GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

Module Code	:	CT083-3-2-OODJ
Intake Code	:	APU2F2411CS(AI)
Lecturer Name	:	Muhammad Huzaifah Bin Ismail
Hand In Date	:	22nd February
Hand Out Date		22nd February
Tutorial No.	:	Lab 38
Group No.	:	Group 21
Group Leader	:	Ong Kai Ying

Student ID	Student Name
TP070517	Kate Lim Jia Yee
TP071290	Manreen Kaur A/P Jagjit Singh
TP086065	Ong Kai Ying
TP072694	Tee Ching Ying

Table of Contents

1.0 Introduction

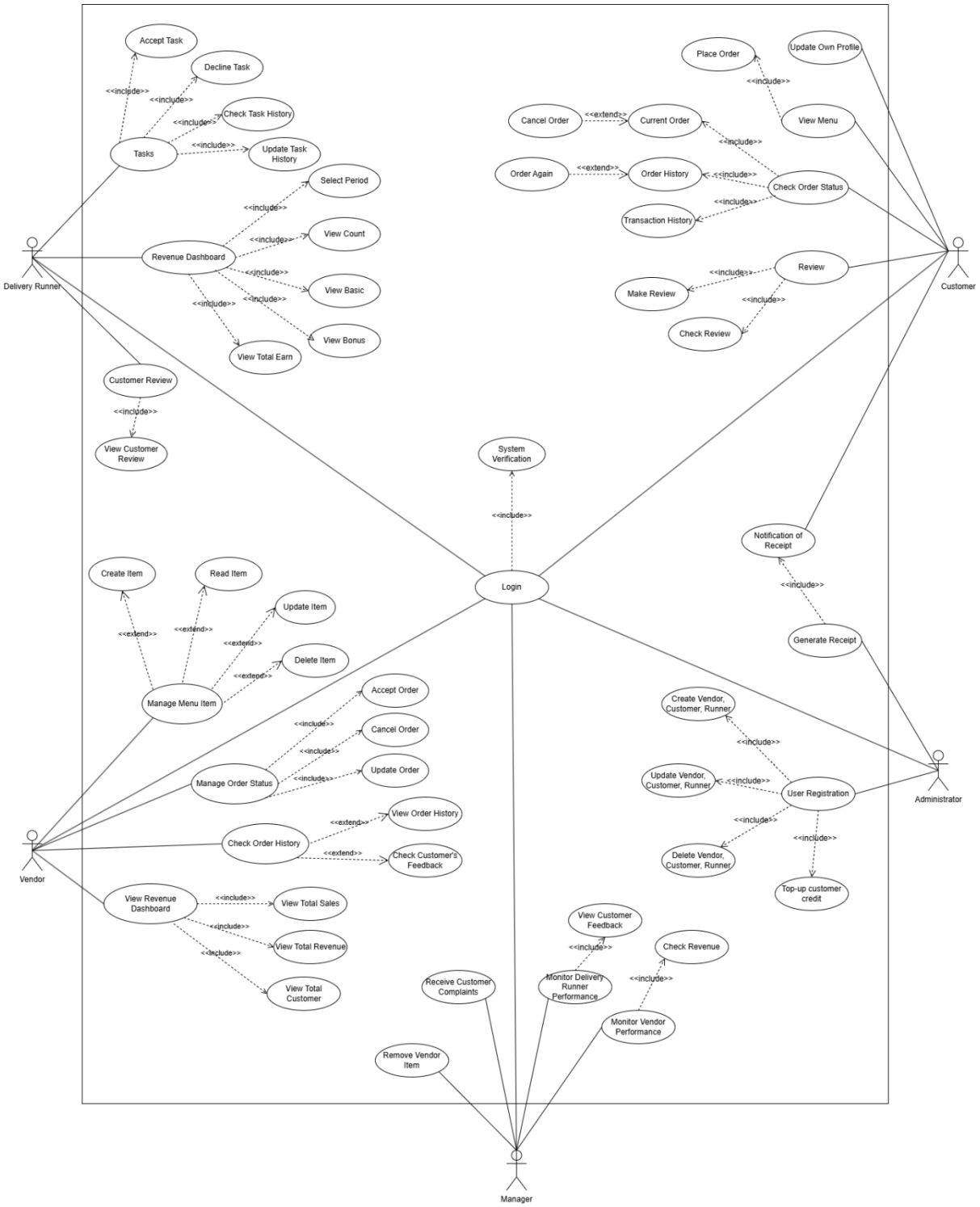
In today's fast-paced world, the way we manage our daily activities is constantly changing, especially in the food industry. Traditional methods of placing orders at food courts are becoming outdated and inefficient. With the increasing demand of online ordering, delivery services, as well as digital payments, people expect faster and more convenient ways to get their food and drinks.

In light of these changing dynamics, a food court management system has become necessary to meet the needs of all users including customers, vendors, delivery runners, manager as well as administrators. This application aims to provide an all-in-one platform that simplifies the process of food ordering, payment, delivery and so on. Even better, this application not only allows customers to easily browse menus, place orders, and select delivery options, but it also integrates a digital wallet for seamless payments and an in-app notification system for real-time updates. Furthermore, vendors can manage orders, track business performance, and even view feedback from customers. Administrators can manage user registrations and overseeing the system in order to ensure smooth operations.

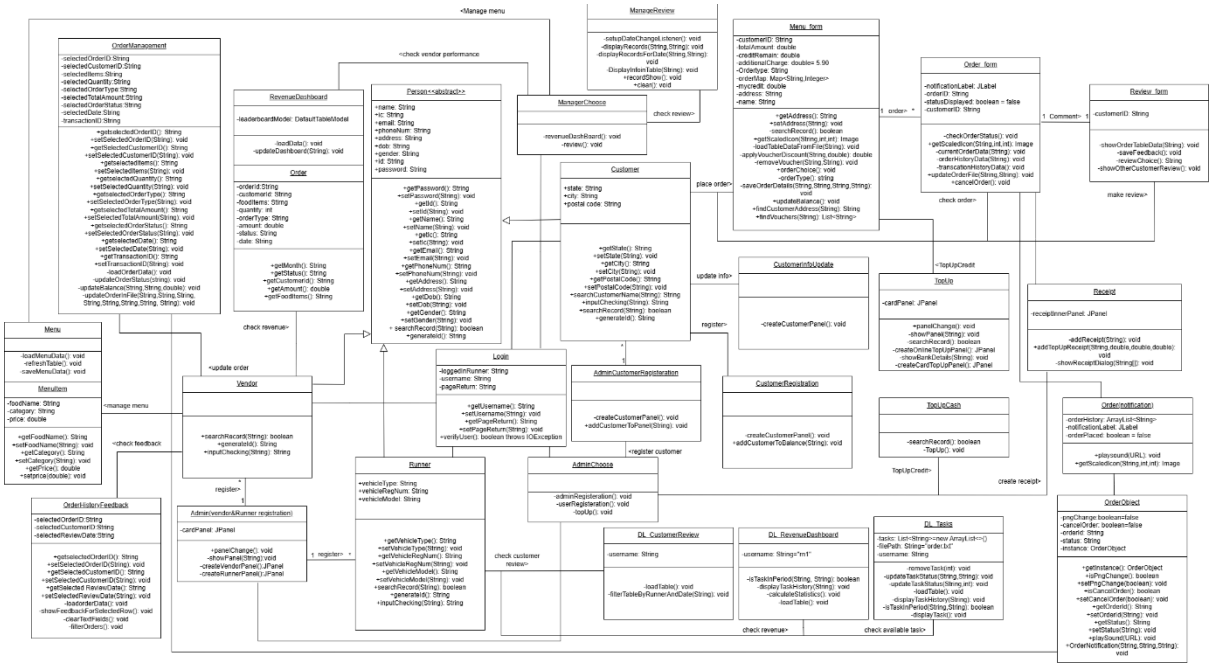
Overall, this management system can play an important role in simplifying the food ordering process, improve business operation and also offer a better experience for all users involved.

2.0 Design Solution

2.1 Use Case Diagram



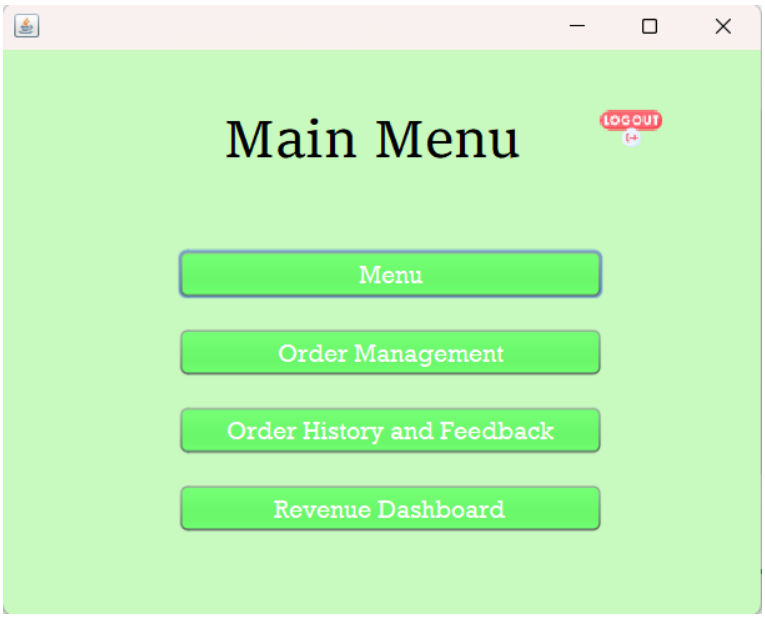
2.2 Class Diagram



3.0 Screenshots of Output

3.1 Kate Lim Jia Yee (Vendor)

Main Menu

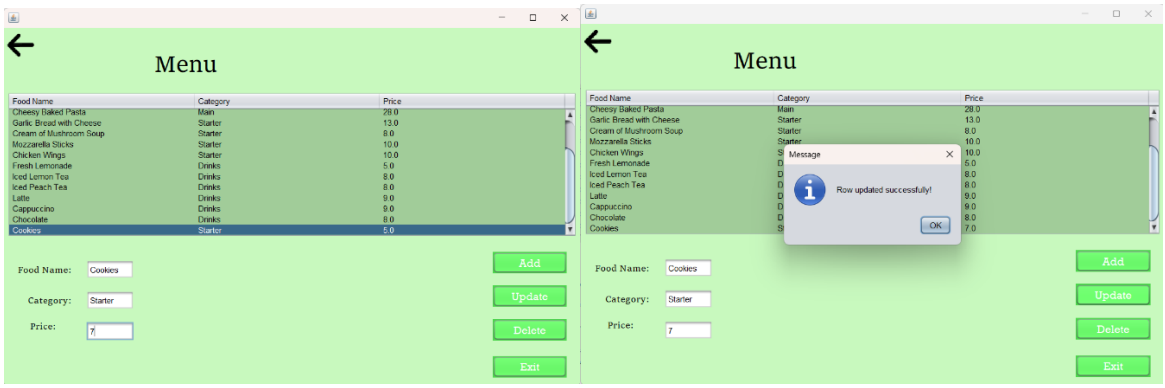


In this main menu page, a main menu interface is developed for vendor to efficiently manage business. This main menu page has four main options, which are Menu Management, Order Management, Order History and Management, as well as Revenue Dashboard. When vendor enter Menu Management page, they can view all available food items, and they have the functionality to add, edit, or even delete items from the menu. Moreover, in order management, vendor can receive customer orders and choose to either cancel or accept their order, with the system notifying customers of any updates regarding their orders. When vendor enter Order History and Management, they can view past orders and customer feedback by day, month or quarter. Lastly, the Revenue Dashboard presents a detailed summary of monthly sales and food rankings, which allow vendor to analyse business performance effectively.

Menu

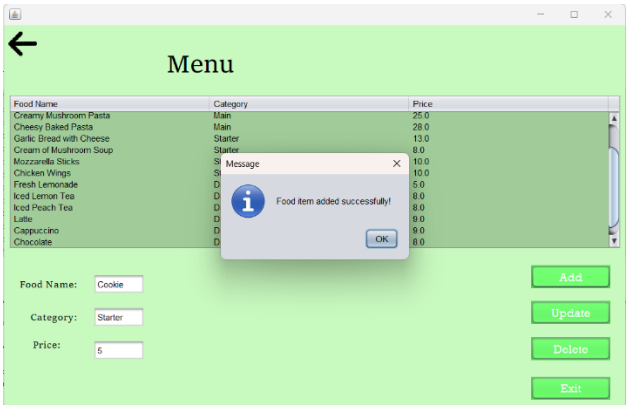
In the Menu Management page, all available food items including starters, drinks and main dishes are displayed in a JTable with three columns which are Food Name, Category, as well as Price. This interface provides a convenient way for vendor to manage the menu. For instance, vendor can simply type the food name, category and price into respective fields and click the Add button to add a new item. Upon doing so, the new item will be automatically added to the JTable and saved in the system. Similarly, vendor can also update an existing item by just selecting it from the table, editing the details and clicking the Update button. Furthermore, items can be removed by selecting them and clicking Delete button.

Update Menu Item



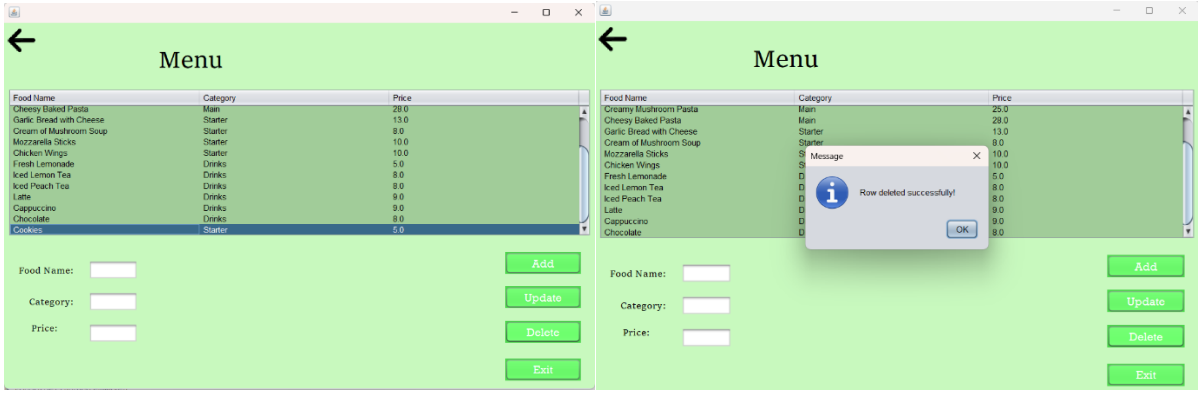
Vendors have the ability to update menu item. For instance, vendor can select the item they wish to update and type the new detail. After that, vendor can then click the button “Update” and a message with text “Row uploaded successfully!” will be shown which indicating vendor has successfully update menu item.

Add Menu Item



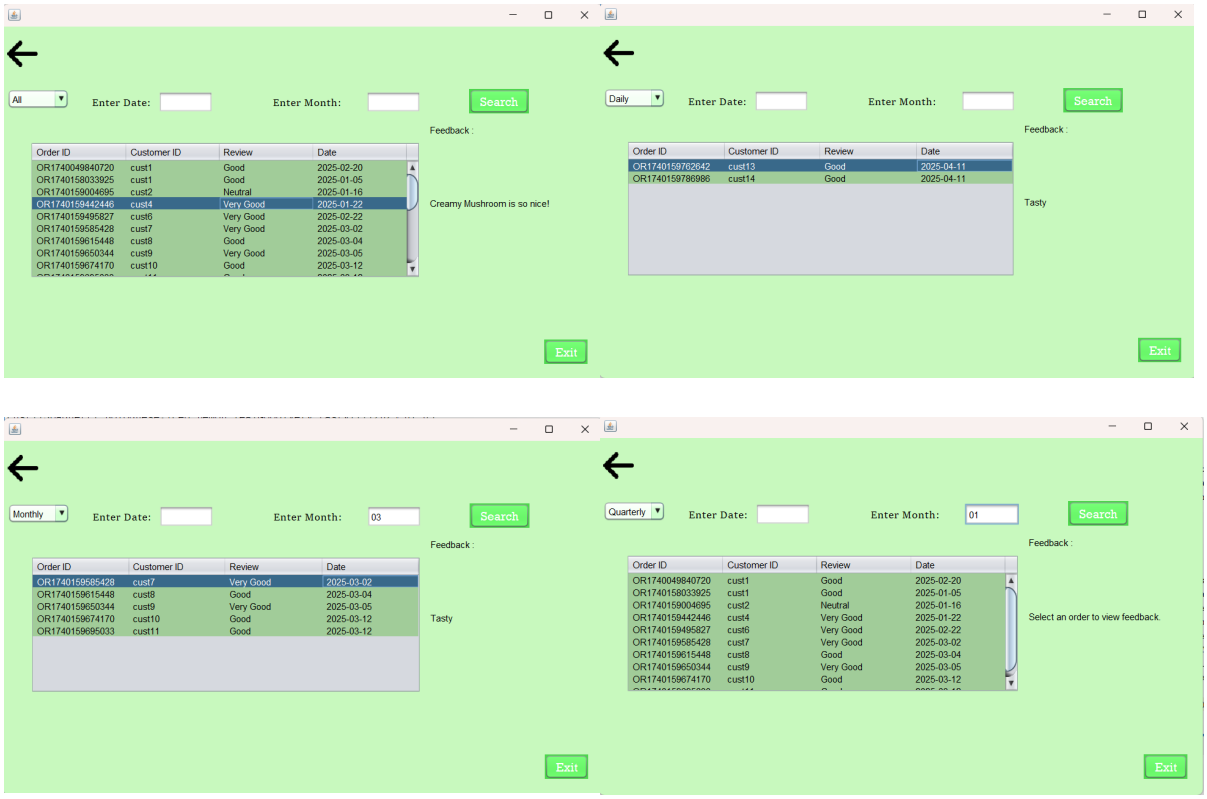
Vendors can also add new item into menu. For instance, vendor can enter the item they wish to add and provide basic information such as food name, category as well as price. After that, vendor can then click the button "Add" and a message with text "Food item added successfully!" will be shown which indicating vendor has successfully added a new item.

Delete Menu Item



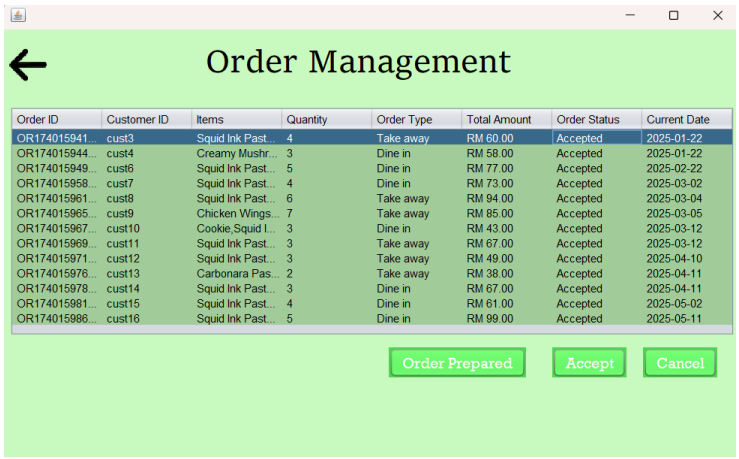
Other than add and update menu item, vendors can also delete an item from the menu. For instance, vendor can select the item they wish to delete and click the button "Delete" and a message with text "Row deleted successfully!" will be shown which indicating vendor has successfully delete an item.

Order History and Review



In this Order History and Feedback page, vendor can review customer order histories and feedback. This page also allows vendor to filter the order history by selecting options to view all orders, daily orders or monthly orders. All customer order records are displayed in a JTable with four columns which are Order ID, Customer ID, Review and Date. In order to view customer feedback, vendor can highlight a specific order from the JTable and the corresponding feedback will be displayed beside the table.

Order Management



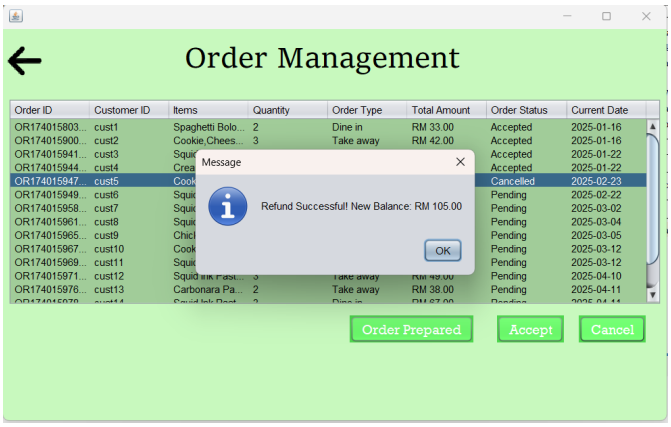
In Order Management page, vendor receive real-time notifications whenever a customer places an order. This page features a JTable displaying key details such as Order Type (Dine in,

Takeaway or Delivery), Total Amount, Order ID and the like. Upon receiving an order, vendor has the option to either Cancel or Accept it. Based on vendor's selection, the order status in the JTable is updated to Cancelled or Accepted. Once the food is prepared, vendor can update the order status as Order Prepared, and the order will be remove from the jTable.

Accept Order

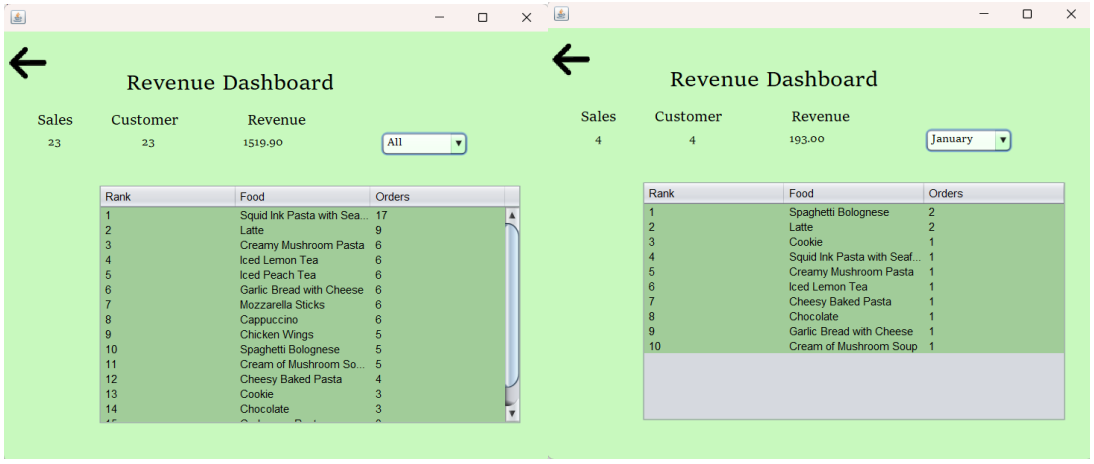
If vendor accepted the order, the order status will change from "Pending" to "Accepted" and inform customer at the same time.

Cancel Order



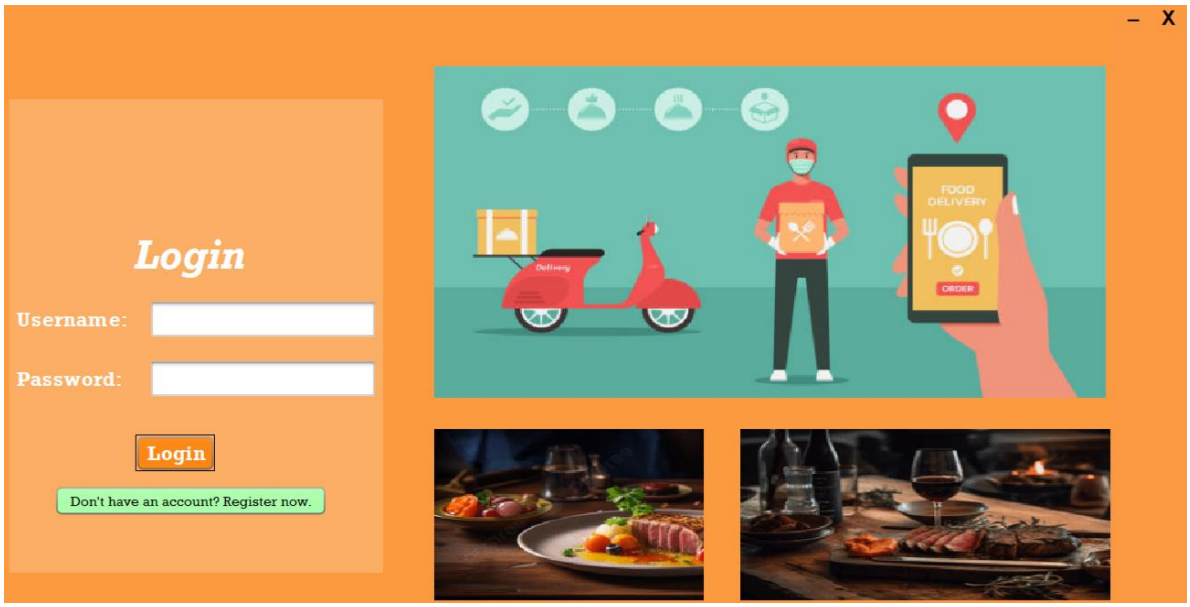
However, if vendor cancelled the order, it will automatically refund the amount to customer and remove from the order management immediately.

Revenue Dashboard

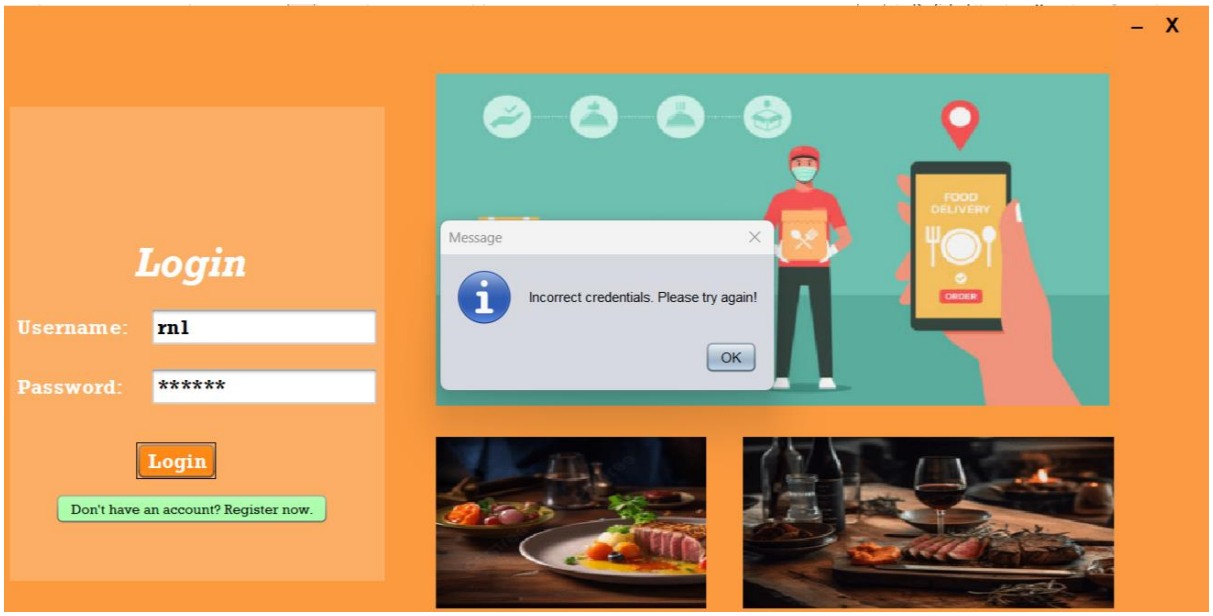


In Revenue Dashboard page, it provides a detailed analysis of the restaurant's performance. Vendor can view key metrics such as Sales, Customers, as well as Revenue either for a specific day or a selected month. The page also features a JTable that organizes menu items by their popularity. For instance, if 40 customers ordered Chicken Bolognese, the table will list Chicken Bolognese as the top-ranked item, along with the total order count. From the image above, we can clearly see that Squid Ink Pasta with Seafood has the highest contribution will is 17 orders.

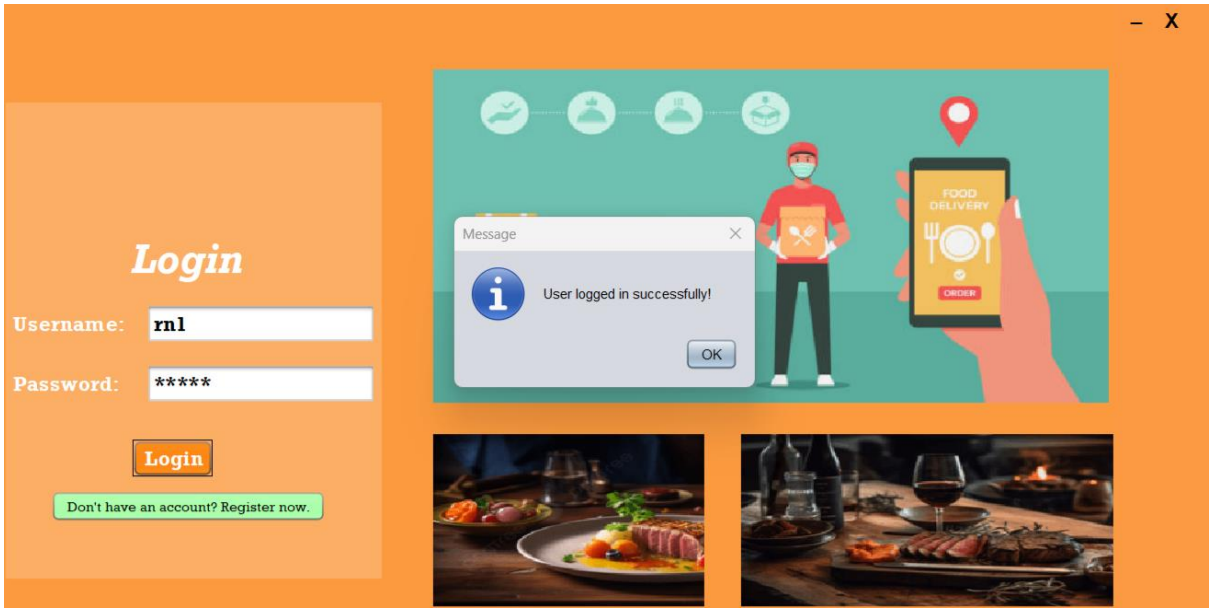
3.2 Manreen Kaur A/P Jagjit Singh (Delivery Runner)



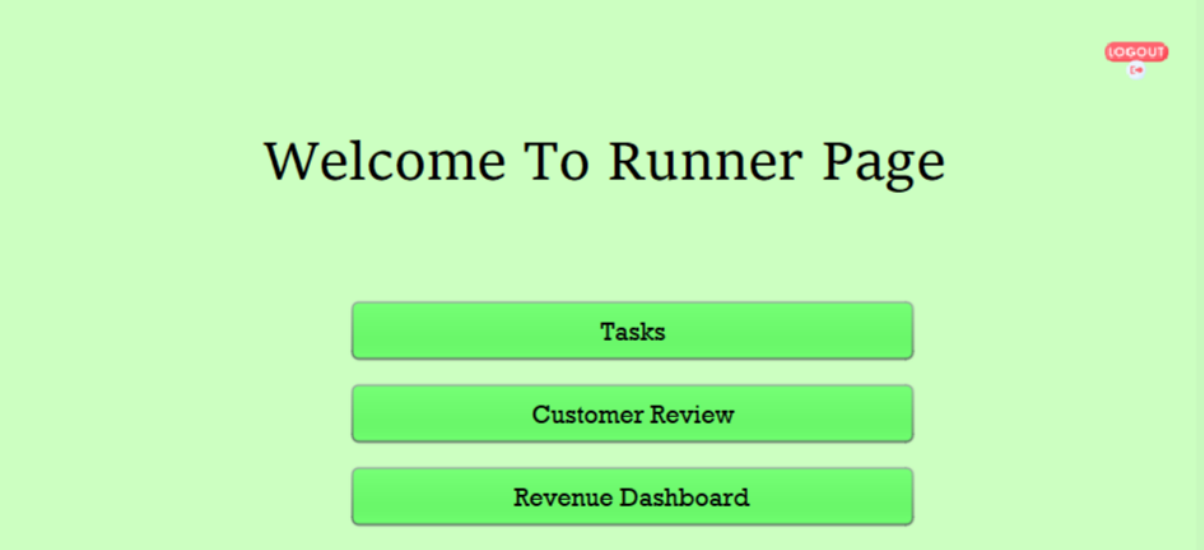
The delivery runner should log in using their own username and password that was made by the admin to access the application. For instance, in the text field, insert the Username and the Password, then click the button "Login".



If the login fails, a popup notification will appear, and the user will need to rewrite the correct credentials to log in.

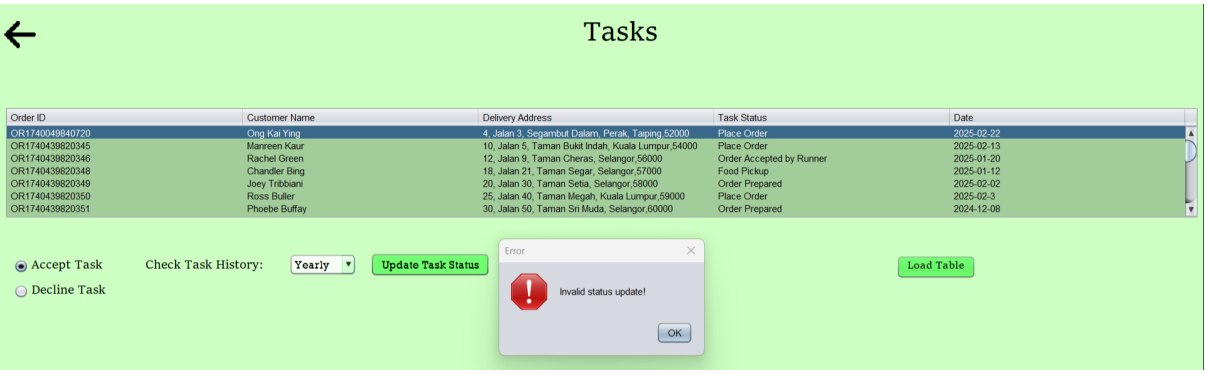


If the login is approved, a popup notification will appear, and the user can click the “OK” button to go to the main page of its delivery runner.



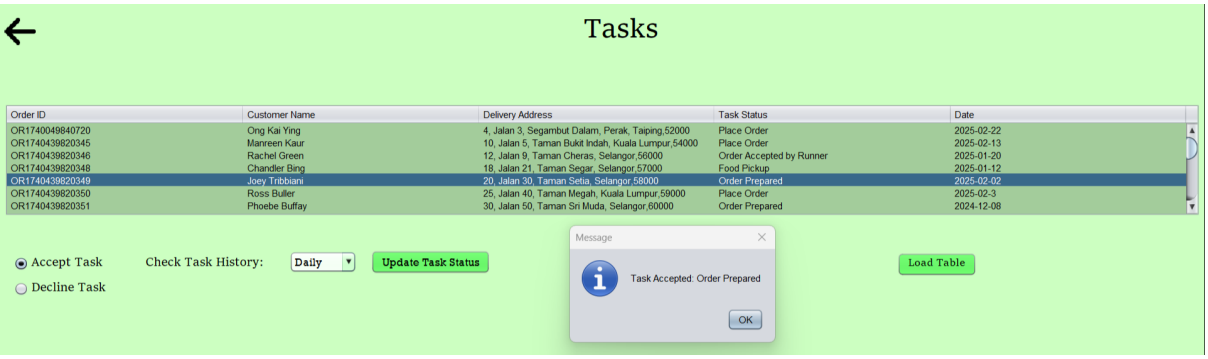
This is called the Main page, the delivery runner can choose to click the “Tasks” button, “Customer Review” button, or “Revenue Dashboard” button to go to its corresponding page. The user can also choose to log out by clicking the “LOGOUT” button, then the delivery runner needs to re-login using its credentials on the Login page.

This is called the Tasks page, the delivery runner can accept or decline tasks, check task history, update task status, and load the table. The table displays OrderID, Customer Name, Delivery Address, Task Status, and Date that is taken from the text file called “orders.txt”.

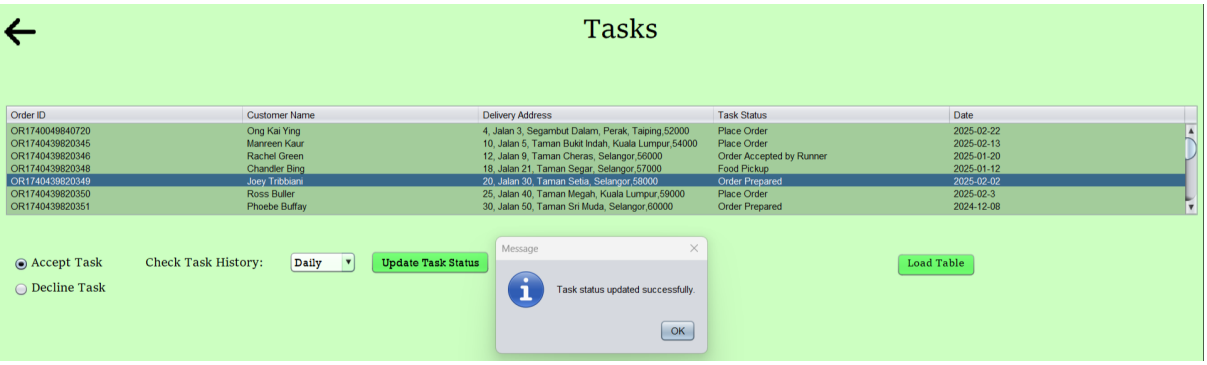


The task status like Place Order can't be modified. It will show a pop-up notification that can't update the status. Logically, the tasks can only be updated based on delivery runner processes such as when the delivery runner accepts the tasks they can update from "Order Prepared" to "Order Accepted by the Runner", to "Food Pickup", to "Delivering", and finally to "Completed" so since "Place Order" is customer's process therefore delivery runner has no access to change it, and the "Order Prepared" can only be saved status by the Vendor when the food is already prepared. Therefore, only after the food has been prepared then only the delivery runner can update its status to accept the task and continue its processes to deliver to the customer's address.

When the delivery runner accepts the task then it will be updated based on delivery runner processes

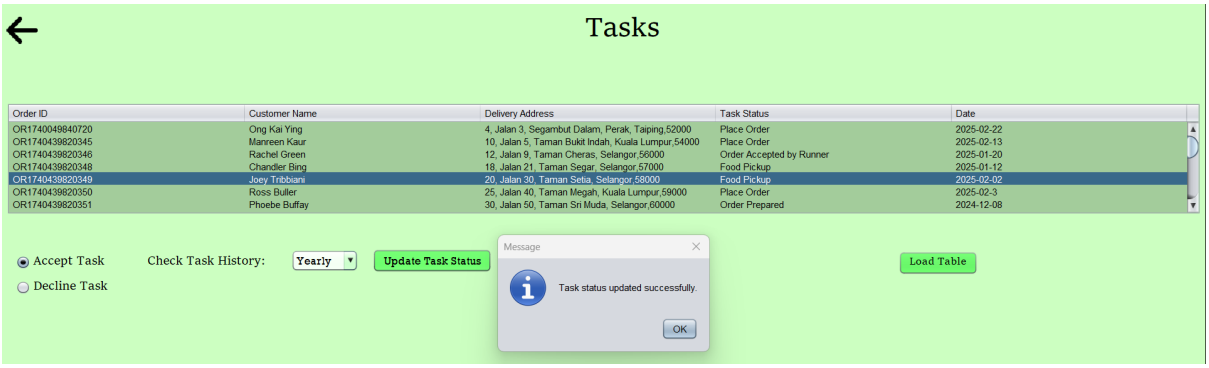


First, click the row to select the task then click the "Accept Task" button to accept the task. Once clicked, it will display a pop-up that the task has been accepted for the Order Prepared.

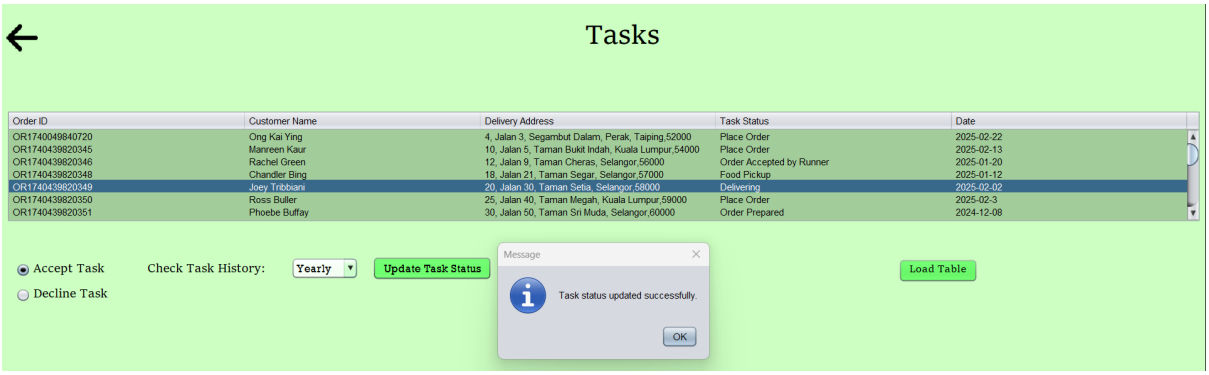


Then click on the "Update Task Status" button to update from "Order Prepared" to the next step "Order Accepted by Runner".

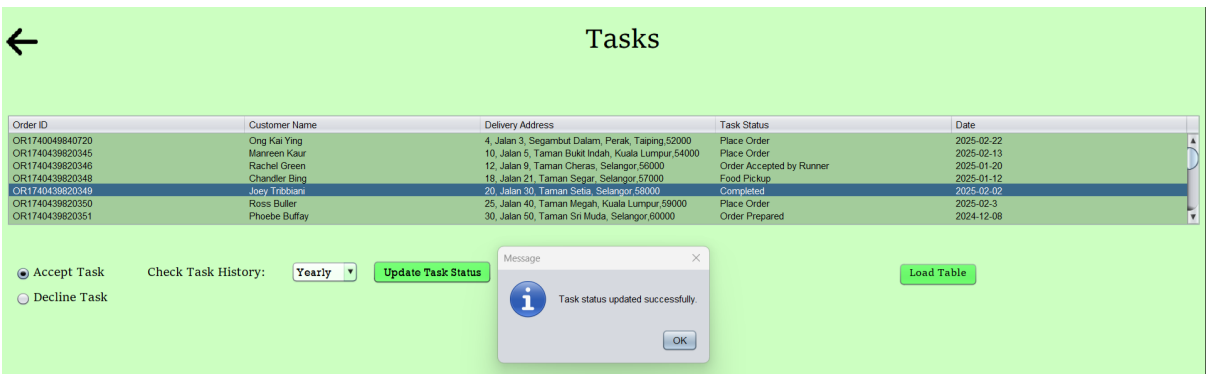
Now the status on the table will be displayed as “Order Accepted by Runner”.



Click on the “Update Task Status” button again to update from “Order Accepted by Runner” to the next step “Food Pickup”.

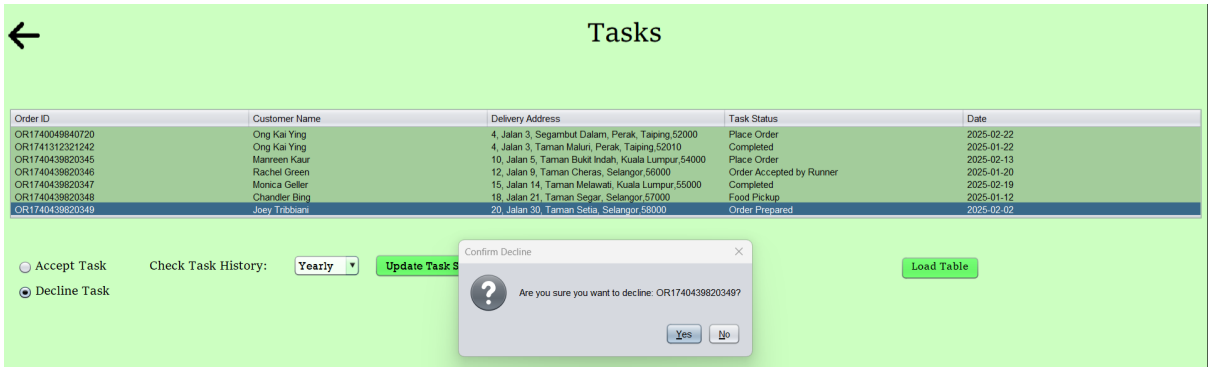


Click on the “Update Task Status” button again to update from “Food Pickup” to the next step “Delivering”.



Click on the “Update Task Status” button again to update from “Delivering” to the next step “Completed”.

Once the task status has been saved to completed it won’t appear on the table, it will only be saved in the text file called “orders.txt”.

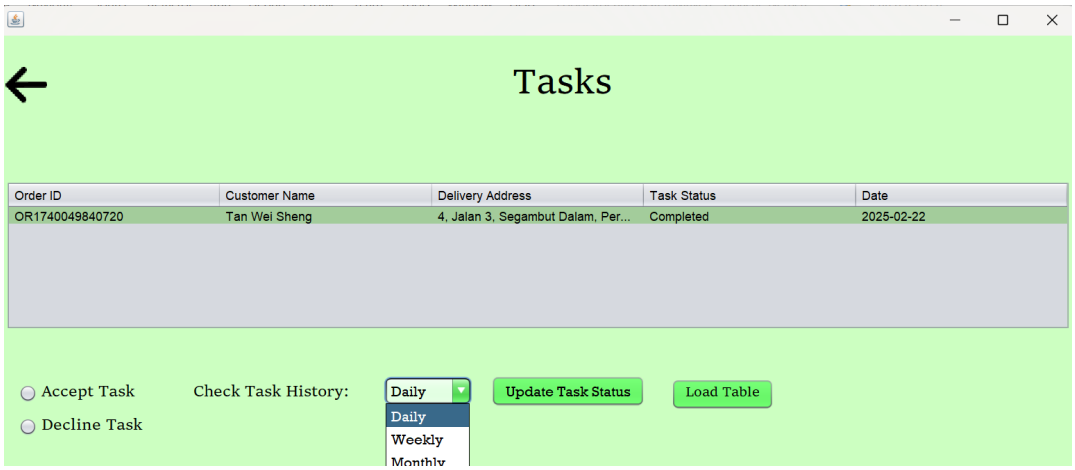


Whereas, if the delivery runner wants to decline the task, then the delivery runner must select the row of the task that wishes to be deleted. Then click on the “Decline” button to decline the task, then click “Yes” if you want to delete otherwise click “No”.

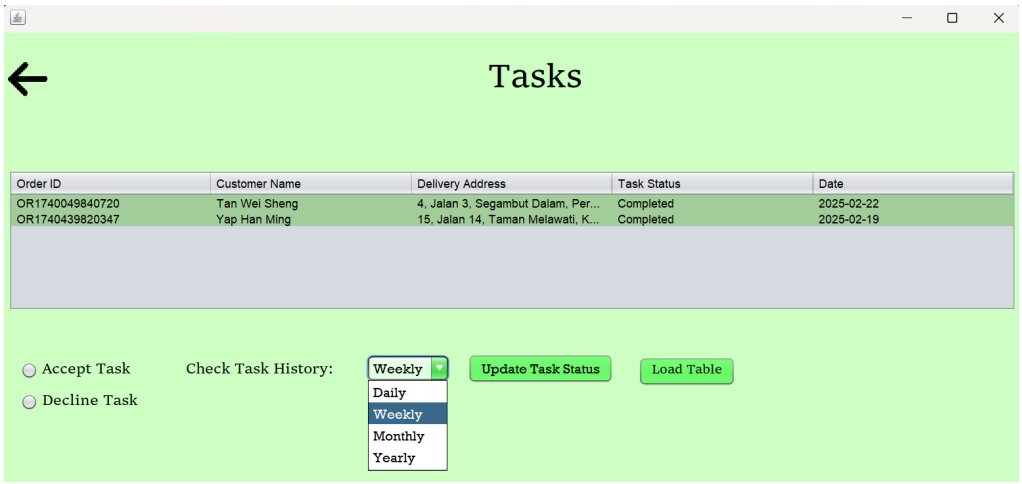
It is displayed that the declined tasks won’t appear on the table.

Select Task History “Daily” “Weekly”, “Monthly”, and “YEARLY”. Based on this table the “Yealy” task history is selected. Therefore, it only displays the Yearly which is this year “2025” all tasks. After completed it won’t appear on the table, it will only be saved in the text file.

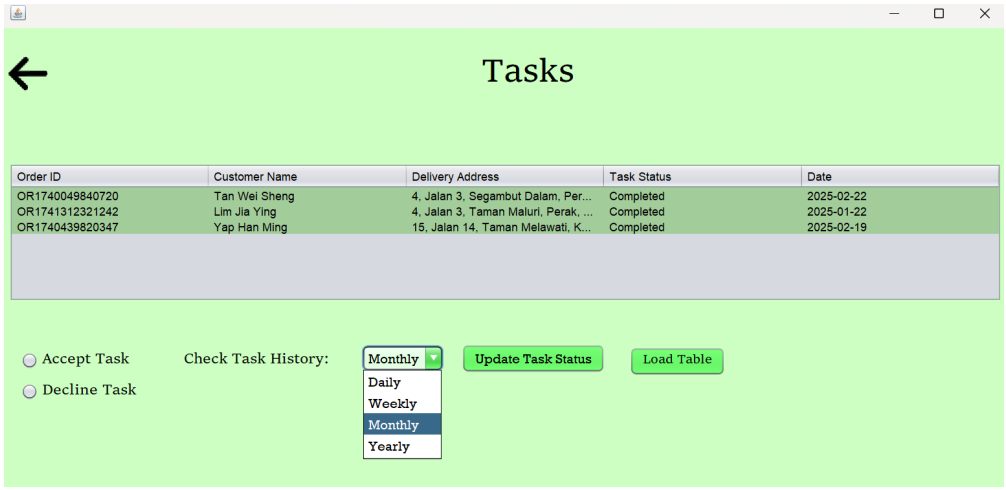
Daily:



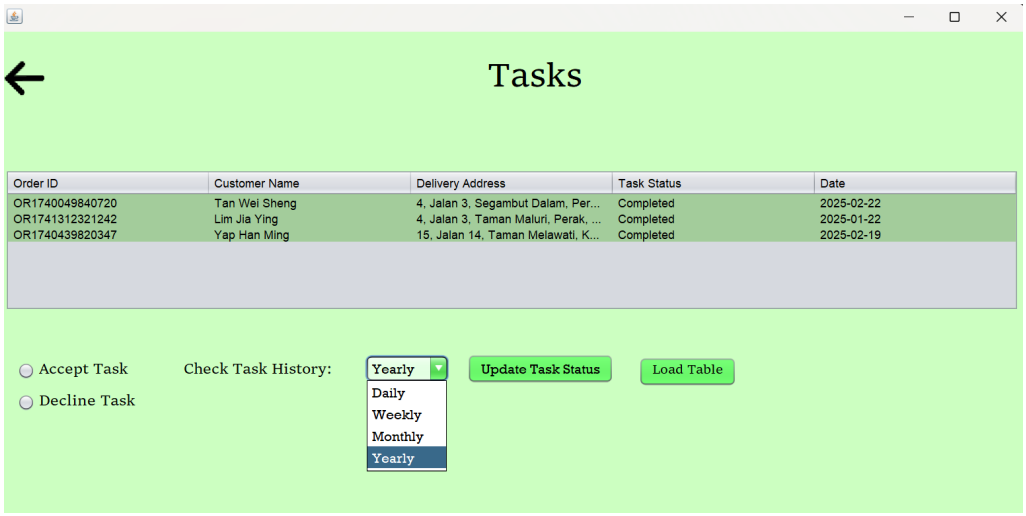
Weekly:



Monthly:

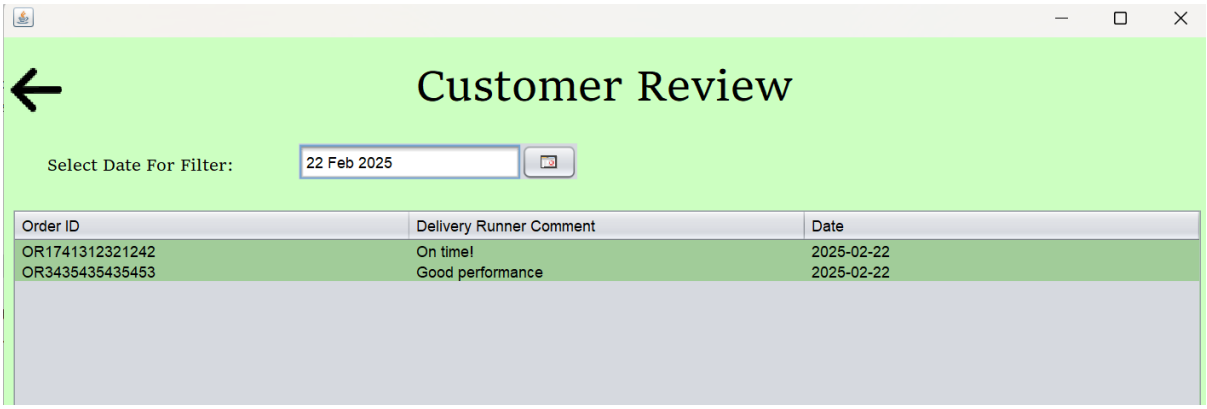


Yearly:

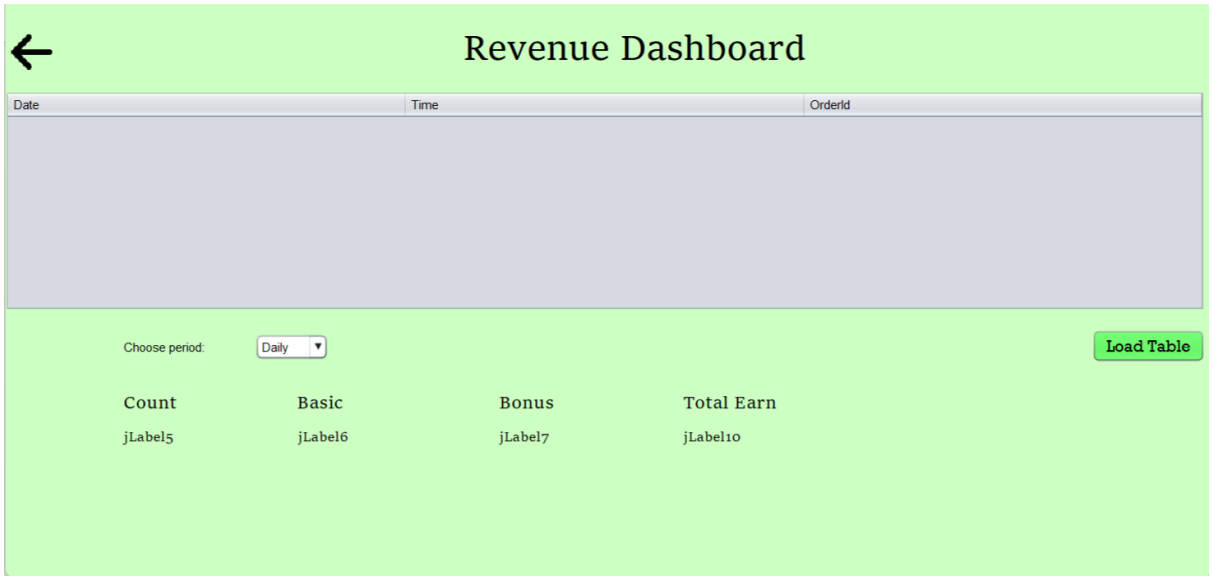


Click on the “Load Table” button the load the table, this will display the data from the text file which is called “orders.txt”.

This is called the Customer Review page, the delivery runner can view the displayed review data based on the selected date from the calendar (“Year”, “Month”, and “Date”). The table displays “OrderID”, “Delivery Runner Comment”, and “Date” which is taken from the text file called “feedbacks.txt”, this text file saves all related reviews.

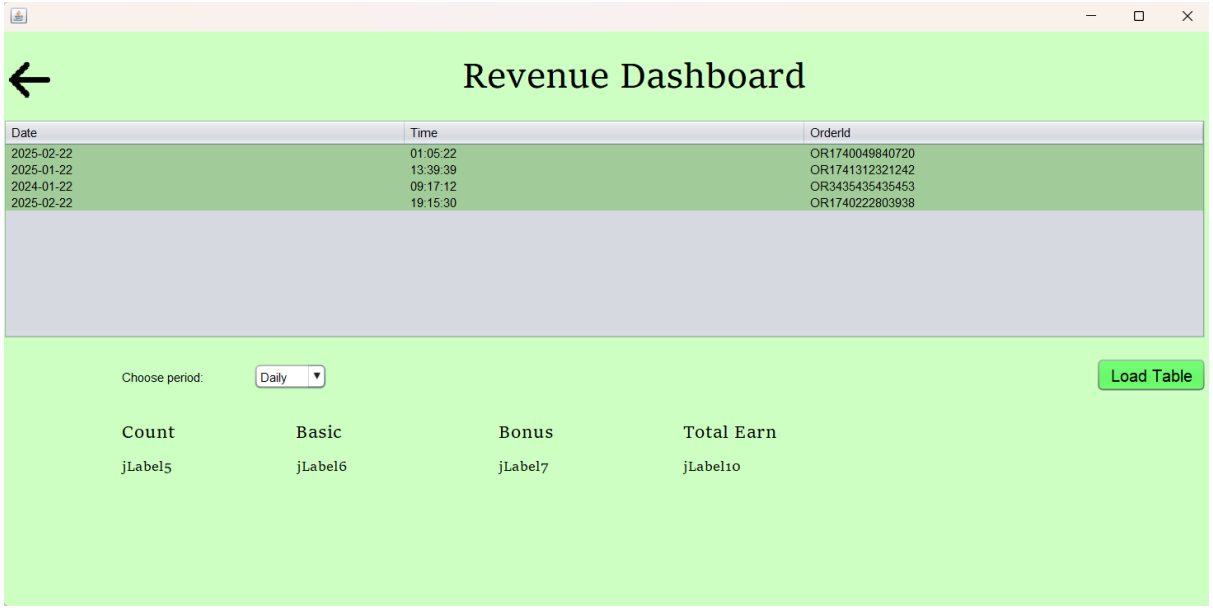


For example, select the date for 2025-02-22. The customer review is displayed accordingly.



This is called the Revenue Dashboard page, the delivery runner can view the displayed task completed data to calculate the profit based on the chosen period from the combo box (“Daily”, “Weekly”, “Monthly” and “Yearly”). The table displays “Date”, “Time”, and “OrderID” which is taken from the text file called “orders.txt”, this text file saves all related completed tasks to calculate its profit on this page. To calculate the profit “Count”, “Basic”, “Bonus”, and “Total Earn” is displayed. So, the count is the number of tasks that have been completed by the delivery runner, the basic is the basic salary, the bonus is when the delivery runner receives it, and the total earned is the sum of the basic and bonus.

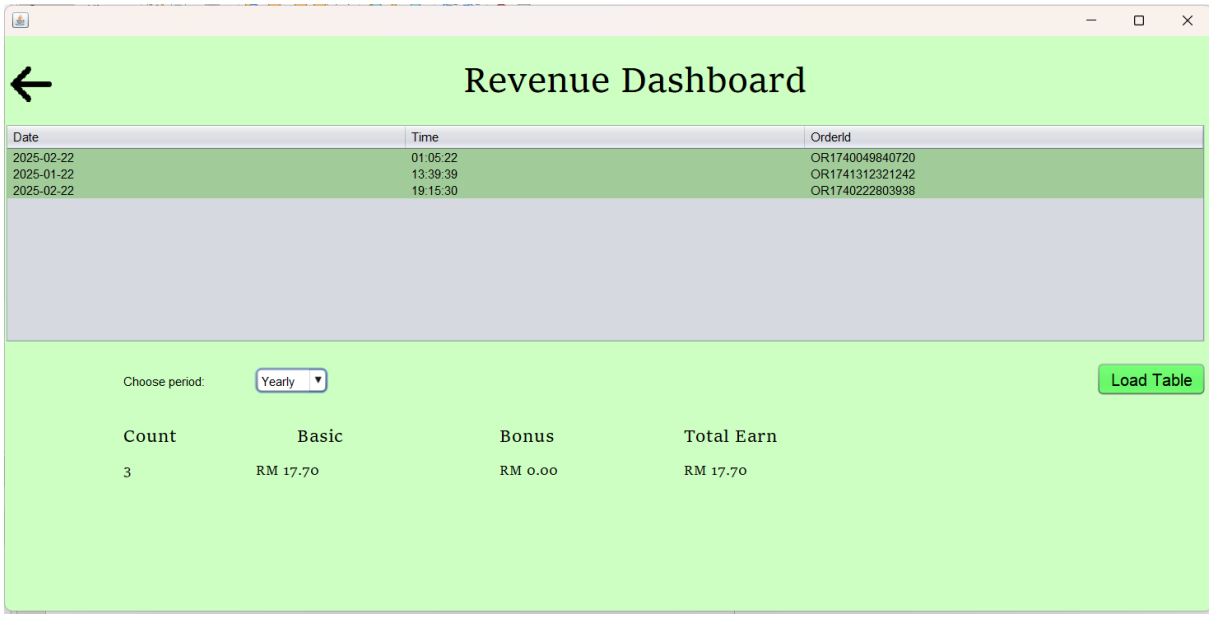
This is the profit made daily.



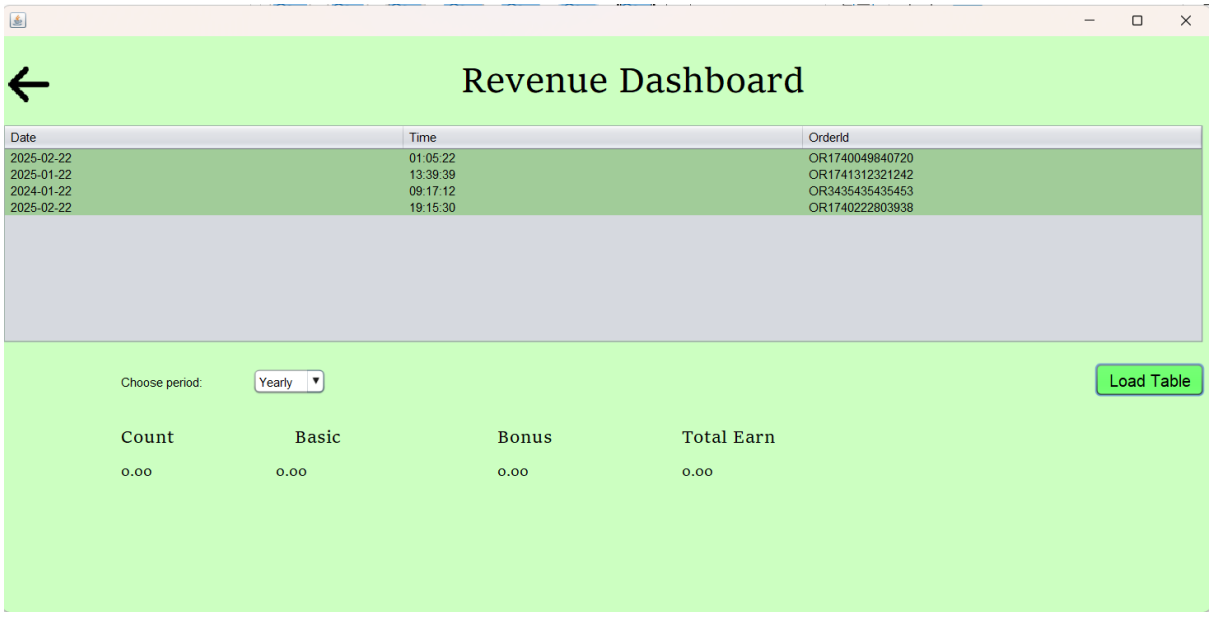
This is the profit made weekly.

This is the profit made monthly.

This is the profit made yearly.



Load the table to display all data for the revenue dashboard.



Each order earns RM5.90. Workers can receive additional bonuses based on their daily, weekly, and monthly performance. If a worker completes at least 15 orders per day and works more than 8 hours (calculated from the first to the last order of the day), they will receive a 1.5% bonus on their total daily earnings. The daily earnings are calculated as 5.9 × total orders, and the bonus is 1.5% of the daily earnings, making the total daily earnings daily earnings + daily bonus.

If a worker meets the daily bonus criteria for all 7 days in a week, they will qualify for an additional 2.0% bonus on their total weekly earnings. The weekly earnings are the sum of

the total daily earnings for 7 days, and the bonus is 2.0% of the weekly earnings, making the total weekly earnings weekly earnings + weekly bonus.

For the highest bonus, if a worker works every day of the month without missing a single day and meets the weekly bonus criteria for all weeks, they will receive an extra 2.5% bonus on their total monthly earnings. The monthly earnings are the sum of the total weekly earnings for the entire month, and the bonus is 2.5% of the monthly earnings, making the total monthly earnings monthly earnings + monthly bonus.

This system rewards consistency, ensuring that workers who put in steady effort over time maximize their earnings.

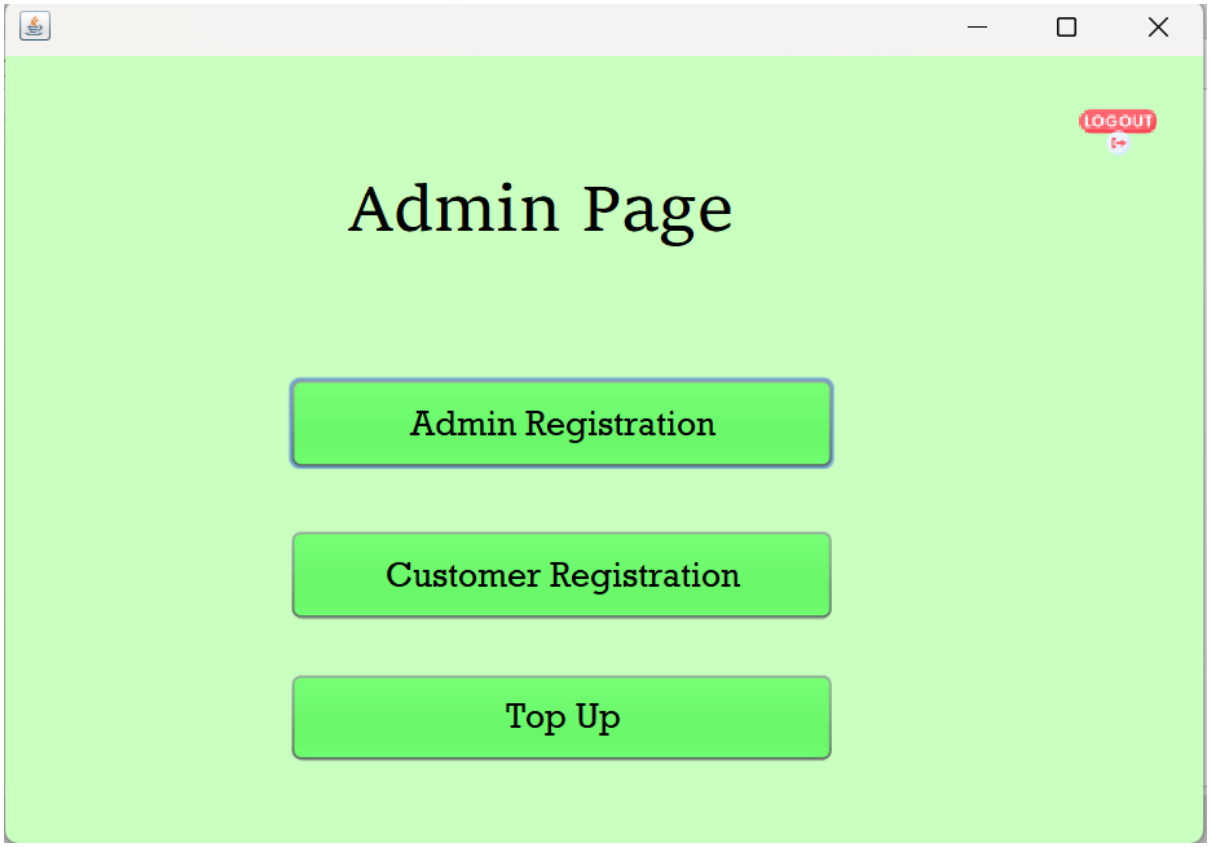
3.3 Ong Kai Ying (Administrator & Manager)

Administrator

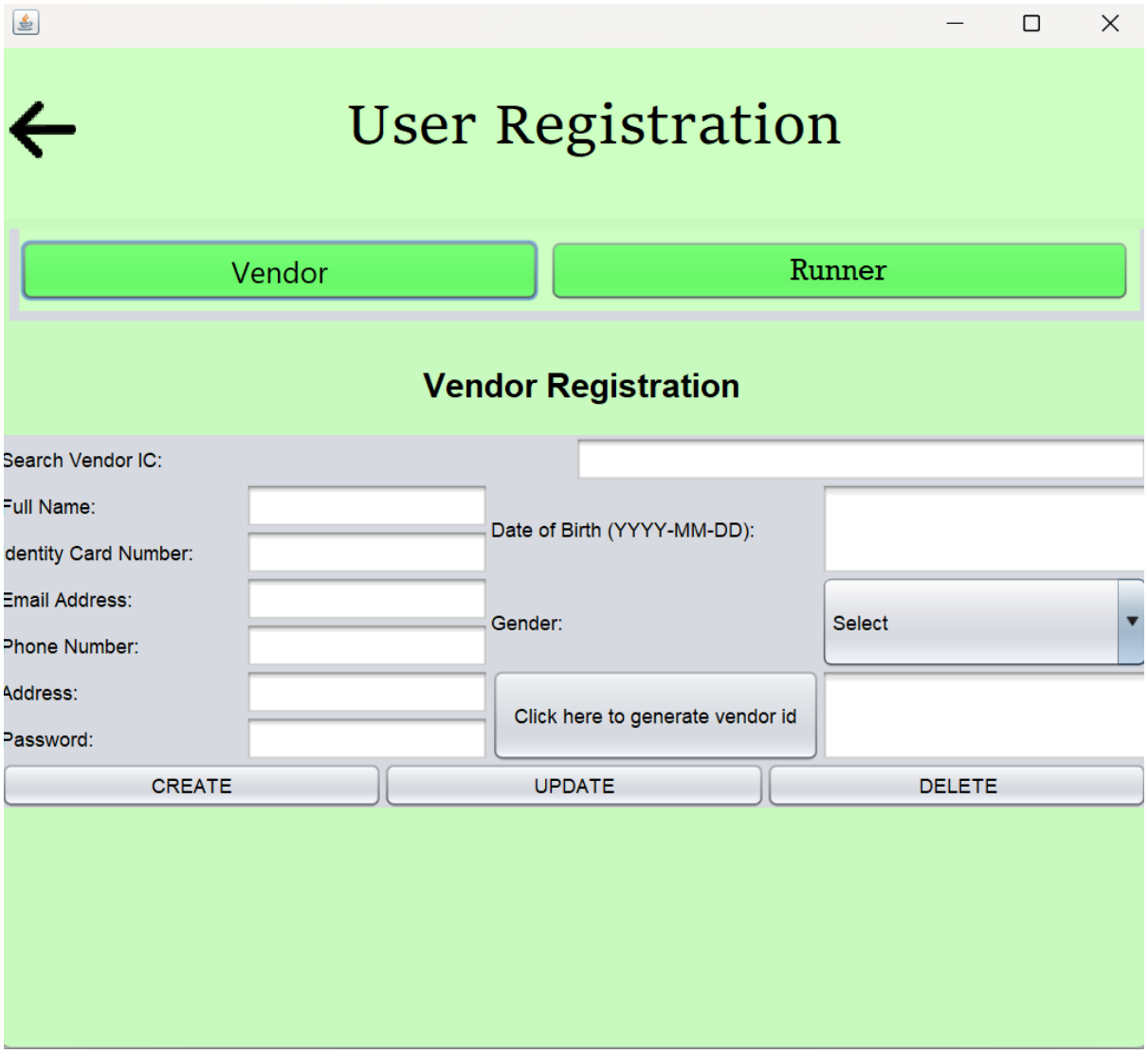
Admin using username “admin123” and password “admin123123” for login into the admin page.

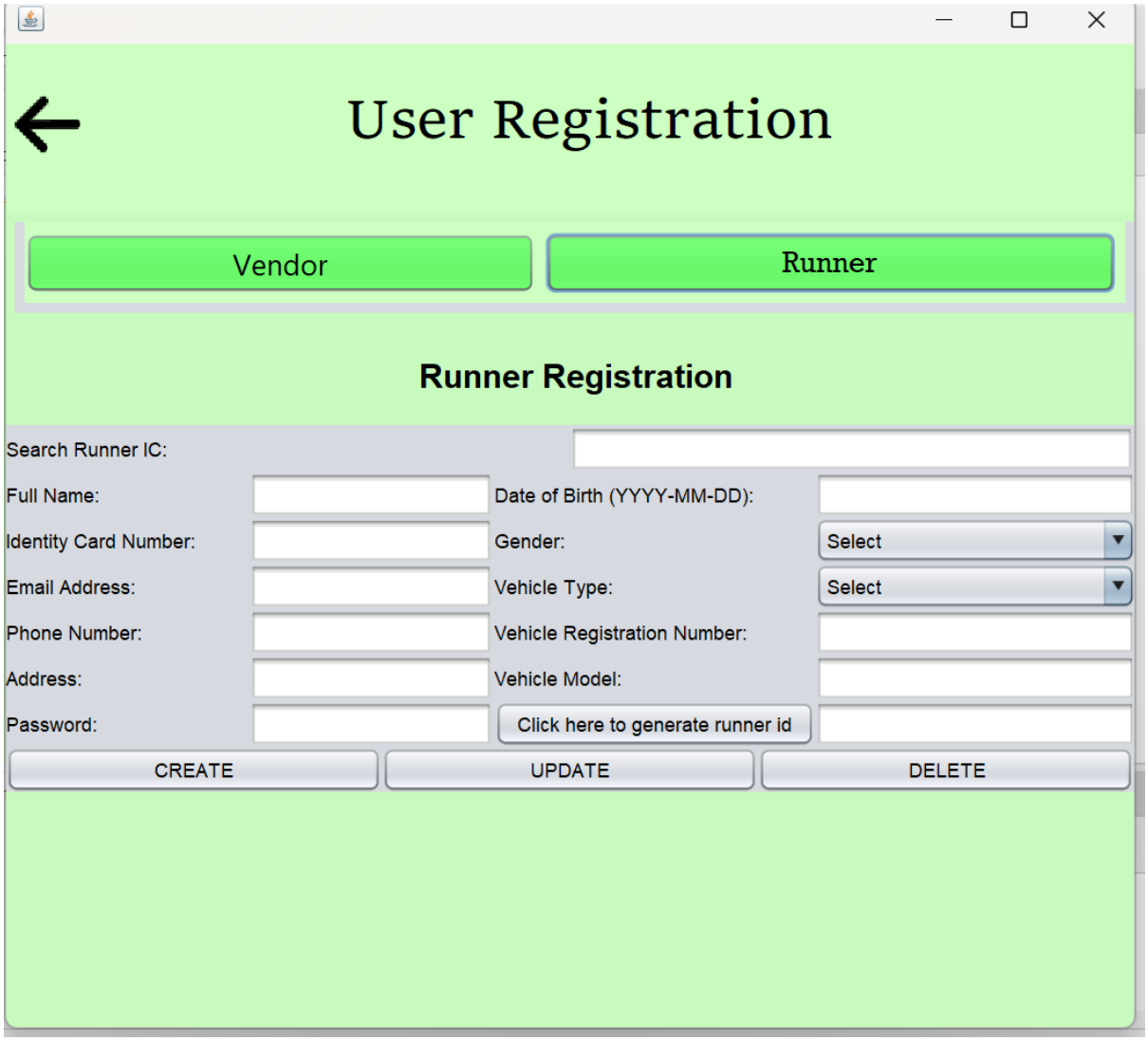
```
public boolean verifyUser() throws IOException {
    String[] customer = {"custRegistration.txt", "11", "10", "12"};
    String[] vendor = {"vendorRegistration.txt", "8", "7", "9"};
    String[] runner = {"runnerRegistration.txt", "11", "10", "12"};
    String[] manager = {"manager", "mng123", "manager123"};
    String[] admin = {"admin", "admin123", "admin123123"};
```

After the admin logs in through the login page, they will be directed to the "AdminChoose" page. This page allows the admin to select tasks such as registering vendor and runner employee accounts, registering customers, and topping up customer balances (for those who require cash top-up).

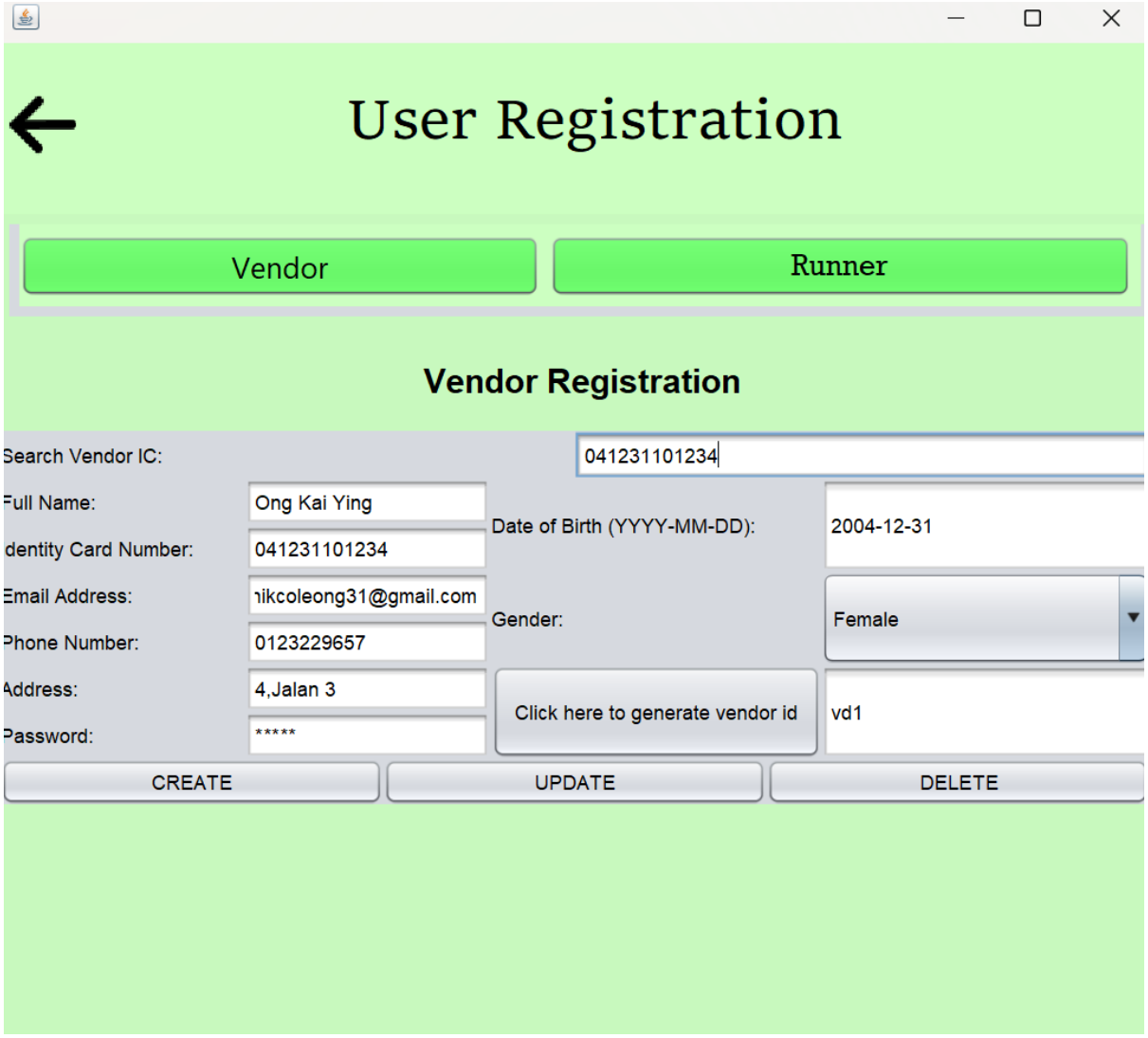


On the "Admin" page for vendor and runner registration, there will be two buttons that allow the admin to choose whether to register a vendor or a runner. If the admin chooses to register a vendor, the data will be stored in vendorRegistration.txt.

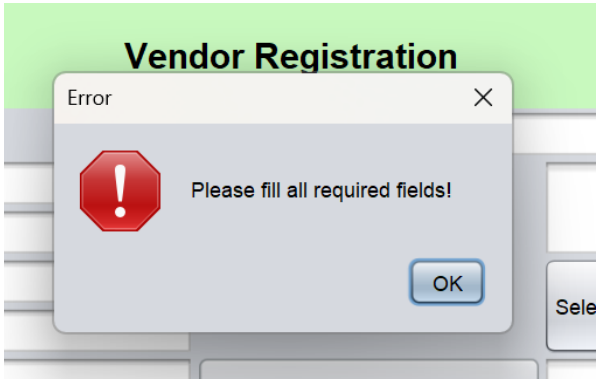


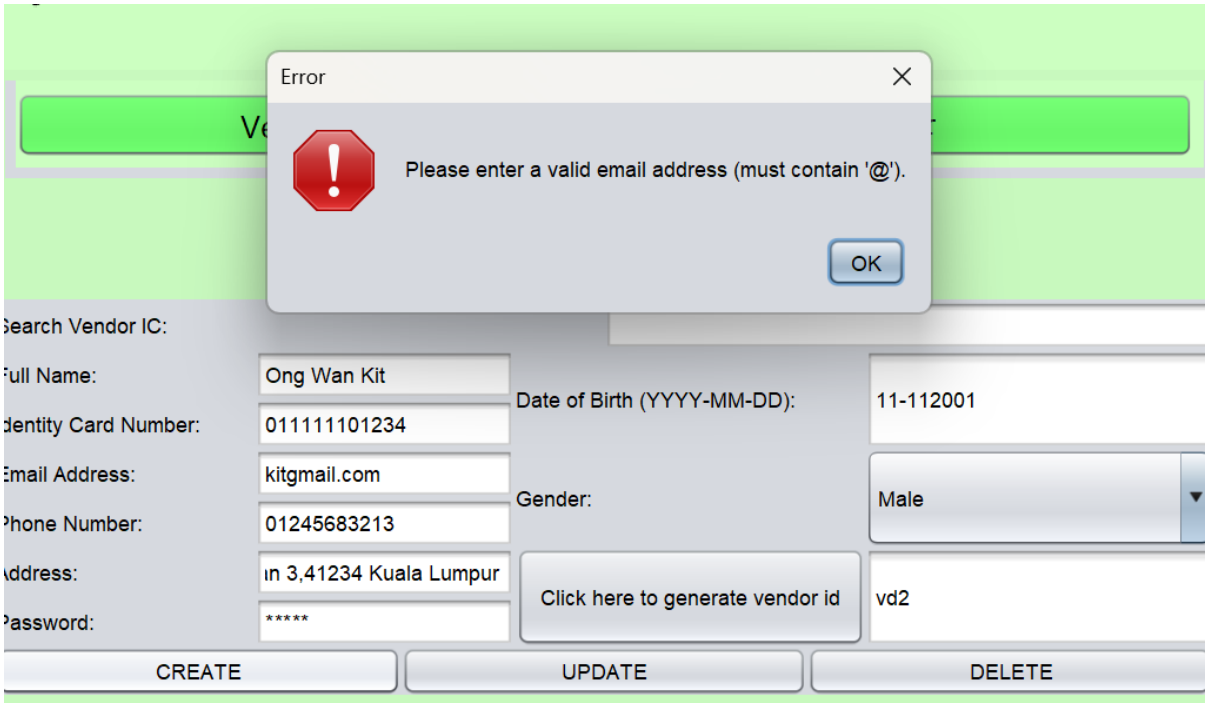


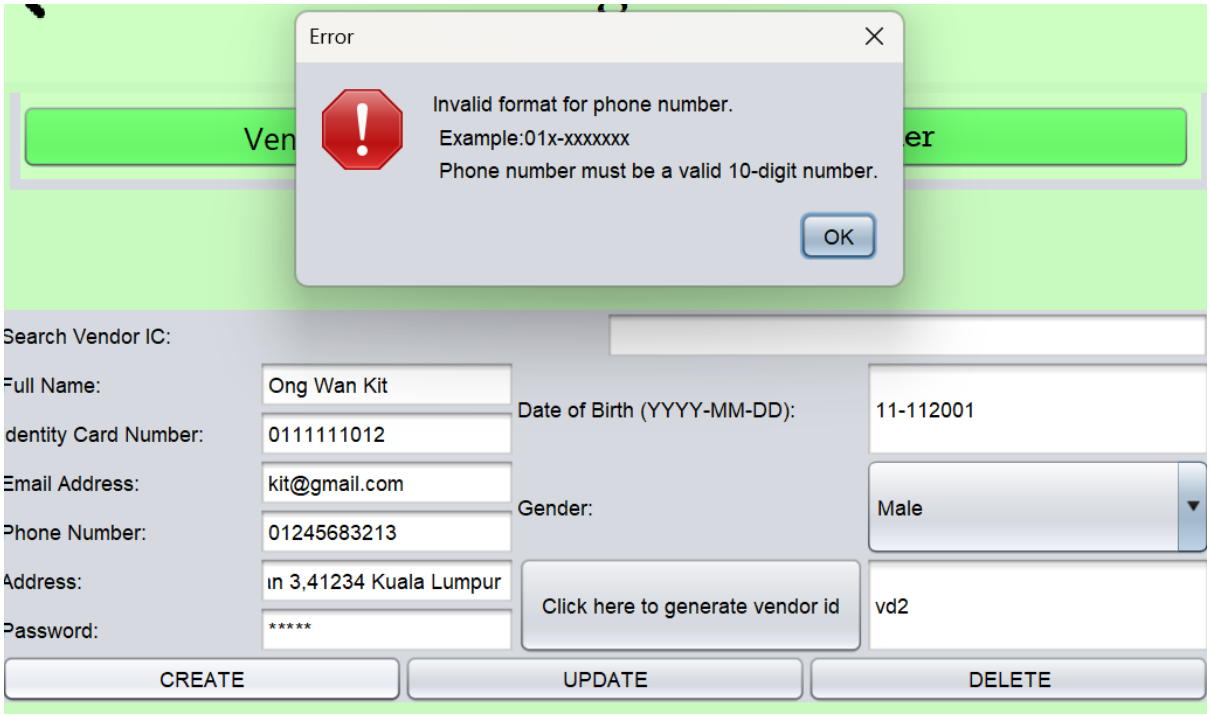
The admin must first create an account before performing updates or deletions. There is a "Generate Vendor/Runner ID" button that automatically generates an ID. If the registration is for a vendor, the ID format will be "vd1", "vd2", and so on, while for a runner, the ID format will be "rn1", "rn2", and so on.



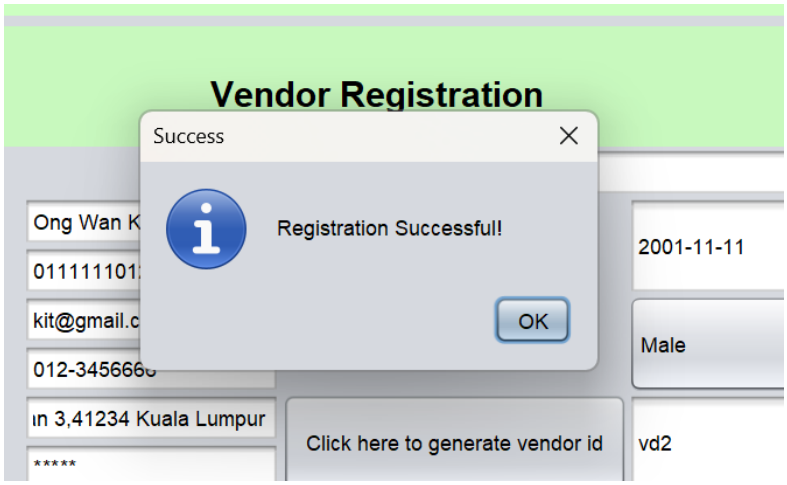
If the admin attempts to create an account without completing all required fields, the system will display an error message prompting them to fill in all necessary information. Additionally, if there are errors in the data format, the system will pop up an error message indicating what is wrong and how to correct it.

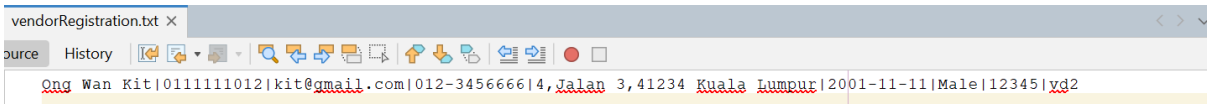




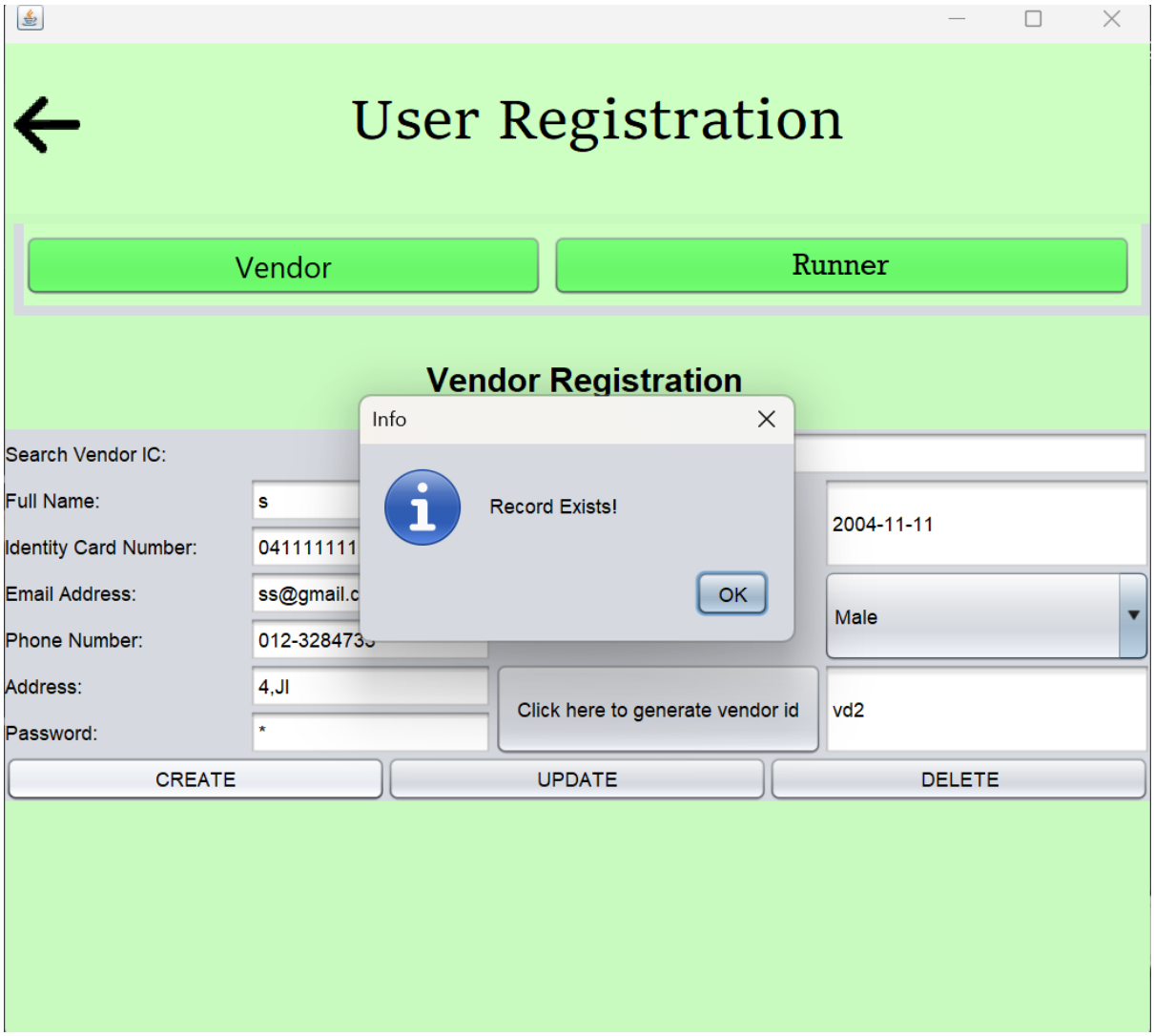


Then click on the create button, the record is stored, pops up a message showing the record successfully added.

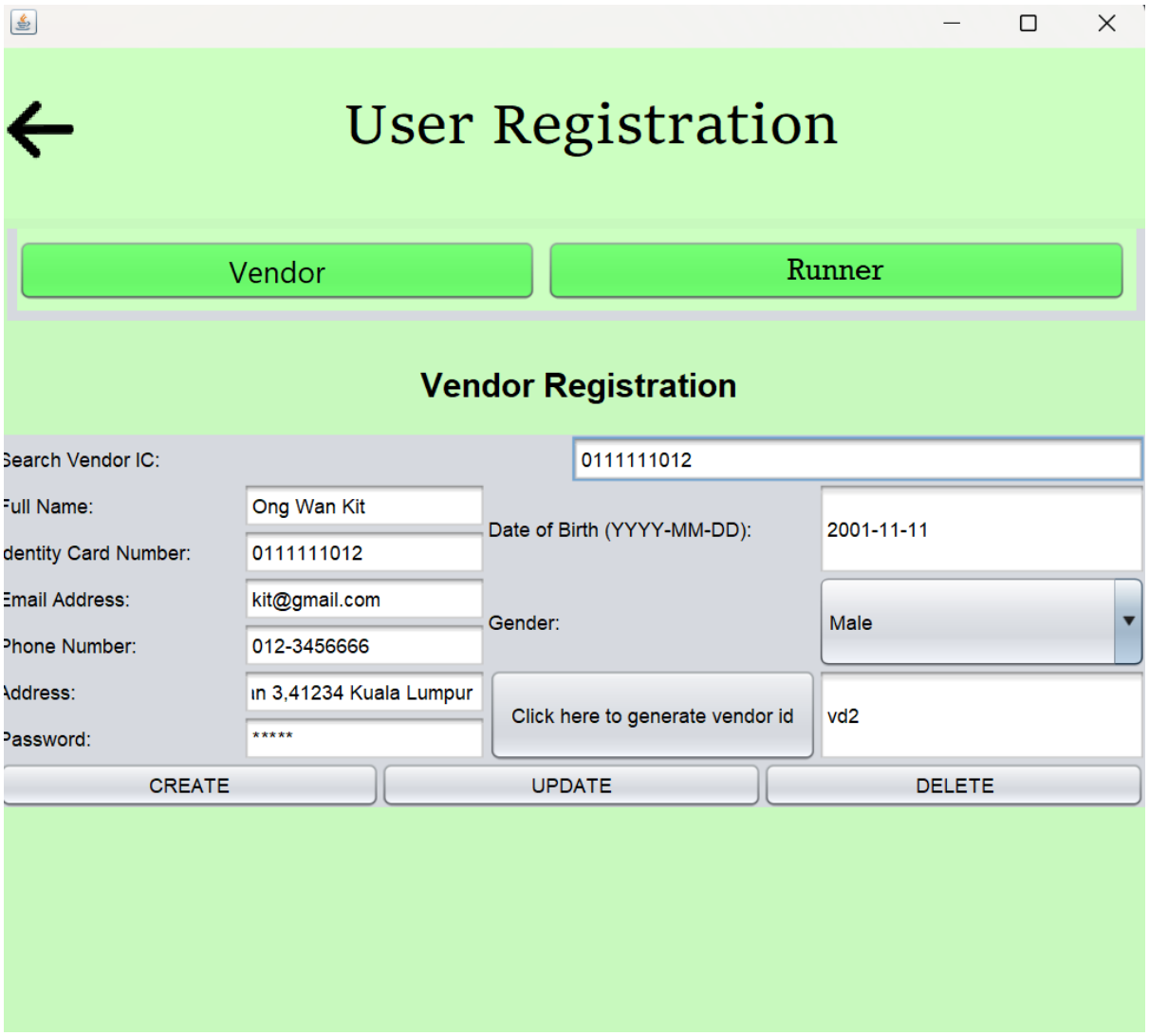




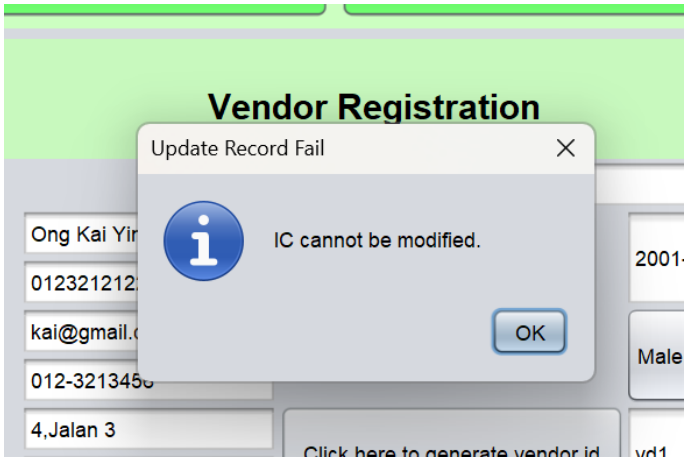
If add the record that exits in the txt file (by checking the ic number), the warning message will pop up.



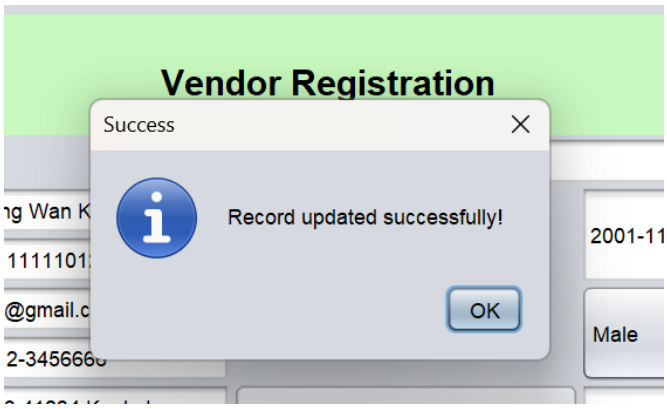
To edit or update information, the admin must search for the IC number and press "Enter", after which the details will be automatically loaded.

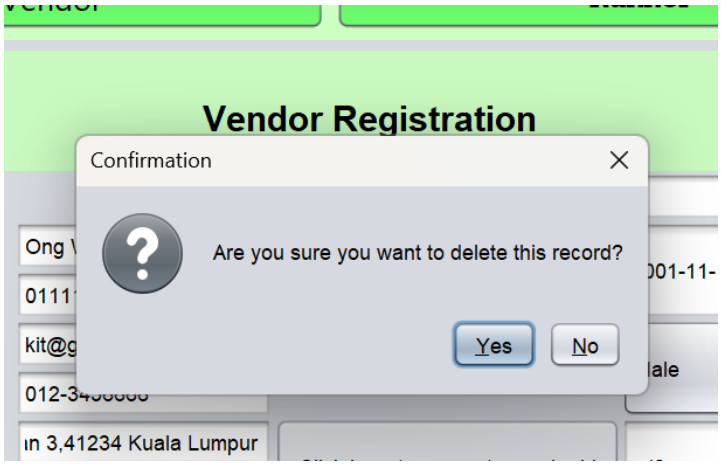


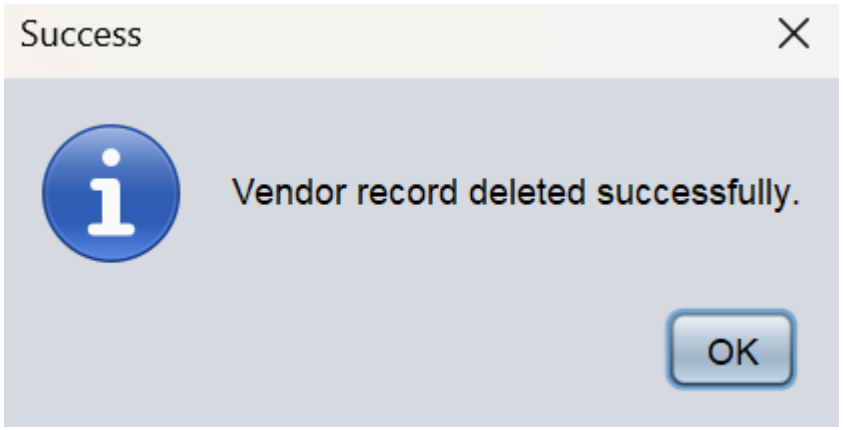
However, the admin is not allowed to update the vendor/runner ID or IC number.



Once the details have been modified, clicking "Update" will save the changes. Similarly, to delete a record, the admin must first search for it before proceeding with deletion.



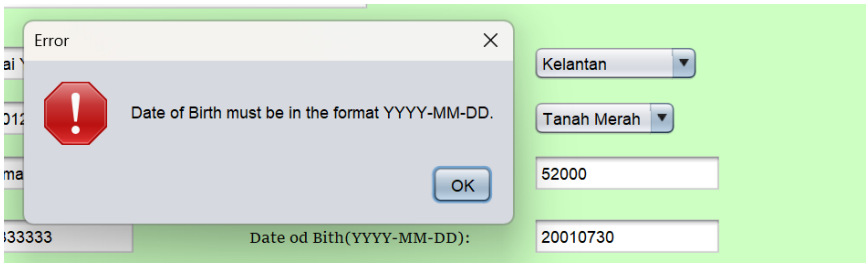




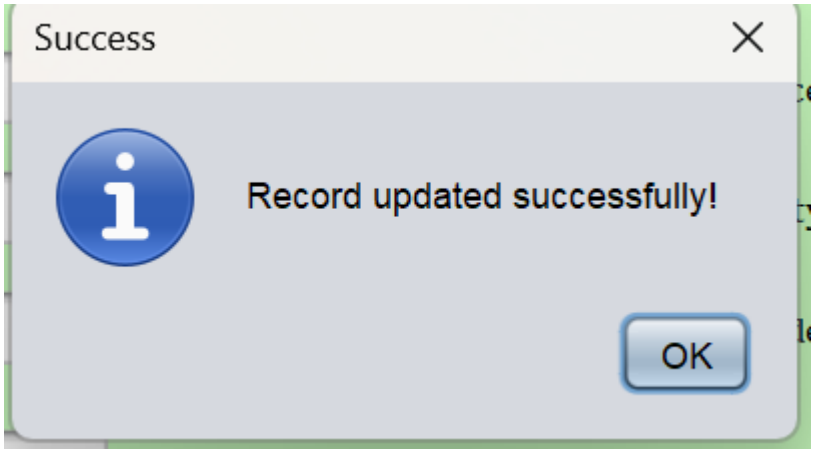
On the " AdminCustomerRegistration " page for customer registration, the admin can create, update, and delete customer accounts. The system provides a "Generate Customer ID" button that automatically generates a unique customer ID in the format "cust1", "cust2", and so on.

If the admin attempts to create an account without completing all required fields, the system will display an error message prompting them to fill in all necessary information.

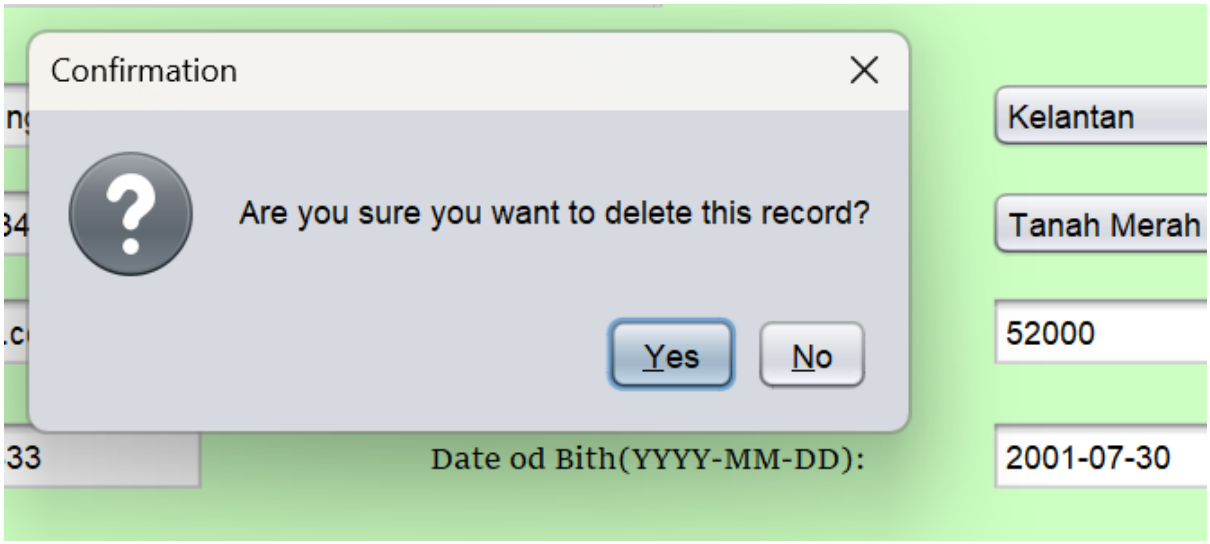
Additionally, if there are errors in the data format, the system will pop up an error message indicating what is wrong and how to correct it.

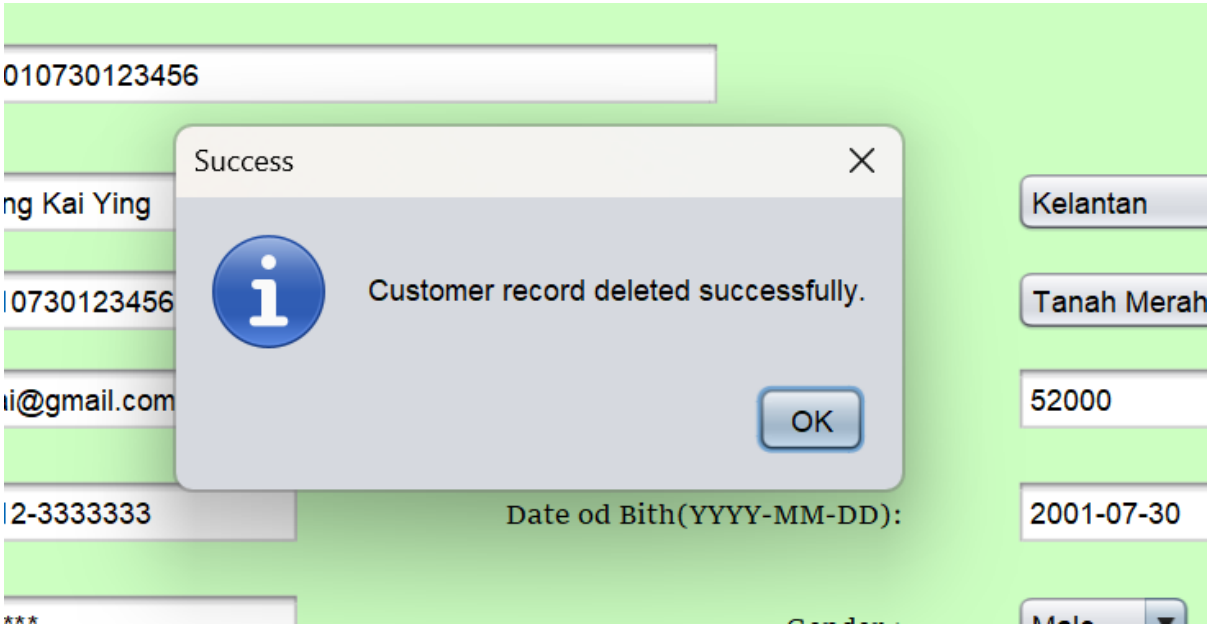


To edit or update customer information, the admin must search for the IC number and press "Enter", after which the details will be automatically loaded. However, the admin is not allowed to update the customer ID or IC number. Once the details have been modified, clicking "Update" will save the changes.

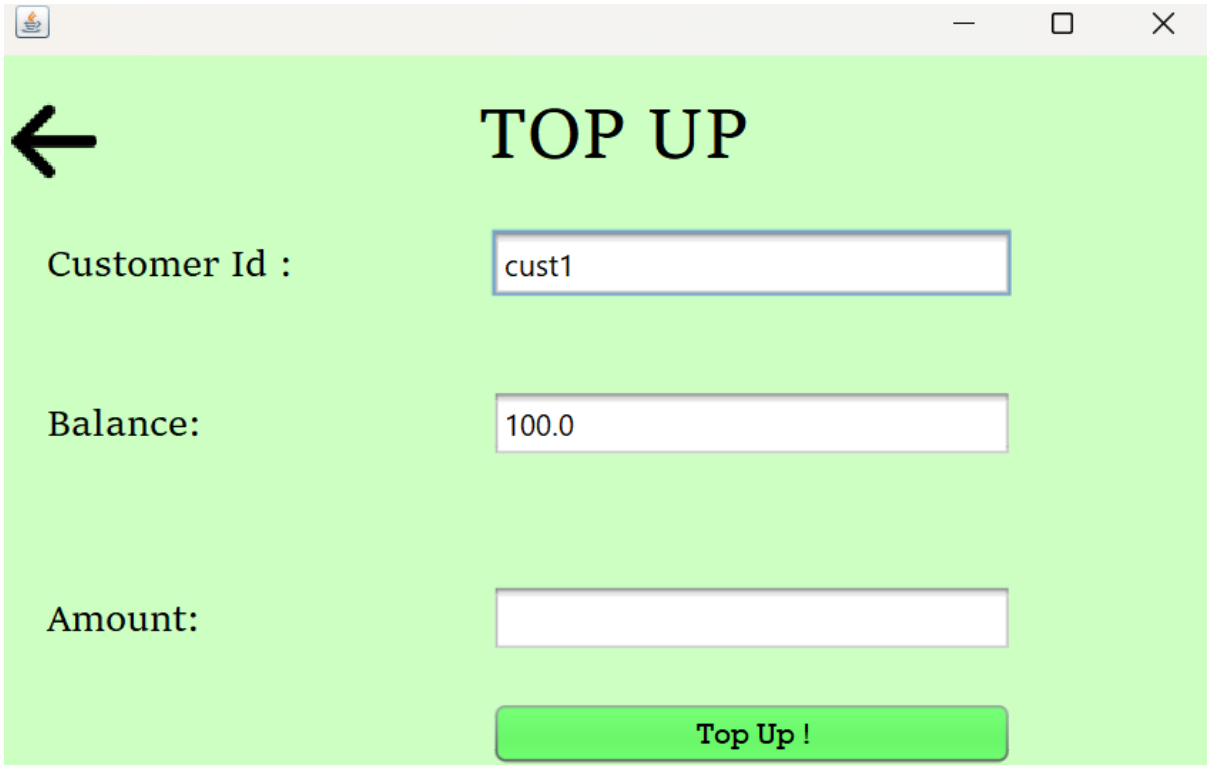


Similarly, to delete a record, the admin must first search for it before proceeding with deletion.



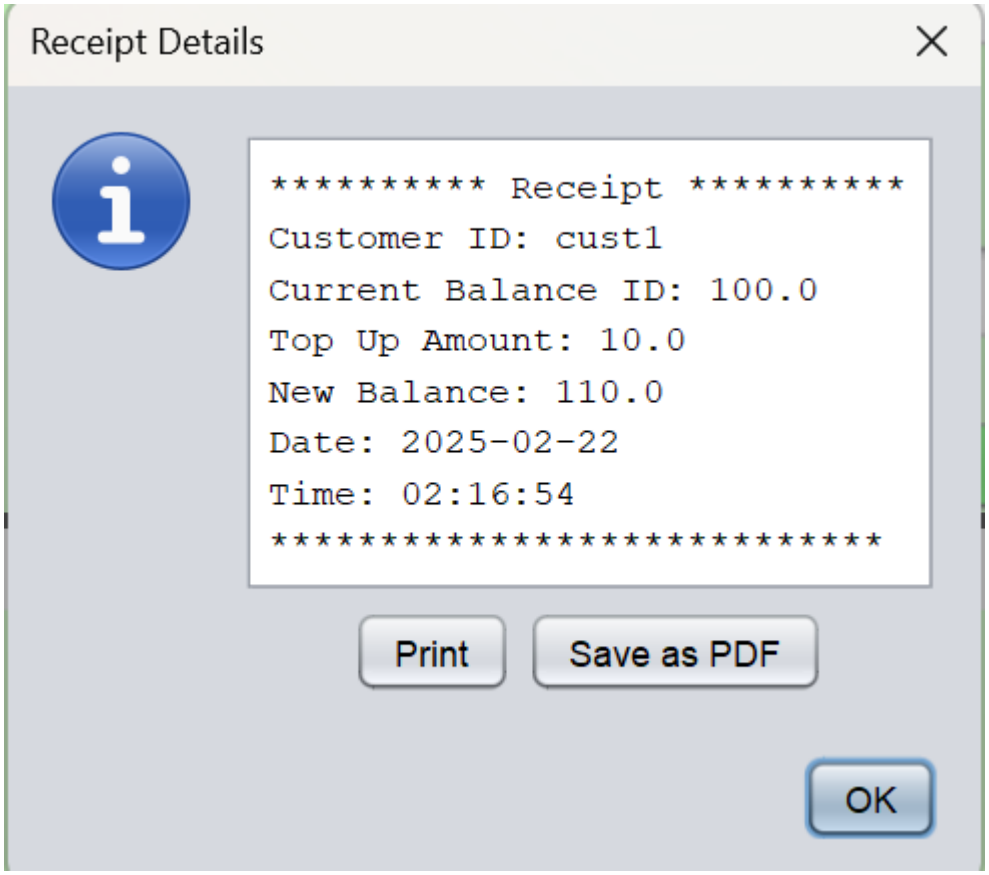


When the admin enters the "TopUpCash" page, it means the customer needs to perform a cash top-up instead of a banking top-up. The admin can assist with the top-up by entering the customer's name and pressing "Enter", which will display the customer's current balance.



After entering the top-up amount, a popup message will confirm the top-up was successful.

Once the transaction is completed, a receipt will be generated. The admin will have the option to either print or save the receipt as an additional feature.

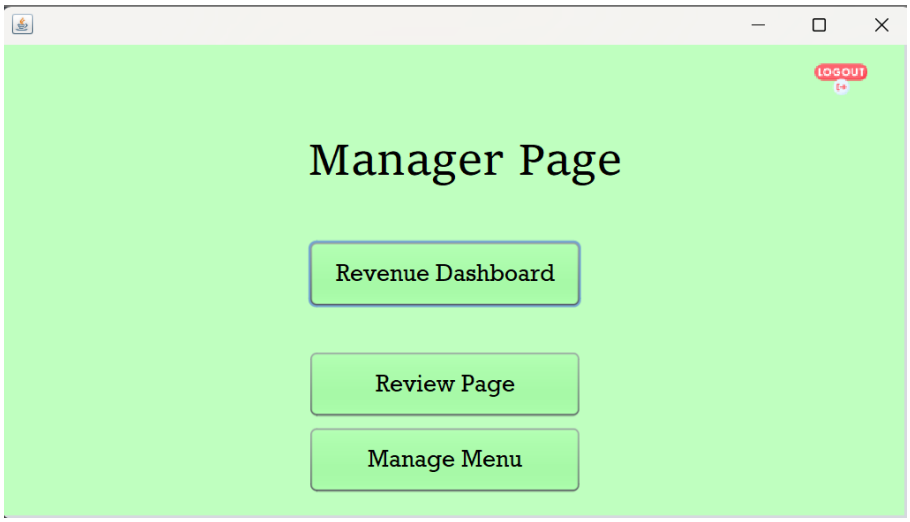

Receipt Details
********** Receipt **********
Customer ID: cust1
Current Balance ID: 100.0
Top Up Amount: 10.0
New Balance: 110.0
Date: 2025-02-22
Time: 02:16:54
****
Print
Save as PDF
OK

Manager

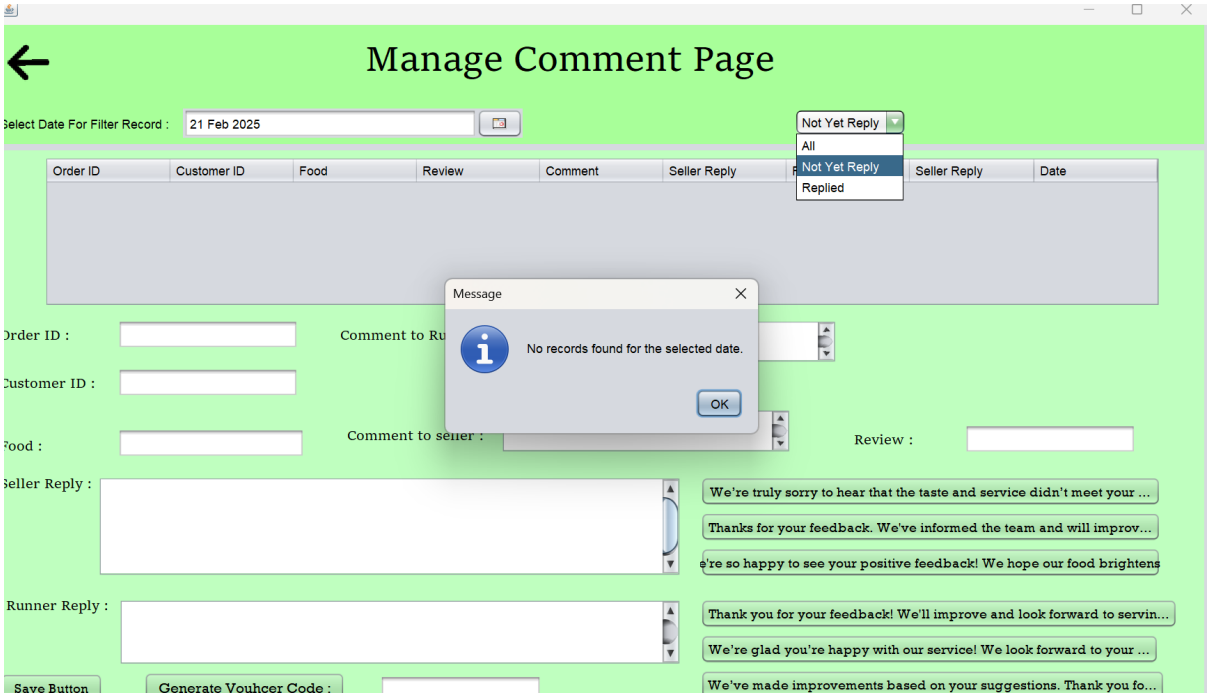
Admin using username “mng123” and password “manager123” for login into admin page.

```
public boolean verifyUser() throws IOException {
    String[] customer = {"custRegistration.txt", "11", "10", "12"};
    String[] vendor = {"vendorRegistration.txt", "8", "7", "9"};
    String[] runner = {"runnerRegistration.txt", "11", "10", "12"};
    String[] manager = {"manager", "mng123", "manager123"};
    String[] admin = {"admin", "admin123", "admin123123"};
```

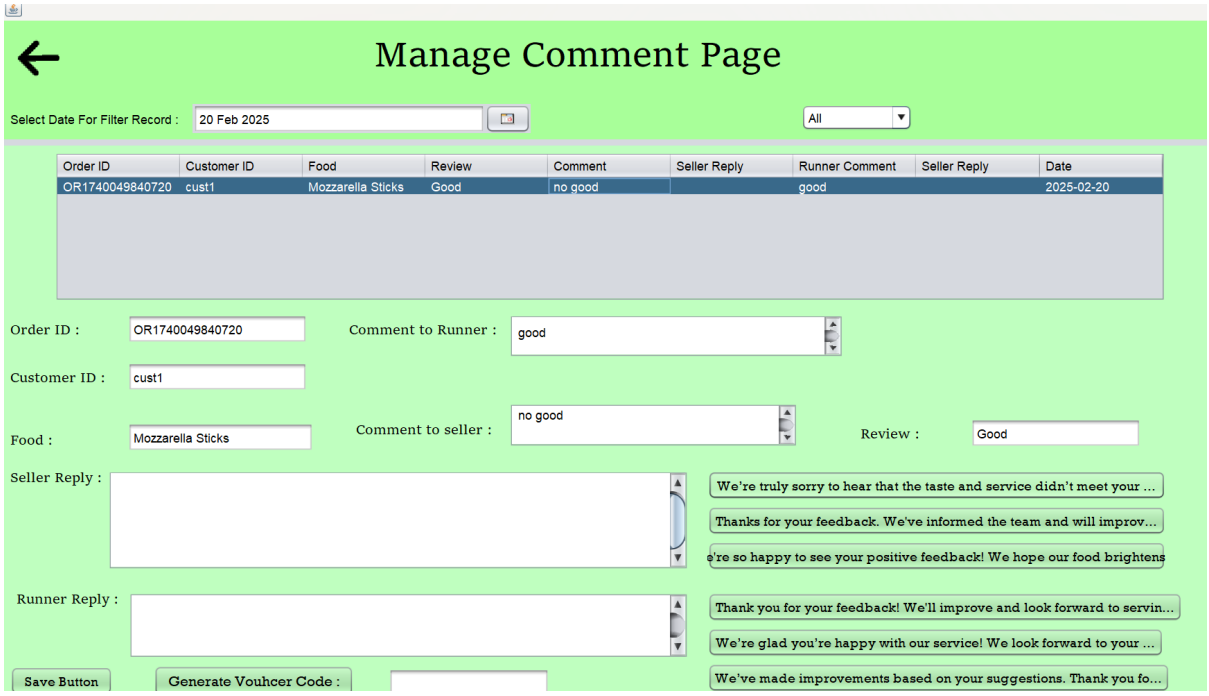
After the managerlogs in through the login page, they will be directed to the "ManagerChoose" page. This page allows the manager to view on the Revenue Dashboard (which is linked to vendor side), review comment page, and manage the menu ((which is linked to vendor side),



When the manager enters the "ManageReview" page, they can view customer reviews for runners and restaurant food. The system also provides a filter option, allowing the manager to filter reviews by date, or filter them based on their status: all reviews, not yet replied, and replied.

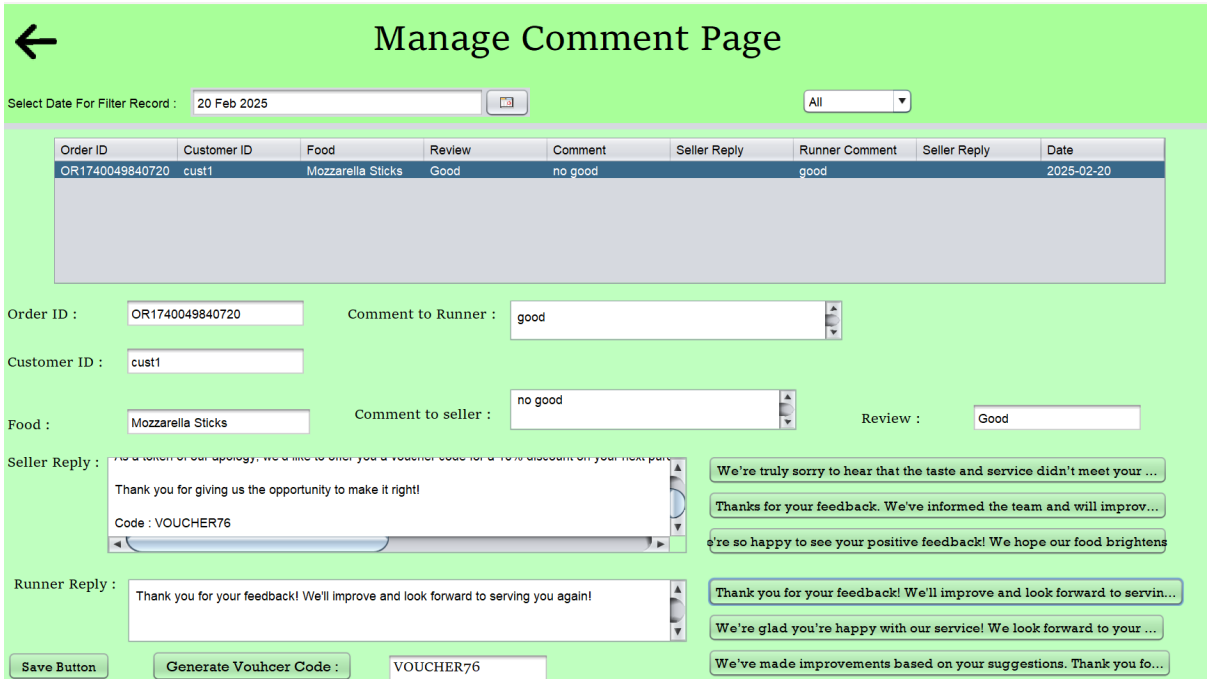


To reply to a customer review, the manager can click on the record in the table, and the review details will be automatically filled in.

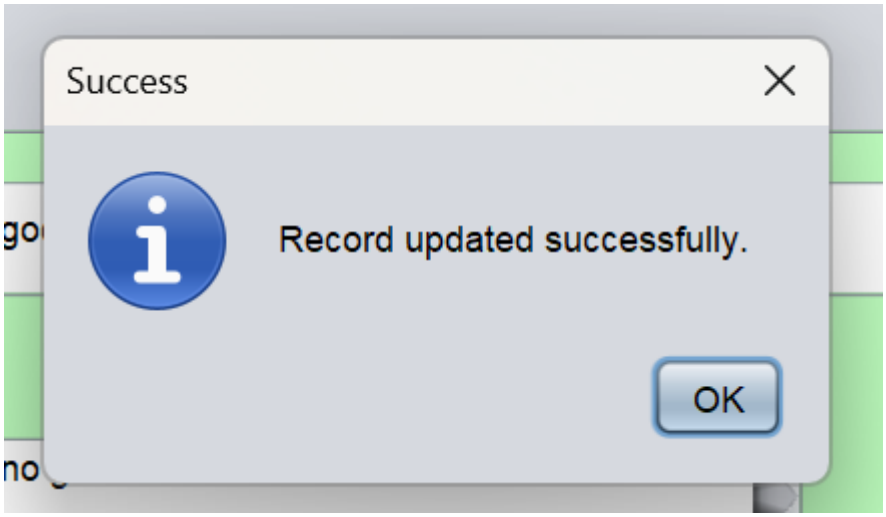


The manager can then write a comment in the designated text fields. There are also preset comment buttons available for quick autofill. If a review contains negative feedback, the manager has the option to offer a voucher to the customer. A voucher code can be generated by clicking the "Generate Voucher" button. Once generated, the voucher code should be

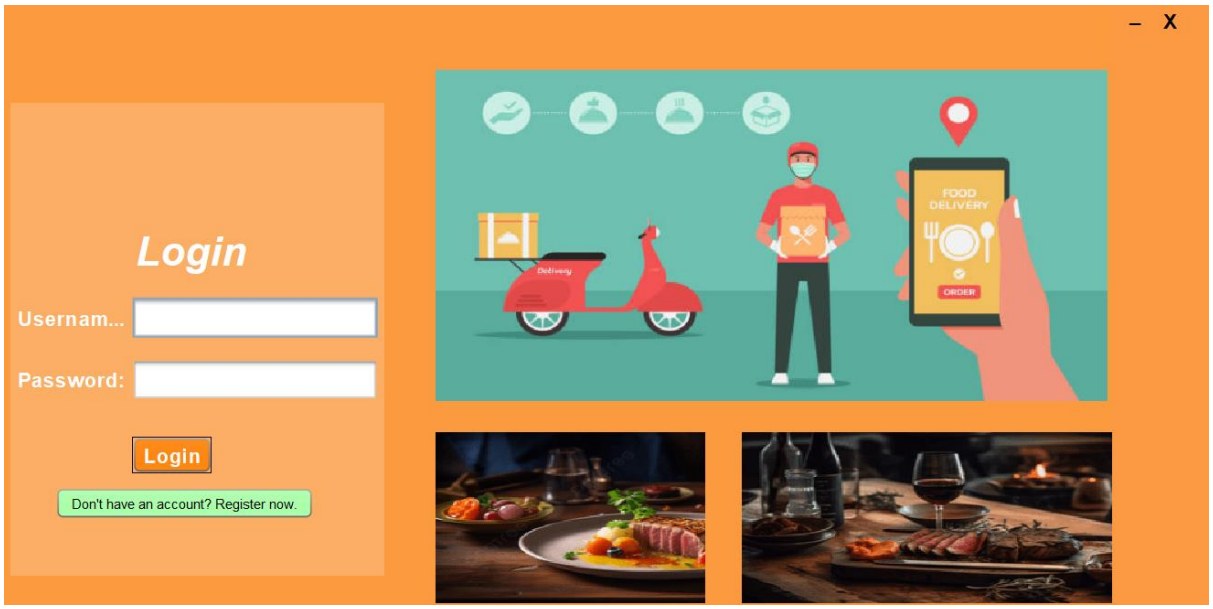
included in the comment section. After finalizing the response, the manager can click the "Save" button to store the reply.



"Successful" message will be pop up after review is saved.



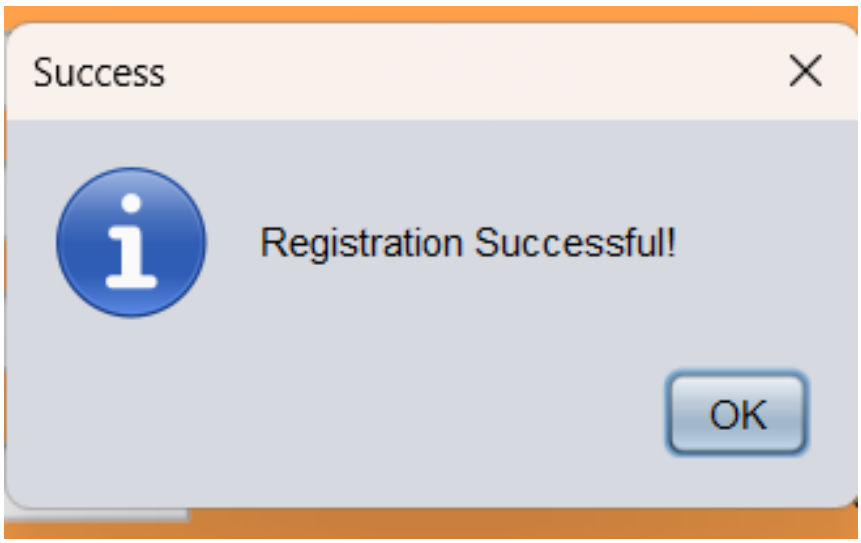
3.4 Tee Ching Ying (Customer)



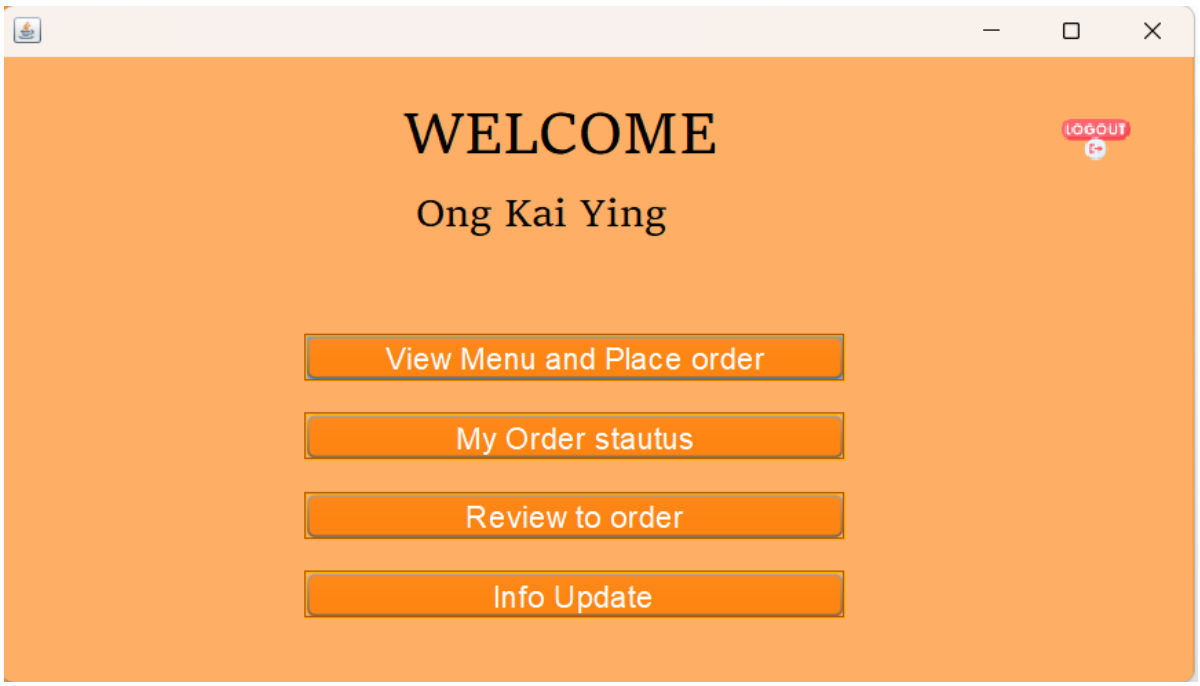
For the customer, they can register an account to login into our food ordering system through the button below the “Login” button.

Customers need to key in their information and choose the create button to validate the account and the information will be saved into customer registration text file. The customer ID will be

generated automatically through clicking the generate button. The clear button is used to clear the information.

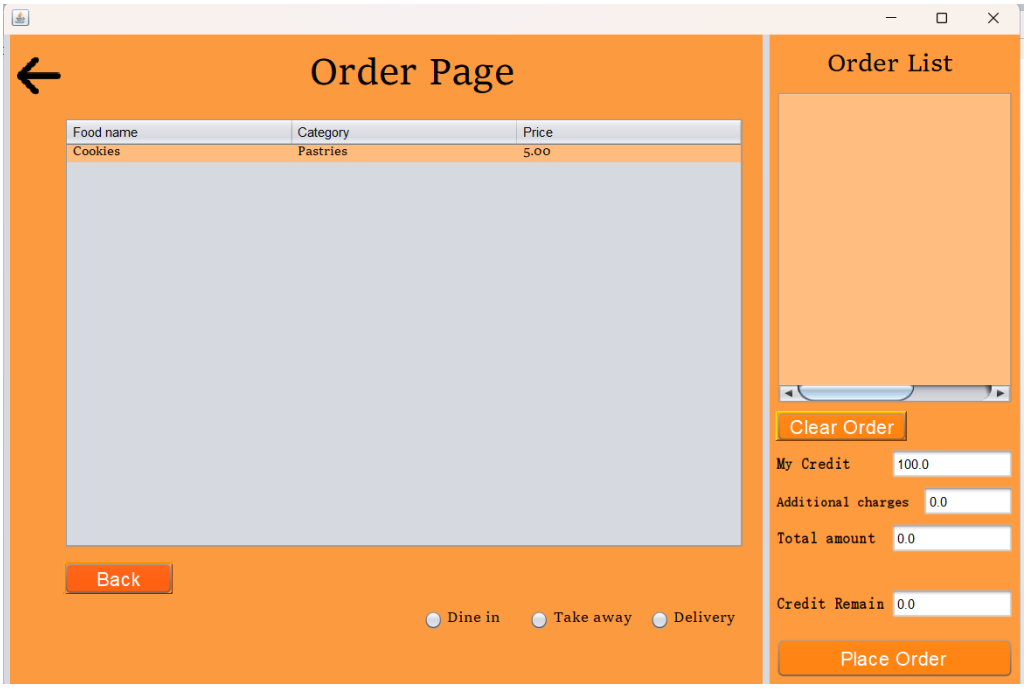


The registration successful message will be shown after they click the create button and they can click the arrow at upper left to back to login page.

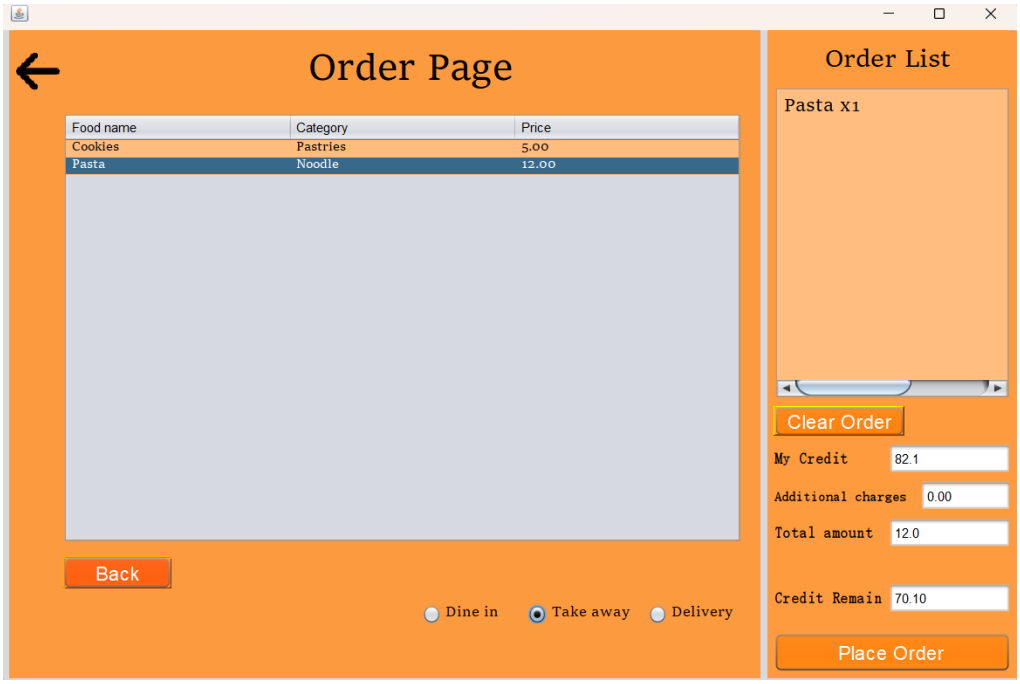


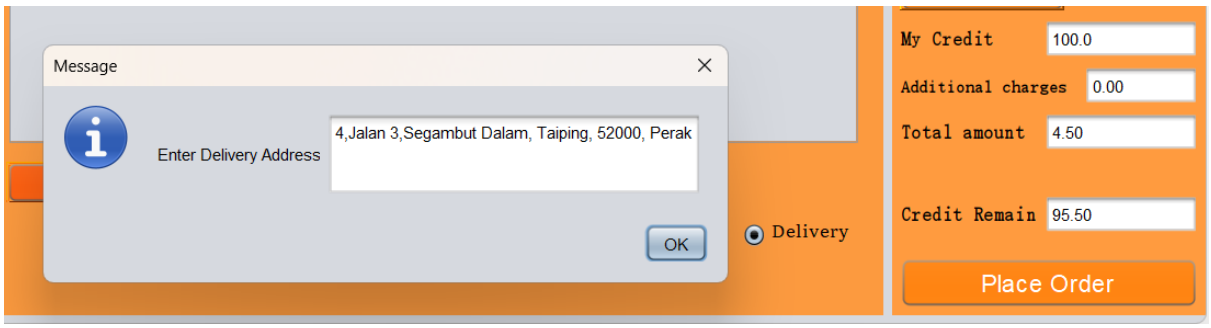
This is the main menu for customer. When customer login successful into this page the label will show the customer's name below the 'welcome' label. There are four button in this form which are "View menu and place order" that bring customer to restaurant menu page to place order, "My order status" that bring customer to the form that they can check their order status, order history and transaction history, "Review to order" which bring customer to the review

page to leave comment to their order and also view other customers' comment and the "Info Update" button which can bring customer to their profile page and made change and update. There is a "logout" located at the upper right of the page to allow customer to logout their account.

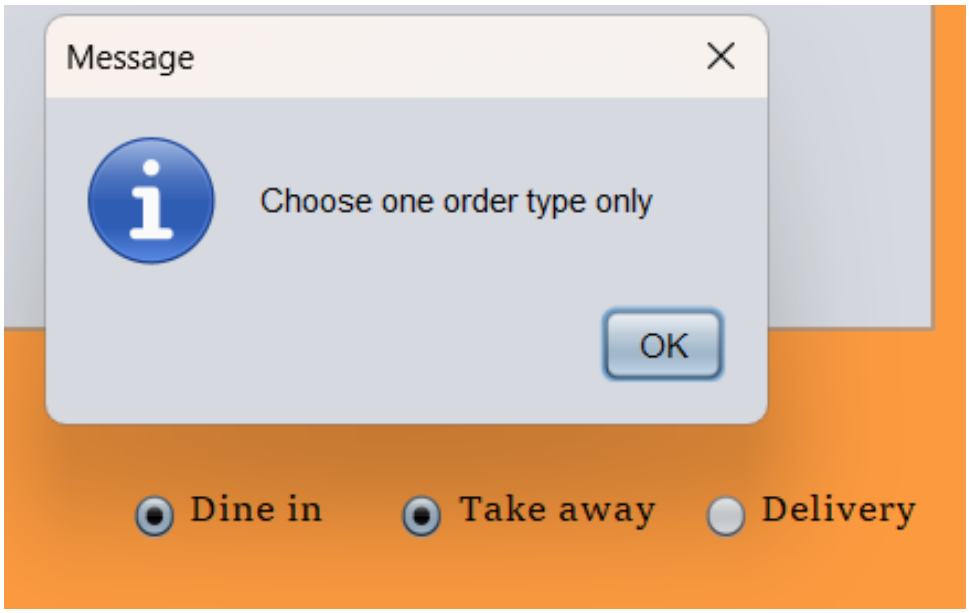


This is the menu form which customer make their order at this page. The food selected will be show together with the quantity. The "Clear Order" button will clear the selected item. The total amount and credit remain of customer will be calculated automatically. The current credit of customer was shown at "My Credit". If the customer choose delivery, the additional amount will be added into total amount which is RM 5.90. If customer choose to dine in or take away, they won't have any additional charges. When "Place Order" button clicked, the order will be added into order txt file. The "Back" button at the bottom is allow user back to customer main page if they don't want to order.

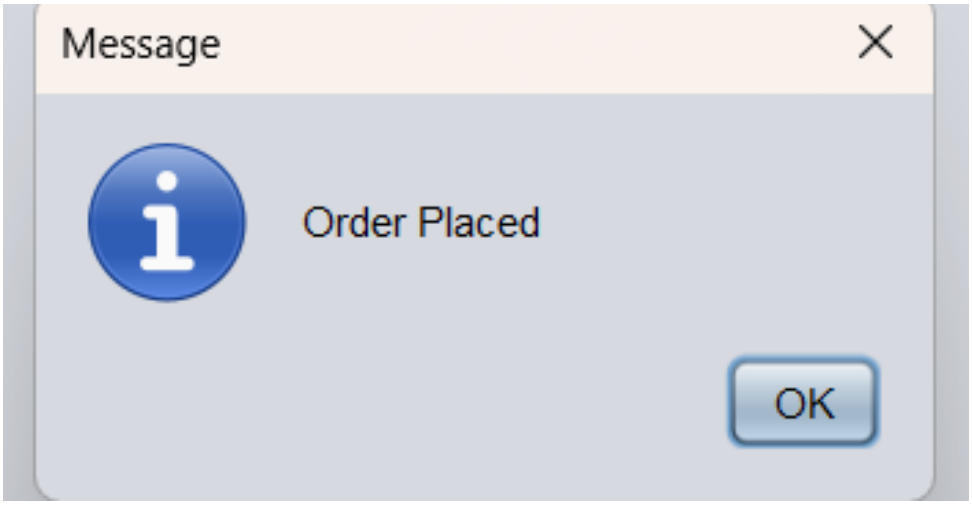




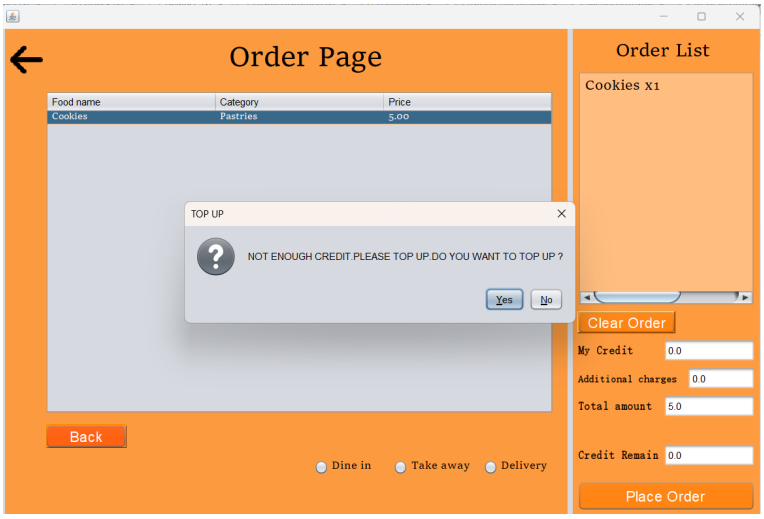
If the customer choose delivery, they need to fill in their address at this message box to save the data into text file together with order information.



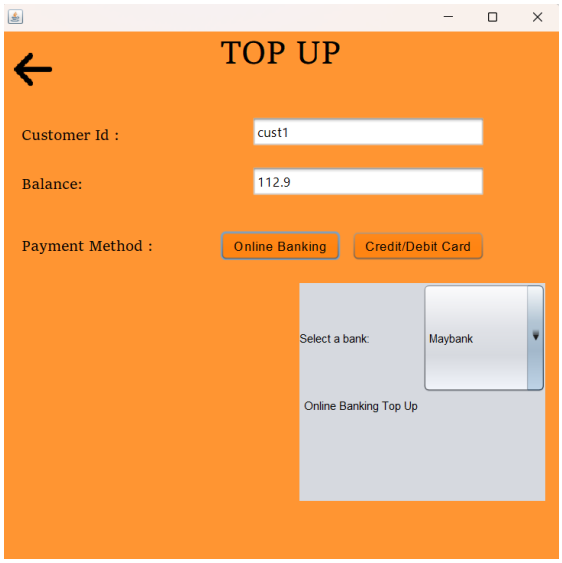
If the customer chooses more that one of order type, the message like figure above will show to remind customer can choose only one type.

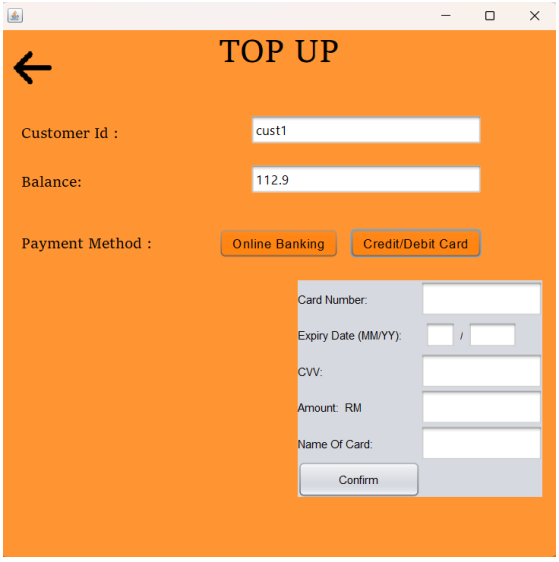


The Order Placed message will show to customer when they click the place order button.

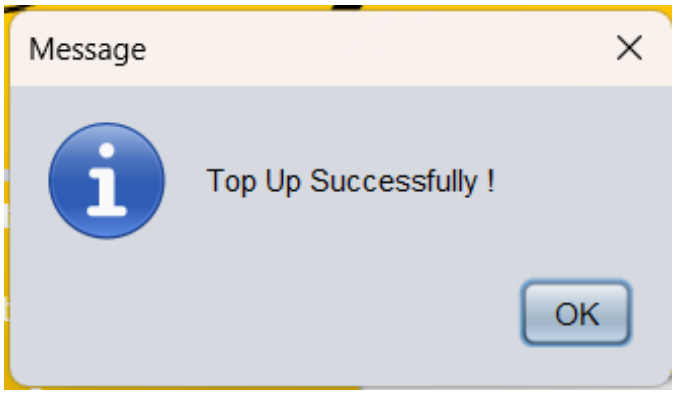


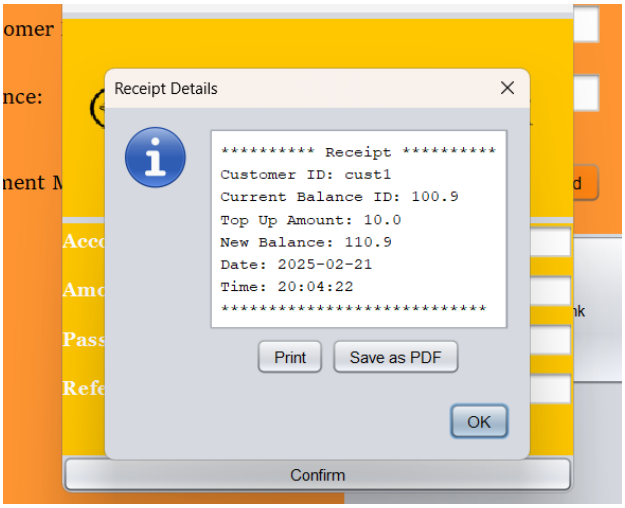
If the customer doesn't have enough credit to pay, a message will be shown to ask customer to top up their amount. If the customer chooses yes, the top up page will show. Customer can choose either online banking or using credit/debit card. As the figure below shown, customer will receive a notify message and a receipt when top up successful.

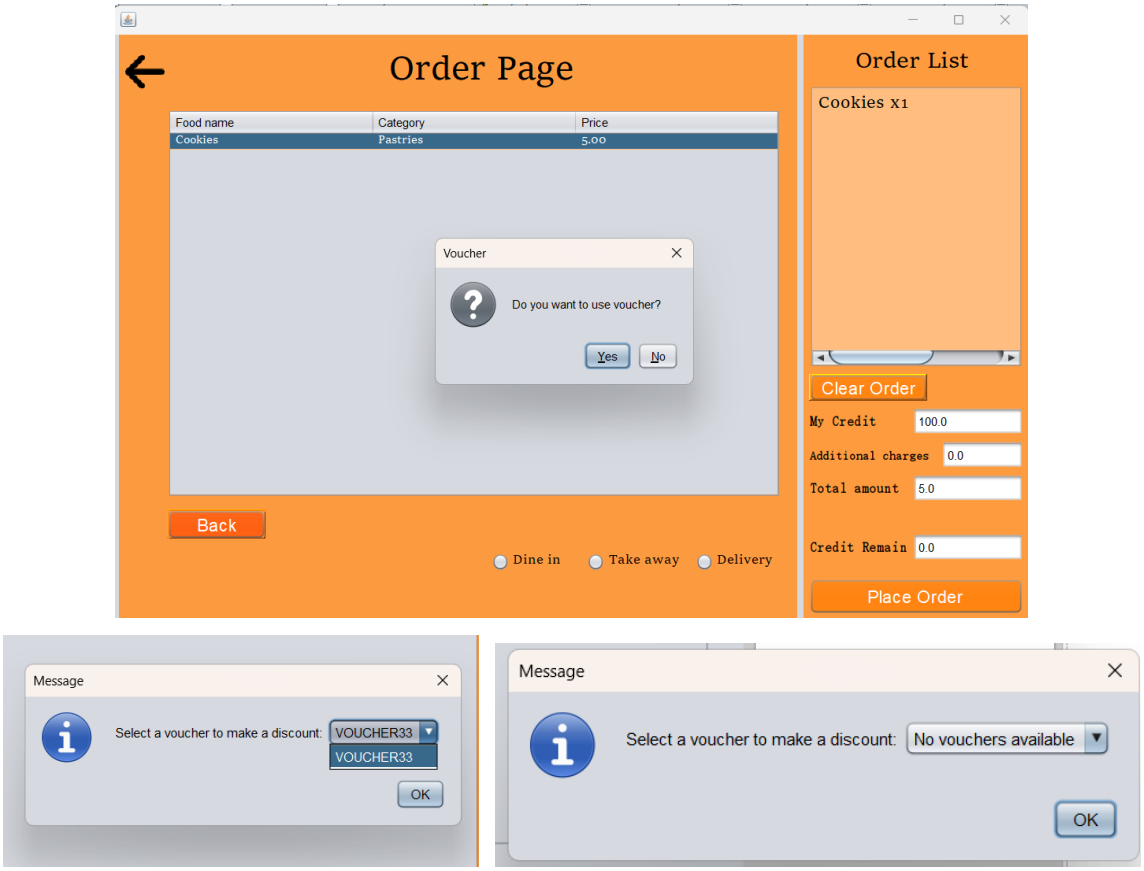

TOP UP
Customer Id :
cust1
Balance:
112.9
Payment Method :
Online Banking
Credit/Debit Card
Select a bank:
Maybank
Online Banking Top Up


TOP UP
Customer Id :
cust1
Balance:
112.9
Payment Method :
Online Banking
Credit/Debit Card
Card Number:
Expiry Date (MM/YY):
CVV:
Amount: RM
Name Of Card:
Confirm

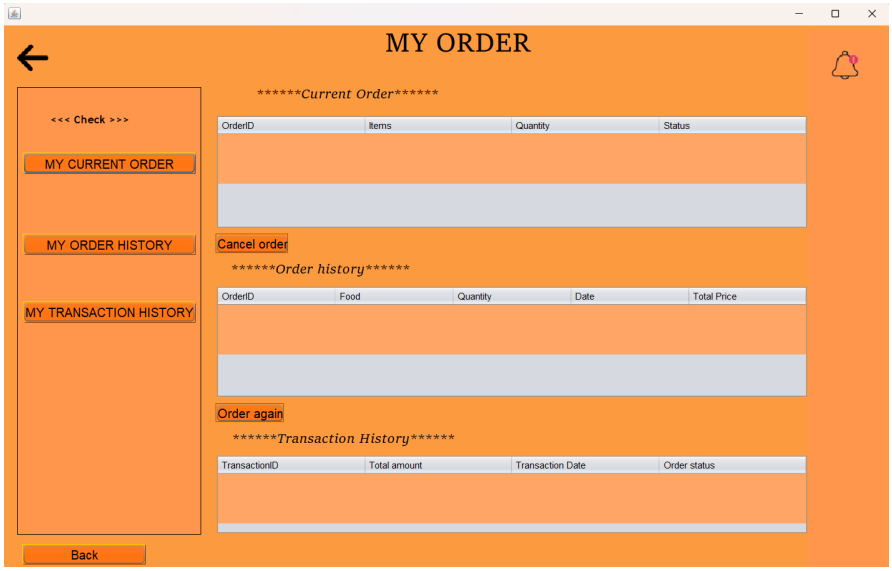

TOP UP
Maybank Page
Maybank
Account Number:
Amount:
Password:
Reference (Optional):
Confirm
RHB Bank
Hong Leong Bank
BSN
UOB


Message
Top Up Successfully !
OK

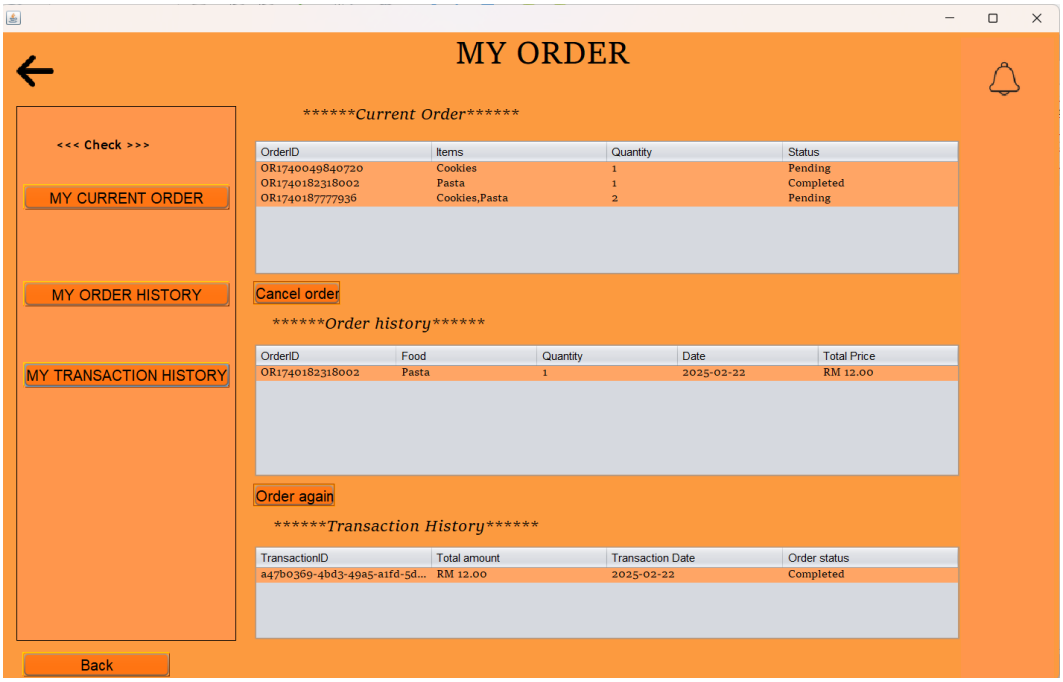

Receipt Details
********** Receipt **********
Customer ID: cust1
Current Balance ID: 100.9
Top Up Amount: 10.0
New Balance: 110.9
Date: 2025-02-21
Time: 20:04:22
****
Print
Save as PDF
OK
Confirm

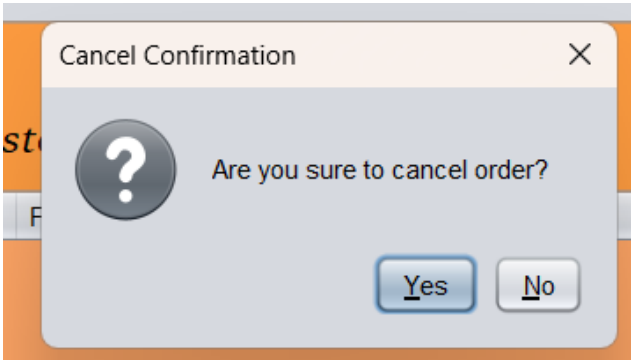


When customer is choosing their food, a message will show to ask customers to use the voucher or not. If customer choose yes, a message box will pop up and customer can choose their voucher from the combo box to be use. If the customer doesn't have voucher, the combo box will show "no vouchers available". After clicking "OK" button the total amount will decrease 10% as its percentage of discount of the voucher.

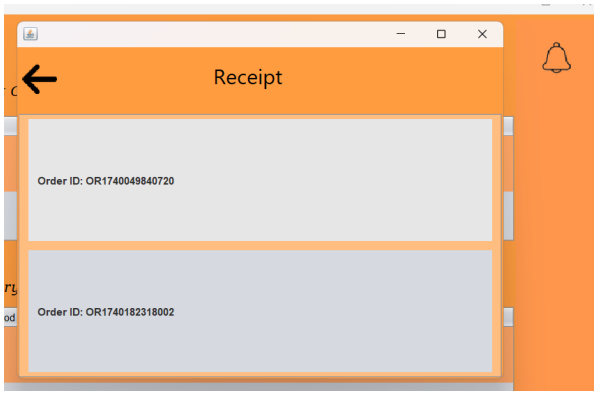


This form shows the order status of the customer. There are three buttons at the left side can be choose by customer. When customer click “My Current order” the first table will show all the order of customer. The “Cancel Order” button can be used to cancel and delete the order from text file and the table when customer choose an order from table and click this button. The message will pop up to ensure the customer want to delete the order. When “My Order History” was clicked, the order history table will show the order history which status is show “Completed”. If user click the “order again” button, the menu form will show to them to place order. When user click “My Transaction History” button, they can see their transaction amount of each order in transaction history table. The “Back” button will bring the user back to customer main page.

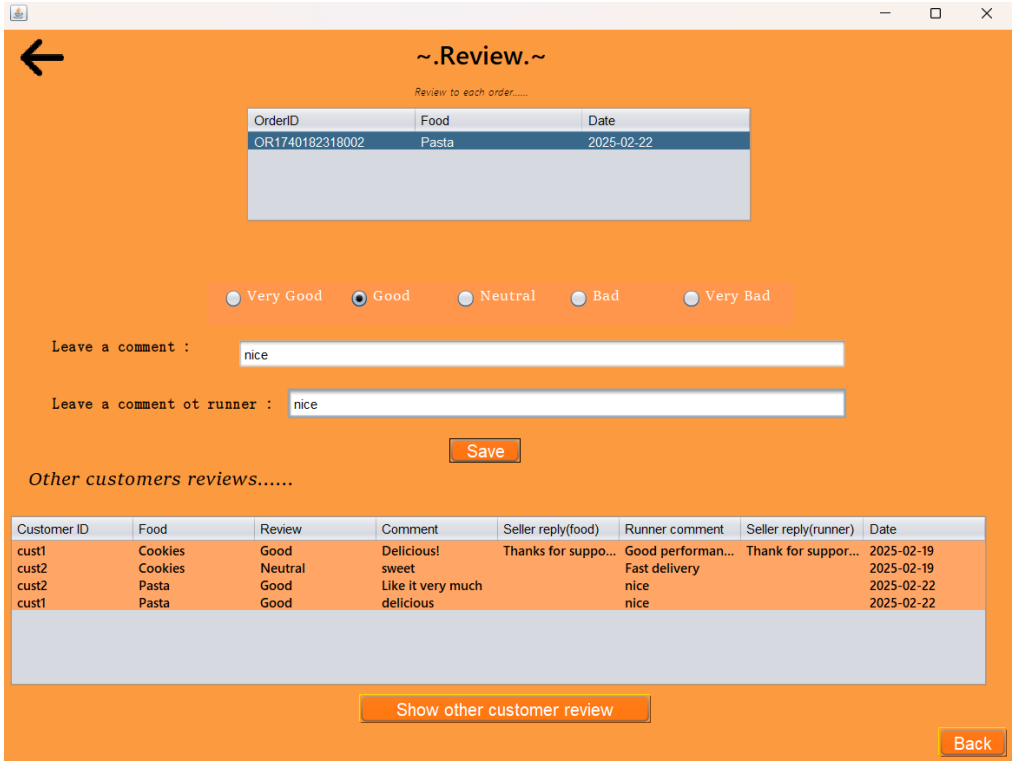


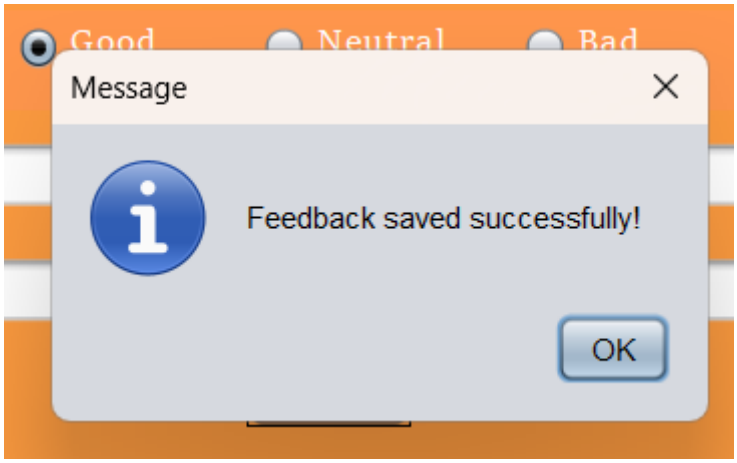


When an order places, the order status will be in pending status and the customer need to wait for the vendor to accept or decline their order. As the vendor accept the order, the customer will receive a notification, and the status will change to “Accepted”.

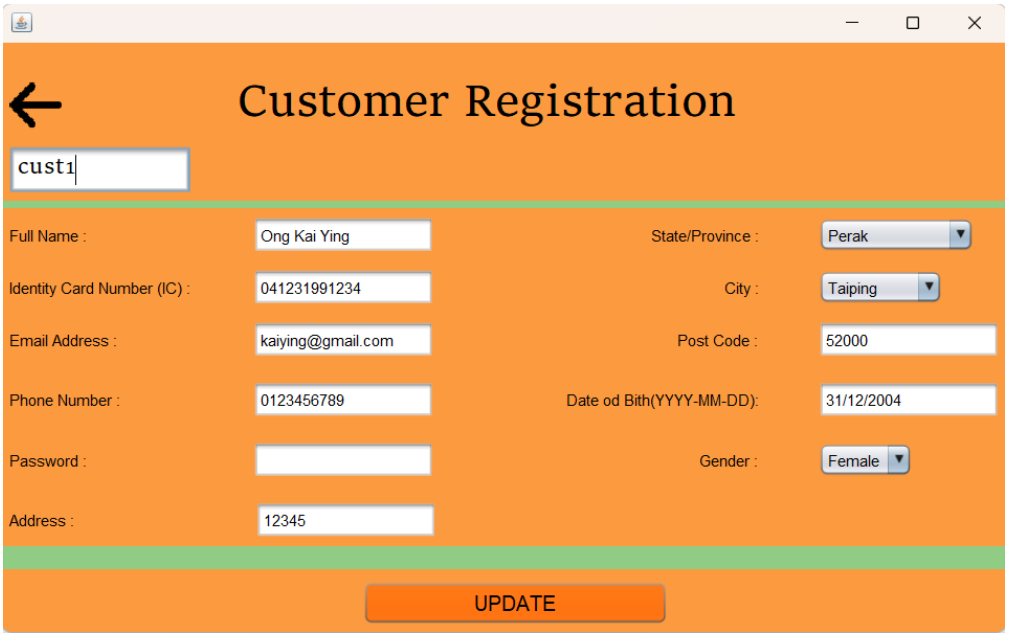


The customer can click the “bell” at the upper right to check their receipt.





This is the review form that customer can give a review to the ordered food and view other customer's comments. Customer can choose which order they want to comment from the table that had shown their order history. They can rate to that order, leave a comment and give comment to runner if they want. The "Save" button will record all the information fill by the customer into the feedback text file. The "Feedback saved successfully!" message will show to user when they save their comment. At the bottom part, the table will show all customer review when the "Show other customer review" button was clicked. In this form also have a "Back" button to allow user back to customer main page.



If user choose to update their information, they will need to come to this page and make modification to their information. After done changing, they need to click the "Update" button to save their change. All the information will be saved into customer registration text file.

4.0 Description and justification of Object-oriented concepts

4.1 Encapsulation

By using the private keyword, we apply encapsulation, ensuring that variables cannot be accessed directly from outside the class. This protects data and allows controlled modifications.

In our system, encapsulation is used in the Person, Customer, and Runner classes. The Person class has private attributes like name, ic, email, phoneNum, address, dob, gender, id, and password. The Customer class adds state, city, and postalCode, while the Runner class includes vehicleType, vehicleRegNum, and vehicleModel.

To access these private attributes, we use getter and setter methods, ensuring data integrity, security, and maintainability. This makes our system more organized and efficient.

```
public abstract class Person {

    private String name, ic, email, phoneNum, address,
            dob, gender, id, password;
```

```
public class Customer extends Person {

    private String state, city, postalCode;
```

```
public class Runner extends Person{
    private String vehicleType,vehicleRegNum,vehicleModel;
```

4.2 Polymorphism

Polymorphism allows one method name to be used for different types of behavior across multiple classes. For example, we used method overriding as we are using the same method name and parameters, but different behavior in a subclass. The polymorphism concept is used in vendor class extends person, which means vendor is a subclass of person. From the polymorphism, we can clearly see that it overrides two methods from Person, which are

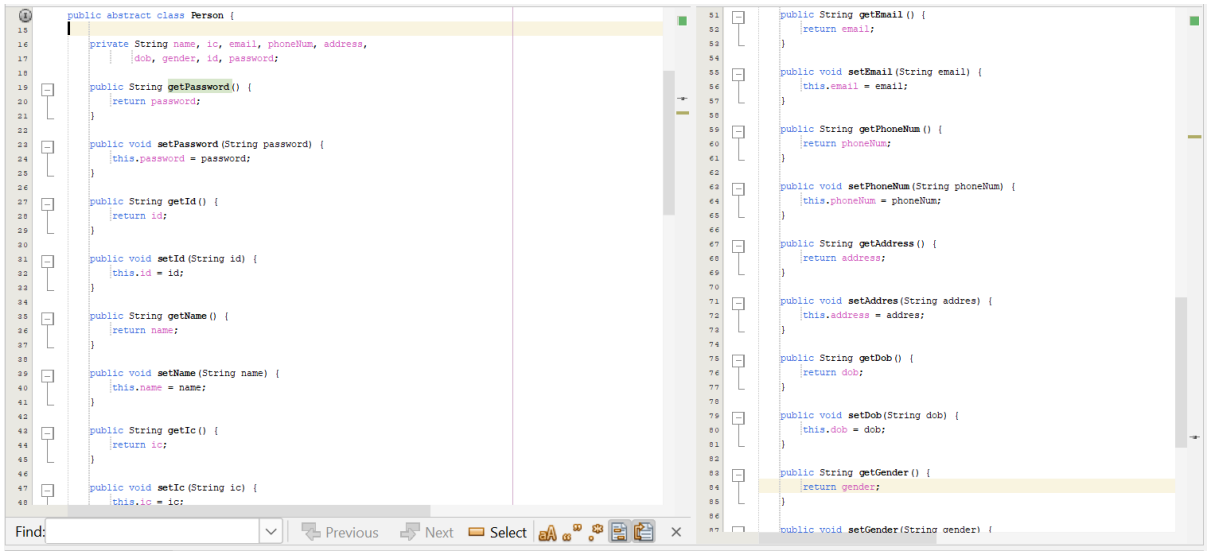
seachRecord(String ic) as well as generateId().

```
44      @Override
45      public boolean searchRecord(String ic) {
46              boolean recordFound = false;
47          try (BufferedReader br = new BufferedReader(new FileReader("runnerRegistration.txt"))) {
48              String line;
49              while ((line = br.readLine()) != null) {
50                  String[] columns = line.split("\\|");
51                  if (columns.length >= 8 && columns[1].trim().equals(ic)) {
52                      recordFound = true;
53                      break;
54                  }
55              }
56          } catch (IOException ex) {
57              JOptionPane.showMessageDialog(null, "An error occurred while reading the file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
58          }
59          return recordFound;
60      }
```

The searchRecord(String ic) in vendor allows them to search for registered vendors in “vendorRegistration.txt” file, which is a unique operation where specifically for vendor only. So, in order to search for registered vendors in the text file, the method will first read the text file line by line, and check if the provided IC number exists. As a result, the method returns “true” if the provided IC is found at “vendorRegistration.txt” which means the vendor is registered successfully. Otherwise, if the system returns “false”, it shows that the IC was not found in the system.

```
62      @Override
63      public String generateId() {
64          int maxId = 0;
65
66          try (BufferedReader br = new BufferedReader(new FileReader("runnerRegistration.txt"))) {
67              String line;
68              while ((line = br.readLine()) != null) {
69                  String[] parts = line.split("\\|");
70                  if (parts.length > 1) {
71                      String currentId = parts[parts.length - 1].trim();
72                      if (currentId.startsWith("rn")) {
73                          try {
74                              int idNumber = Integer.parseInt(currentId.substring(2));
75                              maxId = Math.max(maxId, idNumber);
76                          } catch (NumberFormatException ignored) {
77                          }
78                      }
79                  }
80              }
81          } catch (FileNotFoundException e) {
82          } catch (IOException e) {
83              e.printStackTrace();
84              return null;
85          }
86          return "rn" + (maxId + 1);
87      }
```

Secondly, the generateId() in vendor is implemented to generate a unique vendor ID that follows the format as vd1, vd2, vd3 and so on. By implementing this method, it can now automatically assign a unique vendor ID for every registered vendor. In this method, it will first read our text file to identify the highest existing ID and then generate a new ID for the newly registered vendor. By doing this, it can ensure that every vendor has their own unique ID to prevent duplication.



```
public abstract class Person {

    private String name, ic, email, phoneNum, address,
            dob, gender, id, password;

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getIc() {
        return ic;
    }

    public void setIc(String ic) {
        this.ic = ic;
    }
```

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

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPhoneNum() {
        return phoneNum;
    }

    public void setPhoneNum(String phoneNum) {
        this.phoneNum = phoneNum;
    }

    public String getAddress() {
        return address;
    }

    public void setAddres(String addres) {
        this.address = addres;
    }

    public String getDob() {
        return dob;
    }

    public void setDob(String dob) {
        this.dob = dob;
    }

    public String getGender() {
        return gender;
    }
```

Find: Previous Next Select

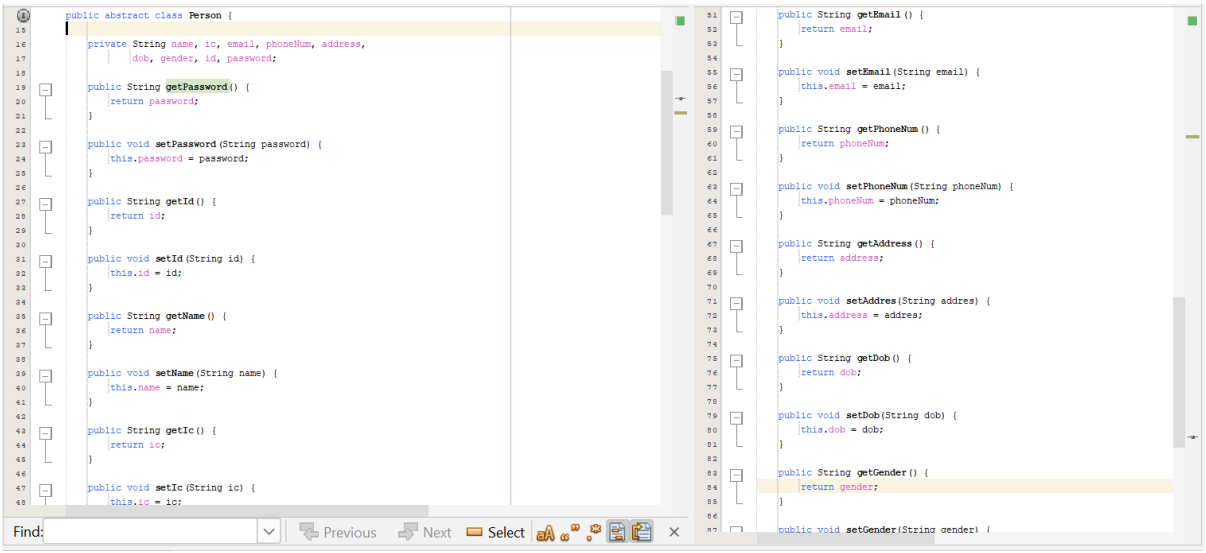
```
    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
```

4.3 Abstraction

A key idea in object-oriented programming (OOP) is abstraction, which focuses on keeping implementation specifics hidden and only showing an object's key attributes or functionalities. By emphasizing on what an object does rather than how it does it, it enables us to construct a simplified model of complicated systems.

In our system, an abstraction class Person was used. The abstraction is achieved through this abstract class and its concrete subclasses which are Customer, Runner and Vendor also.

The Person class is declared as abstract. It cannot be instantiated directly in this system. It contain the common attributes such as name, IC, email, phone number, address, DOB, gender, ID and password and the methods such as getName(), setName(), getPassword(), setPassword() and so on that are shared by all its subclasses. In this abstract class, it also declares the abstract methods which include searchRecord() and generateId(). These two methos do not have an implementation in the Person class and must be implemented by any subclass that extends Person.



```
    public String getDob() {
        return dob;
    }

    public void setDob(String dob) {
        this.dob = dob;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public abstract boolean searchRecord(String ic);
    public abstract String generateId();

}
```

The Customer class extend the Person class and provide concrete implementations for the abstract methods in Person class. Customer class have its own unique attributes and methods such as searchCustomerName() and inputChecking(). The same condition also exist in Runner class with its own attributes like vehicleType, vehicleRegNum and vehicleModel, along with its own inputChecking() method. The Vendor class does not has new attributes but it provide its own implementation of searchRecord() and generateId().

Runner:

```java
public class Runner extends Person{
    private String vehicleType,vehicleRegNum,vehicleModel;

    public String getVehicleType() {
        return vehicleType;
    }

    public void setVehicleType(String vehicleType) {
        this.vehicleType = vehicleType;
    }

    public String getVehicleRegNum() {
        return vehicleRegNum;
    }

    public void setVehicleRegNum(String vehicleRegNum) {
        this.vehicleRegNum = vehicleRegNum;
    }

    public String getVehicleModel() {
        return vehicleModel;
    }

    public void setVehicleModel(String vehicleModel) {
        this.vehicleModel = vehicleModel;
    }
```

```java
@Override
public boolean searchRecord(String ic) {
        boolean recordFound = false;
    try (BufferedReader br = new BufferedReader(new FileReader("runnerRegistration.txt"))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] columns = line.split("\\|");
            if (columns.length >= 8 && columns[1].trim().equals(ic)) {
                recordFound = true;
                break;
            }
        }
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null, "An error occurred while reading the file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return recordFound;
}

@Override
public String generateId() {
    int maxId = 0;

    try (BufferedReader br = new BufferedReader(new FileReader("runnerRegistration.txt"))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] parts = line.split("\\|");
            if (parts.length > 1) {
                String currentId = parts[parts.length - 1].trim();
                if (currentId.startsWith("rn")) {
                    try {
                        int idNumber = Integer.parseInt(currentId.substring(2));
                        maxId = Math.max(maxId, idNumber);
                    } catch (NumberFormatException ignored) {
```

```
public String inputChecking(String name, String ic, String email, String phone, String address,
        String dob, String password, String gender,String vehicleType,String vehicleRegNum,String vehicleModel) {

    if (name.matches(".*\\d.*")) {
        return "Name cannot contain numbers!";
    }

    if (ic.matches("\\d{6}-\\d{2}-\\d{4}")) {
        return "IC cannot contain letters!\n Example:110704-09-7423 \n IC number must be a valid 10-digit number.";
    }

    if (vehicleRegNum.matches("[A-Z]\\s\\d")) {
        return "IC cannot contain letters!\n Example:110704-09-7423 \n IC number must be a valid 10-digit number.";
    }

    if (!email.contains("@")) {
        return "Please enter a valid email address (must contain '@').";
    }

    if (!phone.matches("01\\d-\\d{7}")) {
        return "Invalid format for phone number.\n Example:01x-xxxxxxx \n Phone number must be a valid 10-digit number.";
    }

    try {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        sdf.setLenient(false);
        sdf.parse(dob);  // Will throw an exception if the date format is incorrect
    } catch (ParseException e) {
        return "Date of Birth must be in the format YYYY-MM-DD.";
    }

    return null;
}
```

Vendor:

```
public class Vendor extends Person {

    @Override
    public boolean searchRecord(String ic) {
        // Implementation specific to Vendor
        boolean recordFound = false;
        try (BufferedReader br = new BufferedReader(new FileReader("vendorRegistration.txt"))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] columns = line.split("\\|");
                if (columns.length >= 9 && columns[1].trim().equals(ic)) {
                    recordFound = true;
                    break;
                }
            }
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null, "An error occurred while reading the file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
        return recordFound;
    }

    @Override
    public String generateId() {
        int maxId = 0;

        try (BufferedReader br = new BufferedReader(new FileReader("vendorRegistration.txt"))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split("\\|");
                if (parts.length > 1) {
                    String currentId = parts[parts.length - 1].trim();
                    if (currentId.startsWith("vd")) {
                        try {
                            int idNumber = Integer.parseInt(currentId.substring(2));
                            maxId = Math.max(maxId, idNumber);
                        } catch (NumberFormatException ignored) {
                        }
                    }
                }
            }
        } catch (FileNotFoundException e) {
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }

        return "vd" + (maxId + 1);

    }

    public String inputChecking(String name, String ic, String email, String phone, String address, String dob, String password, String gender) {

        if (name.matches(".*\\d.*")) {
            return "Name cannot contain numbers!";
        }

        if (ic.matches("\\d{6}-\\d{2}-\\d{4}")) {
            return "IC cannot contain letters!\n Example:110704-09-7423 \n IC number must be a valid 10-digit number.";
        }

        if (!email.contains("@")) {
            return "Please enter a valid email address (must contain '@').";
        }

        if (!phone.matches("01\\d-\\d{7}")) {
            return "Invalid format for phone number.\n Example:01x-xxxxxxx \n Phone number must be a valid 10-digit number.";
        }
```

Customer:

```
public class Customer extends Person {

    private String state, city, postalCode;

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getPostalCode() {
        return postalCode;
    }

    public void setPostalCode(String postalCode) {
        this.postalCode = postalCode;
    }
```

```
        while ((line = br.readLine()) != null) {
            String[] columns = line.split("\\|");
            if (columns.length >= 12 && columns[1].trim().equals(ic)) {
                recordFound = true;
                break;
            }
        }
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null, "An error occurred while reading the file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return recordFound;
}
```

```
public String searchCustomerName(String id) {
    String name = "";
    try (BufferedReader br = new BufferedReader(new FileReader("custRegistration.txt"))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] columns = line.split("\\|");
            if (columns.length >= 12 && columns[11].trim().equals(id)) {
                name = columns[0];
                break;
            }
        }
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(null, "An error occurred while reading the file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
    return name;
}

@Override
```

```
public String generateId() {
    int maxId = 0;

    try (BufferedReader br = new BufferedReader(new FileReader("custRegistration.txt"))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] parts = line.split("\\|");
            if (parts.length > 1) {
                String currentId = parts[parts.length - 1].trim();
                if (currentId.startsWith("cust")) {
                    try {
                        int idNumber = Integer.parseInt(currentId.substring(4));
                        maxId = Math.max(maxId, idNumber);
                    } catch (NumberFormatException ignored) {
                    }
                }
            }
        }
    } catch (FileNotFoundException e) {
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }

    return "cust" + (maxId + 1);
}
```

```
public String inputChecking(String name, String ic, String email, String phone, String address,
        String dob, String password, String gender, String state, String city, String postalCode) {

    if (name.matches(".*\\d.*")) {
        return "Name cannot contain numbers!";
    }

    if (ic.matches("\\d{6}-\\d{2}-\\d{4}")) {
        return "IC cannot contain letters!\n Example:110704-09-7423 \n IC number must be a valid 10-digit number.";
    }

    if (postalCode.matches("\\d{5}")) {
        return "PostCode should be 5 ";
    }

    if (!email.contains("@")) {
        return "Please enter a valid email address (must contain '@').";
    }

    if (!phone.matches("01\\d-\\d{7}")) {
        return "Invalid format for phone number.\n Example:012-6743912 \n Phone number must be a valid 10-digit number.";
    }

    try {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        sdf.setLenient(false);
        sdf.parse(dob); // Will throw an exception if the date format is incorrect
    } catch (ParseException e) {
        return "Date of Birth must be in the format YYYY-MM-DD.";
    }

    return null;
}
```

The Customer, Runner and Vendor classes inherit from the Person class. This means they automatically get all the attributes and methods defined in Person. They also override the abstract methods searchRecord() and generateId() to provide their own specific implementations. The Person class abstracts away the common functionality and leaves the specific implementation details to its subclasses. By defining common attributes and methods in the Person class, we can avoid duplicating code across Customer, Runner, and Vendor. It is flexible to use this abstract class and we no need to do modify if we want to add new sub class at the future as long as they implement the required abstract methods.

The abstraction is achieved through the Person abstract class, which defines common attributes and methods while leaving specific implementations to its subclasses. The relationship among the classes is based on inheritance, where subclasses extend the Person class and provide their own implementations for abstract methods. This design promotes code reusability, flexibility, and maintainability

4.4 Inheritance

We have a superclass called Person, which serves as the base class for three subclasses: Customer, Vendor, and Runner. The Person class contains common attributes such as name, IC, email, phone number, address, date of birth, gender, ID, and password.

```
public abstract class Person {

    private String name, ic, email, phoneNum, address,
            dob, gender, id, password;

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }
```

The Customer class extends Person and includes additional attributes specific to customer information: state, city, and postal code.

```
public class Customer extends Person {

    private String state, city, postalCode;

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getCity() {
        return city;
    }
```

The Runner class also extends Person, but it adds attributes related to vehicle details: vehicle type, vehicle registration number, and vehicle model.

```
public class Runner extends Person{
    private String vehicleType,vehicleRegNum,vehicleModel;

    public String getVehicleType() {
        return vehicleType;
    }

    public void setVehicleType(String vehicleType) {
        this.vehicleType = vehicleType;
    }

    public String getVehicleRegNum() {
        return vehicleRegNum;
    }

    public void setVehicleRegNum(String vehicleRegNum) {
        this.vehicleRegNum = vehicleRegNum;
    }
```

The Vendor class inherits directly from Person without adding any extra attributes.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this li
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package domain;

import domain.Person;
import java.io.*;
import java.text.*;
import javax.swing.JOptionPane;

/**
 *
 * @author HP
 */
public class Vendor extends Person {

    @Override
    public boolean searchRecord(String ic) {
        // Implementation specific to Vendor
        boolean recordFound = false;
        try (BufferedReader br = new BufferedReader(new FileReader("vendorRegistration.txt"))) {
            String line;
```

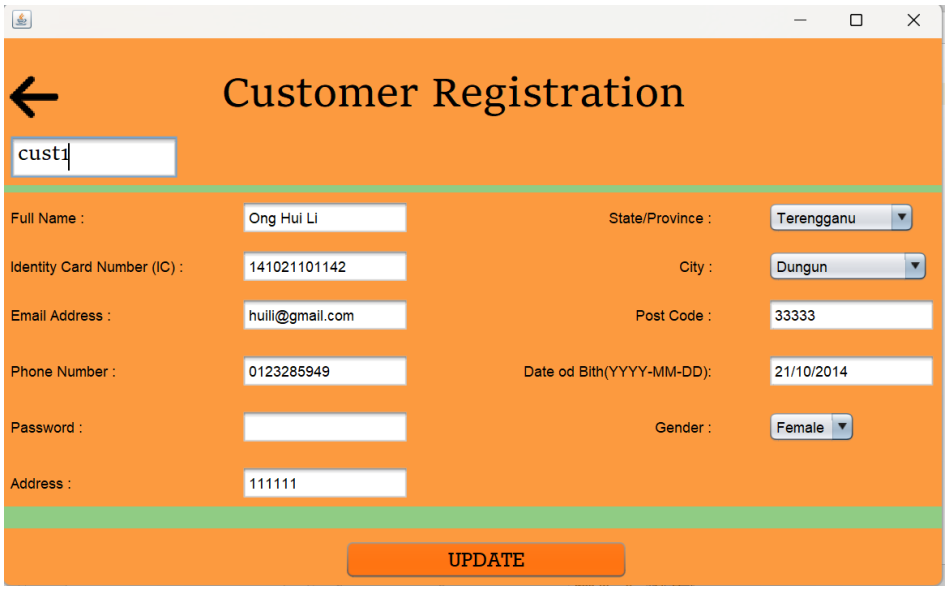
5.0 Additional features

Customer

1. Customer can edit information

Customers can update their personal information by logging into their accounts. Once they successfully log in, the system will hold their customer ID and automatically display their existing details. This allows them to easily review and modify their information as needed.

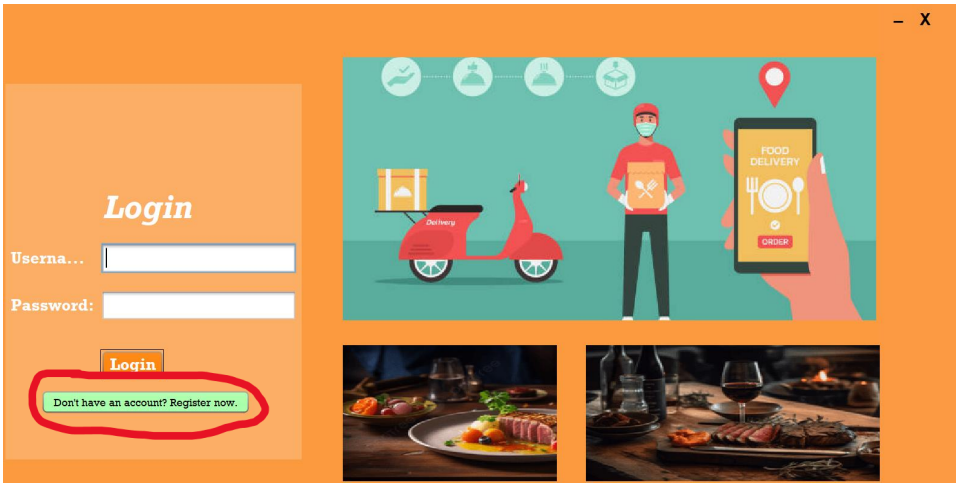
Unlike the admin-assisted update process, this feature gives customers the flexibility to edit their own details at any time, without needing to contact support. This makes the process more convenient, efficient, and user-friendly, ensuring that customers can keep their information up to date with minimal hassle.



2. Customer can register by themselves

An additional feature is implemented in this system where allows customer to register their own accounts directly from the login page, without the need of waiting for admin to create a account. By having this, it can simplify the process of registration as they only need to fill in their basic information such as name, IC number, email, phone number, address, date of birth, as well as gender. Furthermore, customers also required to set their username and password to

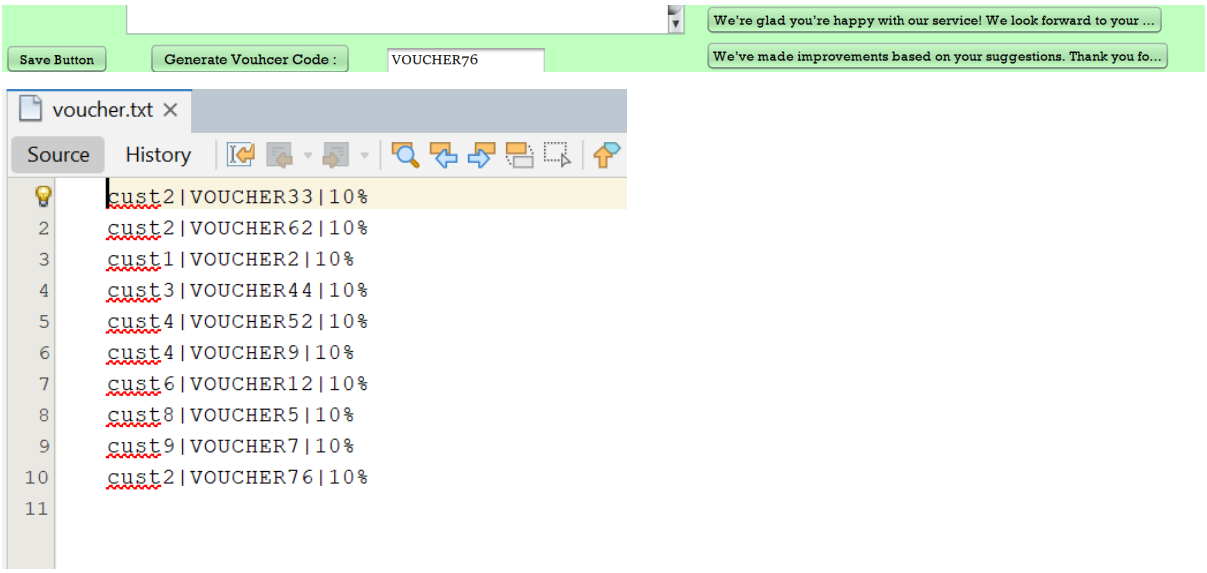
protect their account. By enabling self-registration for customers, it can help to minimize admin's tasks and improves user accessibility.



3. Manager can generate voucher by referring user Id and user can use voucher

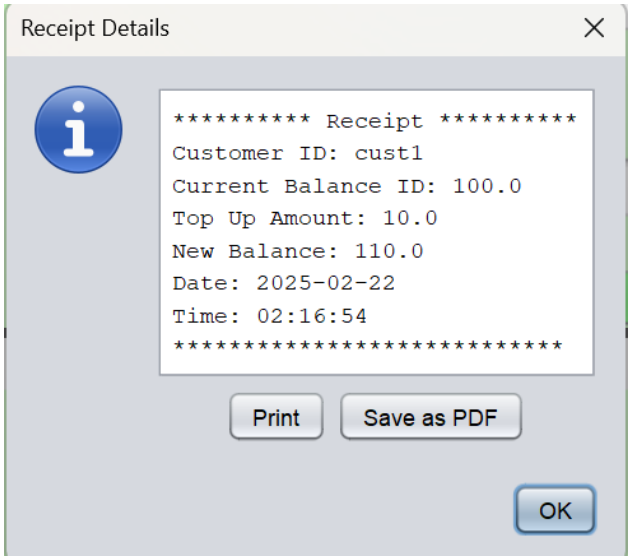
The voucher is given by the manager to customer depends on the situation. This feature is useful for rewarding loyal customers or resolving customer complaints. The voucher code is saved in voucher text file along with customer's ID and the discount percentage. One customer can have many vouchers as each voucher are create with unique voucher ID randomly. The customer receives and use the voucher given to get the discount on their order. Vouchers are tied to the customer's ID, ensuring they can only be used by the intended customer. The system

applies the discount to the total order amount. After a voucher is used, it is removed from the voucher text file to prevent reuse.



4. Receipt can be printed out and save as pdf

After every top-up or order placement, the system will automatically generate a receipt. This receipt will be sent to both the customer and the admin, ensuring that they have a record of the transaction. Customers and admins can click to view the receipt and then choose whether to print it or save it as a PDF for future reference. This feature provides a convenient and organized way to keep track of transactions.



6.0 Limitation and conclusion

Limitation:

Lack of Advanced Data Operations

Compared to a database, text files store data in a basic and unorganized way, making it harder to perform tasks like searching, sorting, or filtering. For example, if we want to find a specific order in a text file, we have to go through the entire file, checking each line. This can take a long time, especially as the file grows in size.

On the other hand, a database makes it easy to search and filter data using commands like SQL, which is faster and more efficient. For example, with a simple command like SELECT * FROM orders WHERE status='accepted', we can quickly get all the accepted orders without manually checking each one. In a text file, however, filtering or searching for specific data takes more time because we need to write extra code to check each line and compare it to certain conditions.

Since we have to do all of this manually in our code, it can take longer to write and increases the chances of errors. Fixing mistakes can also be more difficult. Moreover, as the file grows larger, it becomes slower to process because we must check every line, unlike a database that has built-in tools to handle these tasks more quickly.

Conclusion:

In conclusion, our food court management system offers a robust solution that caters to the needs of customers, vendors, delivery runners, administrators, and managers. By streamlining the food ordering process, providing real-time updates, and offering detailed analytics, the system enhances operational efficiency and user satisfaction. In our system, the customer, runner, vendor, admin and manager are able to login the system and operate the function respectively. Our customer can view food menu, place order, top up their credit by using online banking, credit/debit card or giving cash to admin to top up manually and use voucher to get the discount on their order. Customer are given the choice to dine in, take away or delivery their order. To check the order status or cancel order, the customer is also able to check it in

specific page and they can also check their order history and reorder refer to their order history table in the same page. The customers also able to check their transaction history. In customer review part, the customer can make a rate and comment to their order, and they can see all customer review.

For the vendor, they are able to add, update, and delete the item in the food menu. They will receive an order requirement and notification from customers, and they can decide to accept or decline and update the order status. The vendor also able to check the order history and see the feedback from customers. To check the revenue, vendor can go to revenue dashboard to check according to the monthly revenue.

For the runner, they can view the task and accept or decline the task. After they accepted the task, they need to update the status of the order. They can also choose to check their task history and see their revenue in daily, weekly, monthly or yearly. They also have the access to view the customer comment to them.

The administrator in this system is able to create, read and update the information of vendor, customer and runner. The administrator can help to update the customer's credit manually. An order transaction receipt also will be created by administrator automatically. The receipt will be sent to customer's order status page together with a notification sound.

For the Manager, they have the access to the revenue dashboard to see the performance of the vendor. The main task of manager is to check the review from customer to their order and the runner to solve the customer complaints and check runner performance. They also have the access to remove the not appropriate item in the food menu. All of the information during the system process are save to their respective text file.

Although its reliance on text files for data storage, the system effectively meets the basic requirements of a modern food court management solution, offering convenience and efficiency for all users. Future enhancements could include integrating a database for improved data management and scalability.

7.0 References

Ellis, S. (6 February, 2025). *What is Encapsulation in Java? - A Complete Explanation*. Retrieved from The Knowledge Academy: https://www.theknowledgeacademy.com/blog/encapsulation-in-java/

Gondi. (2 October, 2024). *Java Encapsulation: Principles, Benefits, and Practical Examples*. Retrieved from Medium: https://gondi-sai.medium.com/java-encapsulation-principles-benefits-and-practical-examples-java-no-18-8fbdfa1d3210

Luthra, T. (7 May, 2024). *Polymorphism in Java*. Retrieved from Scaler: https://www.scaler.com/topics/java/polymorphism-in-java/

Obregon, A. (1 March, 2024). *Beginner's Guide to Java Polymorphism*. Retrieved from Medium: https://medium.com/@AlexanderObregon/beginners-guide-to-java-polymorphism-a35ee0d67e35

Pankaj. (4 August, 2022). *Java Read Text File*. Retrieved from Digital Ocean: https://www.digitalocean.com/community/tutorials/java-read-text-file

Pradhan, A. (22 July, 2022). *10-Minute Guide To Abstraction In Jave*. Retrieved from Crio.do: https://www.crio.do/blog/abstraction-in-java/

Sharma, J. (12 October, 2023). *All About Data Abstraction in Java*. Retrieved from Shiksha.com: https://www.shiksha.com/online-courses/articles/data-abstraction-in-java/

Vasani, C. (19 August, 2023). *Guide to Java Methods with Examples*. Retrieved from Medium: https://medium.com/@cpvasani48/guide-to-java-methods-with-examples-71d0c1112be1

8.0 Appendix

Menu Text File

```
Spaghetti Bolognese|Main|25.0
Squid Ink Pasta with Seafood|Main|30.0
Carbonara Pasta|Main|28.0
Creamy Mushroom Pasta|Main|25.0
Cheesy Baked Pasta|Main|28.0
Garlic Bread with Cheese|Starter|13.0
Cream of Mushroom Soup|Starter|8.0
Mozzarella Sticks|Starter|10.0
Chicken Wings|Starter|10.0
Fresh Lemonade|Drinks|5.0
Iced Lemon Tea|Drinks|8.0
Iced Peach Tea|Drinks|8.0
Latte|Drinks|9.0
Cappuccino|Drinks|9.0
Chocolate|Drinks|8.0
Cookies|Starter|5.0
```

Order Text File

```
OR1740049840720|cust1|Cookies|1|Delivery|RM 5.00|Completed|2025-02-22|9fe44728-d57e-4dcf-8df5-2d603c7c1df4|4, Jalan 3, Segambut Dalam, Perak, Taiping,52000|0
OR1741312321242|cust2|Cookies|1|Delivery|RM 10.00|Completed|2025-01-22|9fe44728-d57e-4dcf-8df5-2d603c7c1df4|4, Jalan 3, Taman Maluri, Perak, Taiping,52010|13
OR1324325355444|cust3|Iced Lemon Tea|1|Delivery|RM 8.00|Pending|2025-01-22|9fe44728-d57e-4dcf-8df5-2d603c7c1df4|9, Jalan 7, Taman Tioman, Selangor, Taiping,5
OR3435435435453|cust4|Iced Lemon Tea|2|Delivery|RM 16.00|Completed|2024-01-22|9fe44728-d57e-4dcf-8df5-2d603c7c1df4|7, Jalan 21, Taman Samudera, Shah Alam, Se
OR1740439820345|cust5|Spaghetti Bolognese|1|Delivery|RM 25.00|Food Pickup|2025-02-13|5fe44728-d57e-4dcf-8df5-2d603c7c1df5|10, Jalan 5, Taman Bukit Indah, Kua
OR1740439820346|cust6|Squid Ink Pasta with Seafood|2|Delivery|RM 60.00|Order Accepted by Runner|2025-01-20|3fe44728-d57e-4dcf-8df5-2d603c7c1df6|12, Jalan 9,
OR1740439820347|cust7|Carbonara Pasta|1|Delivery|RM 28.00|Completed|2025-02-19|2fe44728-d57e-4dcf-8df5-2d603c7c1df7|15, Jalan 14, Taman Melawati, Kuala Lumpu
OR1740439820348|cust8|Creamy Mushroom Pasta|1|Delivery|RM 25.00|Food Pickup|2025-01-12|7fe44728-d57e-4dcf-8df5-2d603c7c1df8|18, Jalan 21, Taman Segar, Selang
OR1740439820349|cust9|Cheesy Baked Pasta|2|Delivery|RM 56.00|Order Prepared|2025-02-02|9fe44728-d57e-4dcf-8df5-2d603c7c1df9|20, Jalan 30, Taman Setia, Selang
OR1740439820350|cust10|Garlic Bread with Cheese|1|Delivery|RM 13.00|Pending|2025-02-03|6fe44728-d57e-4dcf-8df5-2d603c7c1df0|25, Jalan 40, Taman Megah, Kuala
OR1740439820351|cust1|Cream of Mushroom Soup|2|Delivery|RM 16.00|Order Prepared|2024-12-08|8fe44728-d57e-4dcf-8df5-2d603c7c1df1|30, Jalan 50, Taman Sri Muda,
OR1740439820352|cust2|Mozzarella Sticks|3|Dine-in|RM 30.00|Order Prepared|2025-02-14|4fe44728-d57e-4dcf-8df5-2d603c7c1df2|null|17:55:30|runnerID|Lim Jia Ying
OR1740439820353|cust3|Chicken Wings|1|Delivery|RM 10.00|Order Prepared|2025-02-03|1fe44728-d57e-4dcf-8df5-2d603c7c1df3|40, Jalan 70, Taman Bukit Jalil, Kuala
```

Feedback Text File

```
OR1741312321242|cust2|Cookies|Good|Tastyyy!|Thanks for support|On time!|Thanks for support|2025-02-22
OR3435435435453|cust4|Iced Lemon Tea|Good|Delicious!|Thanks for supporting!|Good performance|Thank for supporting!|2024-01-22
OR1740439820347|cust7|Carbonara Pasta|Bad|Not worth it|Try better|Late|Do better|2025-02-19
OR1740439820355|cust5|Iced Lemon Tea|Good|Yummy!|Thanks for everything!|Excellent|Thanks for everything!|2024-01-22
OR1740439820357|cust7|Latte|Okay!|Not bad but make it stronger in flavor!|Thanks!|On time|Thanks!|2024-04-09
OR1740439820364|cust4|Creamy Mushroom Pasta|Good|Flavorful!|Thank you|A little late|Thanks|2025-01-11
OR1740439820365|cust5|Cheesy Baked Pasta|Bad|Not worth it, very little cheese|Try better|Very late, late for 1 hour!|Do better|2025-02-21
OR1740439820366|cust6|Garlic Bread with Cheese|Good|Good|Perfect|Fast Delivery!|Perfection|2025-01-31
OR1740439820375|cust5|Chocolate|2|Delivery|Good|Good|Perfect|Fast Delivery!|Perfection|2025-01-22
```

Balance Text File

```
cust1|129.10
cust2|92.0
cust3|0.00
cust4|30.00
cust5|886.00
cust6|34.00
cust7|233.00
cust8|66.00
cust9|77.50
```

Voucher Text File

```
cust2|VOUCHER33|10%
cust2|VOUCHER62|10%
cust1|VOUCHER2|10%
cust3|VOUCHER44|10%
cust4|VOUCHER52|10%
cust4|VOUCHER9|10%
cust6|VOUCHER12|10%
cust8|VOUCHER5|10%
cust9|VOUCHER7|10%
cust2|VOUCHER26|10%
```

Customer Registration Text File

```
Ong Hui Li|141021101142|huili@gmail.com|0123285949|4,Jalan Prima Pelangi 3|Terengganu|Dungun|33333|21/10/2014|Female|111111|cust1
Ong Kai Ying|041231080046|nikcoleong31@gmail.com|012-3229657|4,jalan 3|Kedah|Sungai Petani|24444|31/10/2004|Male|12345|cust2
Lyn Ang|018274018826|kit@gmail.com|0123234578|6,jalan3|Kedah|Sungai Petani|32333|10/10/2011|Female|28437h|cust3
Sam Yong Yi Heng|178263499926|sam@gmail.com|0123727338|8,jalan AmS|Kuala Lumpur|Bukit bintang|36533|14/10/2006|Male|hsy462|cust4
Ian Goln|716354829917|ian@gmail.com|012329288|19,jalan 78|Perak|Sungai Alin|22202|31/1/2008|Male|ssg4|cust5
Loey|921127010814|loey@gmail.com|0138891123|20,jalan durian|Pahang|S Petani|77977|27/11/1992|Male|10v3xo|cust6
Cherry Yap|039823048937|cherry@gmail.com|0169273455|02,jalan B|Johor|Sungai Mati|84400|14/8/2004|Female|yys6|cust7
Sin Xia Lau|789034654782|lau@gmail.com|0117293554|101,jalan SO|Kedah|Jahani Taman|22455|10/10/2011|Female|h0nd3|cust8
Eddy Golden|019803839923|eddy@gmail.com|01178221634|55,jalan XO|Kedah|Sungai P|22222|1/10/1979|Male|yat571|cust9
```

Runner Registration Text File

```
Derick Ong|001208101322|derickong@gmail.com|0121412219|15, Jalan 24/29 |08/12/2000|Male|Car|VGG9676|Toyota|11001|rn1
Wong Pei Pei|990401141622|wongpeipei@gmail.com|0134457812|33, Jalan Impian 5|01/04/1999|Female|Motorcycle|VGB5221|Yamaha|11002|rn2
Tan Chun Kit|970604101778|chunkit@gmail.com|0160221475|19, Jalan Emas 1|04/06/1997|Male|Car|VLV0766|Honda|147596|rn3
Ai Ling|971217143557|ailing97@gmail.com|0123239910|262, Jalan Prima 3a|17/12/1997|Female|Car|JVT7525|Proton|121245|rn4
Joanna Wong|010921014228|joanna28@gmail.com|0136657141|12, Jalan Sentosa|21/09/20041Female|Motorcycle|VH1726|Yamaha|787410|rn5
Brian Tan|001222101776|briantan@gmail.com|0199981252|8, Jalan 16/28|22/12/2000|Male|Motorcycle|VB3696|Yamaha|12458|rn6
Dylan Ooi|971106101334|dylanooi97@gmail.com|0163351019|4, Taman Prima 1|06/11/1997|Male|Car|PBC8141|Proton|85141|rn7
Ethan Lim|980626171662|ethan98@gmail.com|0191140236|32, Jalan Gembira 3|26/06/1998|Male|Car|VLL6617|Honda|12965|rn8
Garry Wong|021209101377|garrywong@gmail.com|0124471205|20, Jalan 34/8|09/12/2002|Male|Motorcycle|VBR9677|Yamaha|101477|rn9
```

Vendor Registration Text File

```
Cherry Hiew|021214101775|cherry02@gmail.com|0121140253|4, Jalan Putih 3|2002-12-14|Female|12225|vd1
Ong Pei Hua|991205141332|peihuaong99@gmail.com|0147785236|2, Jalan Ceria 21|1999-12-05|Female|17471|vd2
Angeline Tan|000321101477|angeline00@gmail.com|0196228351|14, Jalan 10/1|2000-03-21|Female|11740|vd3
Darren Lim|981107142022|darrenlim@gmail.com|0173631402|12, Jalan A124|1998-11-07|Male|32333|vd4
Miko Tan|970913101336|mikotan@gmail.com|0127541331|10,Jalan Pelangi|1997-09-13|Female|96677|vd5
Joan Wong|010711142055|joanwong01@gmail.com|0191358542|3, Jalan 11/12|2001-07-11|Female|23774|vd6
Adrian Ooi|030505101224|adrianooi@gmail.com|0123440085|28, Taman Mewah|2003-05-05|Male|04741|vd7
Janiel Lim|001229147822|janiel00@gmail.com|0123632351|6, Jalan Istimewa 3|2000-12-29|Female|55252|vd8
Desmond Tan|970408101332|desmondtan97@gmail.com|0162885941|9,Jalan A/13|1997-04-08|Male|96988|vd9
```

9.0 Workload Matrix Form

Component	Student 1 Name: Kate Lim Jia Yee TP070517	Student 2 Name: Manreen Kaur A/P Jagjit Singh TP071290	Student 3 Name: Ong Kai Ying TP086065	Student 4 Name: Tee Ching Ying TP072694	Student 5 Name:	Total
a) Use Case Diagram	**25%**	**25%**	**25%**	**25%**		**100%**
b) Class Diagram	**10%**	**10%**	**10%**	**70%**		**100%**
c) Vendor Role Functions	**100%**	**0%**	**0%**	**0%**		**100%**
d) Customer Role Functions	**0%**	**0%**	**0%**	**100%**		**100%**
e) Delivery Runner Role Functions	**0%**	**100%**	**0%**	**0%**		**100%**
f) Administrator Role Functions	**0%**	**0%**	**100%**	**0%**		**100%**
g) Manager Role Functions	**0%**	**0%**	**100%**	**0%**		**100%**
h) Screenshots of output of the program with appropriate explanations	**25%**	**25%**	**25%**	**25%**		**100%**
i) Description and justification of Object-oriented concepts incorporated into the solution	**25%**	**25%**	**25%**	**25%**		**100%**

j) Additional feature	**25%**	**25%**	**25%**	**25%**		**100%**

Video Presentation Link

https://1drv.ms/v/c/76387d1f269abd20/ESskmiXyRXpPn7pFY29n6JgBWSLykOKW50jdK4j3Ad80CA?e=A6cnV1