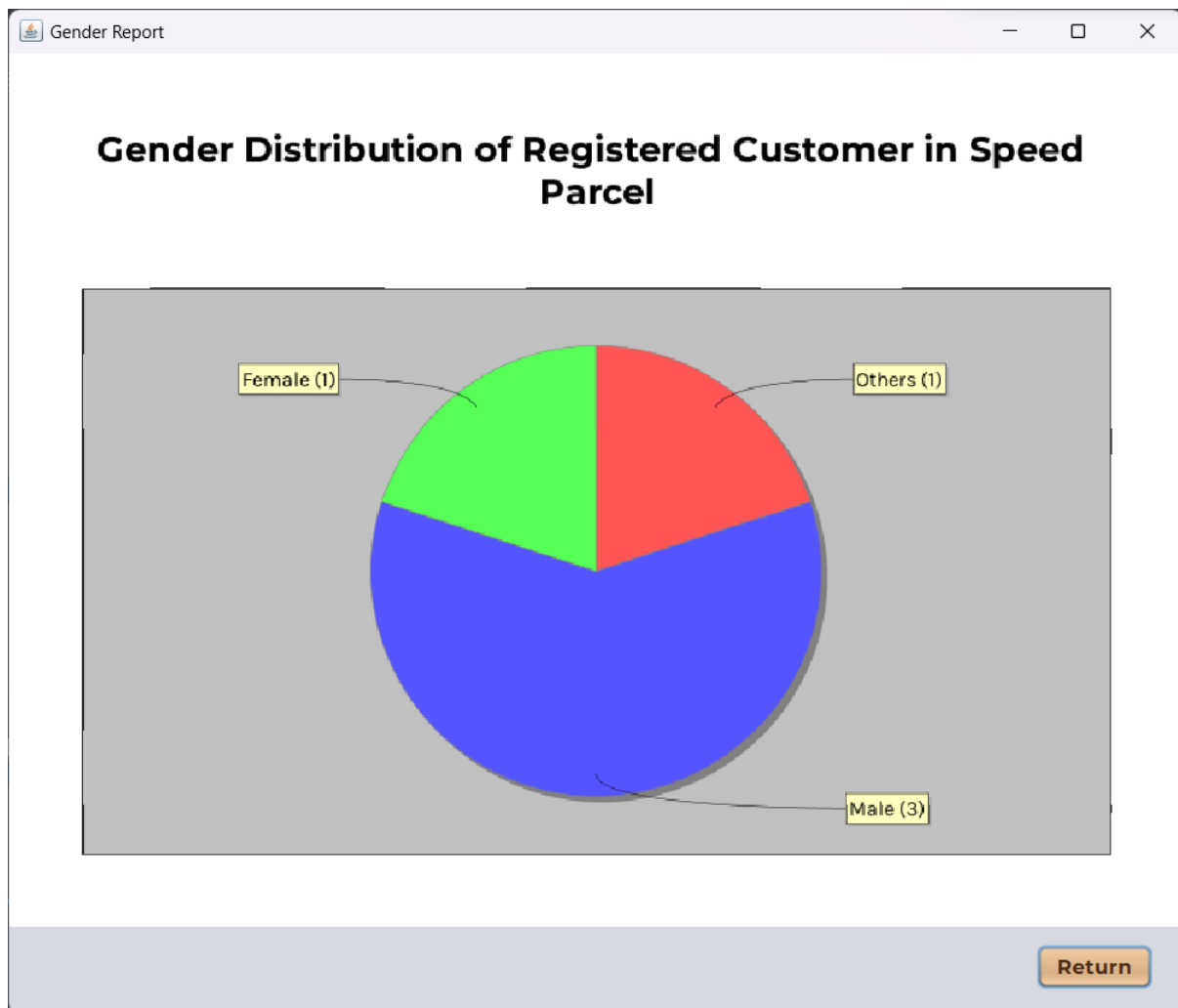
Admin can generate three different reports from the SPOODMS, user gender report, user age report, and category sales report. All these report function can be accessed via Admin Dashboard.



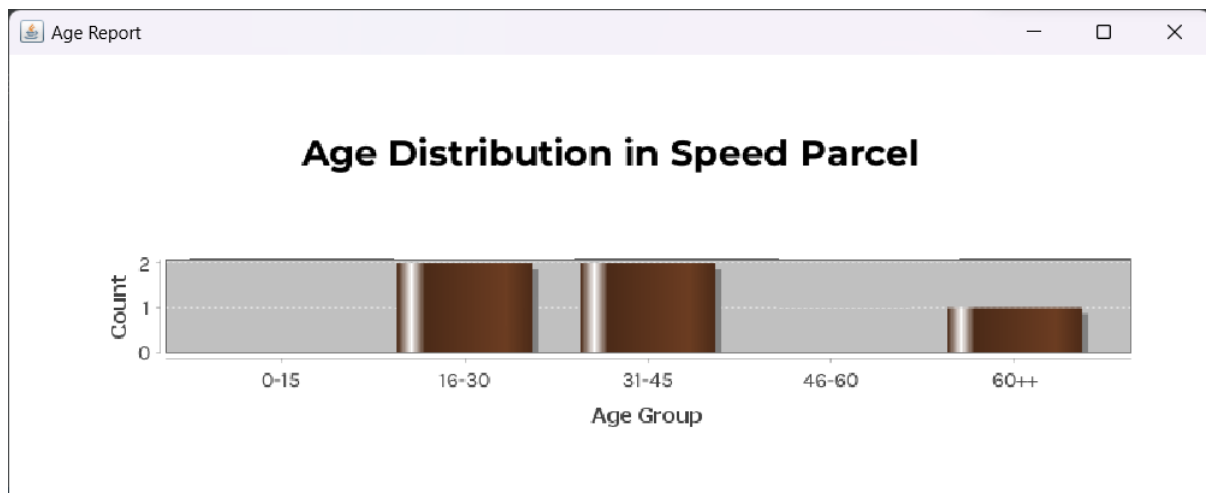
When the admin clicks on category sales report, the system will generate a pie chart that shows the total sales of each category. Total sales are calculated by adding the amount paid for each category.



When the admin clicks on user gender report, the system will generate a pie chart that show all users of SPOODMS including admins, delivery staffs, and customers into three categories of gender, Male, Female and Others.



When the admin clicks on user age report, the system will generate a bar chart that shows all users of SPOODMS including admins, delivery staffs, and customers into different age groups. Noted that the height of the bar chart depends on the highest count, thus the bar chart display can be ignored while it still returns the correct value.



6.3 Delivery Staff

6.3.1 Staff Dashboard

When delivery staff login to the SPOODMS with their delivery staff username and password, they will be redirected to their own Staff Dashboard. Each delivery staff will have a different Staff Dashboard because they will have their own order deliveries.

Welcome back! Delivery_Alpha

[Log Out](#)

Functions

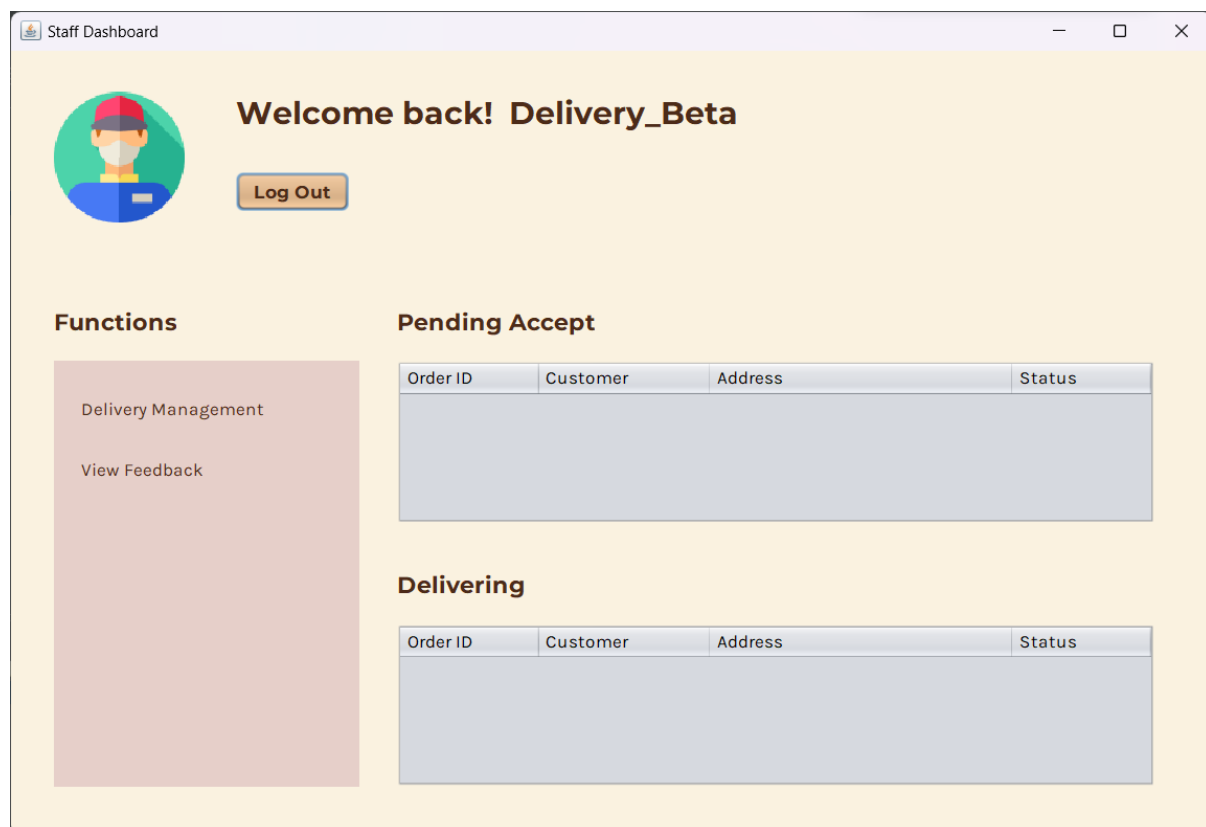
- Delivery Management
- View Feedback

Pending Accept

Order ID	Customer	Address	Status

Delivering

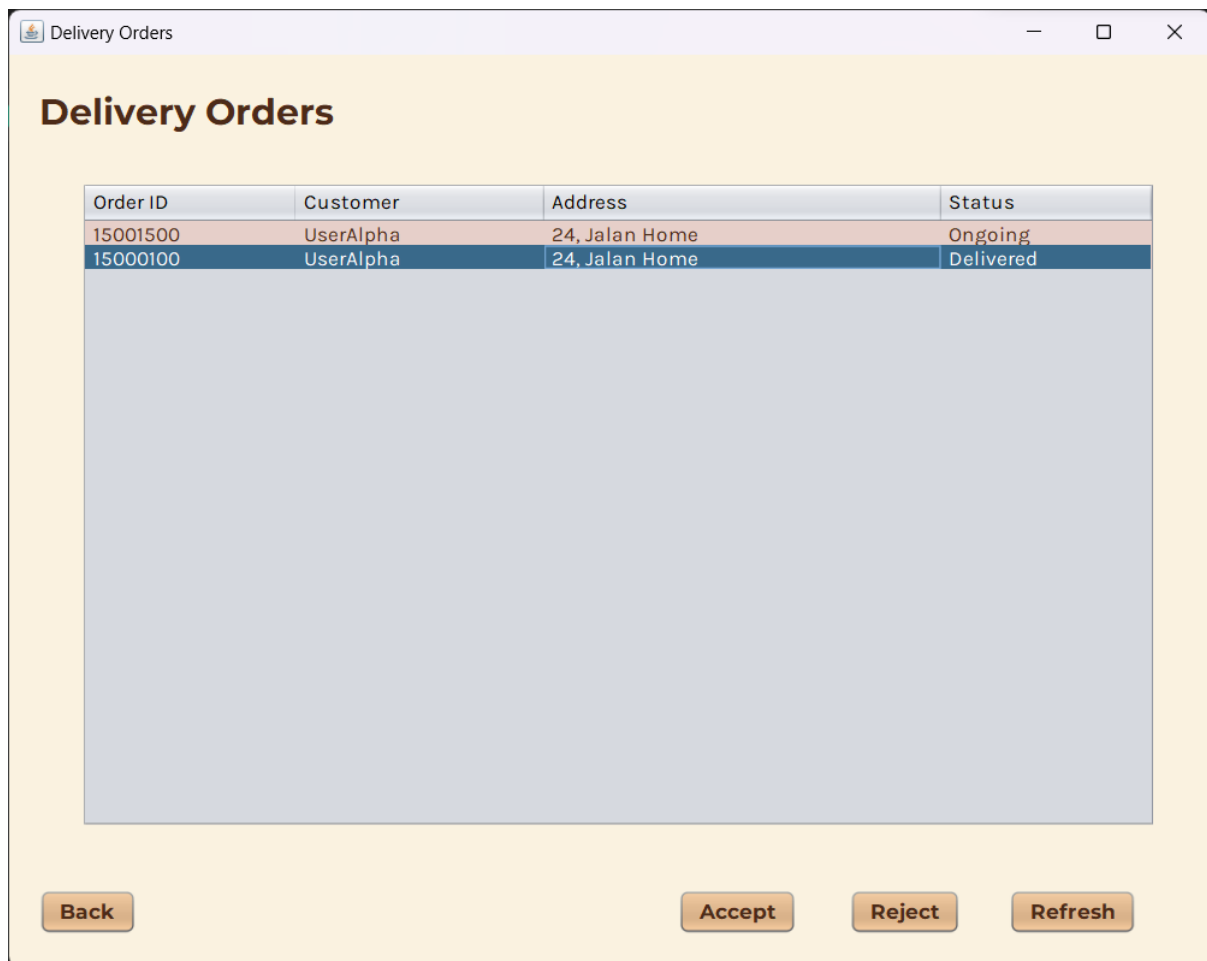
Order ID	Customer	Address	Status
15001500	UserAlpha	24, Jalan Home	Ongoing



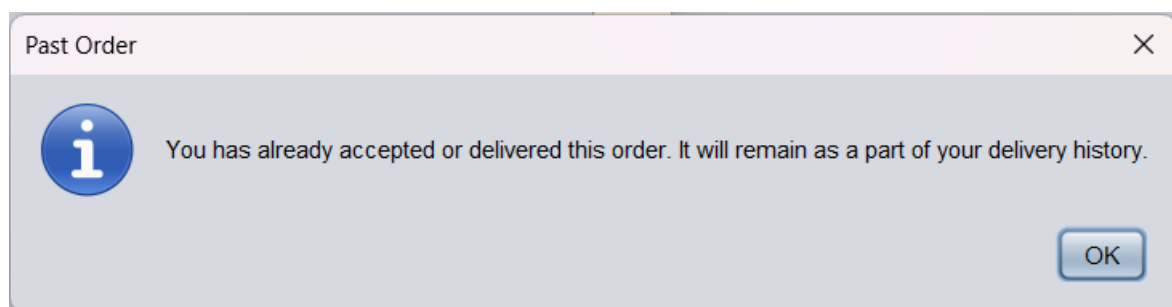
Delivery staff can view their order deliveries directly from their dashboard without the need to navigate into the Delivery Management Display. Pending accept will be their new incoming orders which they will need to accept before making a delivery. Delivering are all delivery orders that the delivery staff has accepted and currently making the errands.

6.3.2 Delivery Management

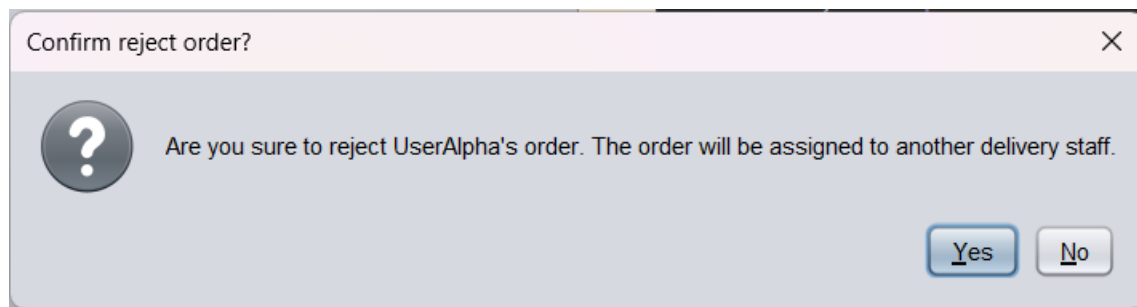
If the delivery staff wishes to manage their delivery orders, he or she can navigate to the Delivery Management Display. The delivery staffs will have their exclusive delivery orders which has been assigned under their name in display. They will have the option to accept or reject delivery orders assigned by the system admin.



When they choose to accept an Ongoing or Delivered order, the system will prompt that the selected order has been accepted or delivered by the delivery staff.



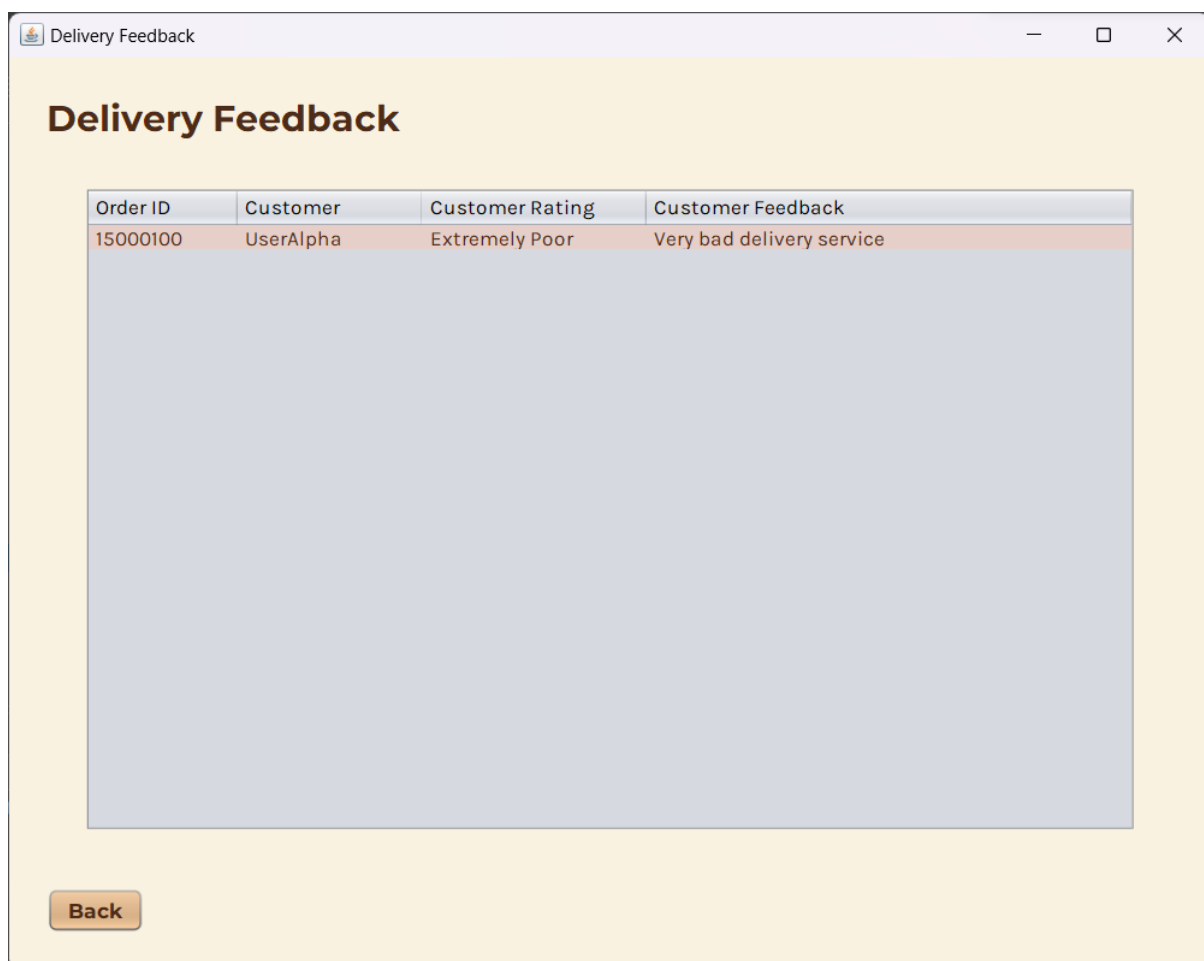
If the delivery staff is unable to make a delivery order, they can choose to reject the delivery orders by pressing the Reject button. The system will prompt for confirmation whether they want to reject the selected customer's order.



After accepting or rejecting any delivery order, the delivery staff will need to refresh the table to view their orders has been updated or reopen the Delivery Management Display by navigating back to the Staff Dashboard and back.

6.3.3 View Feedback

Whenever a delivery staff has completed a delivery order and the customer has confirmed that they received, paid for the parcel and made a feedback, delivery staff can see the customers' feedback on each delivery errand that is exclusively for them. They will not see other delivery staffs' feedbacks from customers because delivery staffs have their own Staff Dashboard.

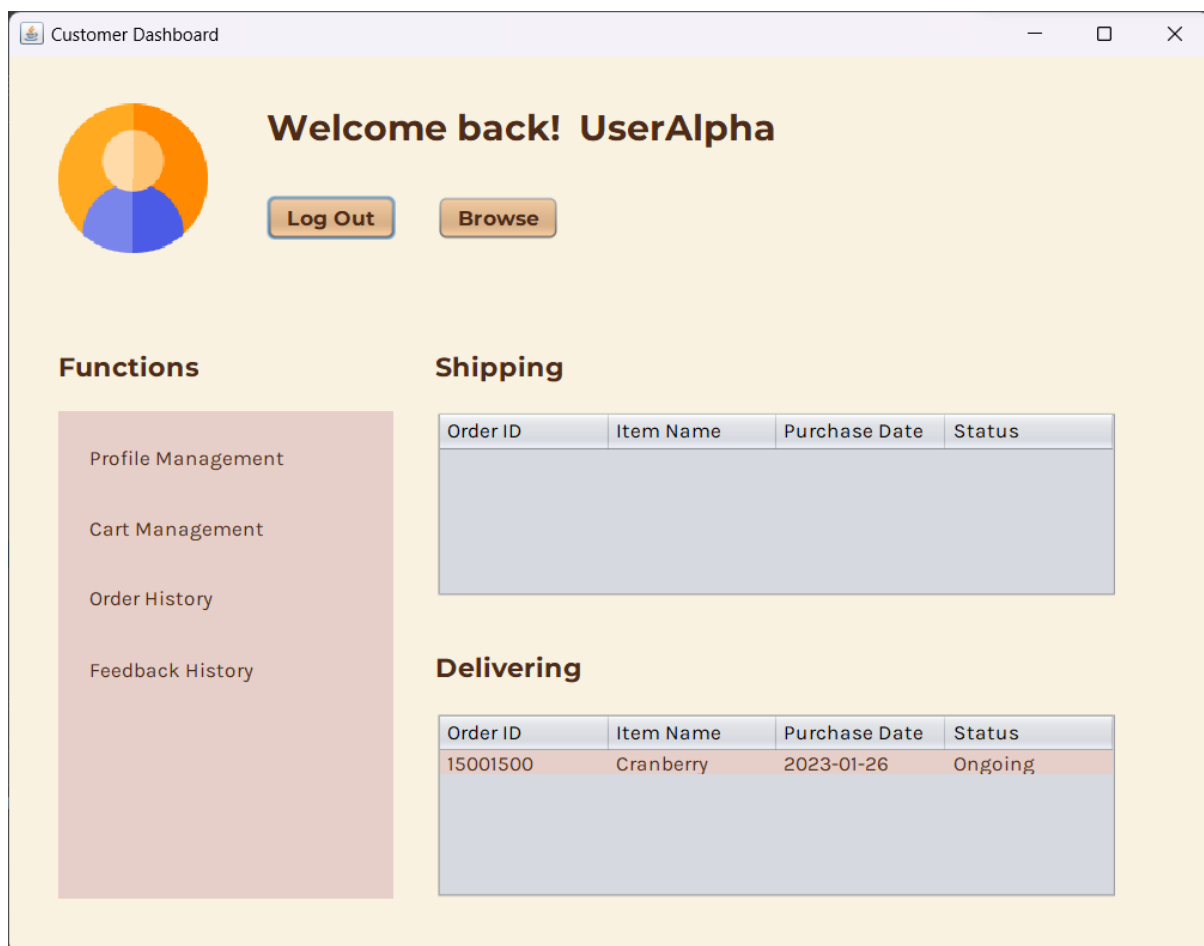


6.4 Customer

6.4.1 Customer Dashboard

All registered customers will have their own personal Customer Dashboard. Customers can view their delivery orders easily from their dashboard. There will be two tables available for customers to keep track of their delivery orders, one is shipping where their delivery orders are pending for delivery, another one is delivering where their delivery orders are out for delivery. The table will also show their purchased date which is the date when they placed the order.

Other than that, customers will also have access to their own profile management, cart management, order history and feedback history. Customer can press the Browse button to be redirected to the Online Speed Mall where they search for different items to purchase.



6.4.2 Online Speed Mall

When the customer wants to search for items or make a purchase, the customer can press the Browse button and it will open up the Online Speed Mall.

The screenshot shows the 'Online Speed Mall' application window. The title bar says 'Online Speed Mall'. The main content area has a header with the title 'Online Speed Mall' and the tagline 'Great deals always come with great delivery speed'. Below the header, there are two main sections: 'Search' and 'List'. The 'Search' section on the left contains a text input field for 'Enter Name:', a 'Search' button, a 'Filter Category:' dropdown menu, and a 'Details' section. The 'Details' section has input fields for 'Item Name:', 'Price (RM):', a text area for 'Description:', and a 'Quantity:' spinner set to 0. There is an 'Add To Cart' button and a 'Back' button at the bottom left. The 'List' section on the right is a large empty box with a header table.

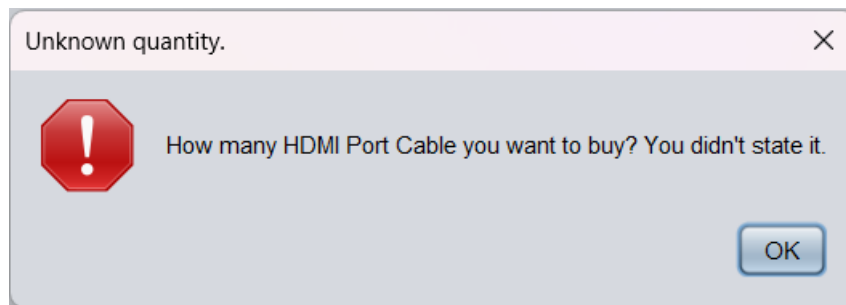
Item Name	Price (RM)	Category	Description
-----------	------------	----------	-------------

Customer can search for their items or filter through the categories available in the Online Speed Mall.

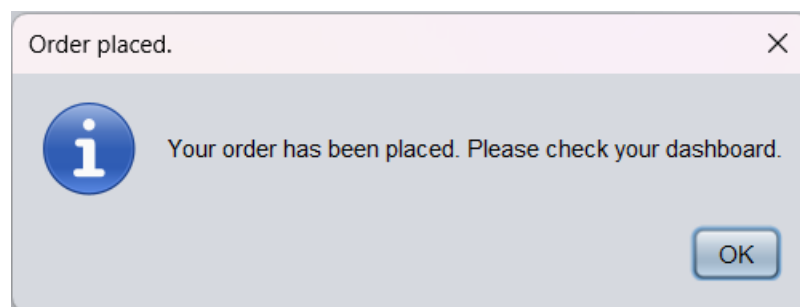
This screenshot shows the same 'Online Speed Mall' application window, but now the 'List' section on the right contains data. The 'Search' section is identical to the previous screenshot, with the 'Filter Category' dropdown set to 'Food and Beverages'. The 'List' section now displays a table with four rows of data.

Item Name	Price (RM)	Category	Description
Cranberry	4.90	Food and Beverages	Organic from Australia
Strawberry	3.90	Food and Beverages	Local from Cameron Highlands
Blueberry	5.90	Food and Beverages	Organic from New Zealand
Blackberry	7.90	Food and Beverages	Organic from Japan

If the customer wants to make a purchase, he or she will need to select the item and input the quantity he or she want to purchase. If not, the system will prompt that no quantity is stated, and the order will not be taken.

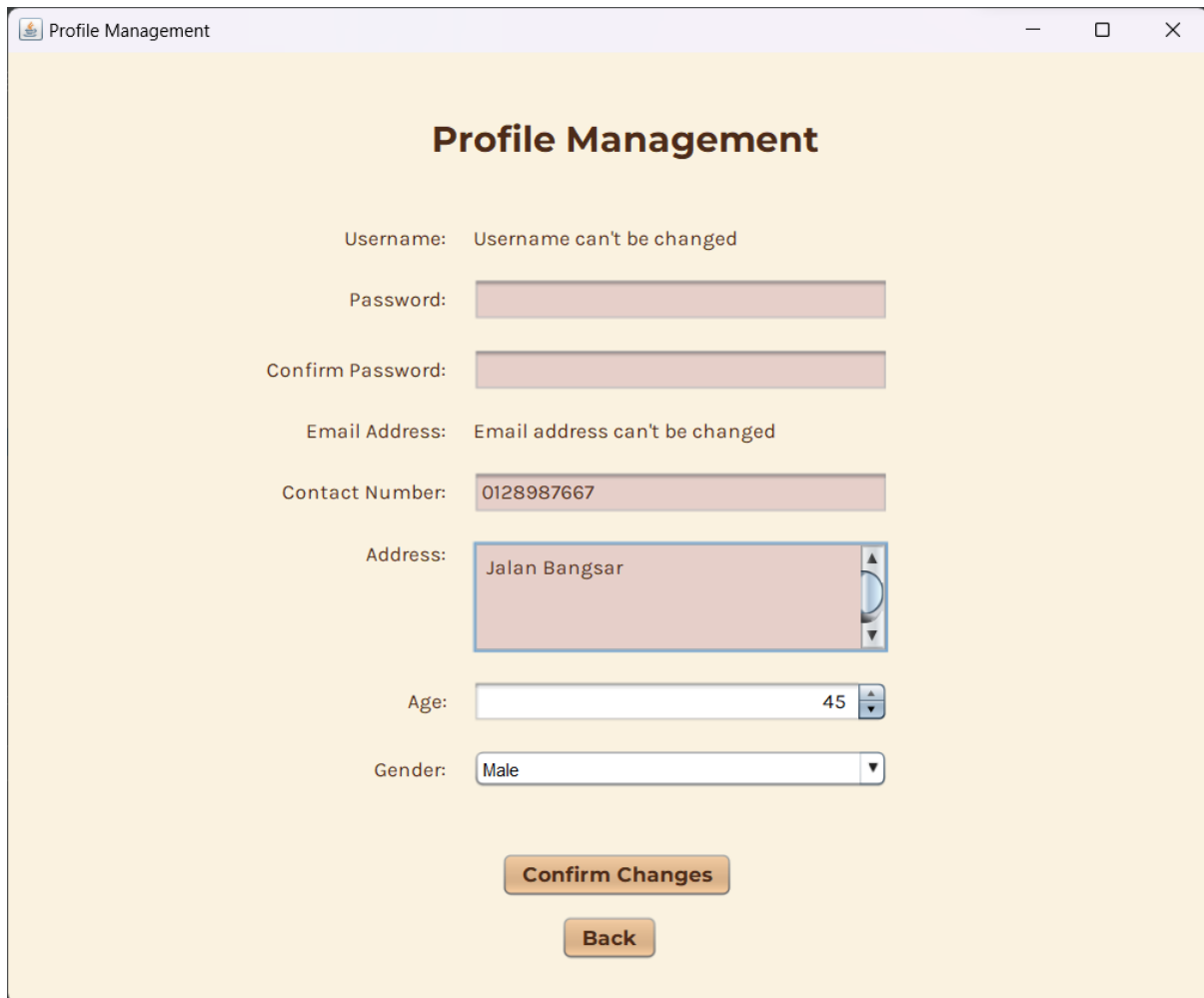


When the customer has inputted a quantity for the item he or she wished to purchase, a random cart ID within the range of 1 million will be generated. Noted that the cart ID might be duplicated but it will be in 1 of 1 million chance to have two similar cart IDs.



6.4.3 Profile Management

If the customer wants to change his personal details, he or she can navigate to the Profile Management Display. The customer can change most of the personal details except for username and email address. If the customer insists to change their username and email address, assuming there is a customer service for SPOODMS, they can request the system admin to change it.

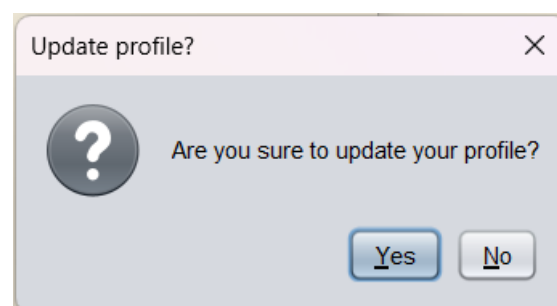


The screenshot shows a window titled "Profile Management" with a light yellow background. The title bar includes a small icon, the text "Profile Management", and standard window controls (minimize, maximize, close). The main content area is titled "Profile Management" in bold black text. Below the title, there are several form fields:

- Username:** A text field containing "Username can't be changed".
- Password:** A text field with a light pink background.
- Confirm Password:** A text field with a light pink background.
- Email Address:** A text field containing "Email address can't be changed".
- Contact Number:** A text field containing "0128987667".
- Address:** A text area containing "Jalan Bangsar".
- Age:** A text field containing "45".
- Gender:** A dropdown menu with "Male" selected.

At the bottom of the form, there are two buttons: "Confirm Changes" and "Back".

By default, the old personal information will be displayed on each fields respectively. When the customer done changing his or her personal information, he or she can press the Confirm Changes button to update their profile details. The system will prompt the customer for confirmation before updating and check for any empty fields like previous displays.



6.4.4 Cart Management

When the customer wants to view what items do he or she have, the customer can navigate to the Cart Management Display. They can search for their item or just press the Search button to list out all items. Customer can also edit the quantity of item or delete them. When the customer

is editing the item quantity, the system will disable the purchase button until the customer save their changes.

The screenshot shows a web application window titled "Cart Management". It features a search bar on the left with a "Search" button. Below the search bar is a "Details" section with input fields for "Cart ID" (3023400), "Item Name" (Cranberry), "Price (RM)" (4.90), "Quantity" (5), and "Total Price (RM)" (24.50). There are "Edit", "Save", and "Delete" buttons at the bottom of the details section. On the right, a "List" table displays the following data:

Cart ID	Item Name	Price (RM)	Quantity	Total Price (RM)
3023400	Cranberry	4.90	5	24.50
3650632	HDMI Port Ca...	39.90	2	79.8

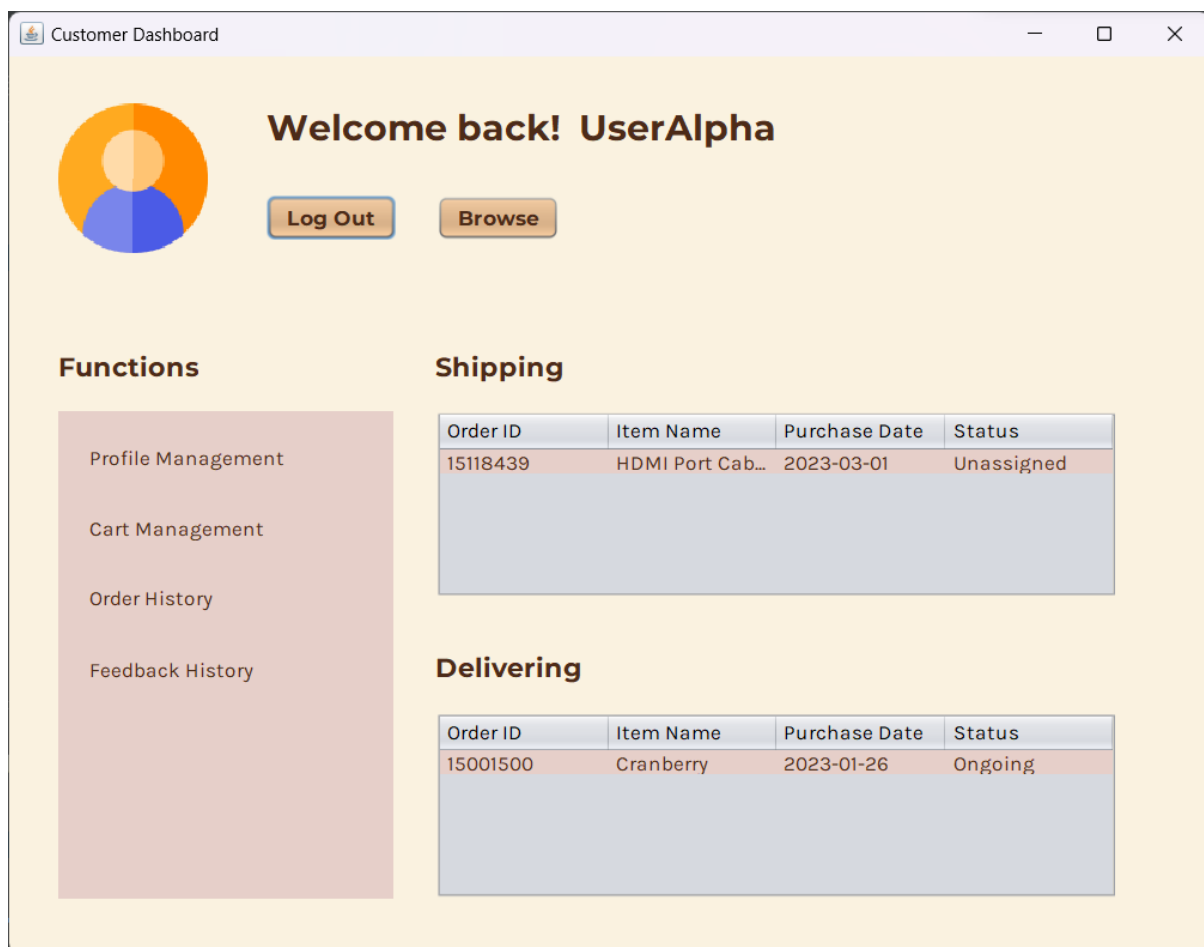
At the bottom of the window, there is a "Back" button and a "Purchase Selected Item" button.

If they want to check out their cart item, they will need to select the row of the item that they want to check out for delivery, then press the Purchase Selected Item button. The system will prompt for confirmation to avoid any mis-click from customer. After purchasing the item, the item will be saved in their order history and prepared for shipping.

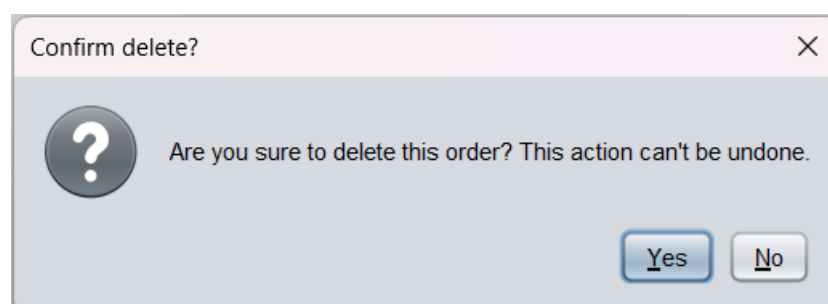
A confirmation dialog box titled "Confirm order?" with a close button (X). It contains a question mark icon and the text "Are you sure to place order?". At the bottom, there are "Yes" and "No" buttons.

A success message dialog box titled "Success" with a close button (X). It contains an information icon (i) and the text "Order has been placed. Please view your purchase detail in payment history.". At the bottom, there is an "OK" button.

The customer also views from their dashboard about their newly purchased item easily.

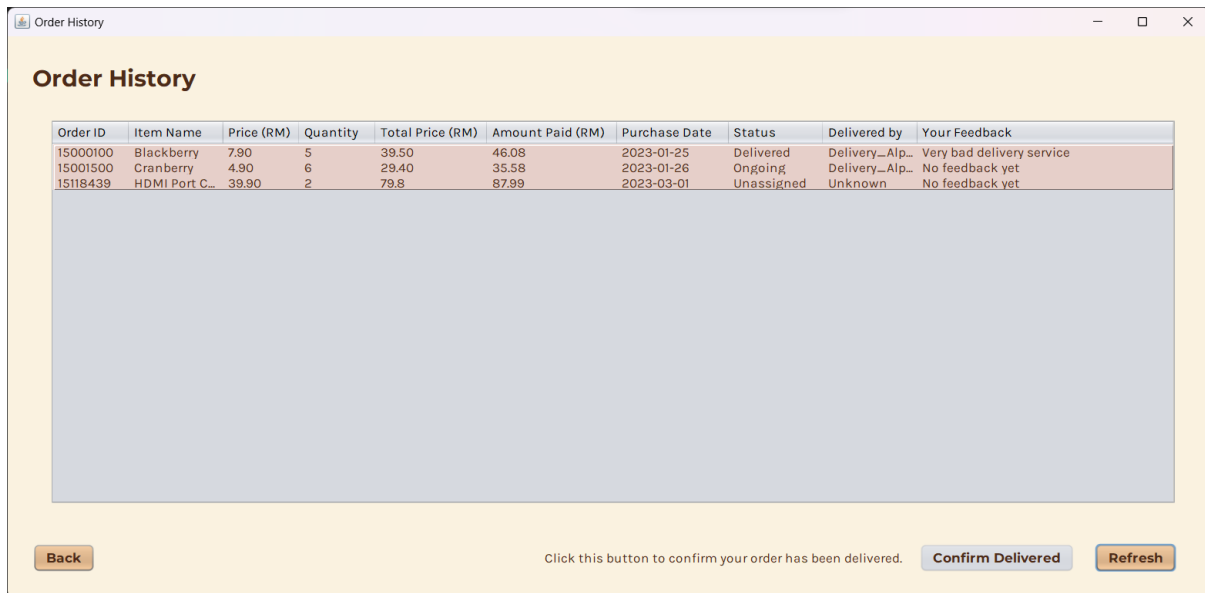


If the customer wants to delete the cart item, they just need to select the row and press the Delete button. The system will also prompt for confirmation before deleting the item from the customer's cart.



6.4.5 Order History

If the customer wants to check his or her order history, he or she can navigate to the Order History Display to view all orders. By default, the order history is written in such way that is arranged from past date to today's date.



Order History

Order ID	Item Name	Price (RM)	Quantity	Total Price (RM)	Amount Paid (RM)	Purchase Date	Status	Delivered by	Your Feedback
15000100	Blackberry	7.90	5	39.50	46.08	2023-01-25	Delivered	Delivery...Alp...	Very bad delivery service
15001500	Cranberry	4.90	6	29.40	35.58	2023-01-26	Ongoing	Delivery...Alp...	No feedback yet
15118439	HDMI Port C...	39.90	2	79.8	87.99	2023-03-01	Unassigned	Unknown	No feedback yet

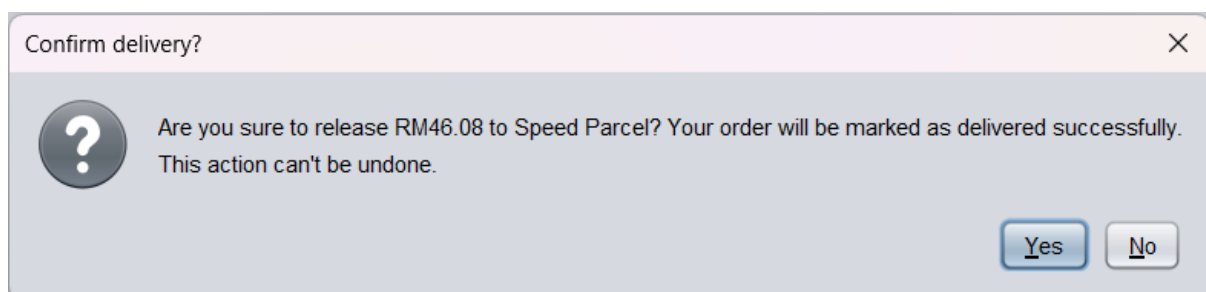
Back

Click this button to confirm your order has been delivered.

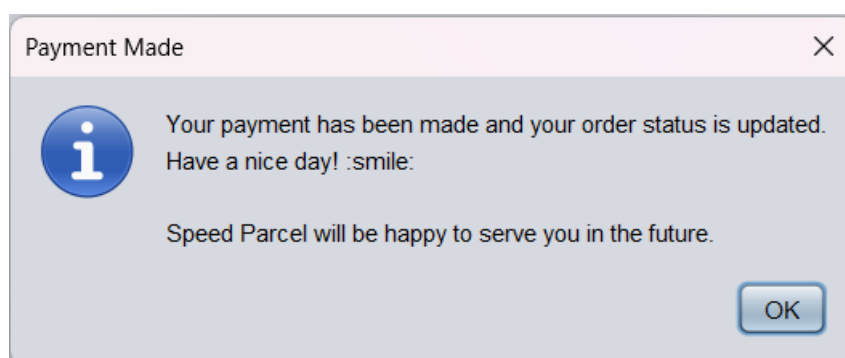
Confirm Delivered Refresh

When the customer has received his or her parcel, the customer can press the Confirm Delivered button to update the status to “Delivered”. This is to ensure that the customer has confirmed receipt of his or her parcel before releasing the payment, so that problems like dishonest delivery staff or damaged parcel can be avoided. Customer can also press the Refresh button to refresh the table to see their updated order status.

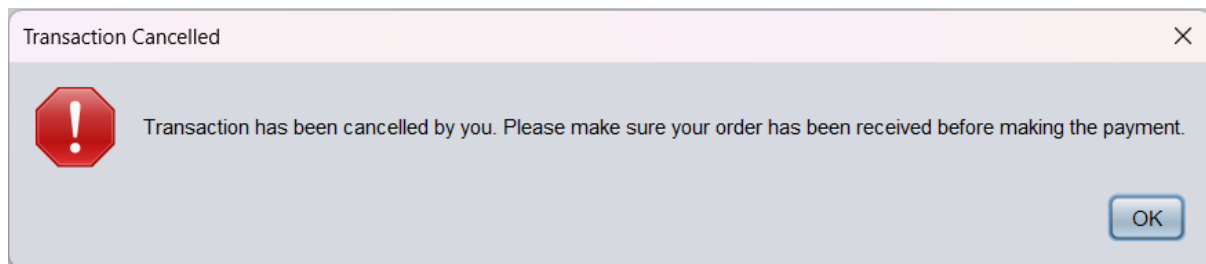
Before confirming the order, the system will prompt the customer for confirmation.



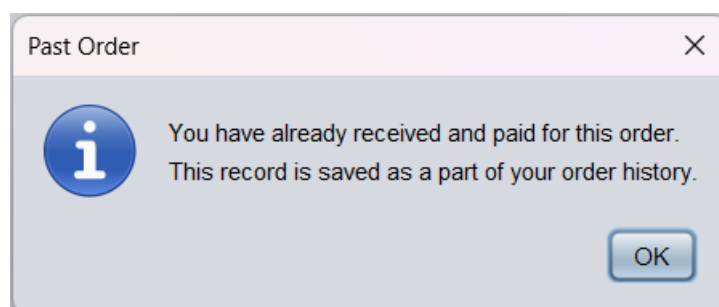
If the customer presses Yes, it will release the payment to Speed Parcel and the payment is done.



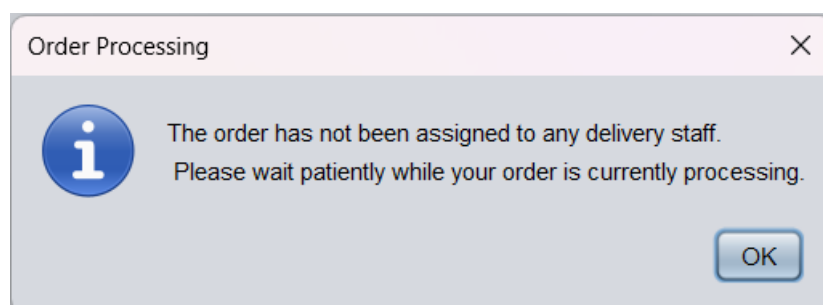
If the customer presses No, it will cancel the transaction and the customer will not pay for the order, thus the status will remain as “Ongoing”.



If the customer presses the Confirm Delivered on an order that has been delivered, the system will prompt the customer that the selected order has been delivered and paid previously.



If the customer presses the Confirm Delivered on an order that hasn't been assigned to any delivery staff, the system will prompt the customer to wait patiently as the order is still under process.



6.4.6 Feedback History

When the customer confirms an order delivery, dummy feedback will be generated but there will be no context of the rating and feedback. Customer can navigate to the Feedback History Display to give their rating from the scale of 1 to 5 and write their review about the delivery service or the item quality.

Feedback History

Feedback History

Order ID	Item Name	Amount Paid	Purchase Date	Rating	Your Feedback
1500010...	Blackberry	46.08	2023-01-25	Extremely Poor	Very bad delivery service
1500150...	Cranberry	35.58	2023-01-26	Mediocre	Average
15118439	HDMI Port Ca...	87.99	2023-03-01	No rating yet	No feedback yet

Rating:

Extremely Poor

Feedback:

Submit

Back

Refresh

Feedback History

Feedback History

Order ID	Item Name	Amount Paid	Purchase Date	Rating	Your Feedback
1500010...	Blackberry	46.08	2023-01-25	Extremely Poor	Very bad delivery service
1500150...	Cranberry	35.58	2023-01-26	Mediocre	Average
15118439	HDMI Port Ca...	87.99	2023-03-01	No rating yet	No feedback yet

Rating:

Good

Feedback:


Submit

Back


Refresh

After customer submitted their rating and feedback, he or she can press the Refresh button to refresh the table and view their updated rating and feedback which will overwrite the dummy feedback. The system will prompt for the update confirmation because customer can't edit the feedback after it is submitted.

Confirm submit?


 Are you sure to submit your feedback now? You can't change it anymore

Success

 Your feedback has been submitted. Please refresh the page to view it.
NOTE: If it contains sensitive contents, admins will not hesitate to delete your feedback and take actions upon you.

Feedback History

Order ID	Item Name	Amount Paid	Purchase Date	Rating	Your Feedback
1500010...	Blackberry	46.08	2023-01-25	Extremely Poor	Very bad delivery service
1500150...	Cranberry	35.58	2023-01-26	Mediocre	Average
15118439	HDMI Port Ca...	87.99	2023-03-01	Good	Quite on time

Rating: 

Feedback:

If the customer selects a row that has been given a rating and feedback, the system will not enable the rating JSlider and feedback JTextArea.

Feedback History

Order ID	Item Name	Amount Paid	Purchase Date	Rating	Your Feedback
1500010...	Blackberry	46.08	2023-01-25	Extremely Poor	Very bad delivery service
1500150...	Cranberry	35.58	2023-01-26	Mediocre	Average
15118439	HDMI Port Ca...	87.99	2023-03-01	Good	Quite on time

Rating:

Feedback:

Submit

Back

Refresh

7.0 Conclusion

Speed Parcel Online Order and Delivery Management System (SPOODMS) might not be a 100% user-friendly or having the top functionalities, but it provides enough information to demonstrate the advantages of object-oriented programming with Java. SPOODMS can be improved by implementing more functionalities in future versions since it is object-oriented so it is easily scalable and adjustable without tampering the main structure of the system.

8.0 References

Half., R. (2023, January 5). 4 Advantages of Object-Oriented Programming. *Robert Half Talent Solutions*. Retrieved from: <https://www.roberthalf.com/blog/salaries-and-skills/4-advantages-of-object-oriented-programming>

9.0 Appendix

9.1 JFreeChart

An external library called JFreeChart version 1.0.19 is used to generate bar graph and pie chart in Speed Parcel Online Order and Delivery Management System. This library is versatile to generate other graphs like line graphs, scatterplot, etc. More details can be found at <https://www.jfree.org/jfreechart/>.

It is worth knowing that JFreeChart will need to be downloaded and imported into the project locally or else the project can't be run. The installation guide and importing steps can be seen from <https://www.youtube.com/watch?v=aBONSQ44cnk>.

[illegible]