OODJ -112024

Object Oriented Design with Java

Group Assignment (Group 18)

Topic: Food Ordering System

Lecturer Name: Muhammad Huzaifah Ismail

Group Member: Liew Hao Shean (TP070311)
Chin Xuan Han (TP070331)
Chong Jia Jun (TP070683)
Tan Po Yeh (TP070780)

Date Assigned: 28 November 2024

Due Date: 22 February 2025

Intake Code: APU2F2411CS (AI)

Table of Contents

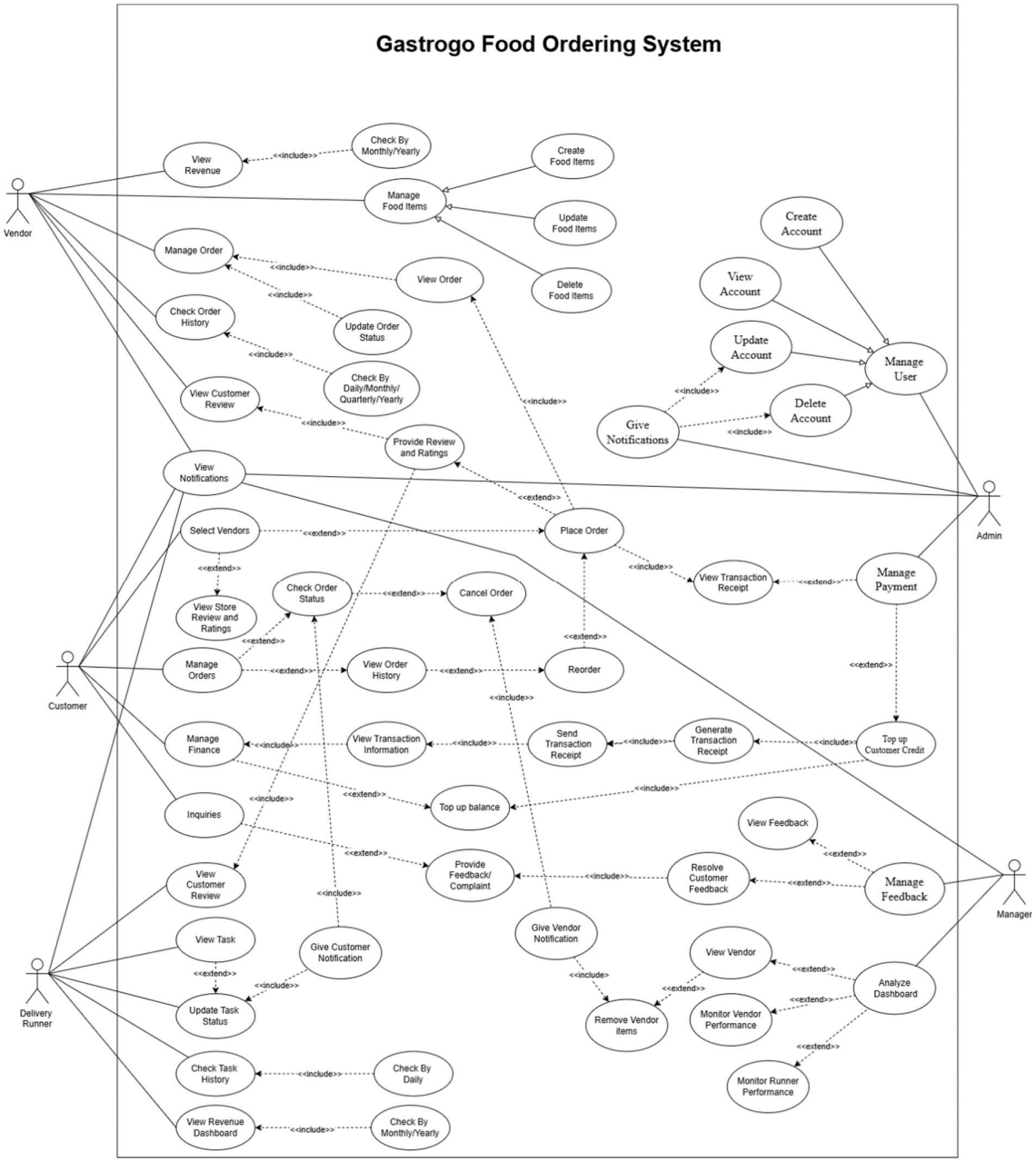
Workload Matrix

Component	Liew Hao Shean	Chin Xuan Han	Chong Jia Jun	Tan Po Yeh	Total
a) Use Case Diagram	**50%**	-	-	**50%**	**100%**
b) Class Diagram	-	**50%**	**50%**	-	**100%**
c) Vendor Role Functions	-	-	-	**100%**	**100%**
d) Customer Role Functions	-	**100%**	-	-	**100%**
e) Delivery Runner Role Functions	**100%**	-	-	-	**100%**
f) Administrator Role Functions	-	-	**100%**	-	**100%**
g) Manager Role Functions	**100%**	-	-	-	**100%**

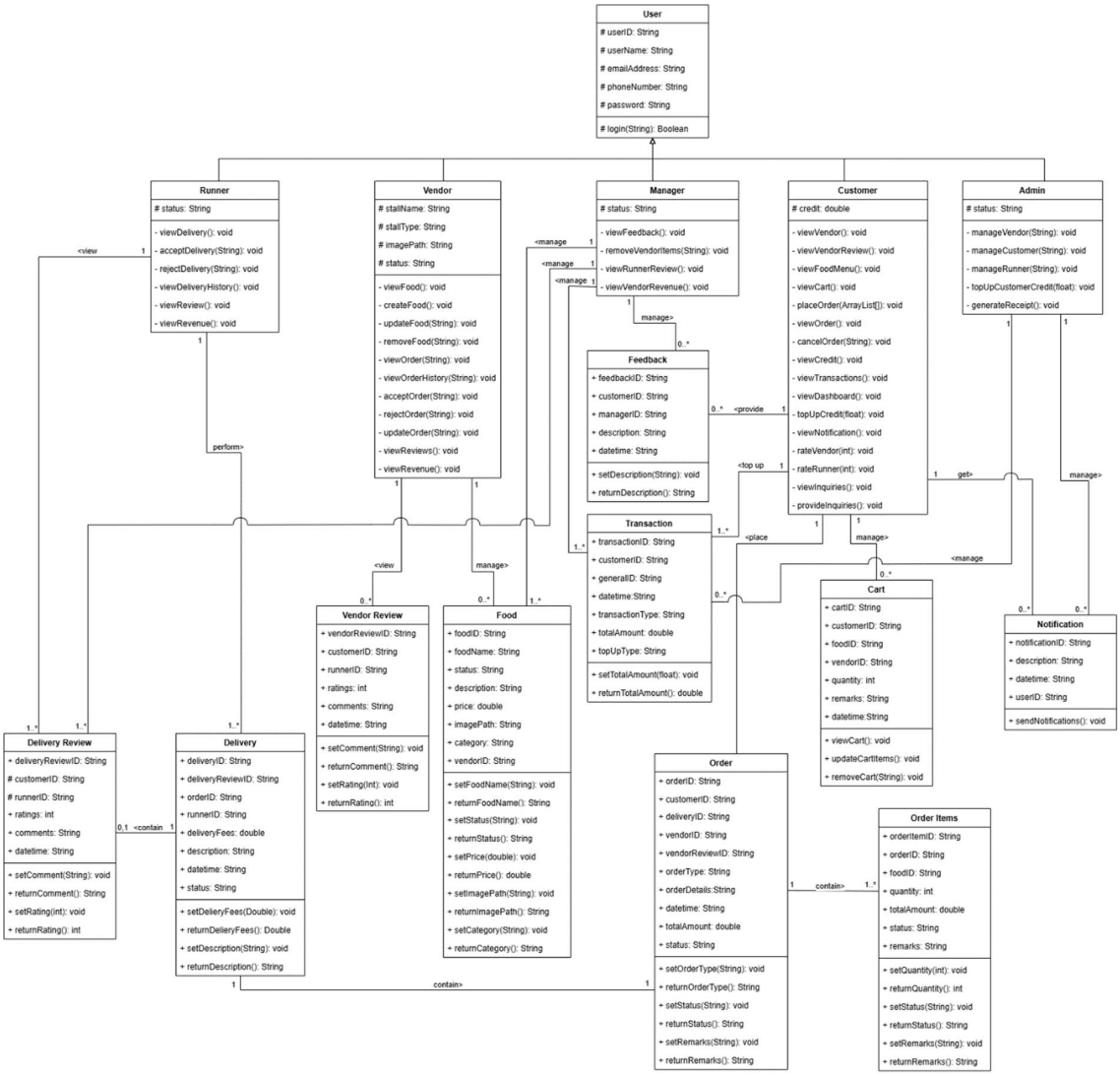
h) Screenshots of output of the program with appropriate explanations	**25%**	**25%**	**25%**	**25%**	**100%**
i) Description and justification of Object-oriented concepts incorporated into the solution	**25%**	**25%**	**25%**	**25%**	**100%**
j) Additional feature	**20%**	**40%**	**20%**	**20%**	**100%**

1.0 Design Solutions

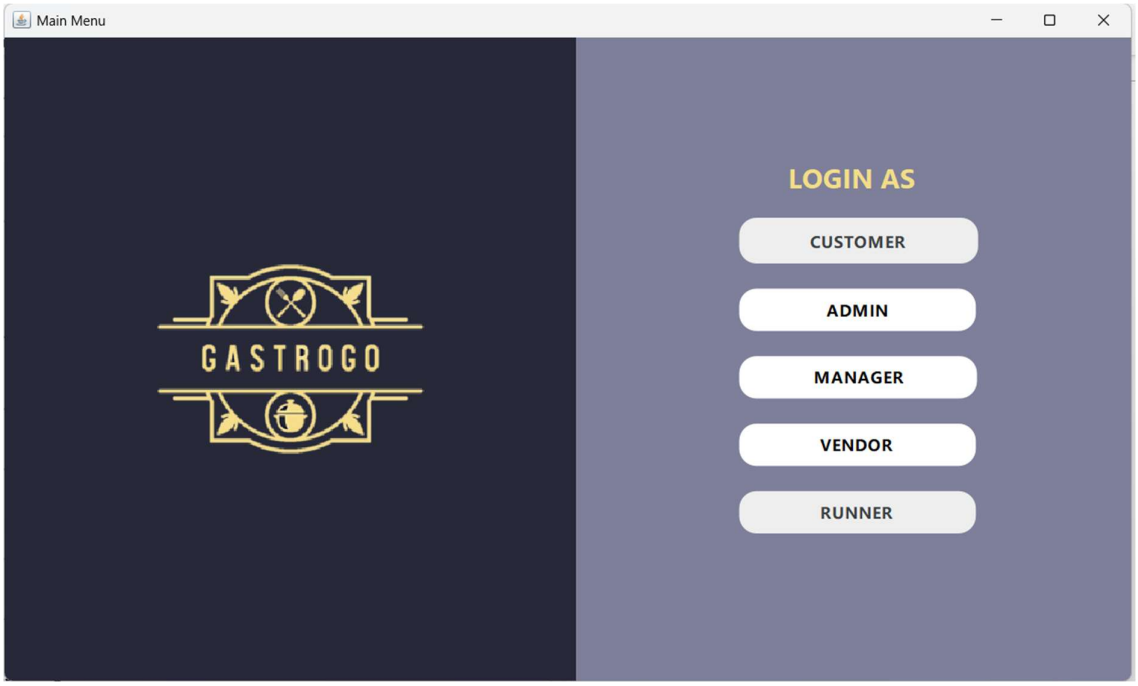
1.1 Use Case Diagram



1.2 Class Diagram



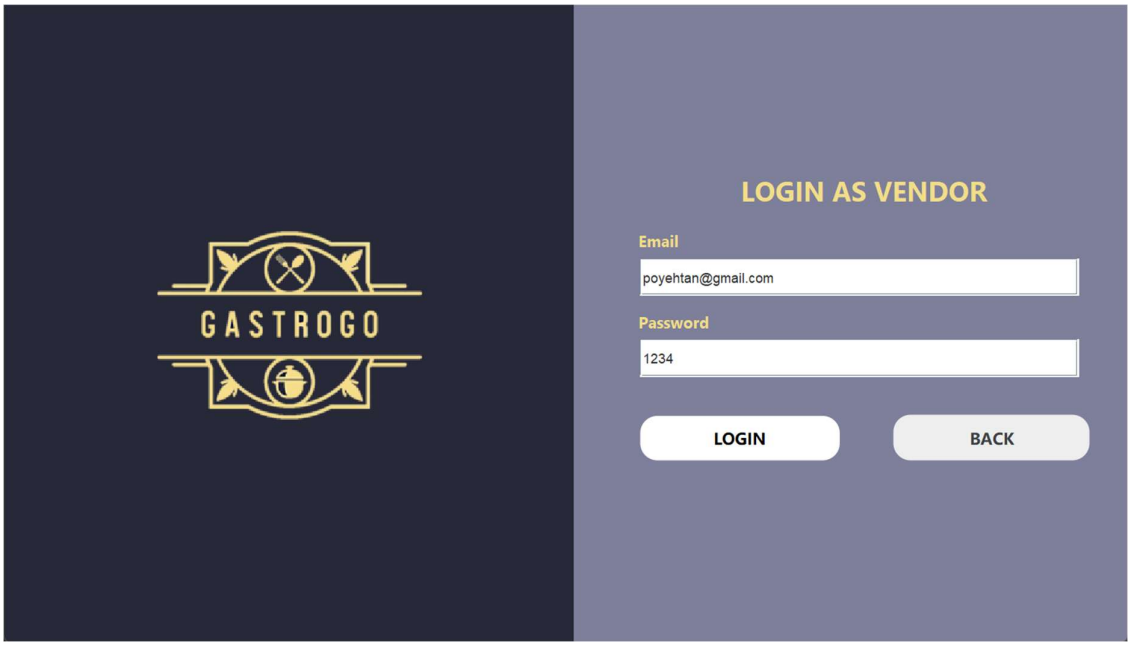
2.0 Program Outputs



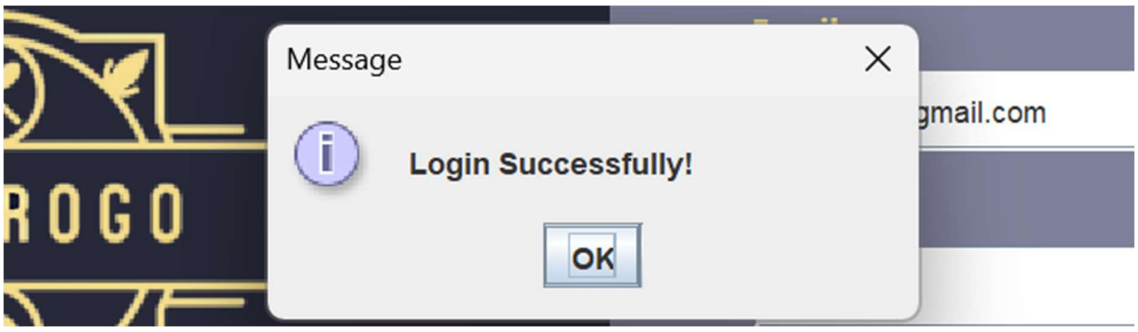
2.1 Vendor

2.1.1 Vendor Login Page

To login into the vendor page, user need to first click on the vendor button on the start page. Once the user being directed to the vendor login page, they will need to fill in the correct email and password in order to access the vendor page. After filling the email and password, click the login button. If the below are the vendor login page of the system.



If the credentials entered are correct, the user will be redirected to the vendor system where they can manage their account, view orders, update product listings and etc. If the credentials are incorrect, an error message will be displayed prompting the user to re-enter their email and password.

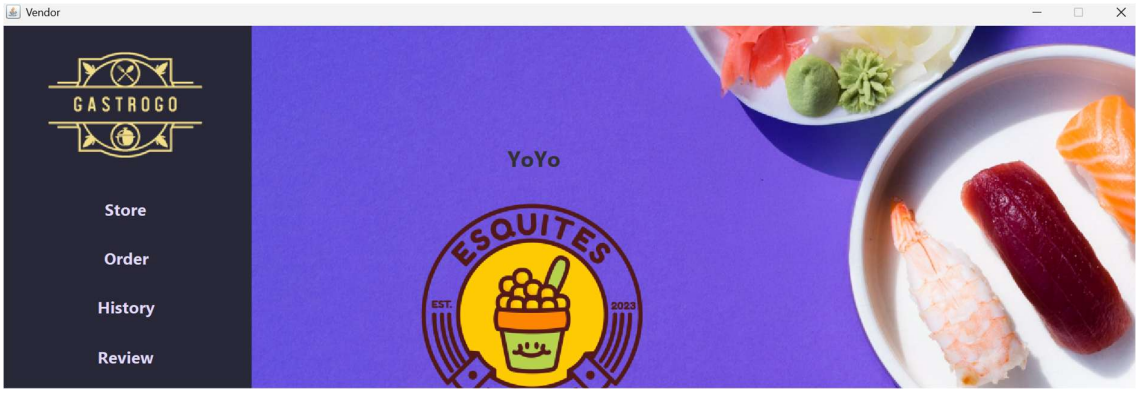


If the user decide to go back to the start page, user can just simply click the back button located besides the login button.

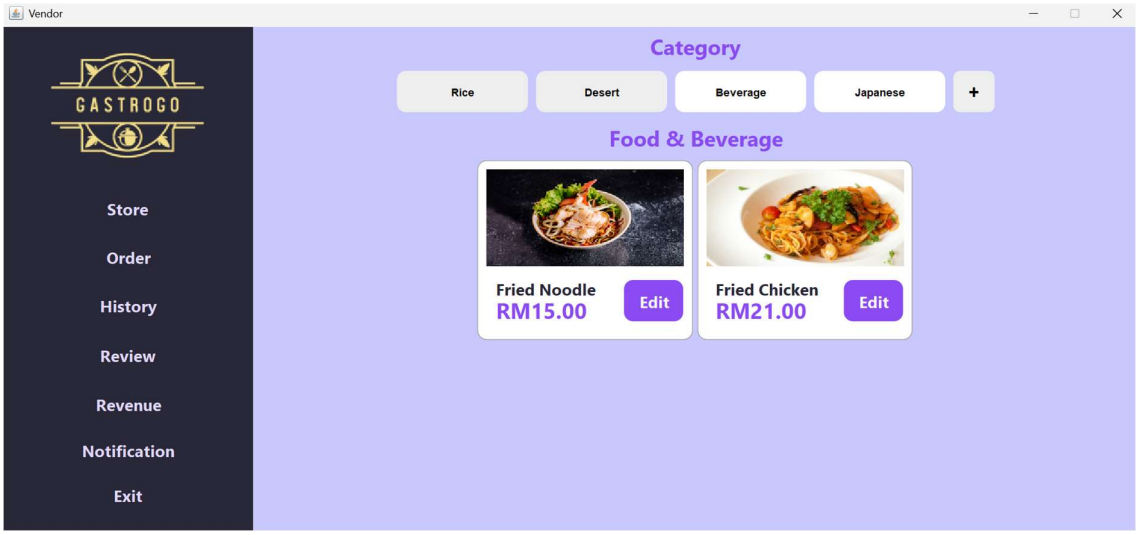
2.1.2 Vendor Home Page

This picture below show the vendor home page where this is the first page vendor will see after login. On this page, vendor can check whether they are logged into the correct account and confirm that the business phone number, digital logo and store owner details are accurate.

2.1.2 Vendor Store

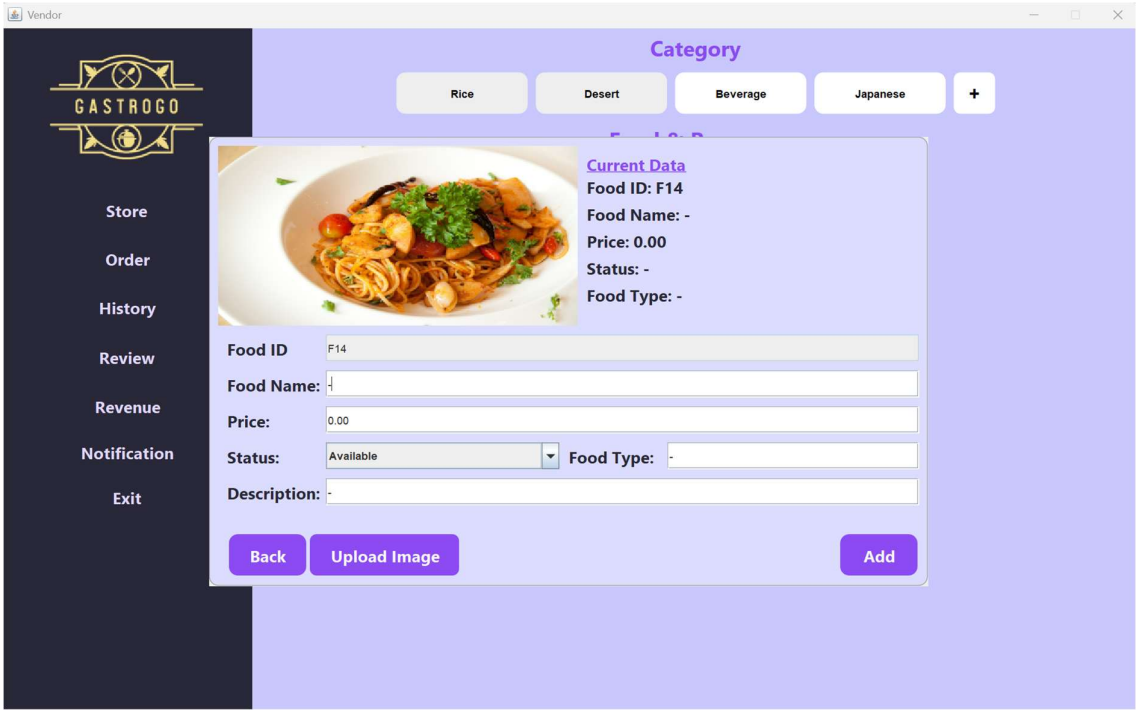


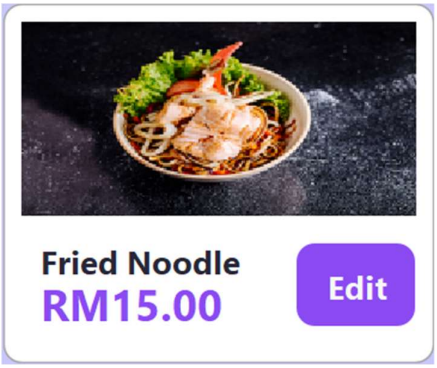
To edit the vendor product listing, vendor will need to click the button called 'Store' located on the left panel to access the page. Once success, the vendor will be directed to a page where all the food in the store will be displayed. Below are the picture of the vendor store page:



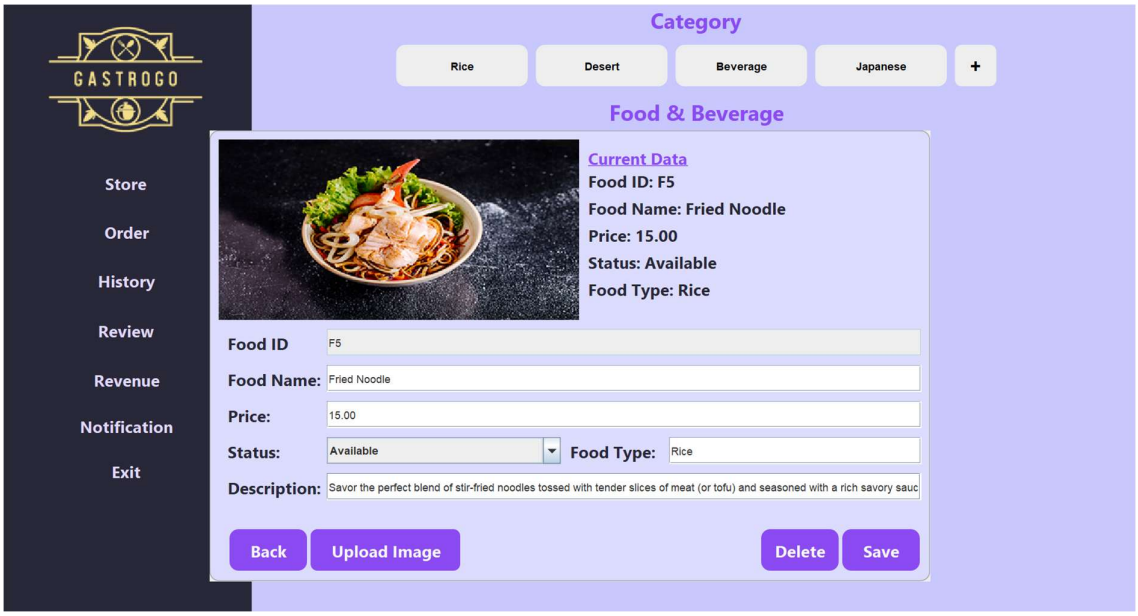
On this page, the system will automatically generate all the food categories available in the store. This allows vendors to easily find and navigate through their food listings. In the same row as the food category buttons, there is a plus '+' button, which is used to create a new food product for the store.

When the vendor clicks the plus button, a draggable pop-up window will appear. The vendor can fill in the food details such as name, price, description, image, category, etc. Once completed, the vendor can click the Add button located at the bottom right corner of the pop-up to save the food item to the store.

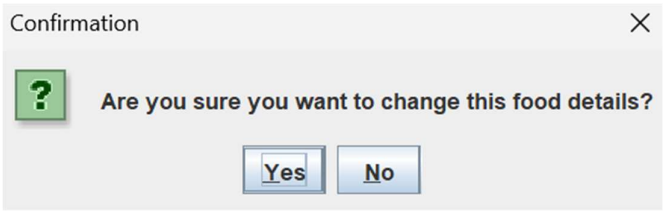




If vendor want to edit an existing food product, simply click the edit button in the bottom right of the food block. A pop-up windows will occur like the picture shown below. The pop-up will have all the current data of the food product. The vendor can directly change the data of the food product and click the save button on the bottom left corner.



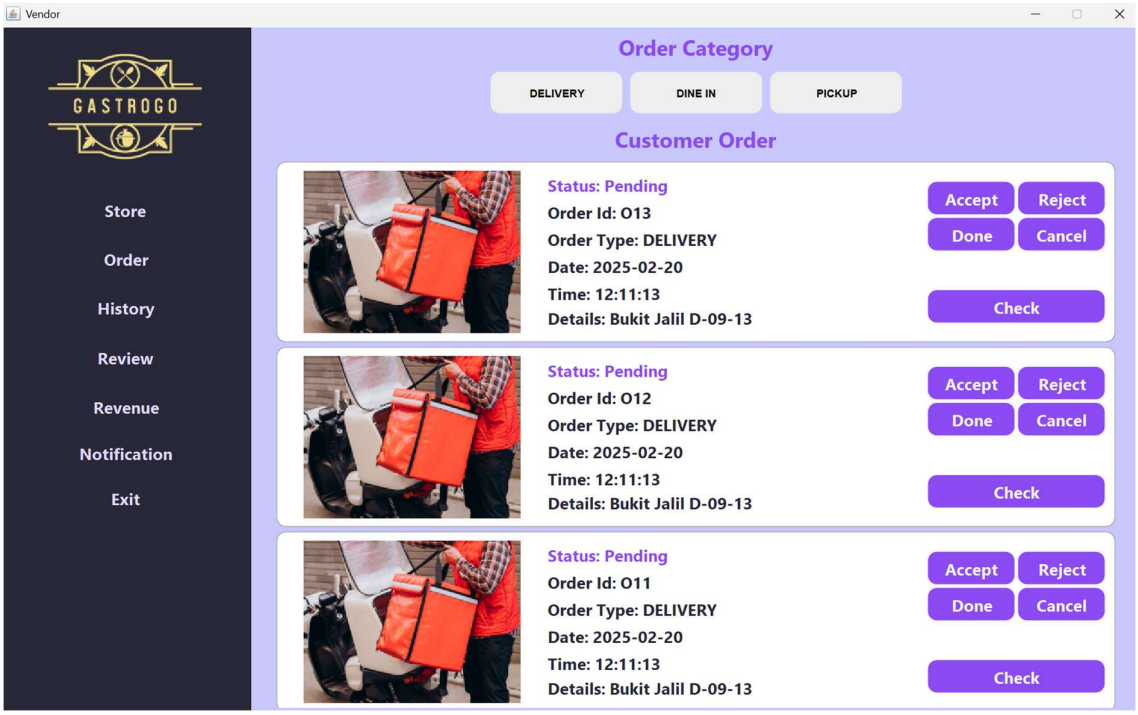
A confirmation panel will be shown to ensure that the vendor wants to save the changes. Once confirmed, the updated details will be applied to the food product and reflected in the store's listing. If the vendor chooses to no, no changes will be made, and the original information will remain intact.



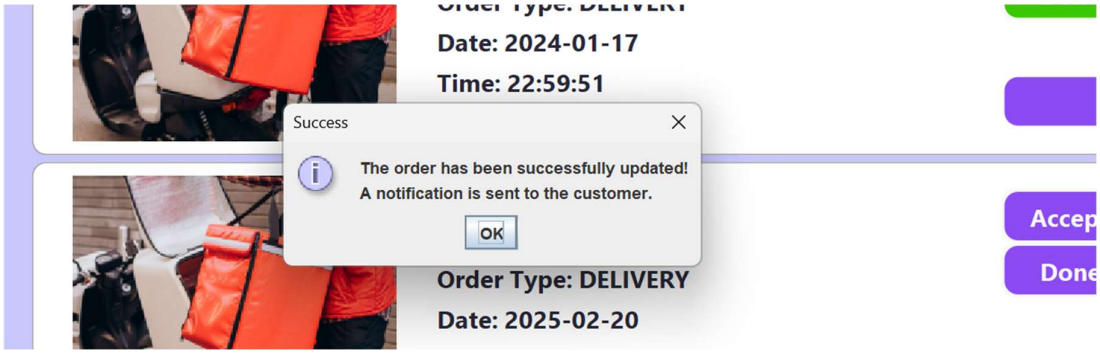
2.1.3 Vendor Order Page

To check orders, the vendor needs to click the Order button located on the left panel of the page. Once clicked, the vendor will be directed to the Order page, where they can browse orders using the order category buttons. There are three types of order categories: Delivery, Dine-In, and Pick-Up.

This page will only display orders with a 'Pending' status for the vendor to review and manage.

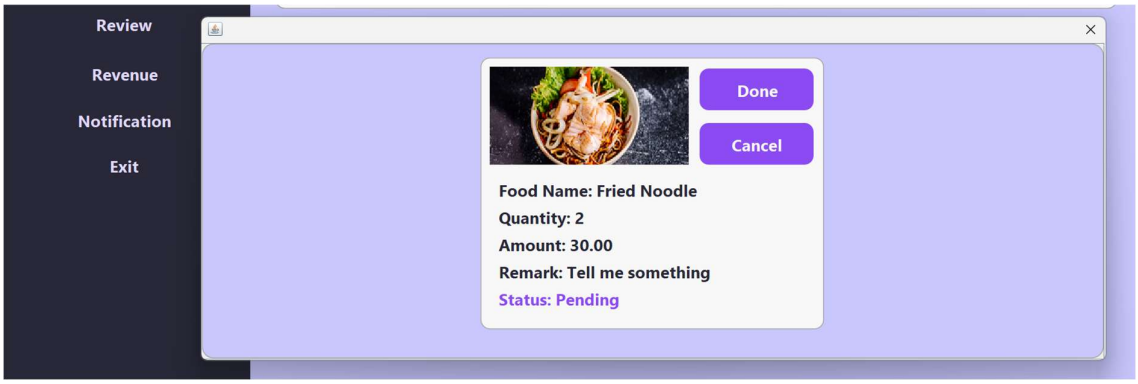


There is four buttons (Accept, Reject, Done, Cancel) for the vendor to update the status of the order. Once the vendor clicks these buttons, the order status will be updated and a notification will be sent to the customer.

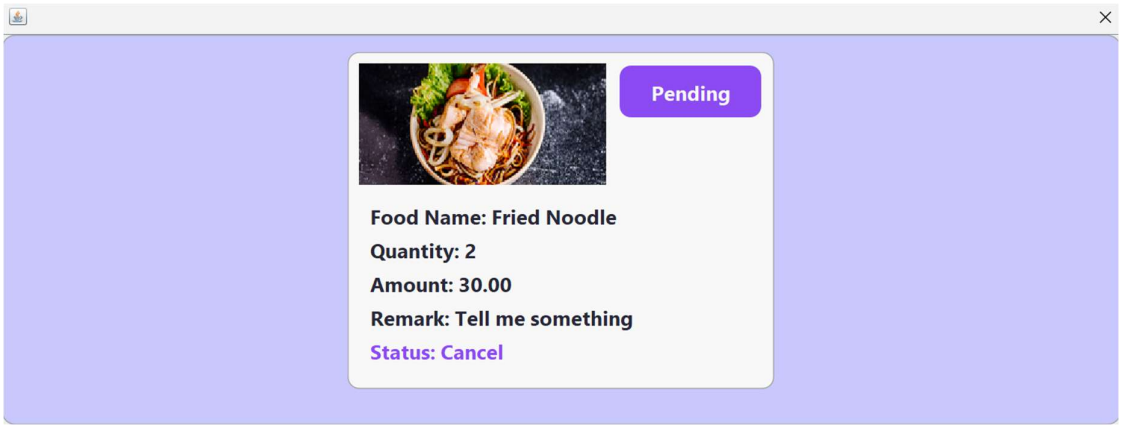


On the order block, the vendor can also click the check button to check what's the customer order. A draggable and resizable panel container with the food block details will be shown

along with the food details and quantity so that the vendors can check the order easier. The vendors can also click the done or cancel button to cancel certain food or update the food status to done.

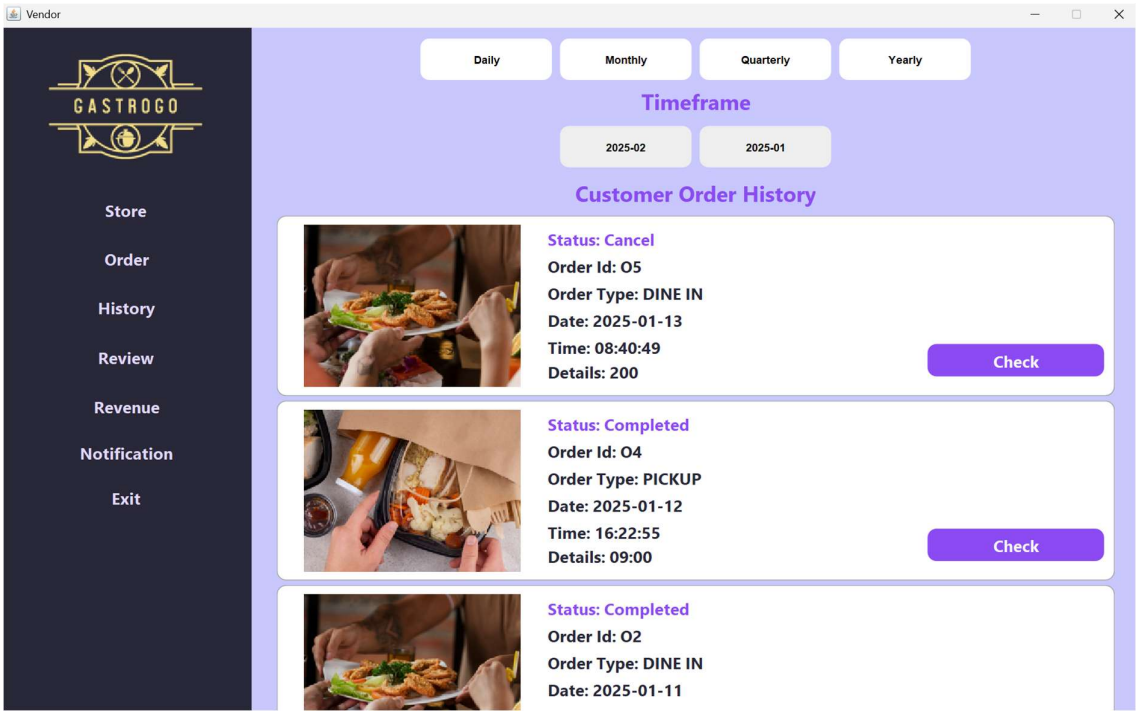


If the vendor happen to mis-clicked the done or cancel button. They can always revert the order status by clicking the 'Pending' button to change the status.

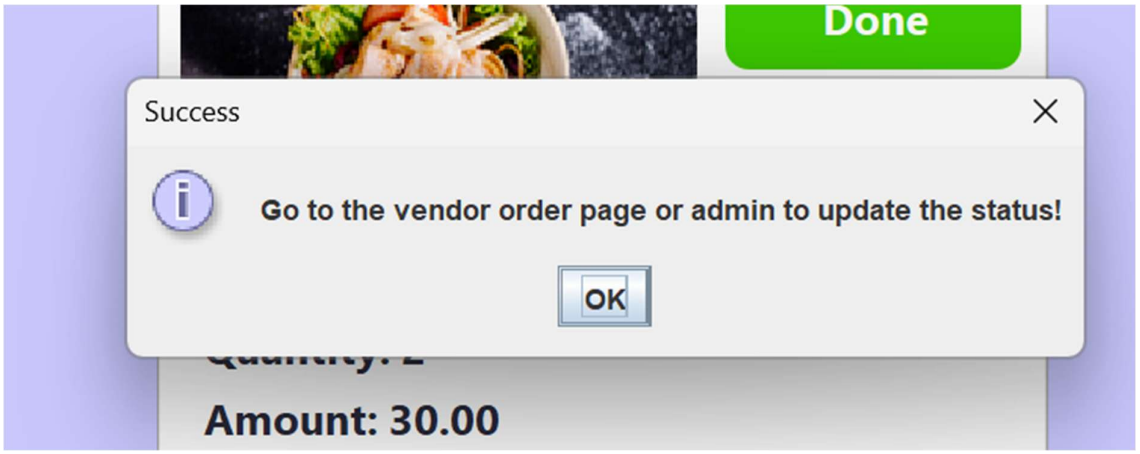


2.1.4 Vendor Order History Page

To view the order history, click the history button on the left panel. The system will then direct the user to the order history page. On this page, it works similar to the order page but on this page, the vendors are not able to change the order status. The order displayed here will be the status that is not in pending.



The user can also check the order items in the order but they are not allowed to edit the order items status in this page.



2.1.5 Vendor Review Page

To check the customers' reviews, vendor simply click the review button on the left panel. Then, the vendor will be able to check all the customer review. The vendor can view detailed feedback for each order, including the customer's rating and written comments.

2.1.6 Vendor Notification Page

For the notifications page, vendor can also receive notifications from others like admin, manager or the system generated message such as order notifications like the figure shown below. Everytime when customer place an order, vendors will receive a notification alert. System will display out the notifications block detailed with date and content.

2.1.7 Vendor Revenue Page

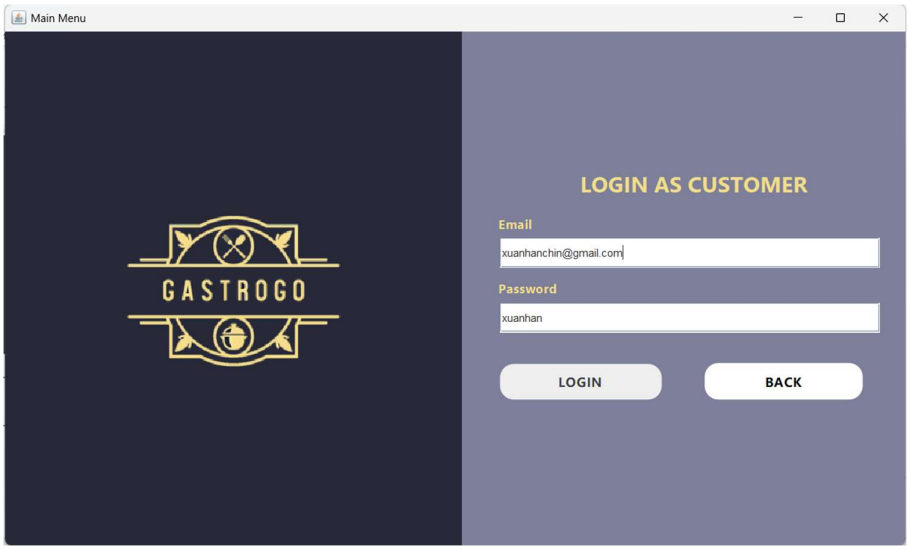
For the revenue page, vendors able to access this page by clicking the revenue button on the side bar.

From here, vendors can directly view their total revenue, which is prominently displayed at the top in Malaysian Ringgit (RM). Alongside this, the page shows the total number of orders completed. Additionally, vendors can view their average rating, which reflects customer satisfaction. A monthly revenue graph is also available, offering a visual representation of revenue trends for the selected year.

Vendors can either filter the revenue result by using the drop-down menu to select the month or year to analyze revenue trends over time, allowing them to identify patterns and adjust their strategies accordingly.

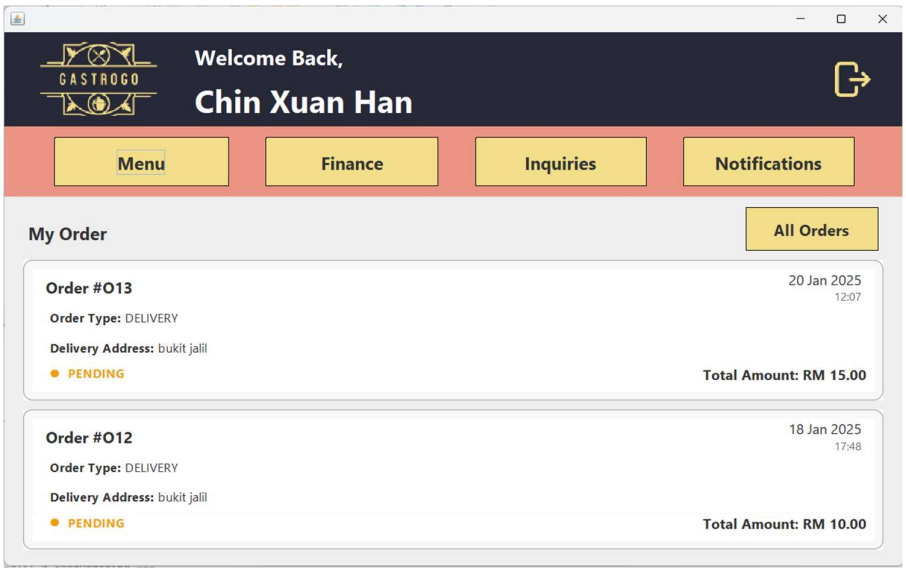
2.2 Customer

2.2.1 Customer Login



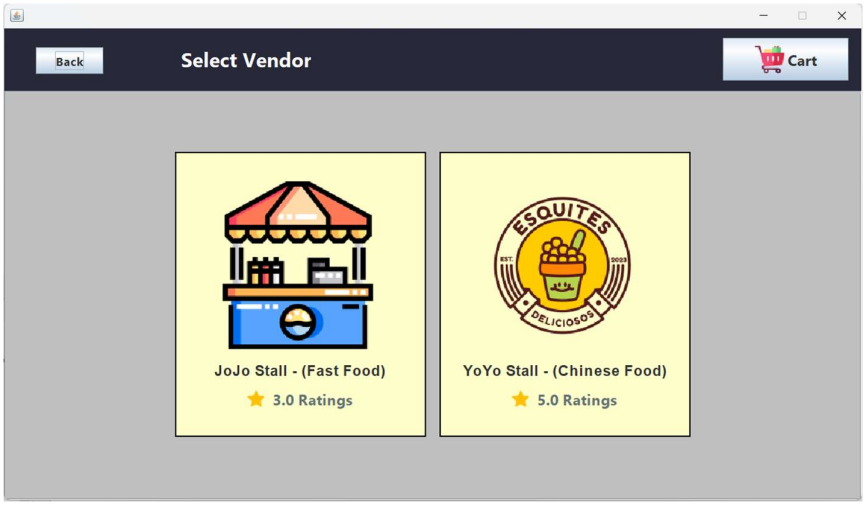
This is the login interface that provides email and password text field for the users to sign in their own account by clicking on the login button. While the back button allows the user to back to the user role selection interface.

2.2.3 Customer Homepage



This is the main page of customer that shows the user's name, current orders as well as order again suggestions, navigation button such as menu, finance, inquiries, notifications, orders button, and logout button.

2.2.4 Vendor Selection

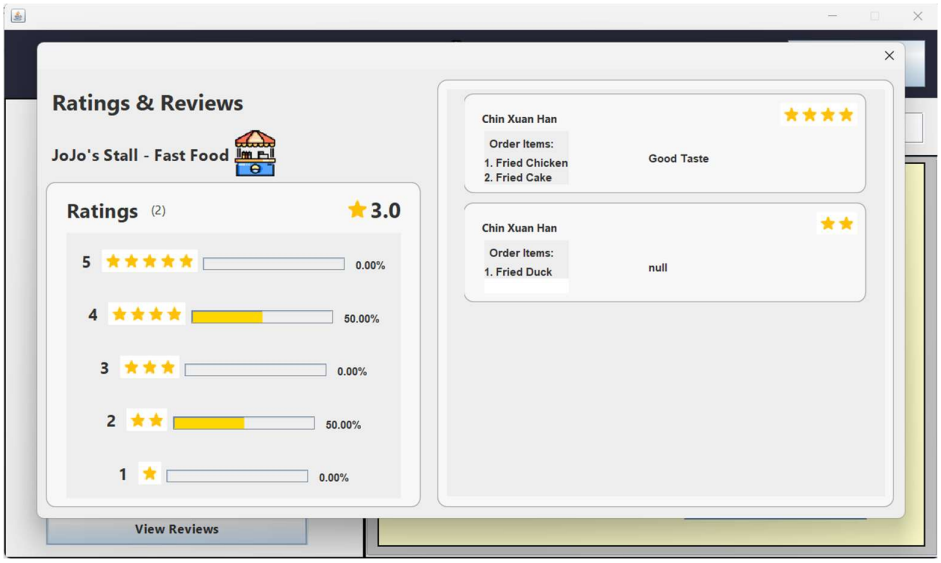


This is the vendor selection interface after clicking on the menu button. The customers can view every vendor in the food court. Additionally, there is a button to navigate to the cart interface and a back button to navigate back to the main page.

2.2.5 Vendor Stall Details

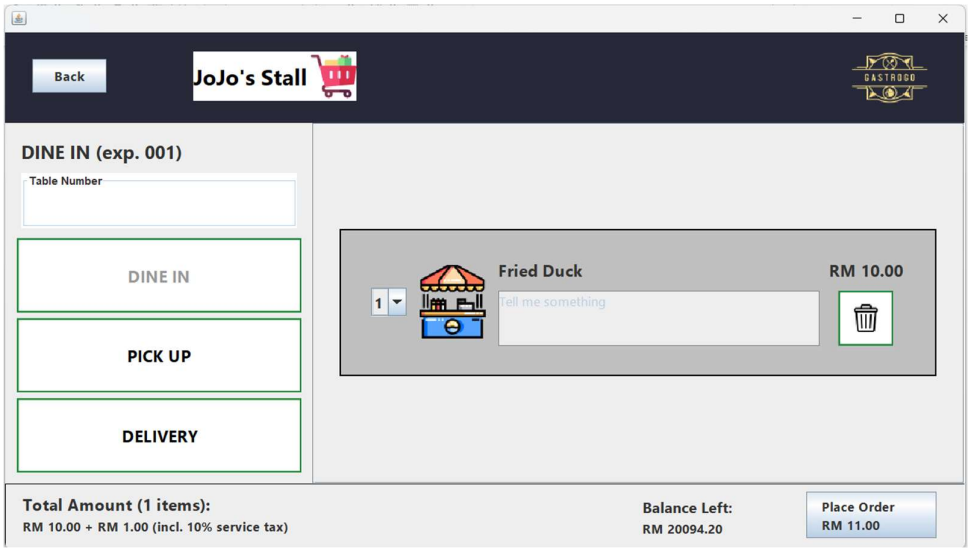
This is the vendor stall information that shows the vendor's information after clicking on the vendor panel. The customer can search for their food by inputting the query in the text field and the system will filter out the food. The food details included name, type, price, description, quantity selection, and the add cart button. By selecting the quantity and clicking on the added cart button, the customer can submit their remarks, and the food will add into the cart. Furthermore, the customer can click on the view review button to view the review and ratings that are given by the other customers.

2.2.6 Vendor Review



This is the ratings and review interface for the vendor. It shows the overall rating and distribution of each rating, and all the customer reviews for each order.

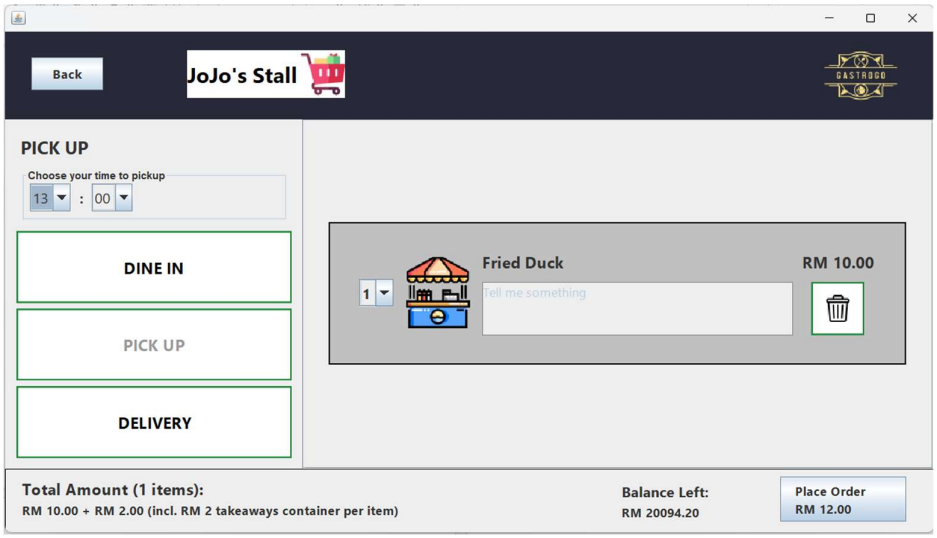
2.2.7 View Cart (Dine In)



This is the cart interface that shows the food which is added by the customer.

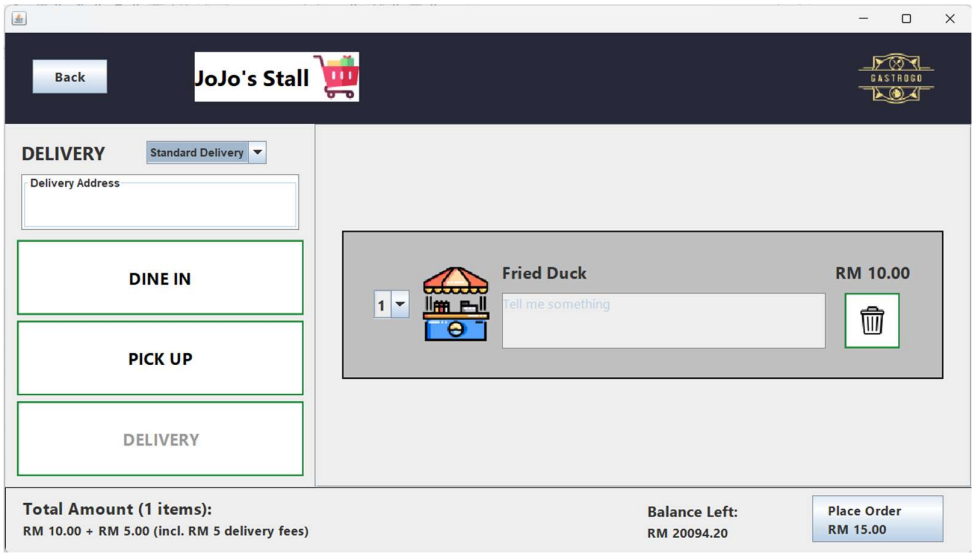
For dine in option, The table number is required to fill up before placing order. 10% of service tax is included into the total amount.

2.2.8 View Cart (Pick Up)



For the pickup option, the pickup time must be between the working hours. The additional charge for takeaways is RM 2 for each item.

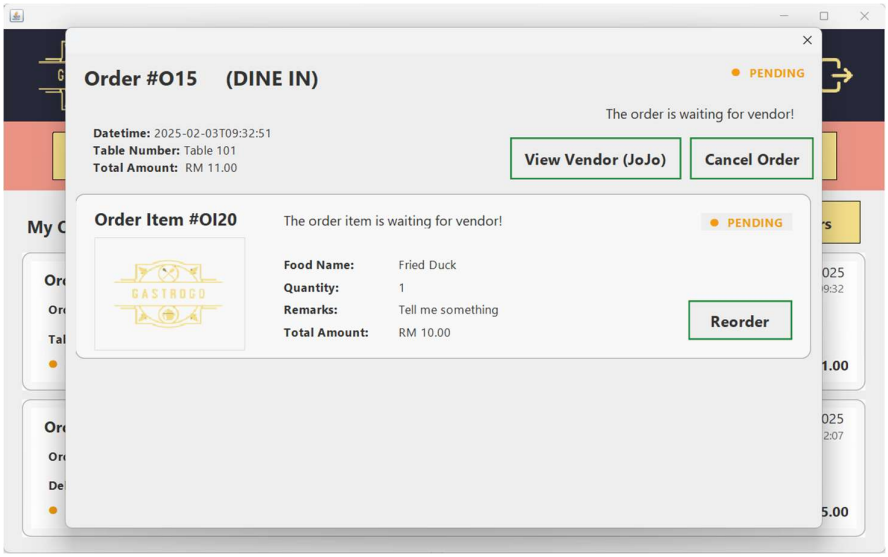
2.2.9 View Cart (Delivery)



For delivery option, the delivery address is required to be filled up in Bukit Jalil area. There is an additional option for standard delivery and fast delivery which charges different fares.

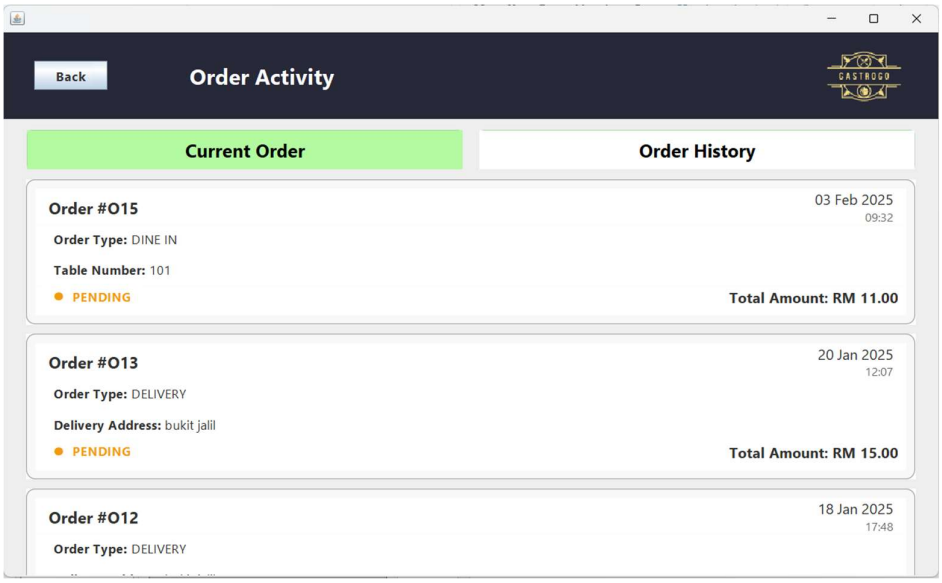
The customer can modify the quantity of food added and remove the food. Additionally, the balance left and the total amount to pay will be calculated for the customers if the balance is sufficient or not. In concisely, the cart interface will show different total amounts for each of the options.

2.2.10 View Order Details

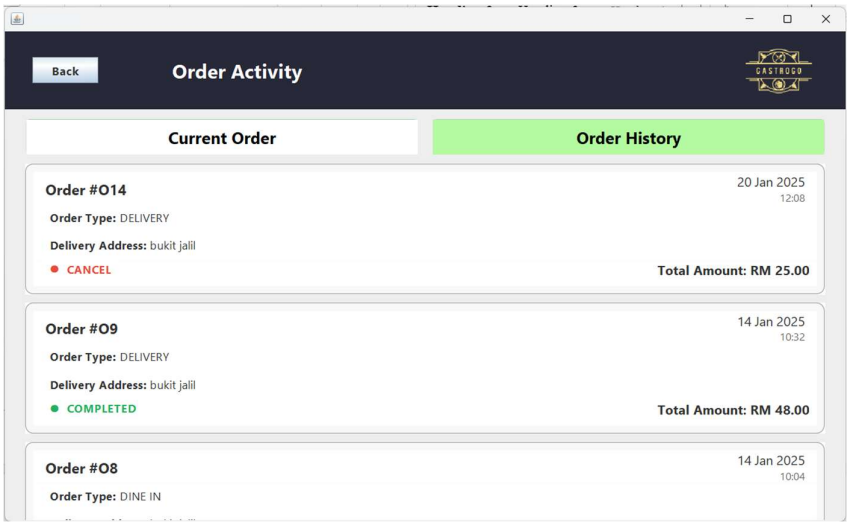


The customer can view the order details after they place order. Under this page, they can view vendor stall, cancel the order when the order status is pending, and reorder the order items.

2.2.11 View All Order (Ongoing)

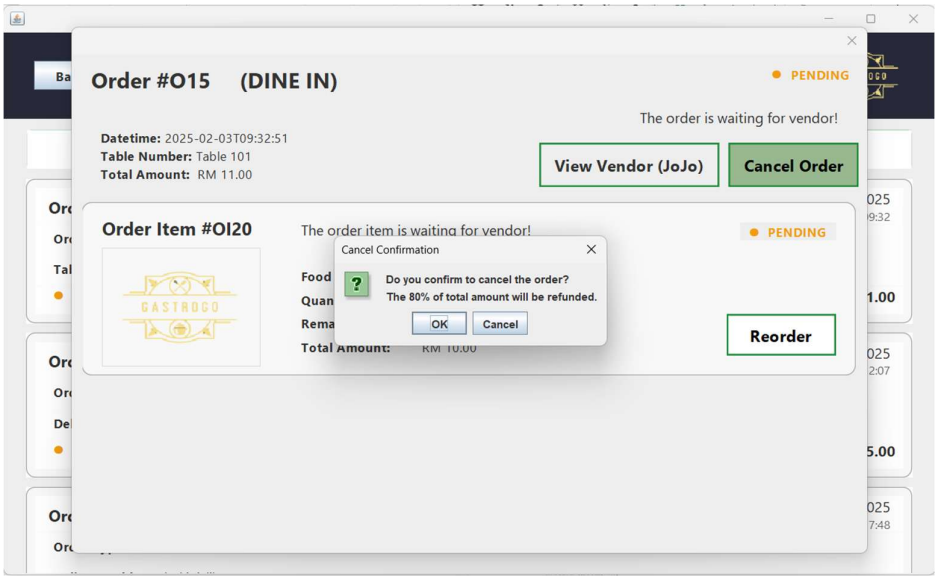


2.2.12 View All Order (History)



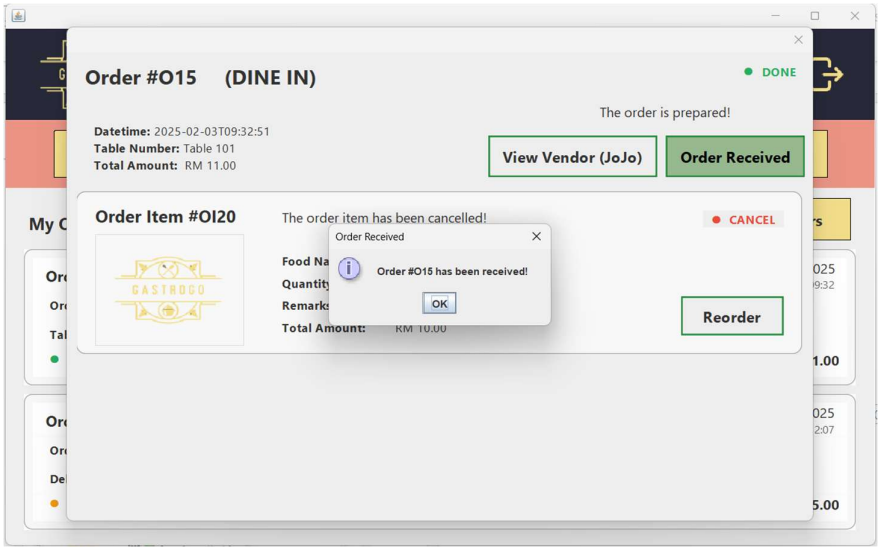
This is the page of all orders that are divided into current order and order history. All the order status that is under pending or done will categorize into current order while for the order history, the order status will be cancelled and completed. By clicking on the order panel, all the order information will display as the image on 2.2.10 view order details.

2.2.13 Cancel Order



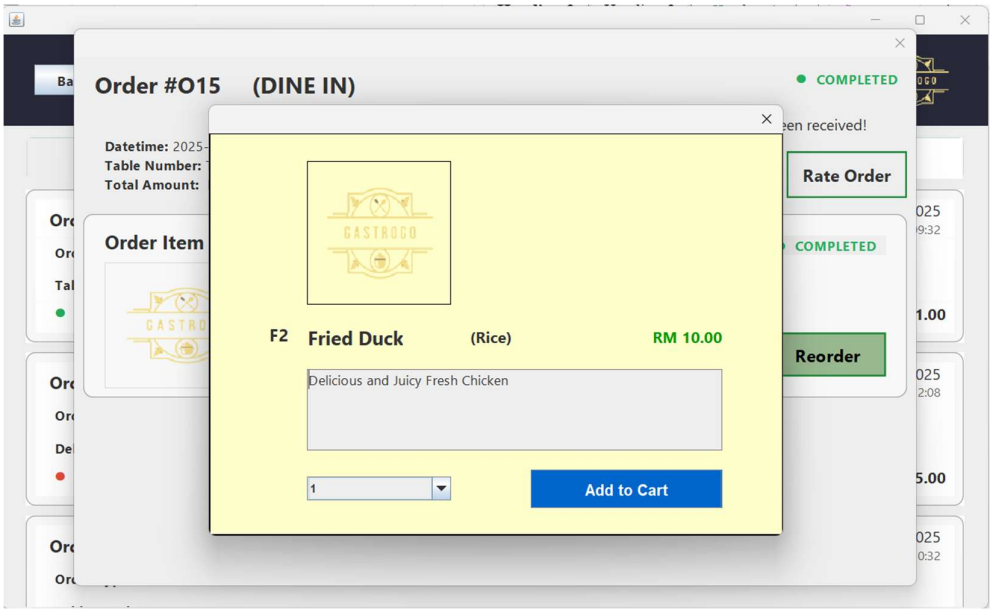
This is the page of view order details that the customers click on the cancel order button and pop up a confirmation message to validate the intention. If the customer confirms to cancel the order, the order status will be changed into cancel and the 80% of total amount will be refunded to the customer's balance.

2.2.14 Receive Order



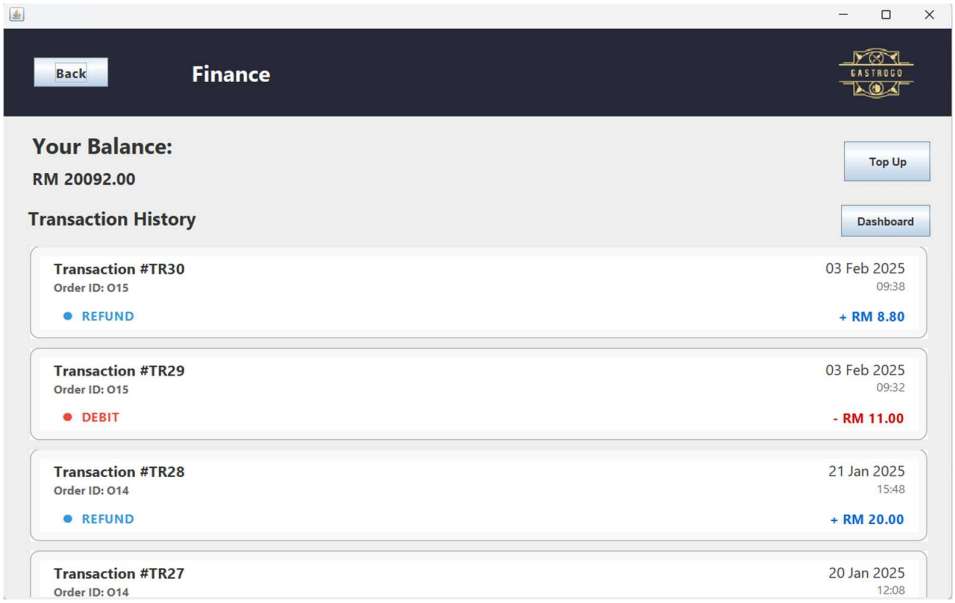
This is the view order details page that validates whether the order is received by the customer when the order status is under done status.

2.2.15 Reorder Food



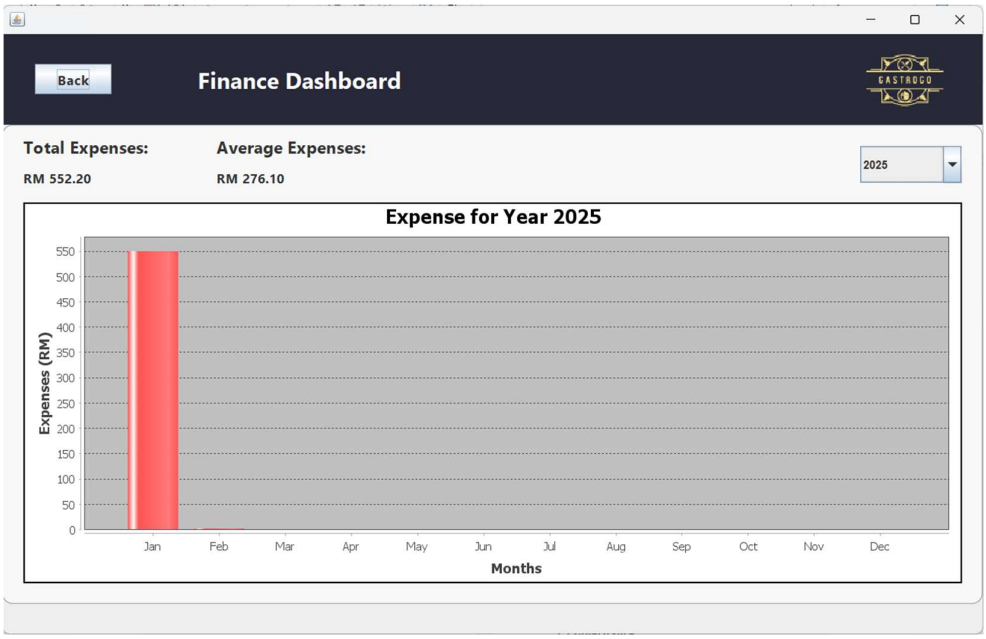
The customer can reorder the food from the view order details page by clicking on order item panel's button. The food details' window will be pop up and the customer can add the food into cart.

2.2.16 View Finance



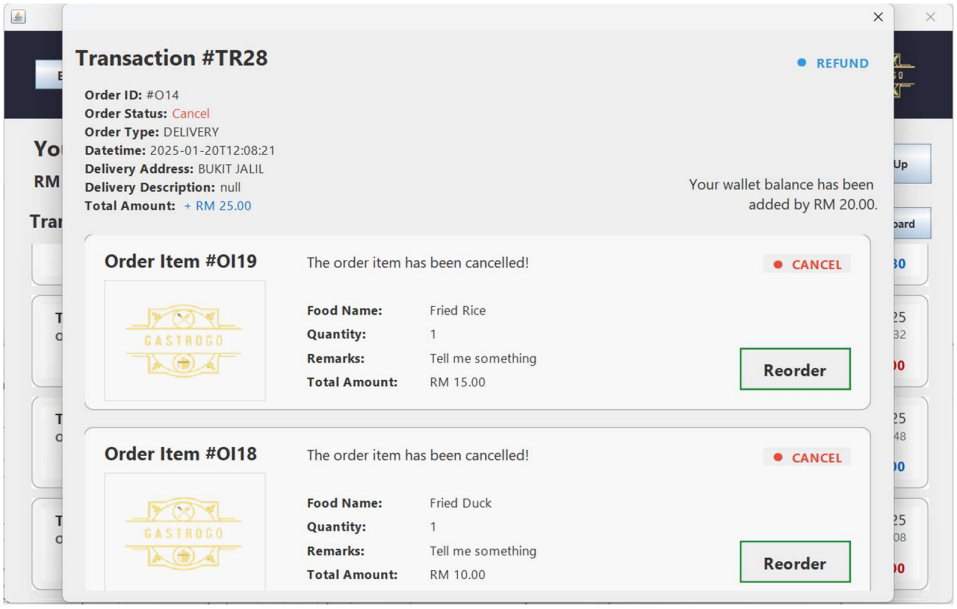
This is the finance page where the customer can view their current balance, transaction history, dashboard and online top up.

2.2.17 View Finance Dashboard



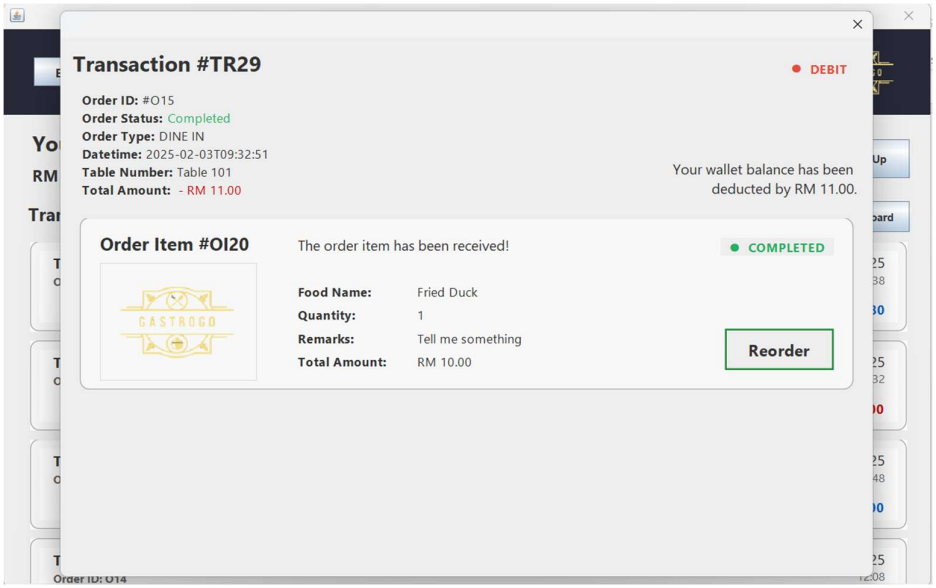
This is the customer finance dashboard that shows the bar chart of expenses and the total and average expenses for each year. Additionally, the average expense is calculated based on the current month in a certain year.

2.2.18 View Transaction (Refund)



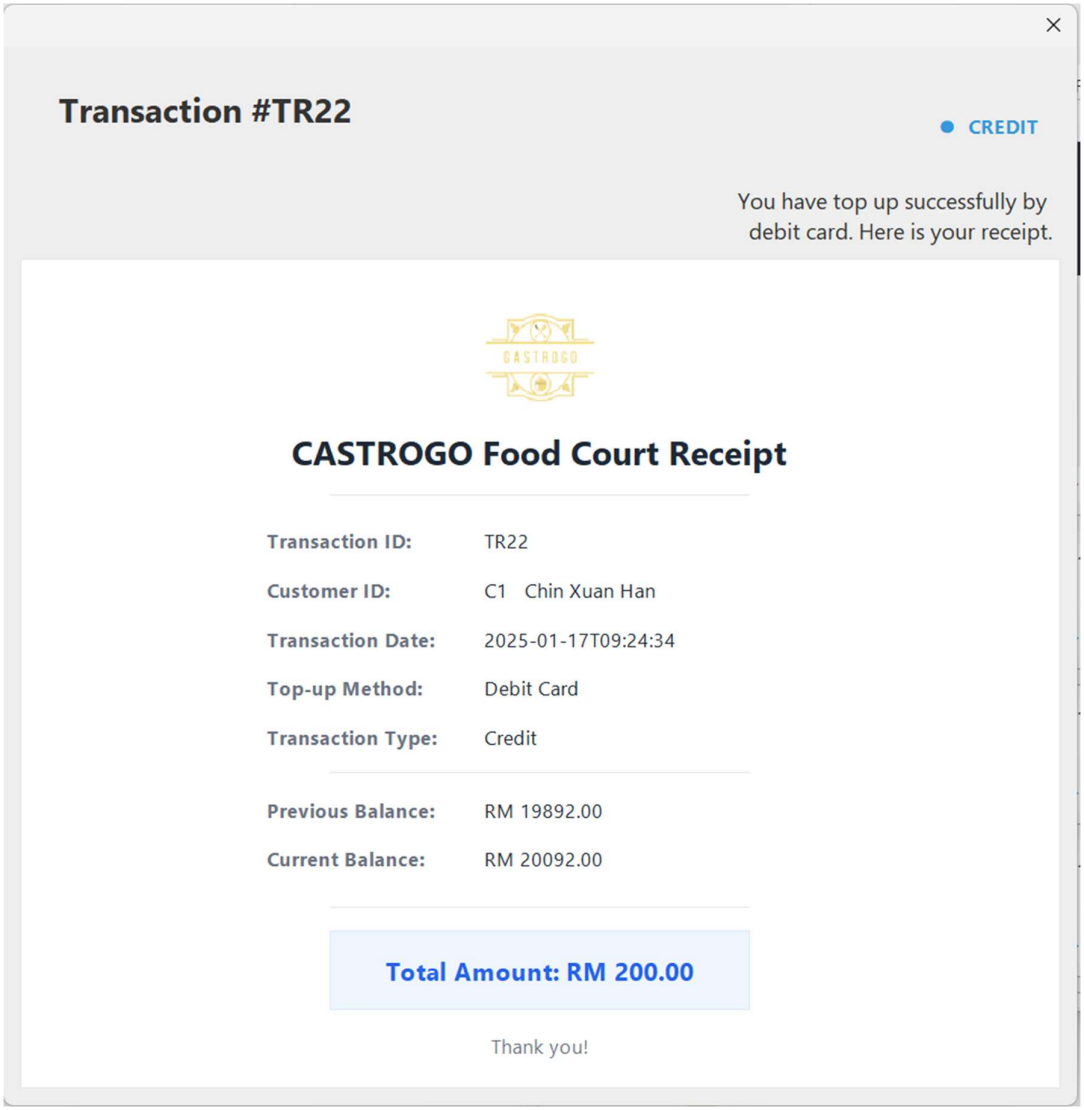
After the customer clicks on the transaction panel, the transaction's details window will be displayed. This is the transaction's details which is refund status that showed the cancellation of order.

2.2.19 View Transaction (Debit)



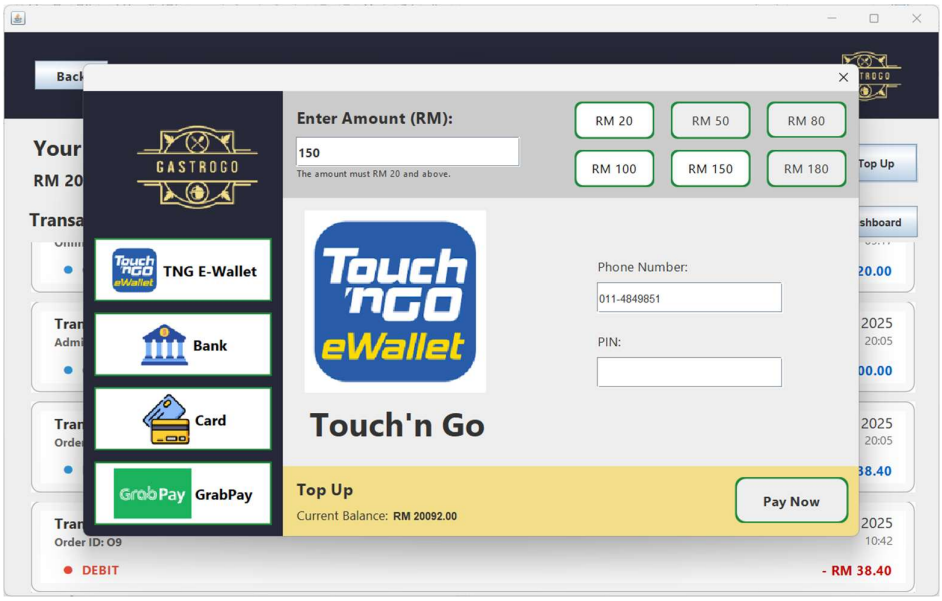
This is the transaction's details which is debit status when the customer place order.

2.2.20 View Transaction (Credit)



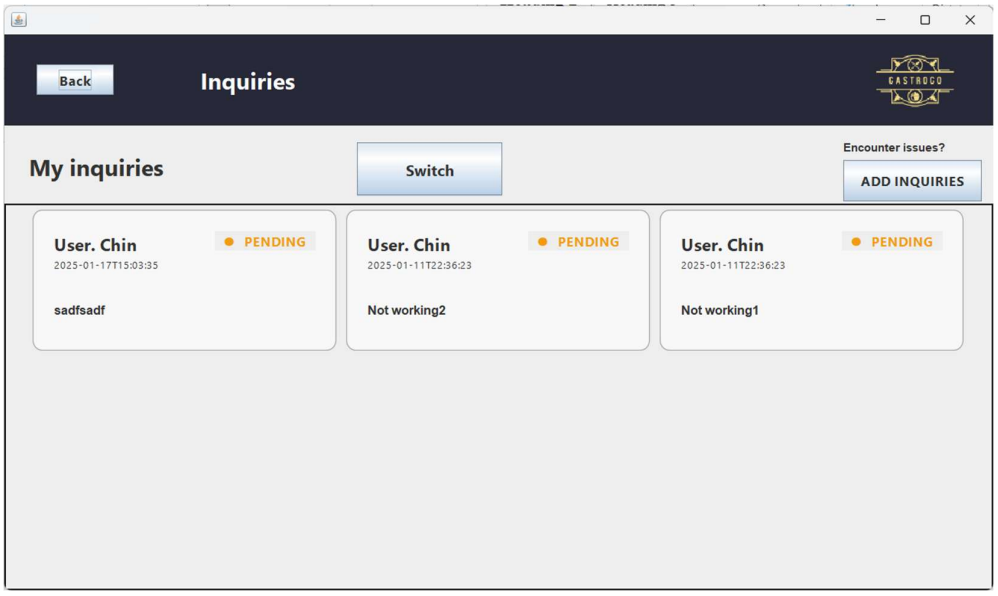
This is the transaction detail which is credit status that shows the receipt with all the payment information for the customer after they top up their balance.

2.2.21 Online Top Up



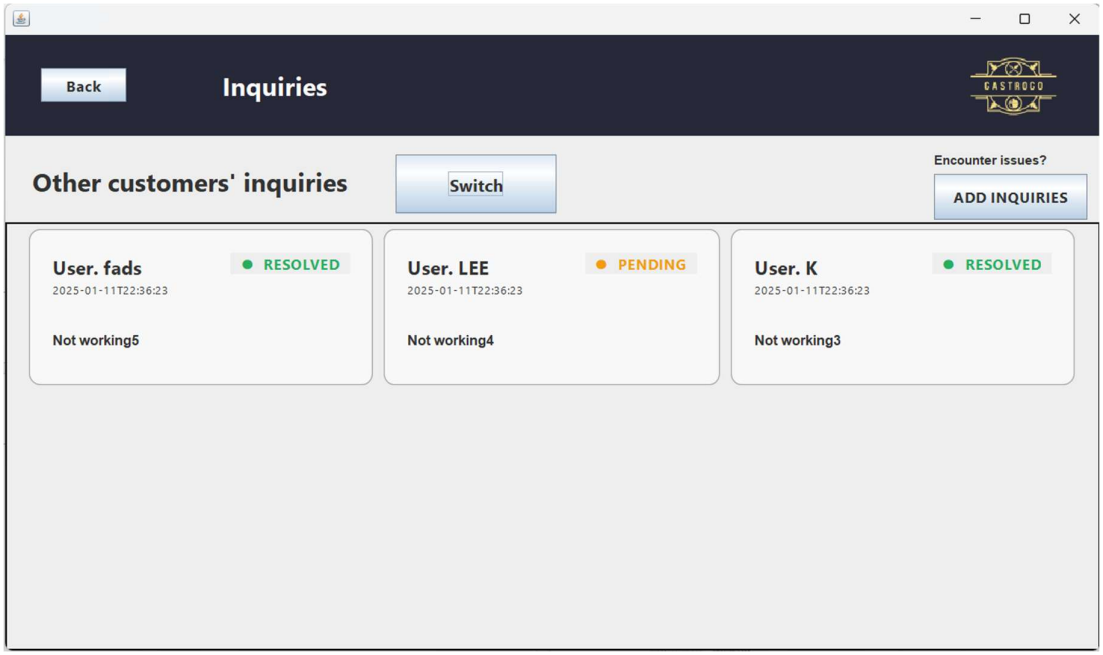
This is the online top up interface that the customer can top up their balance by choosing the amount via e-wallet, bank and card.

2.2.22 View Inquiries (My Inquiries)



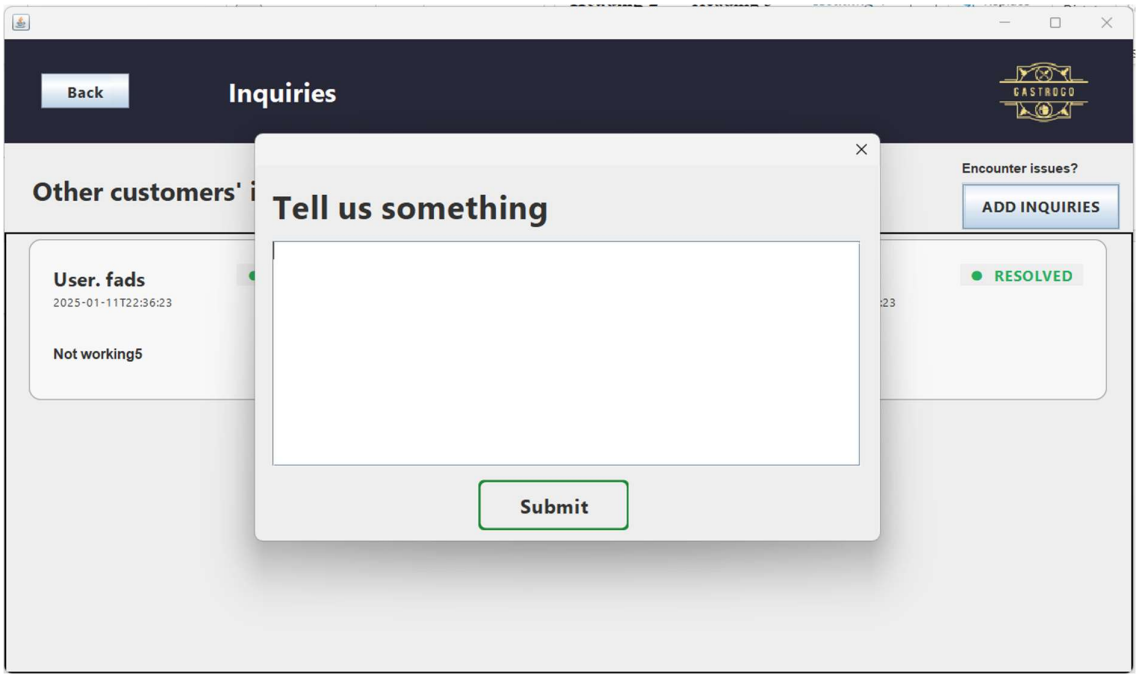
This is the inquires interface that can view the customer's previous and current inquiries.

2.2.23 View Inquiries (Other Inquiries)



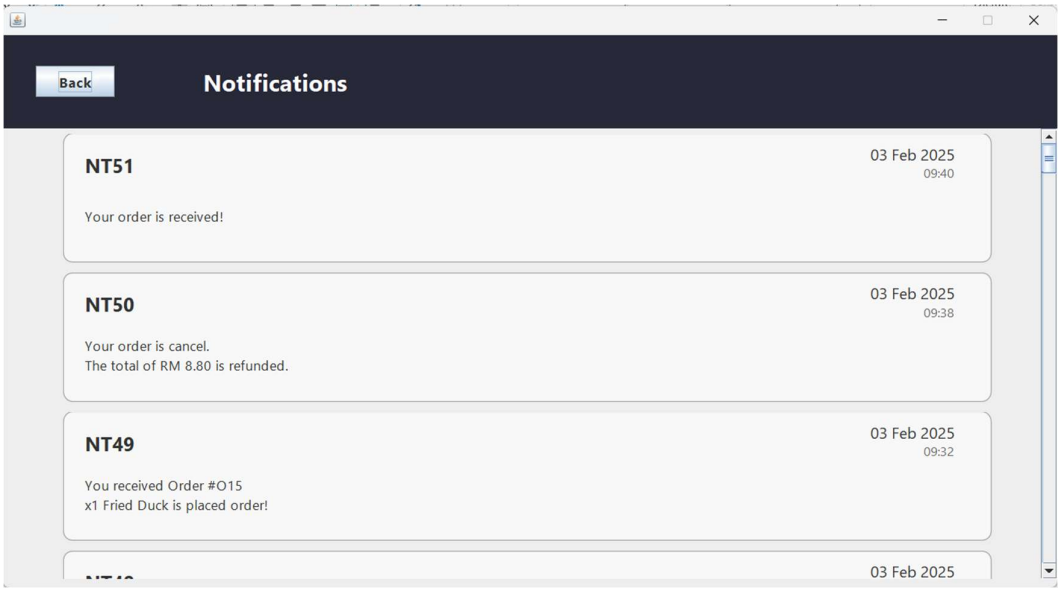
This is the inquires interface that can view the other customers' previous and current inquiries.

2.2.23 Submit feedback or complaints



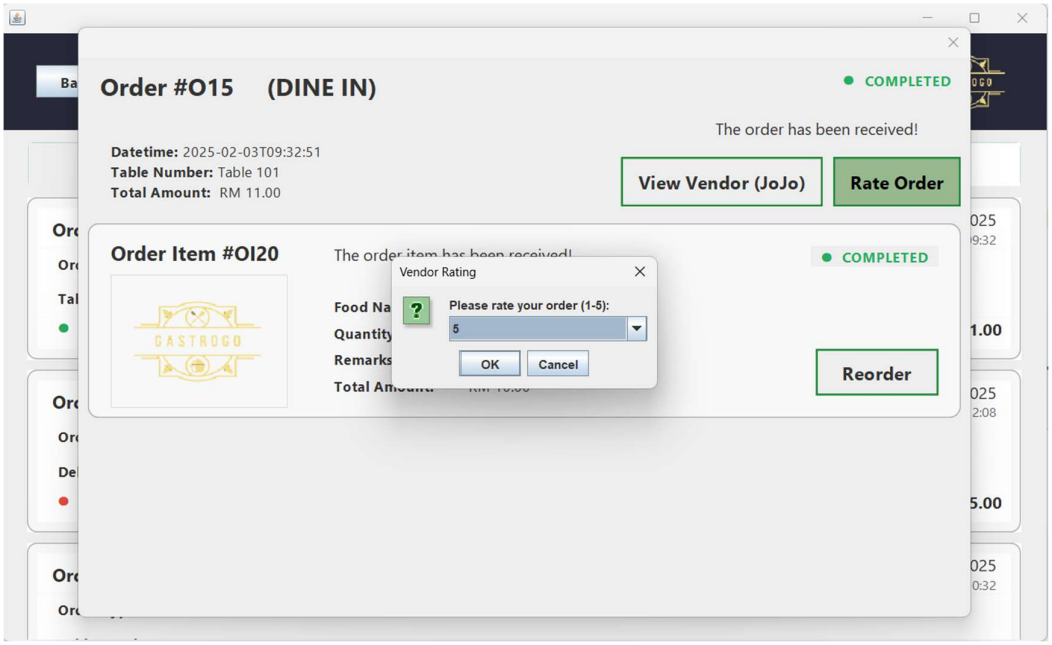
This is the pop-up window that allowed the customer to submit their feedback or complaints.

2.2.24 View Notifications

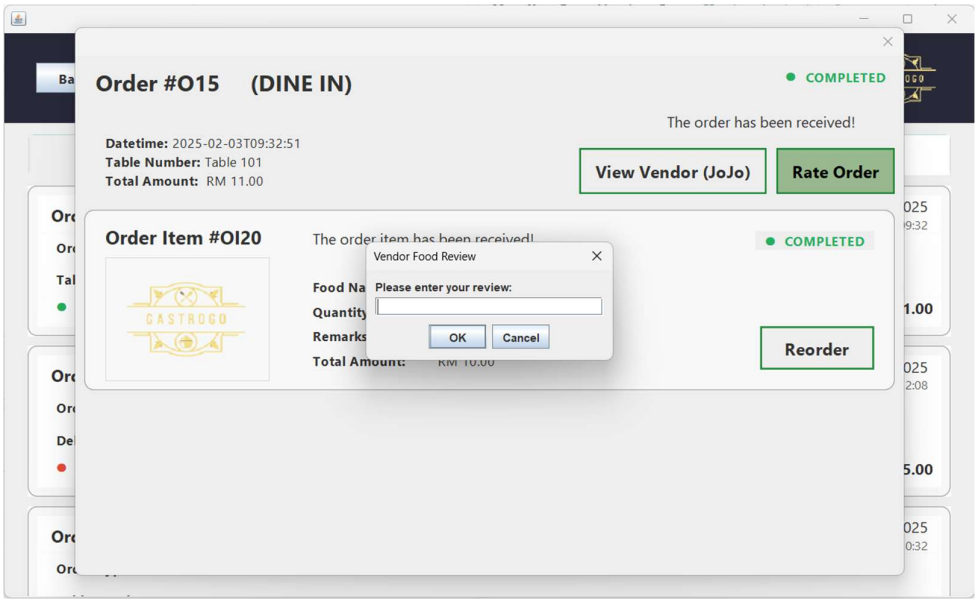


This is the notification interface that shows all the notification for the customer's activity in the system.

2.2.25 Rate Order

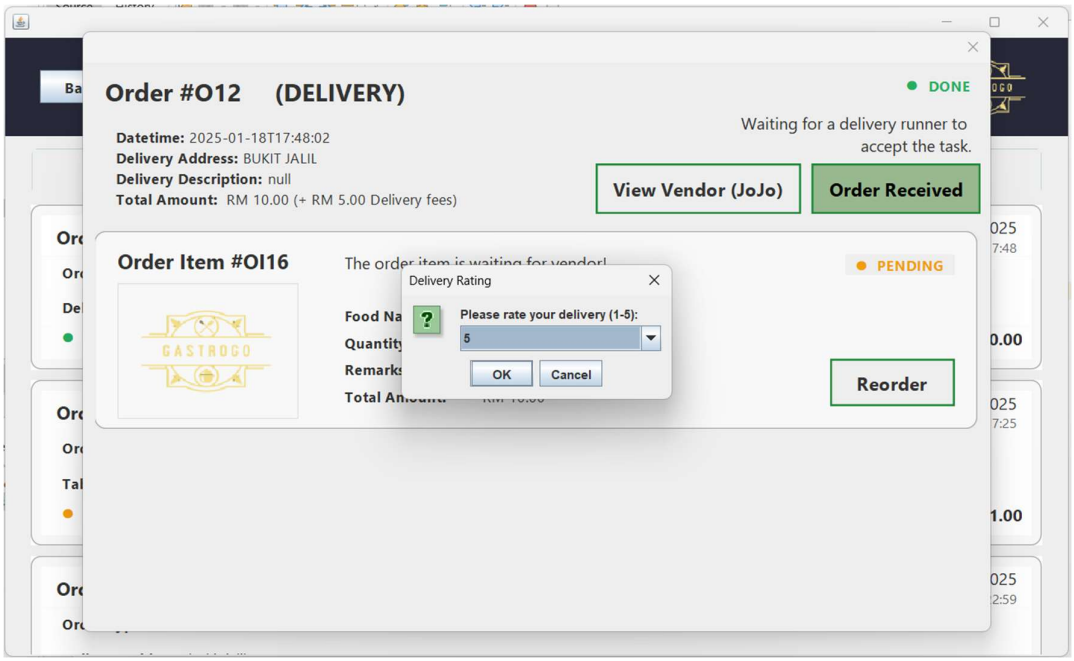


2.2.26 Submit Vendor Review

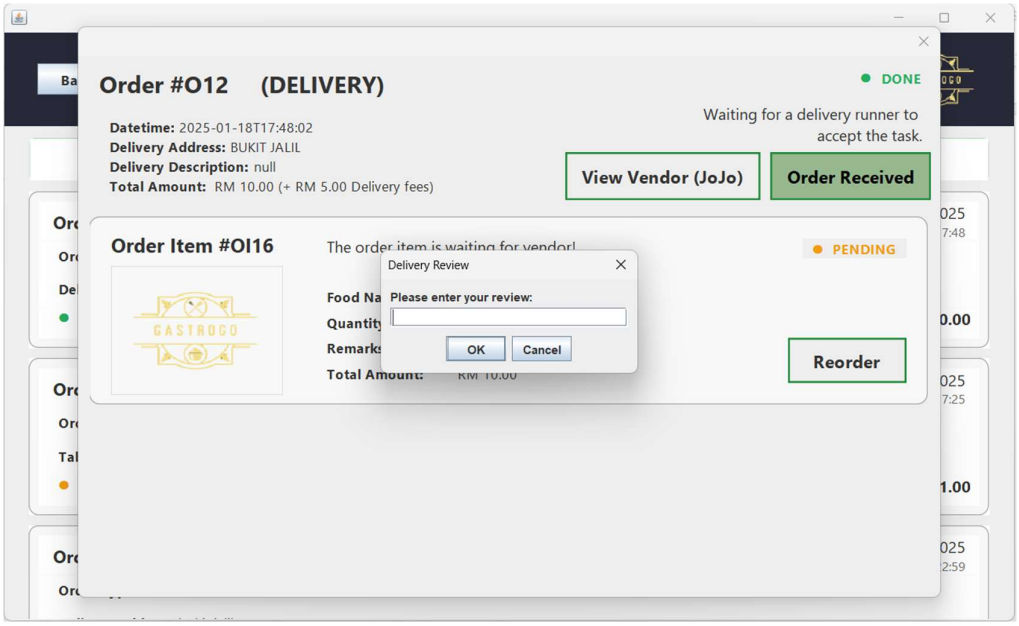


This is the view order detail interface that the customer can rate and submit review to the order for the vendor after the customer receives their order.

2.2.27 Rating Delivery



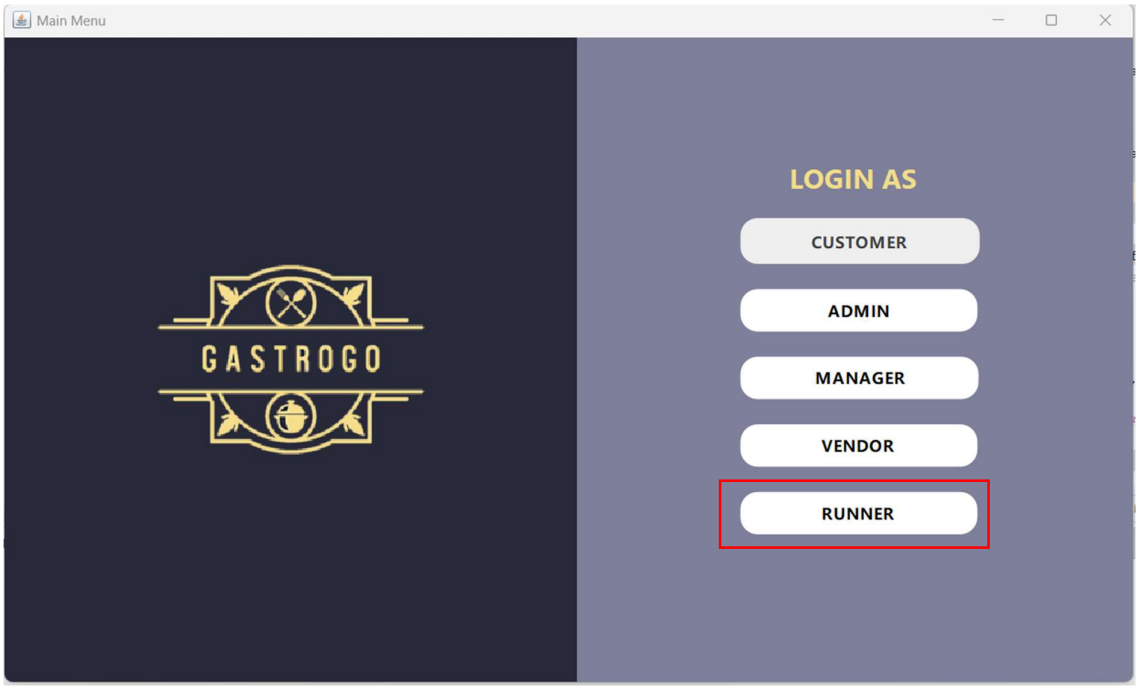
2.2.28 Submit Runner Review



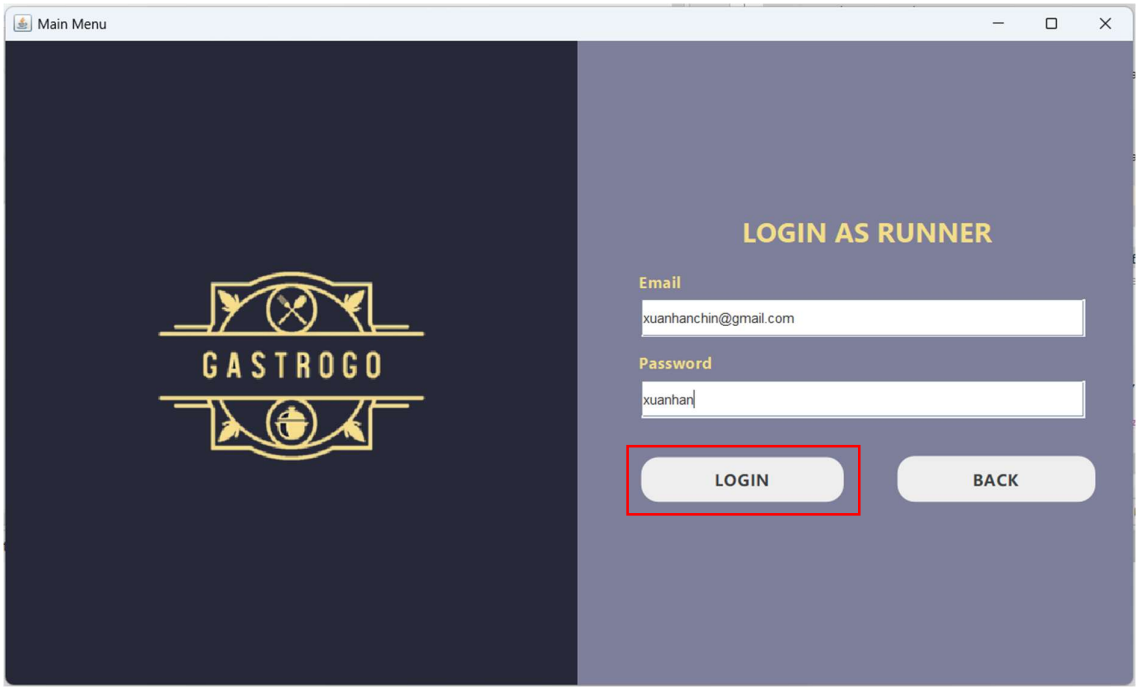
For the delivery order, the customer can rate and submit review to the runner after clicking on the order received button.

2.3 Delivery Runner

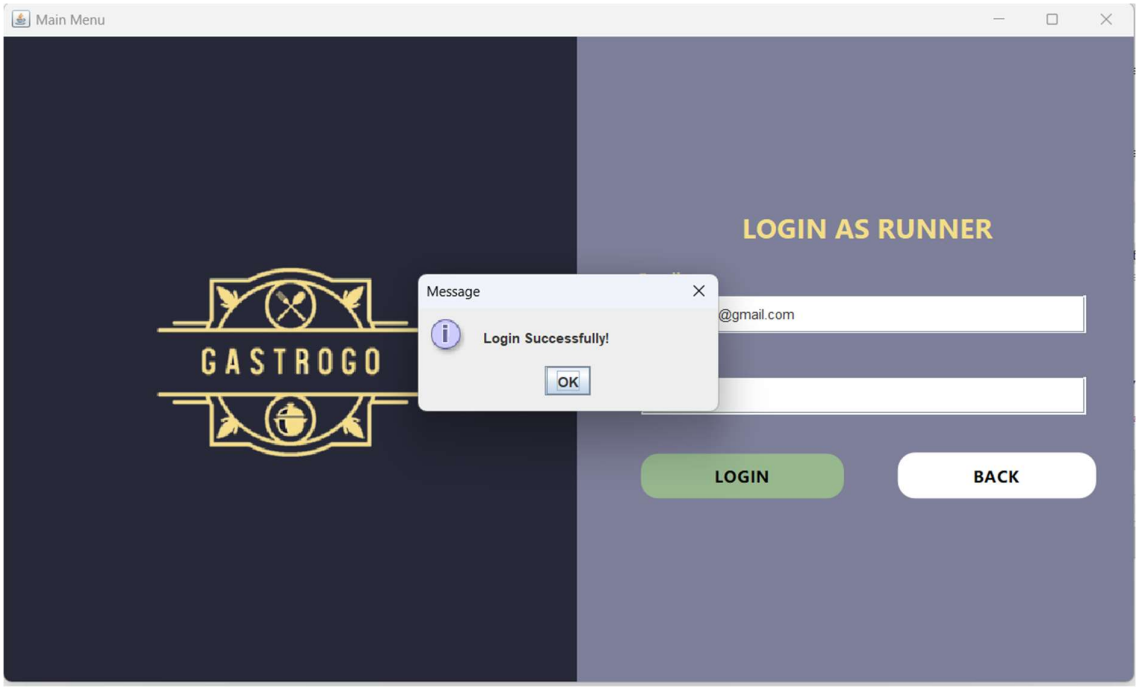
2.3.1 Runner Login



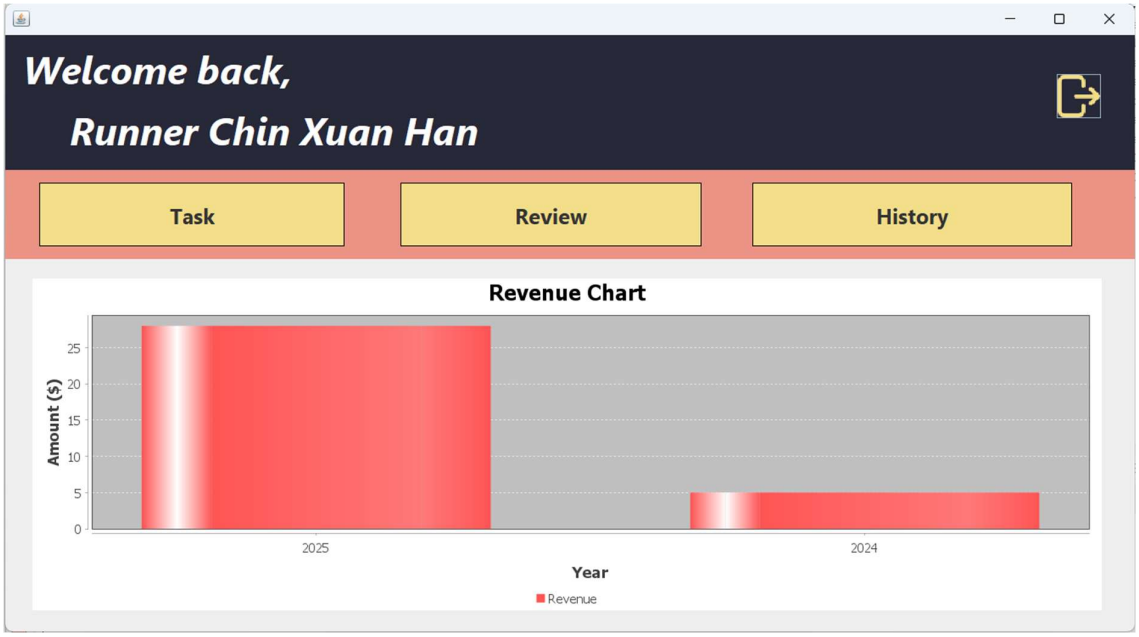
Firstly, the runner will need to choose his/her role by clicking on the runner button.



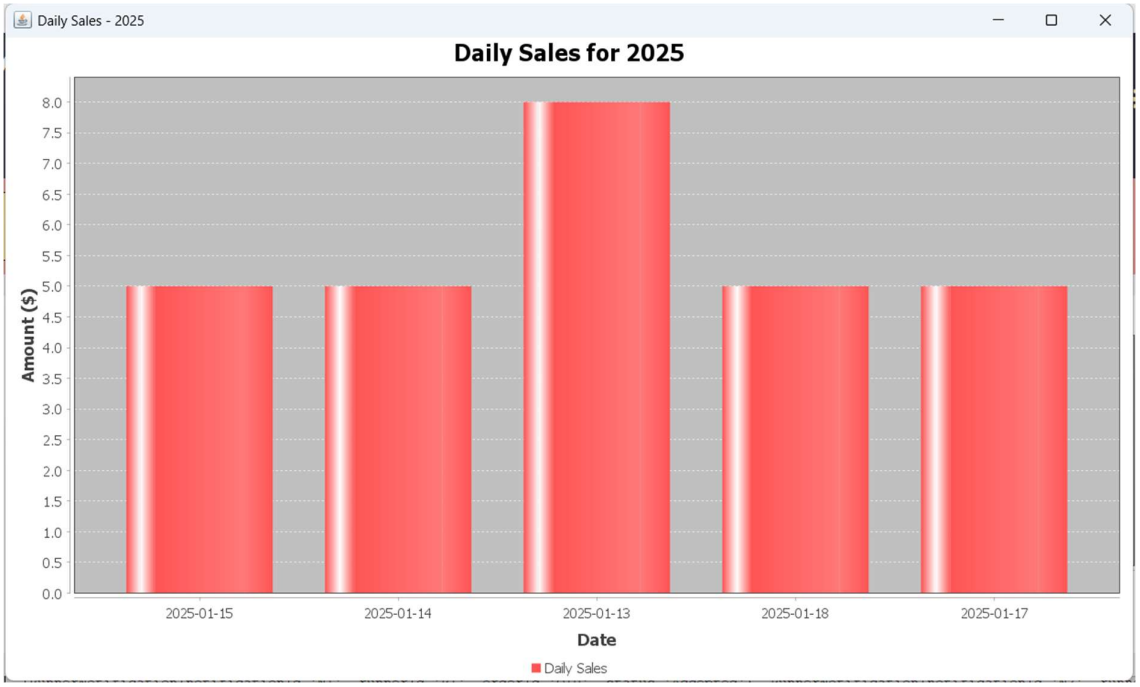
After entering his/her credentials, the runner can click on the login button to log into his account.



2.3.2 Runner Revenue Dashboard



After logging into the runner's account, he/she can view his/her yearly revenue directly. These charts represent the annual revenue for the runner.

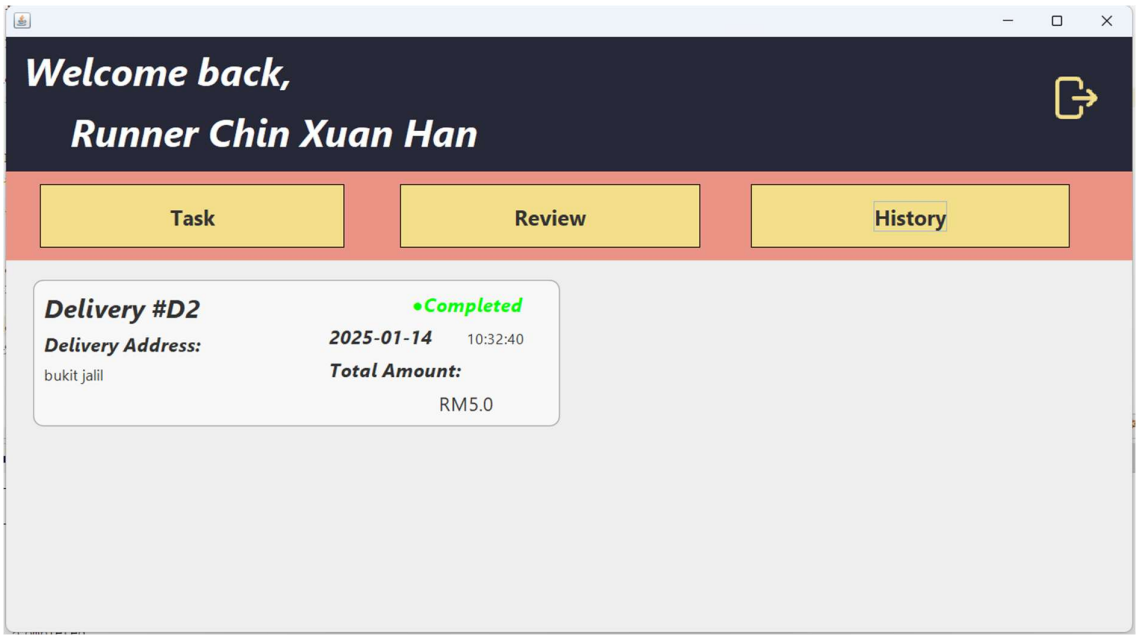


When the runner clicks on the year bar chart, our system will show the daily revenue of the runner for the whole year.

2.3.3 Runner Customer Review

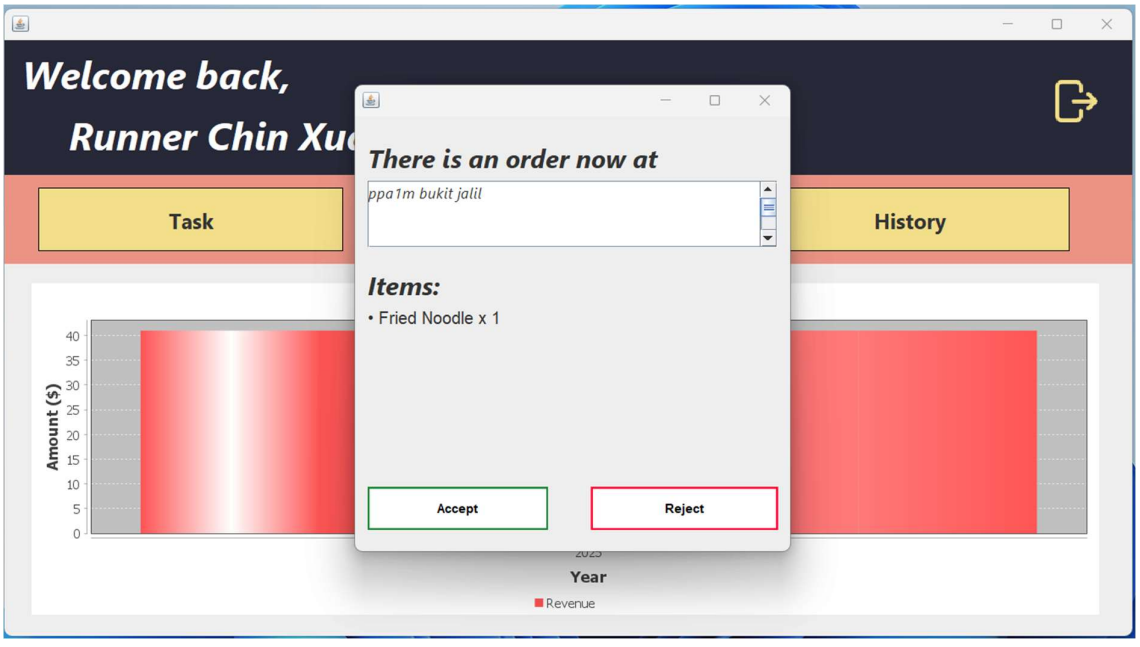
By clicking on the review button, the runner can view all the delivery reviews such as the reviewId, ratings and also description.

2.3.4 Runner Task History

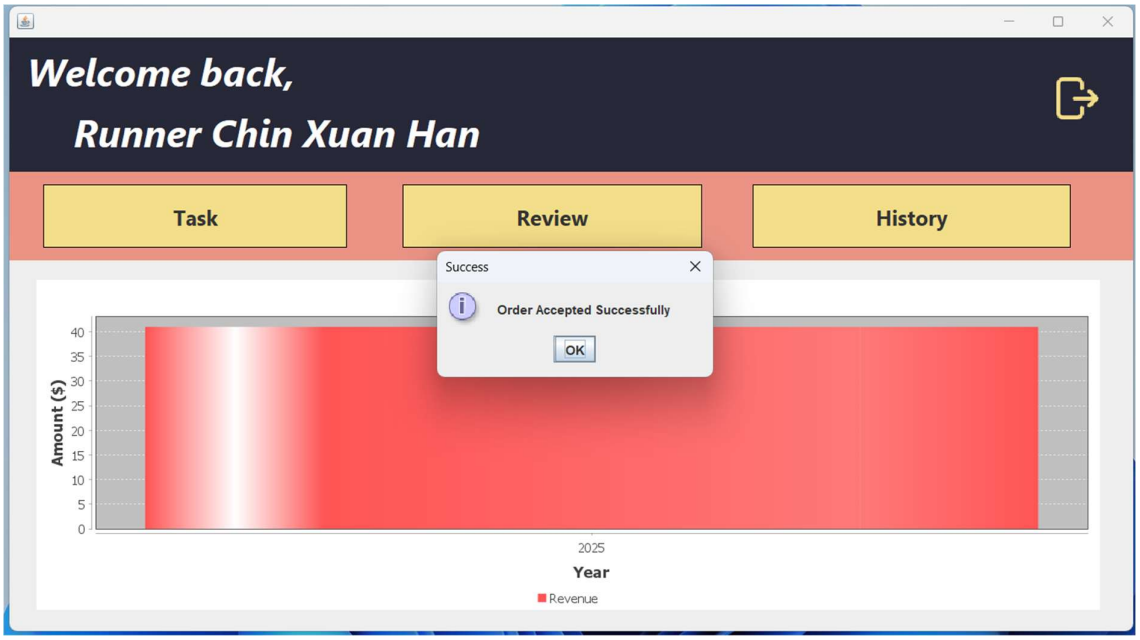


The runner can also view all of his/her task history by clicking on the history button.

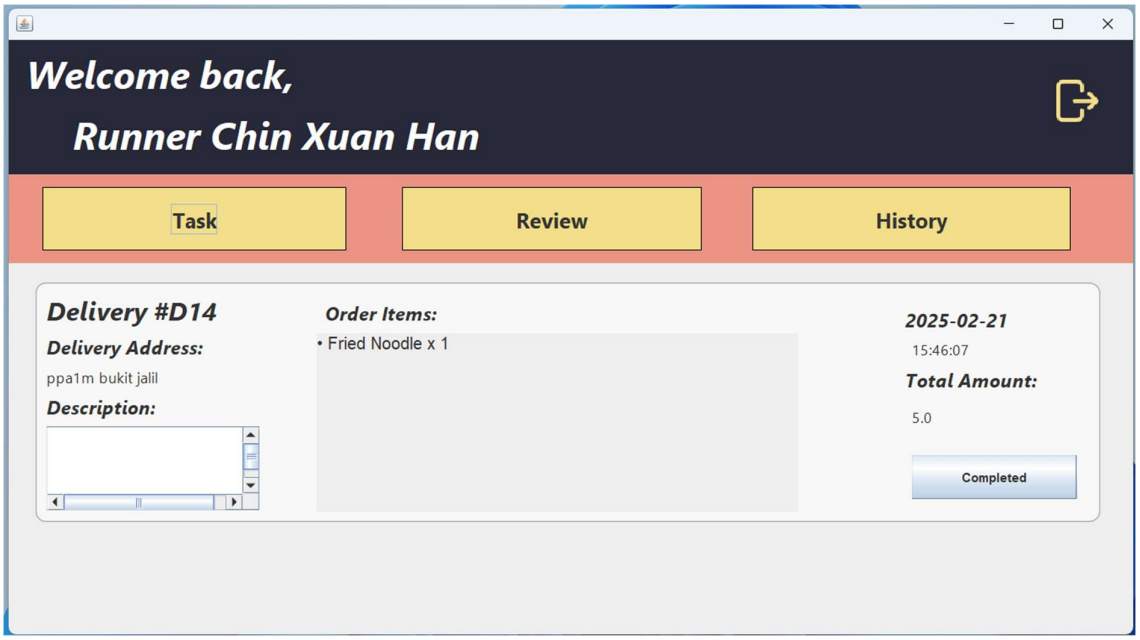
2.3.5 Runner Accept Task



After the customer makes a delivery order, our system will send a pop-up notification to an available runner so that he/she can accept or reject the order. In this case, the runner will click on the accept button to accept the order.

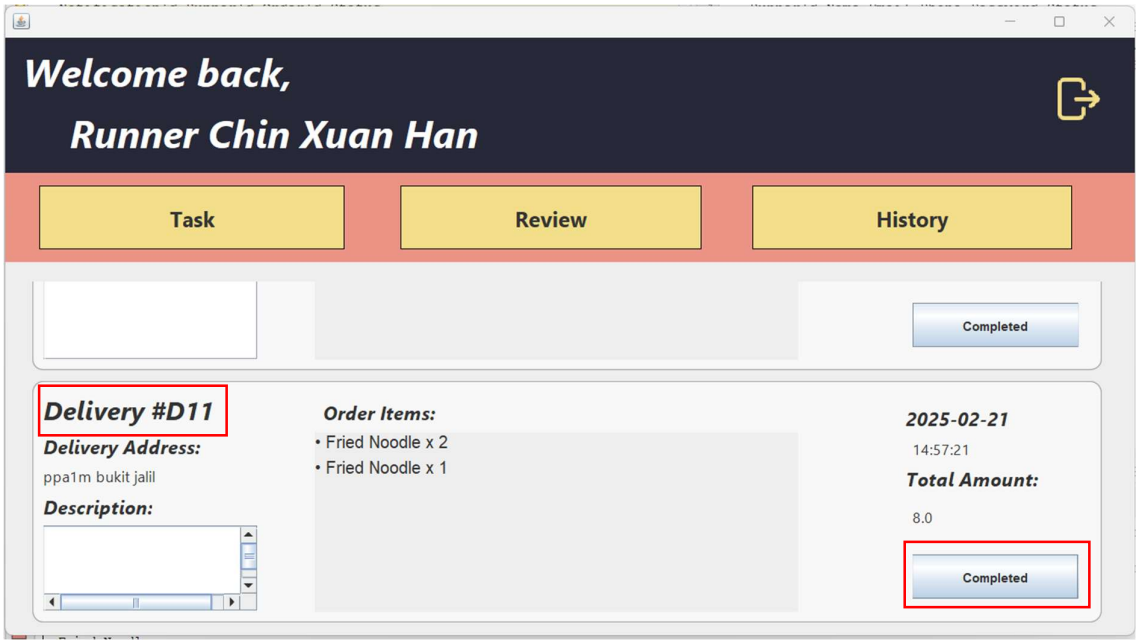


After accepting the order, the system will pop up a dialog box to inform that the order has been accepted by the runner.

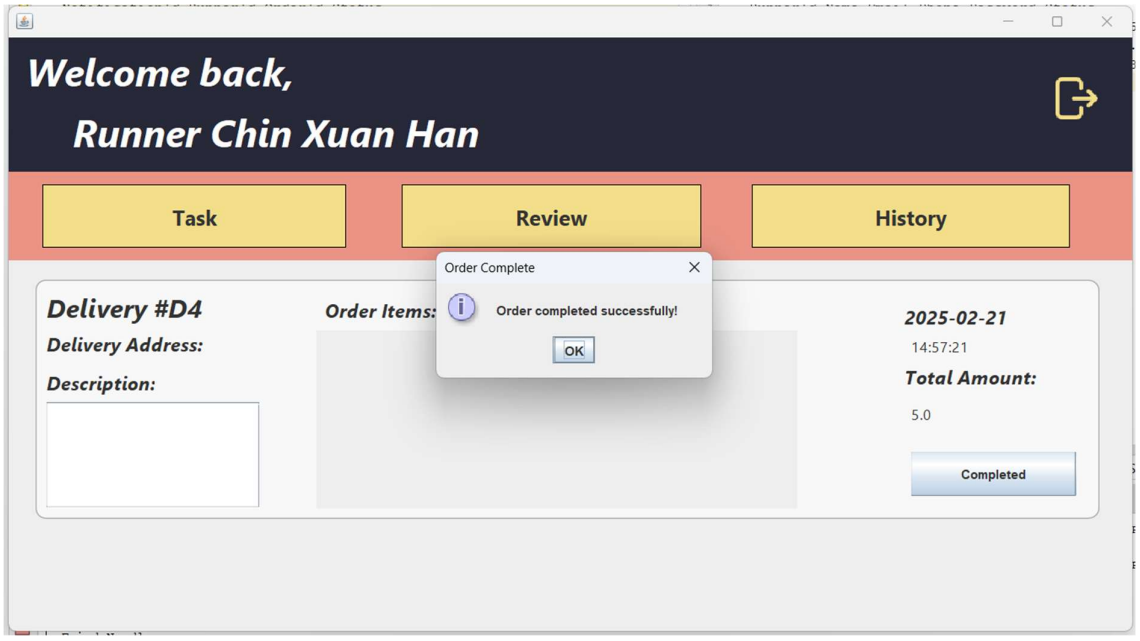


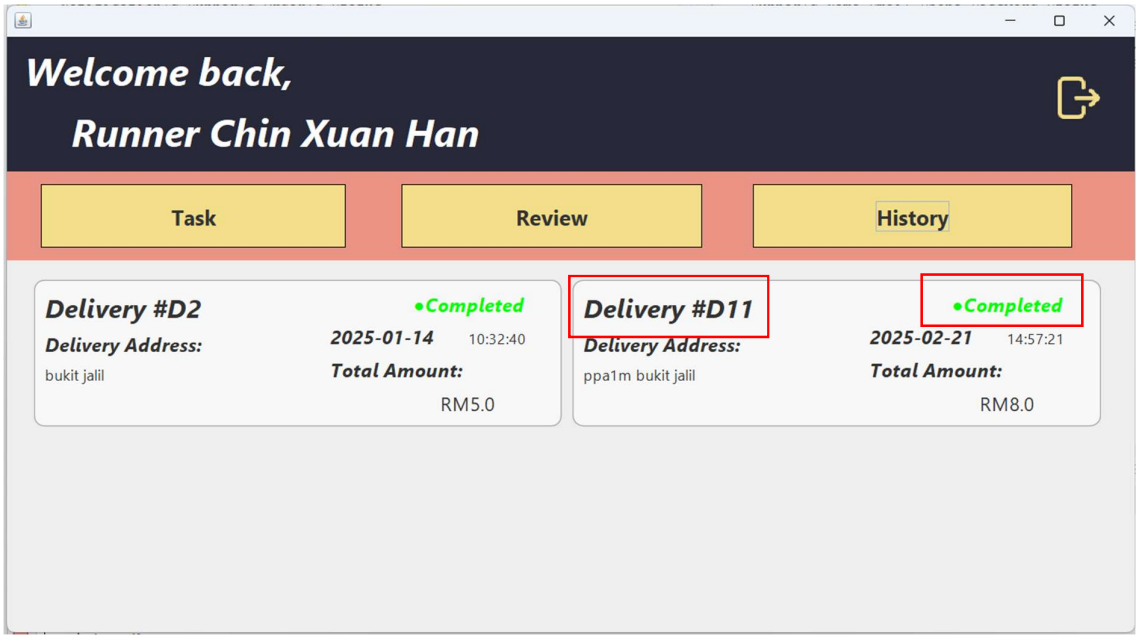
By clicking on the task button, the runner can view the details of the delivery order and update the status of the delivery order.

2.3.6 Update Task Status



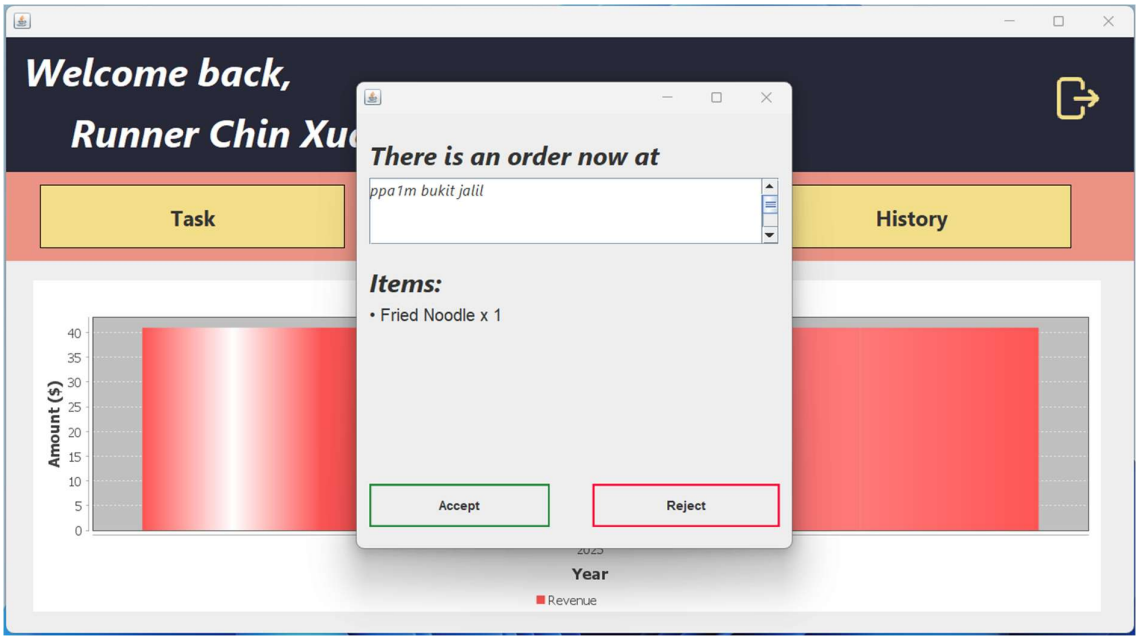
By clicking on the completed button, which means the runner has successfully delivered the order, the status of the delivery will be updated to complete.



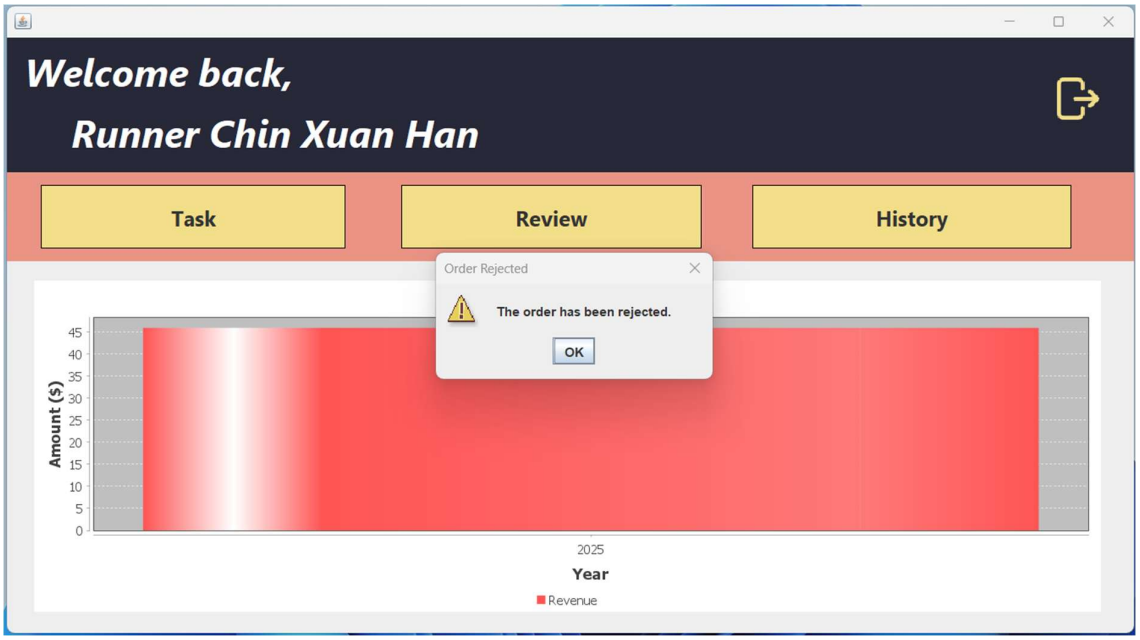


The image above shows that the status of the delivery order has been updated to complete and it will be stored into the history task panel of the runner.

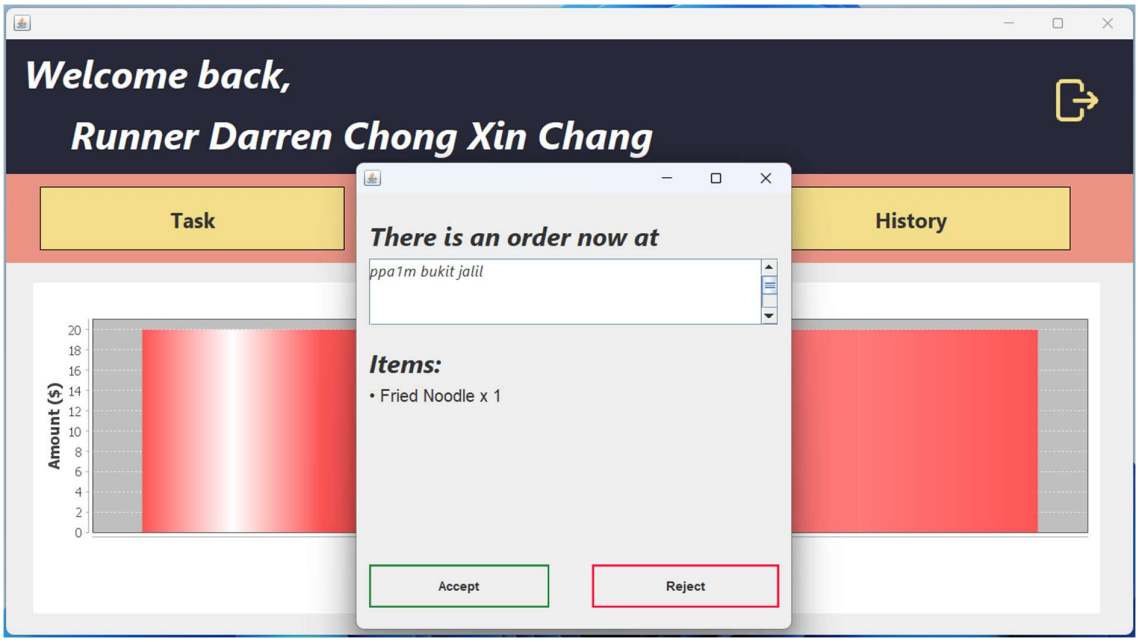
2.3.7 Runner Decline Task



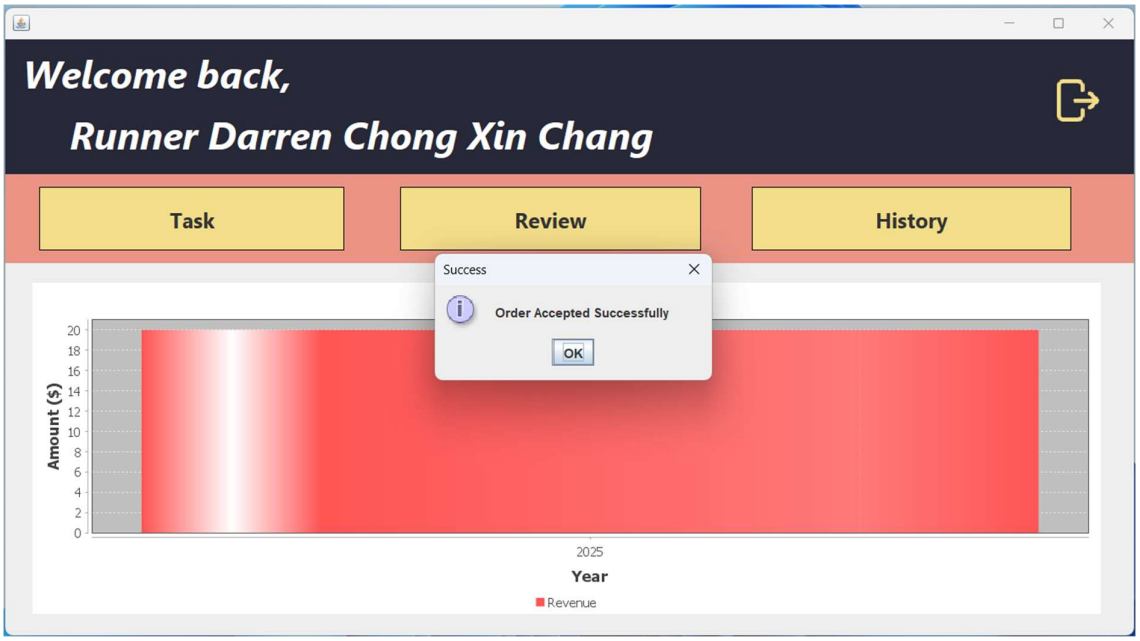
The runner can also choose to reject the order by clicking on the reject button of the notification box.

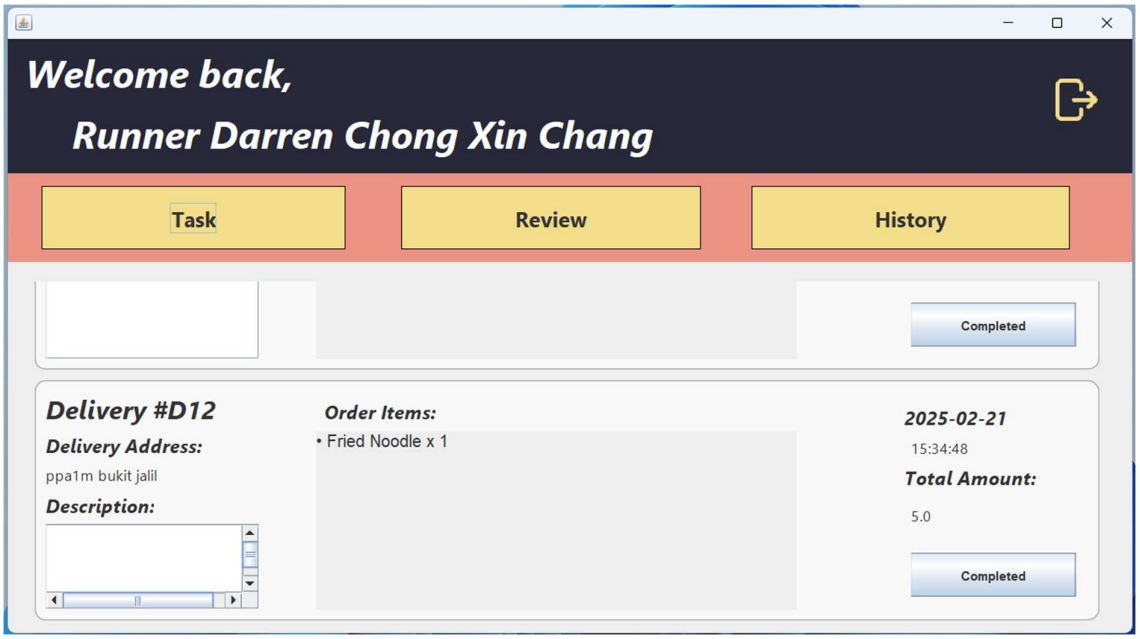


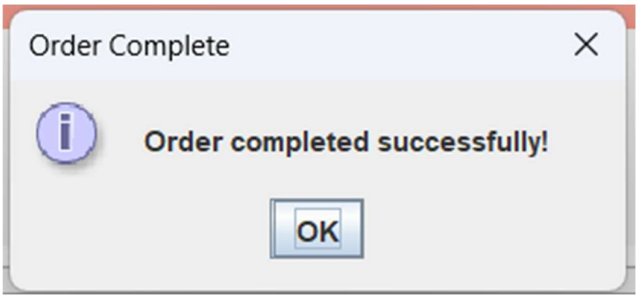
After the delivery is rejected by the first runner, our system will find a runner which has a status of available.

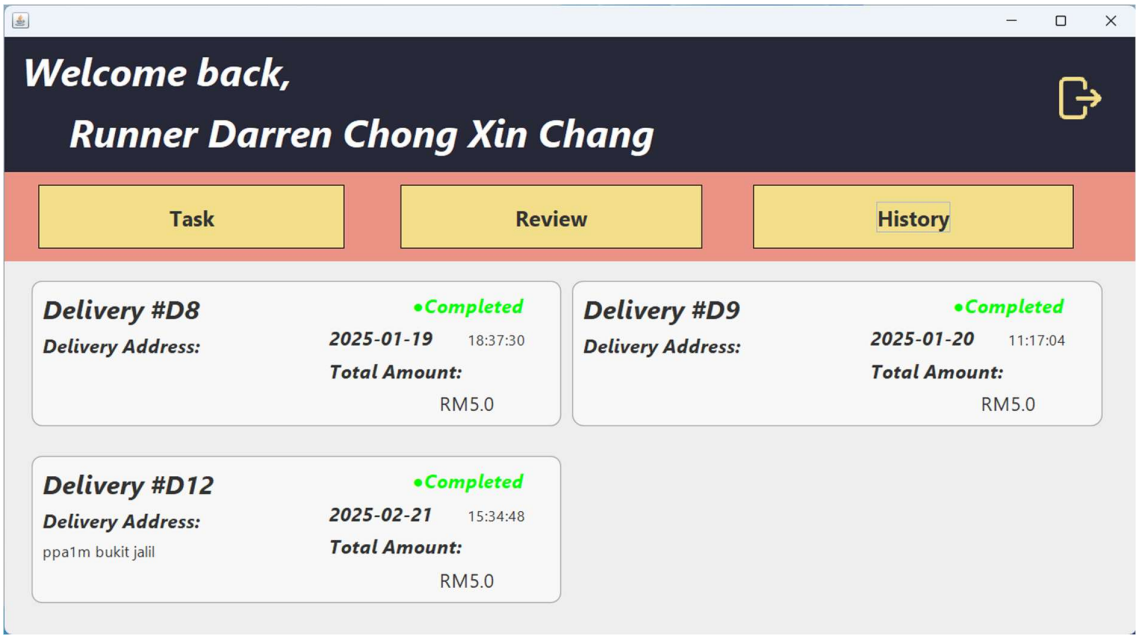


The image above shows that the notification is sent to another driver. Hence, the driver can accept the delivery order as mentioned earlier.


Welcome back,
Runner Darren Chong Xin Chang
Task
Review
History
Success
Order Accepted Successfully
OK
Amount ($)
20
18
16
14
12
10
8
6
4
2
0
2025
Year
Revenue

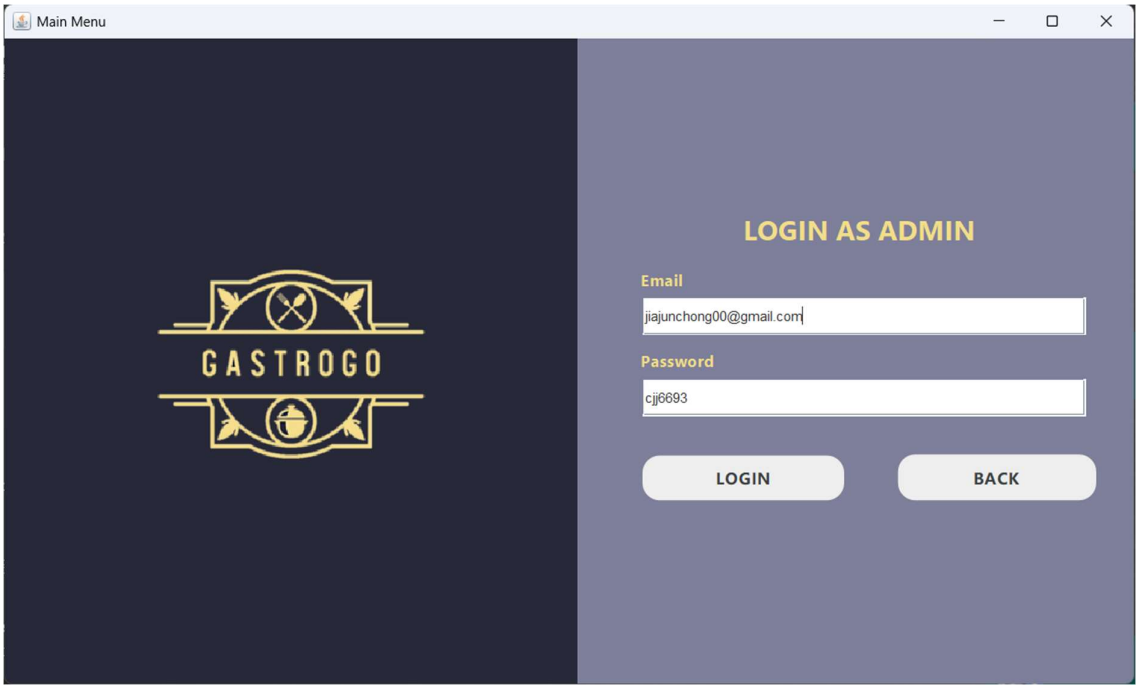

Welcome back,
Runner Darren Chong Xin Chang
Task
Review
History
Completed
Delivery #D12
Delivery Address:
ppa1m bukit jalil
Description:
Order Items:
• Fried Noodle x 1
2025-02-21
15:34:48
Total Amount:
5.0
Completed


Order Complete
Order completed successfully!
OK


Welcome back,
Runner Darren Chong Xin Chang
Task
Review
History
Delivery #D8
•Completed
Delivery Address:
2025-01-19 18:37:30
Total Amount:
RM5.0
Delivery #D9
•Completed
Delivery Address:
2025-01-20 11:17:04
Total Amount:
RM5.0
Delivery #D12
•Completed
Delivery Address:
ppa1m bukit jalil
2025-02-21 15:34:48
Total Amount:
RM5.0

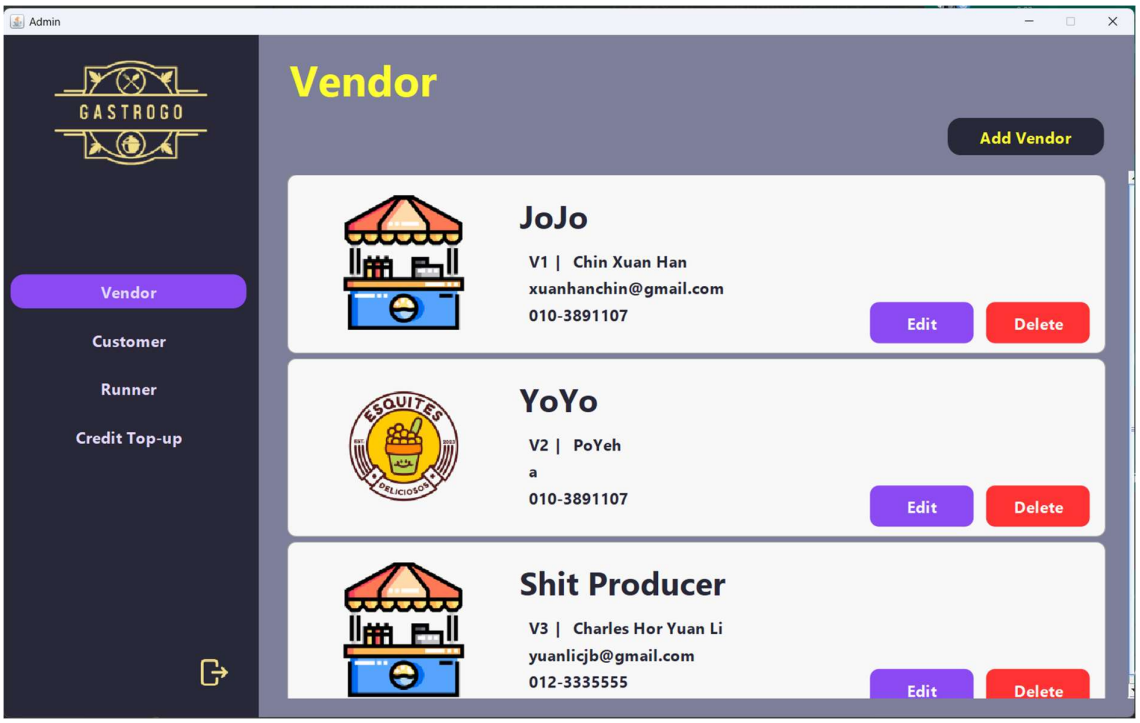
2.4 Administrator

2.4.1 Admin Login Page



To login into the admin page, admin needs to first click on the admin button on the start page. Once the admin being directed to the admin login page, they will need to fill in the correct email and password in order to access the admin page. After filling the email and password, just click the login button. If the credentials entered are correct, the admin will be redirected to the admin page. If the credentials are incorrect, an error message will be displayed prompting the user to re-enter their email and password.

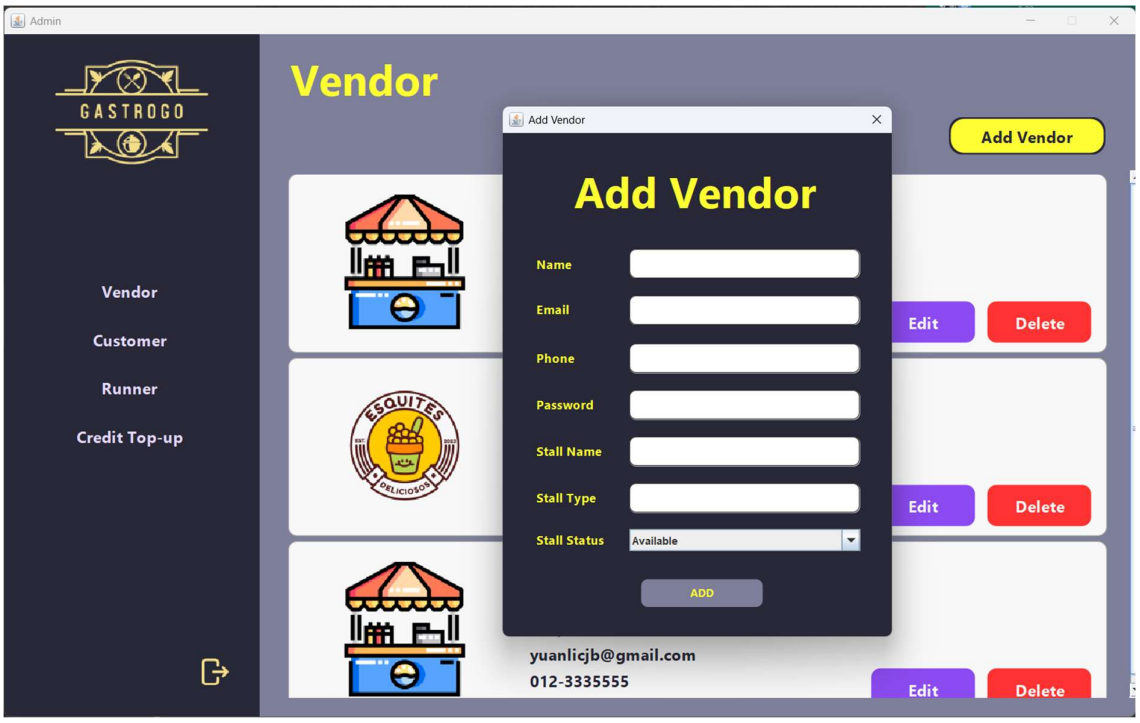
2.4.2 Vendor List Page



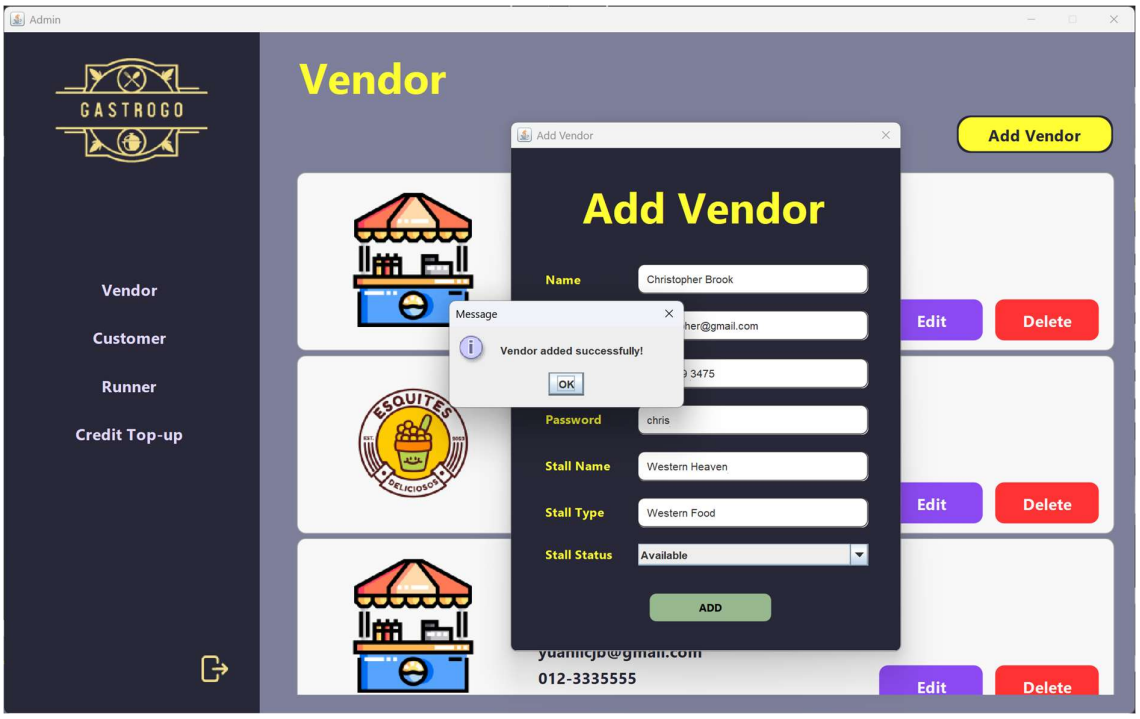
When the admin successfully login, they will be redirected to the default page for admin which is the Vendor List Page. This page will list out all the vendor stored in vendor database with their information including stall name, vendor ID, vendor name, email and contact number. There is also a stall's profile picture for every vendor for better recognition. Admin can scroll the page down in order to view more vendor.

In this default page, there is a navigation panel for admin to select which page they want to be directed to. At the right lower part of the panel, there is a logout logo for admin to sign out from the system.

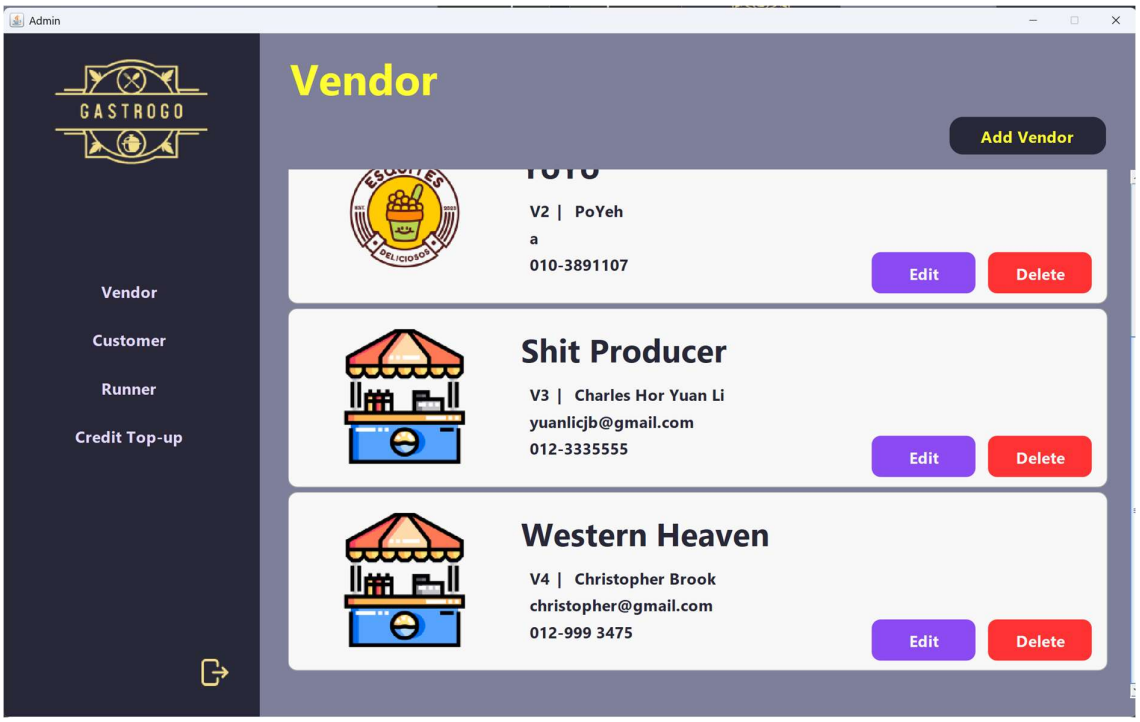
2.4.3 Add Vendor



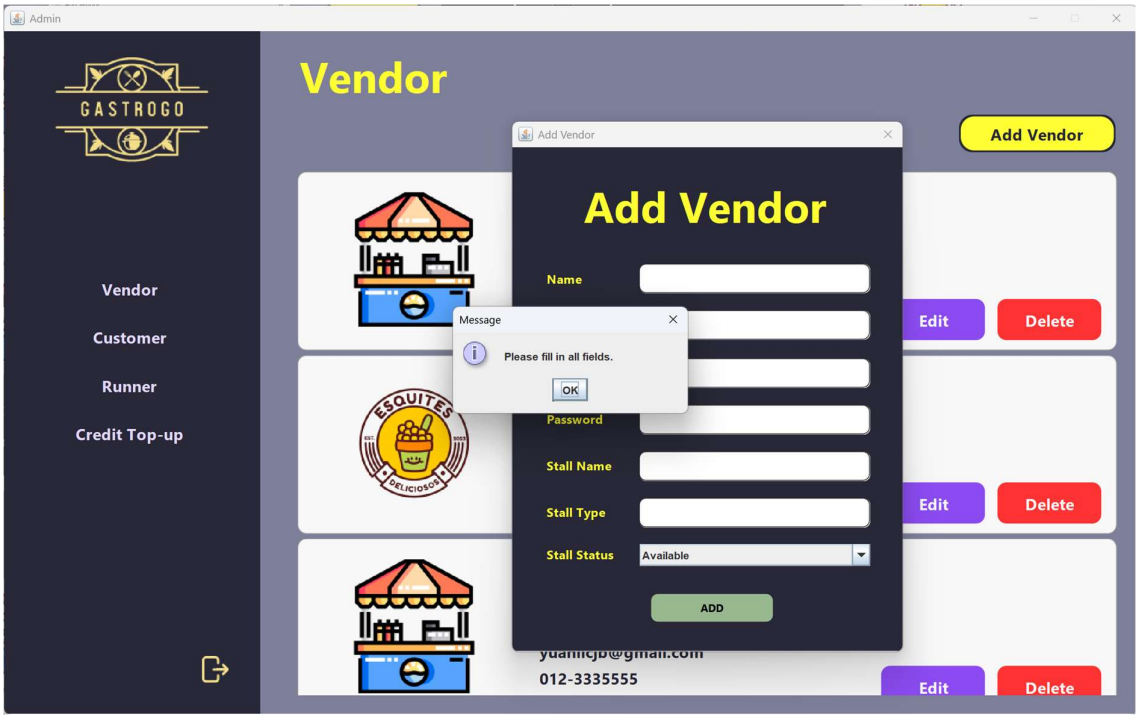
Admin can add new vendor by clicking on the Add Vendor button and there will be a pop-up window for admin to enter the data for the new vendor. Admin is required to fill in all the data inside the blanks to add the new vendor.



Once all the data was filled in, admin just needs to click on ADD button to add new vendor into database. A new vendor ID will be created for the new vendor. There will be a message showing admin that the new vendor has been added successfully.

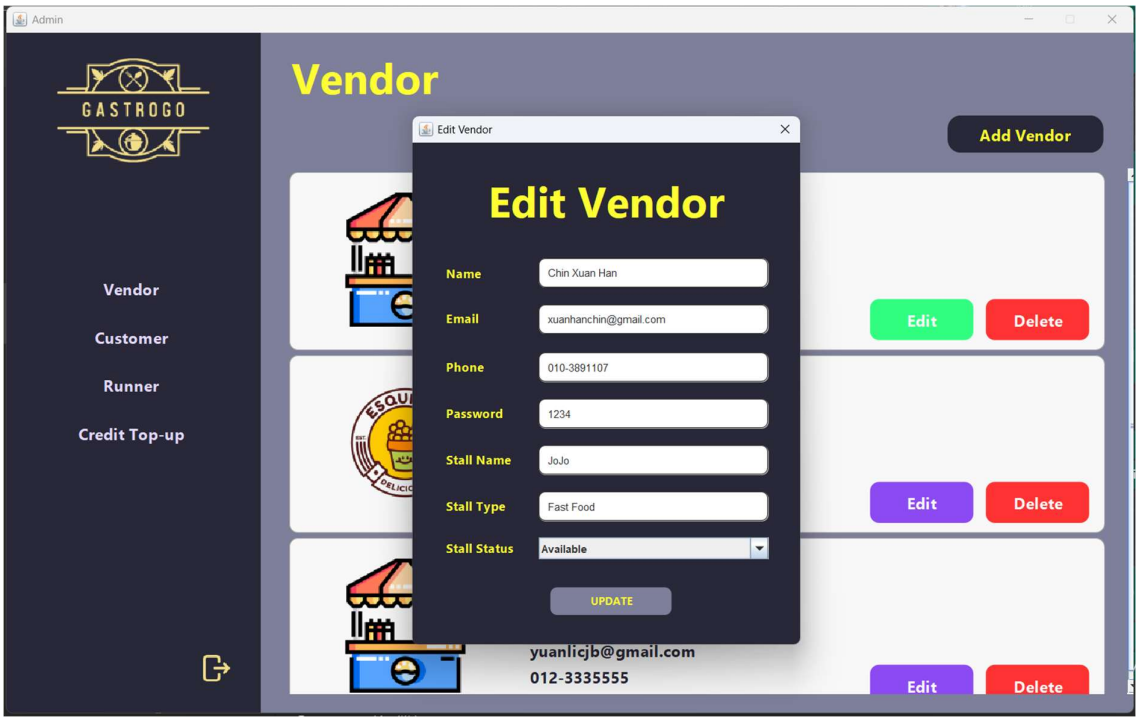


Admin can view the information of new vendor by clicking the Vendor from navigation panel to refresh the vendor list page.

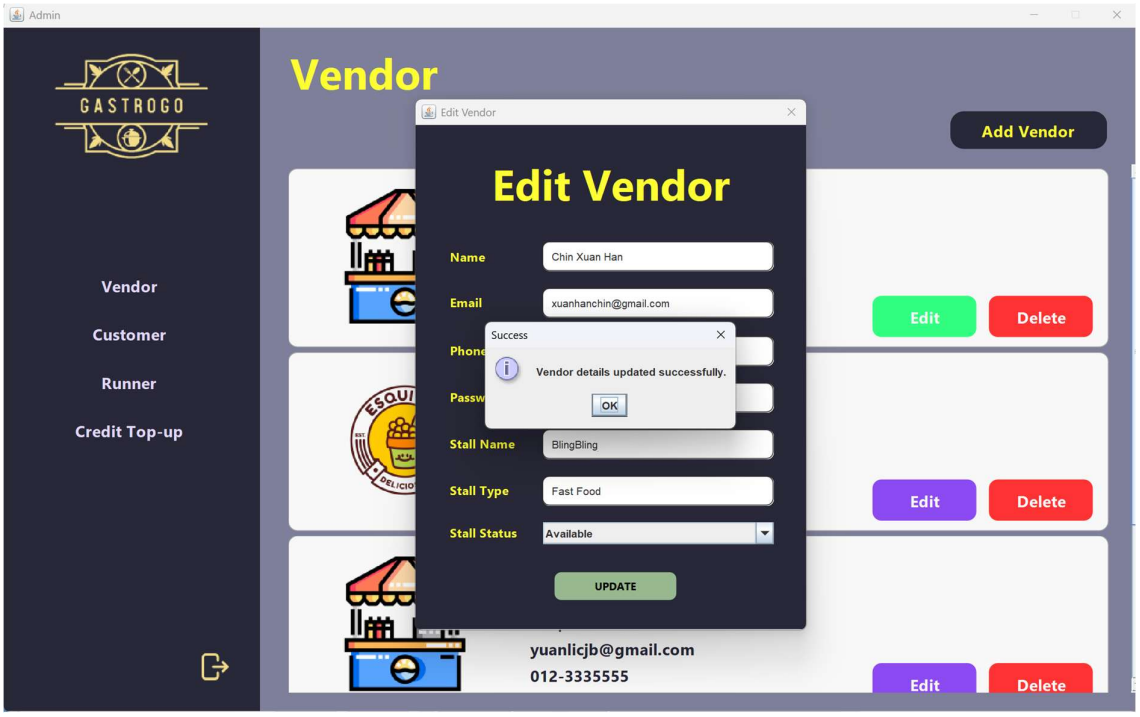


If there is any field hasn't been filled, a message will be showed to admin to ask them fill in all the blank fields.

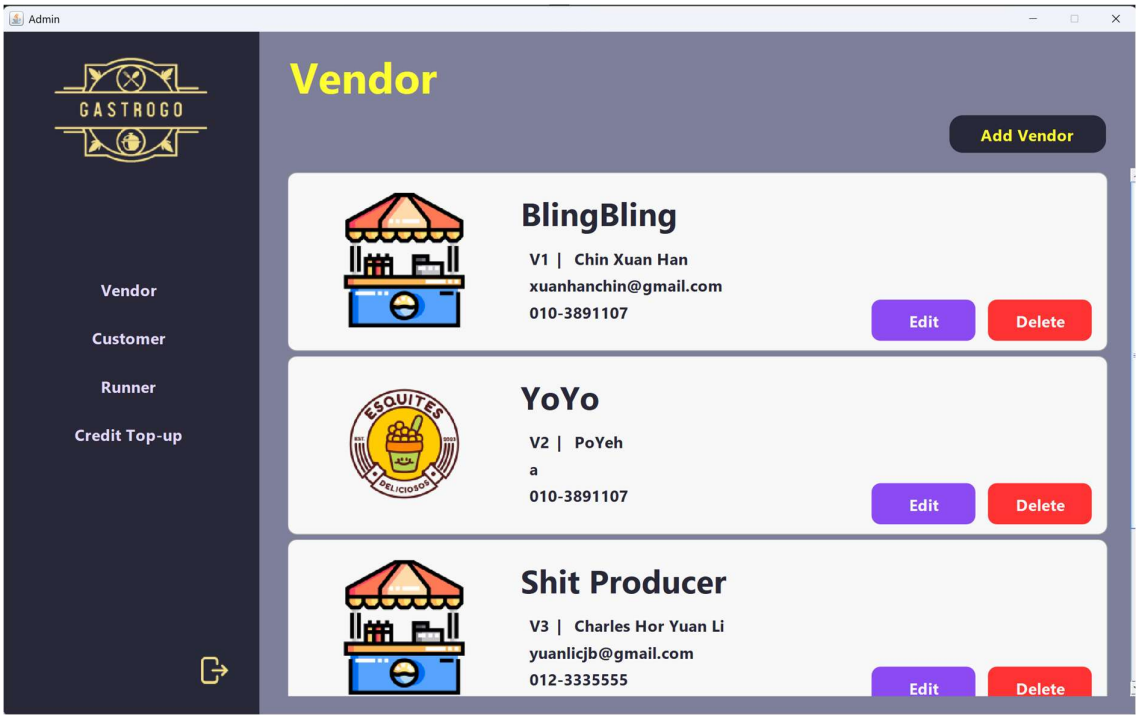
2.4.4 Edit Vendor



Admin can also edit the data for a specific vendor by clicking on Edit button from the vendor block that they want to modify. All the data from that vendor will be show at every field. Admin can modify the data and then click on UPDATE button to update the new data to the database.

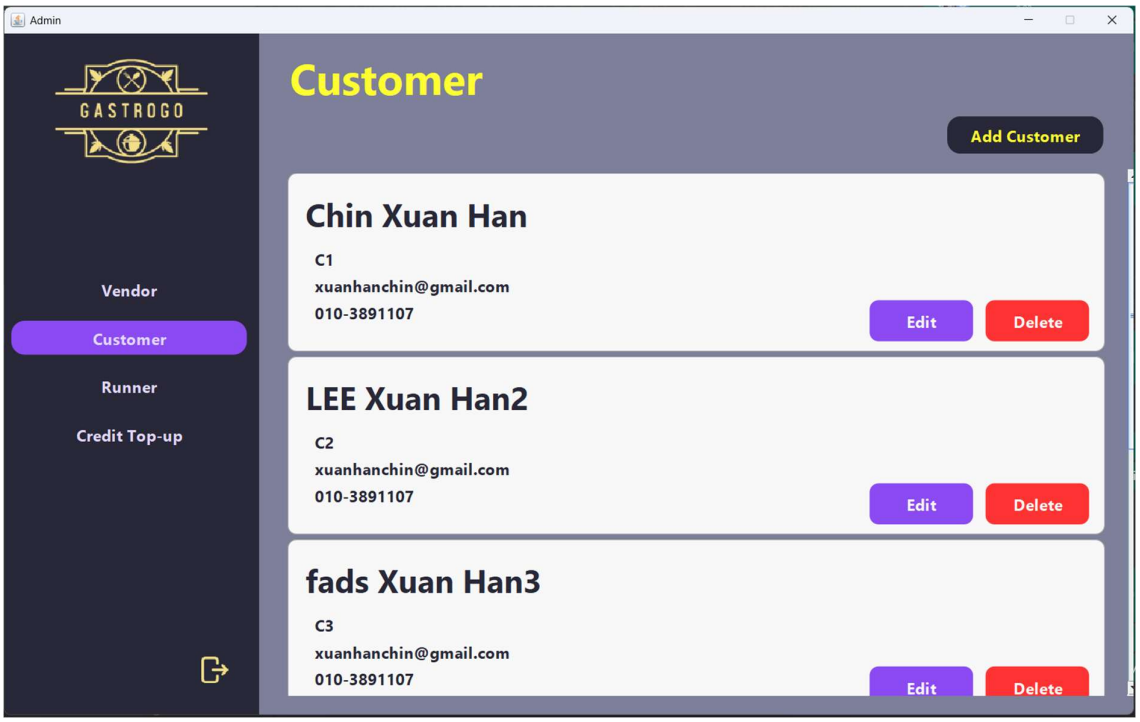


Once the data updated successfully, there will be a message prompt to admin about the success of updating data.



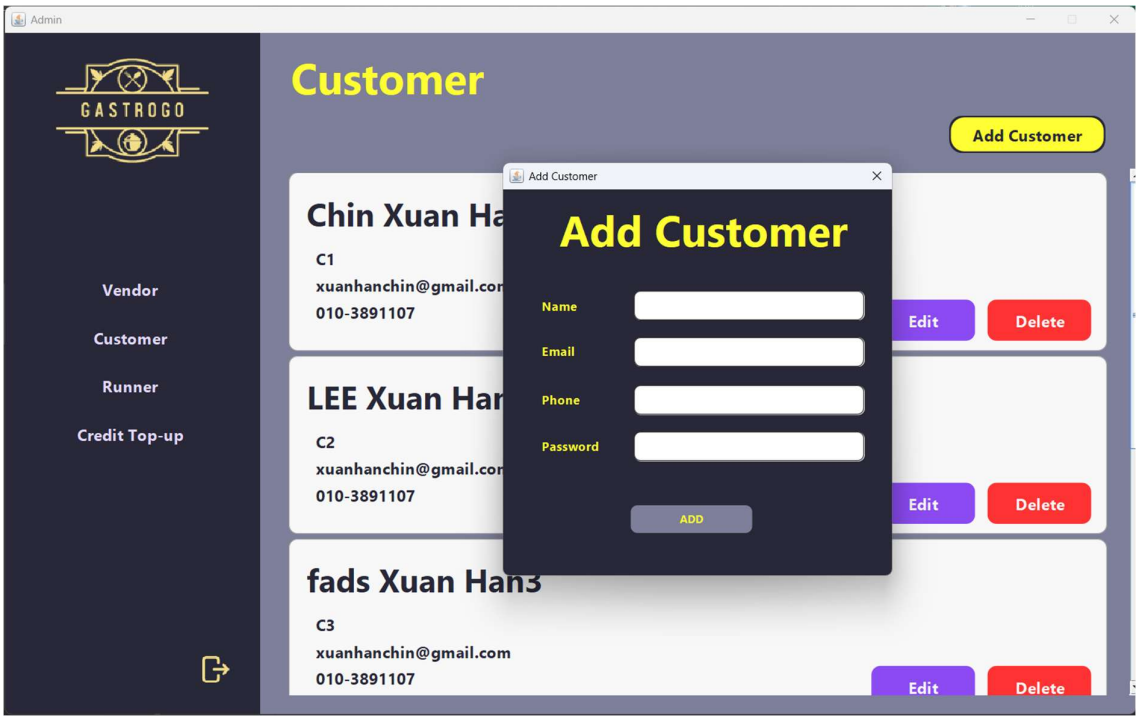
Same with the step in Add Vendor, admin needs to refresh the Vendor List Page by clicking on the Vendor again from the navigation panel to view the modified data.

2.4.5 Customer List Page

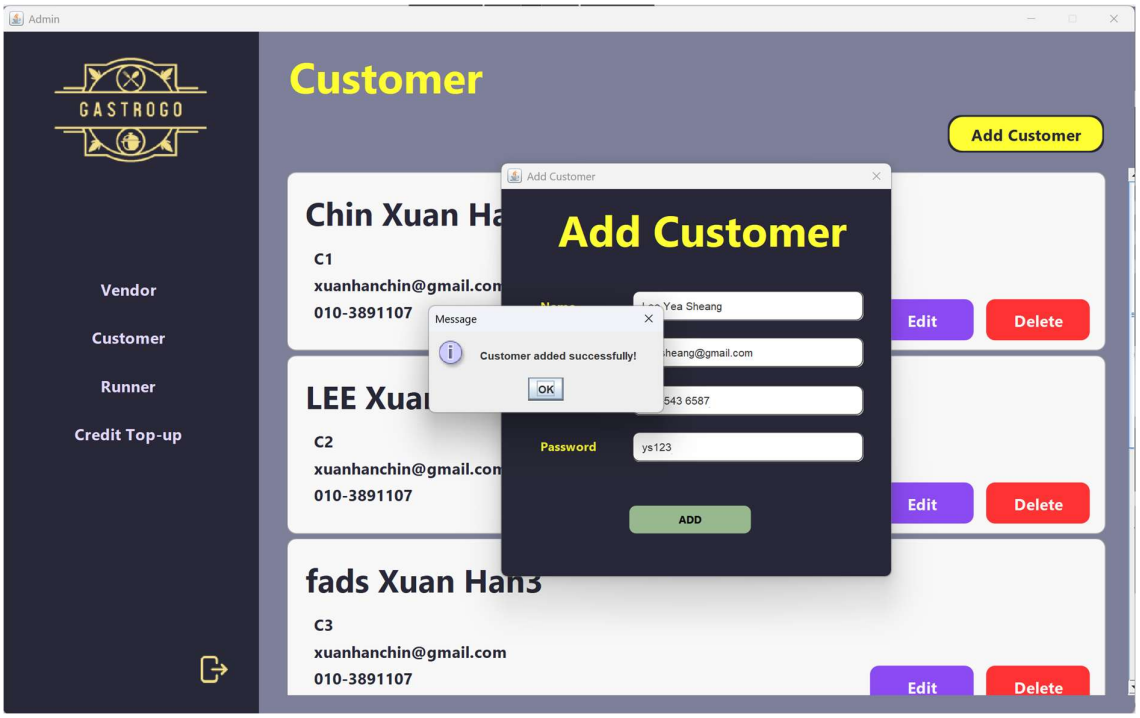


To turn to the Customer List Page, admin has to click on Customer from the navigation panel. Same with the Vendor List Page. There is a scroll function for admin to scroll down to view more customer lists. This page will list out all the customers stored in the customer database with their details including name, customer ID, email and contact number.

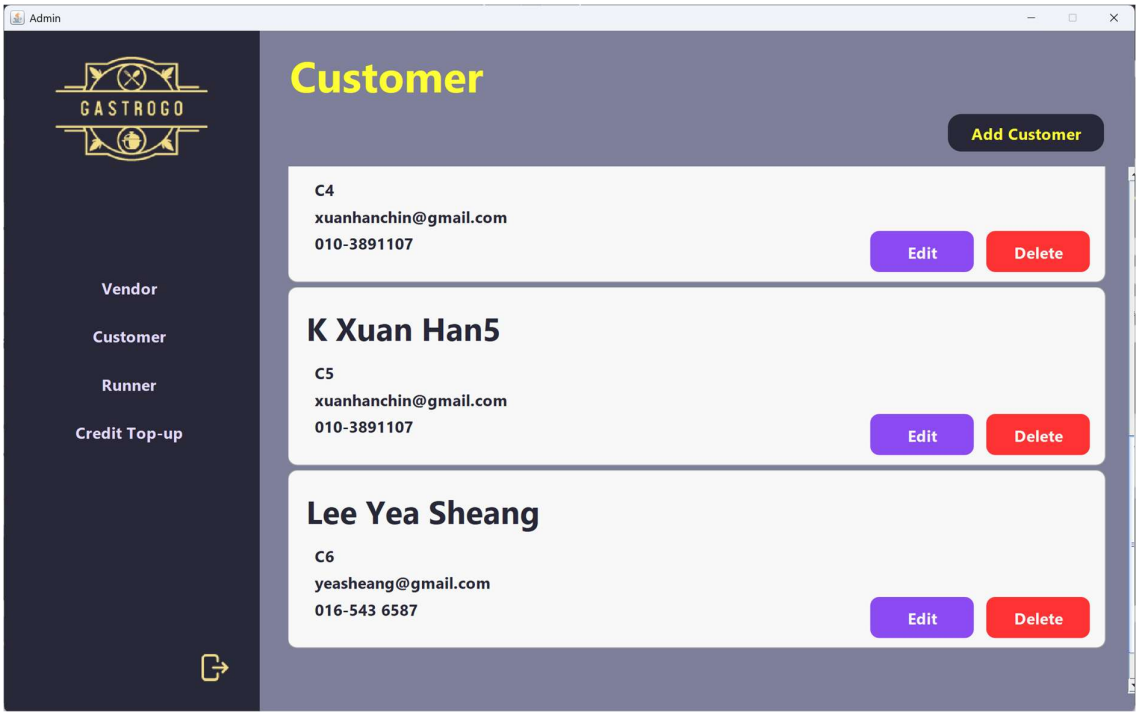
2.4.6 Add Customer



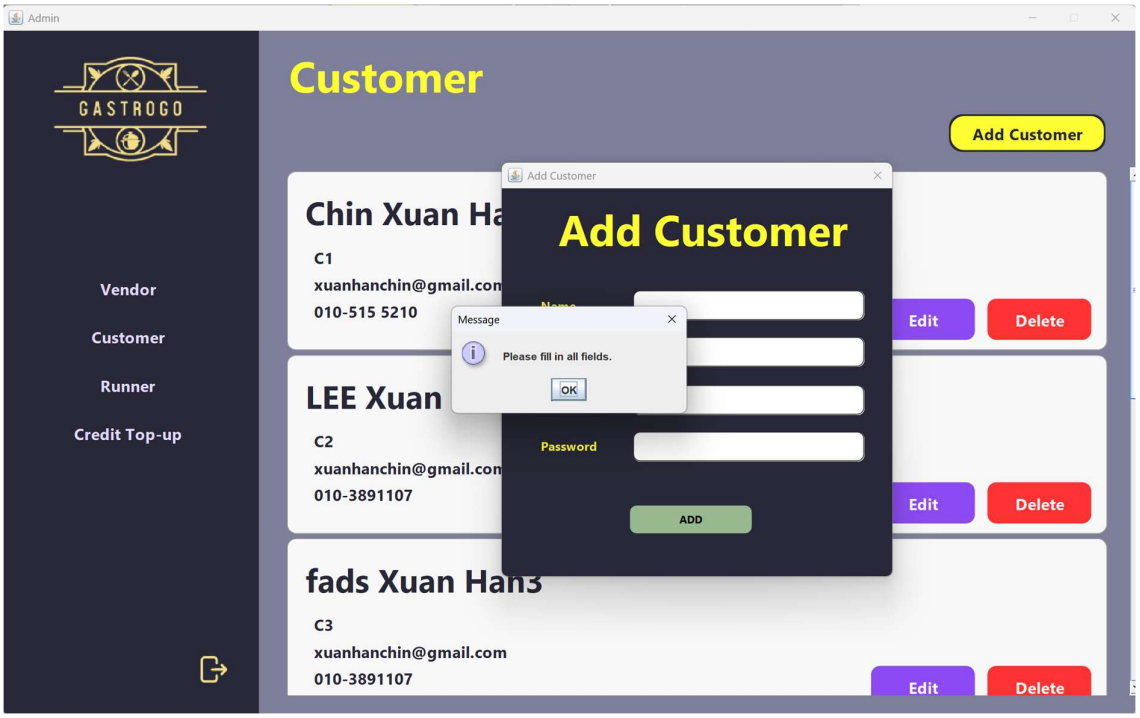
Admin can creates an customer account for the new customer by clicking on the Add Customer button and there will be a pop-up window for admin to key in details for customer. Same as Add Vendor, all fields must be filled to create the new account.



Once all of the customer details are filled in, admin can click on ADD button to add customer data into the database.

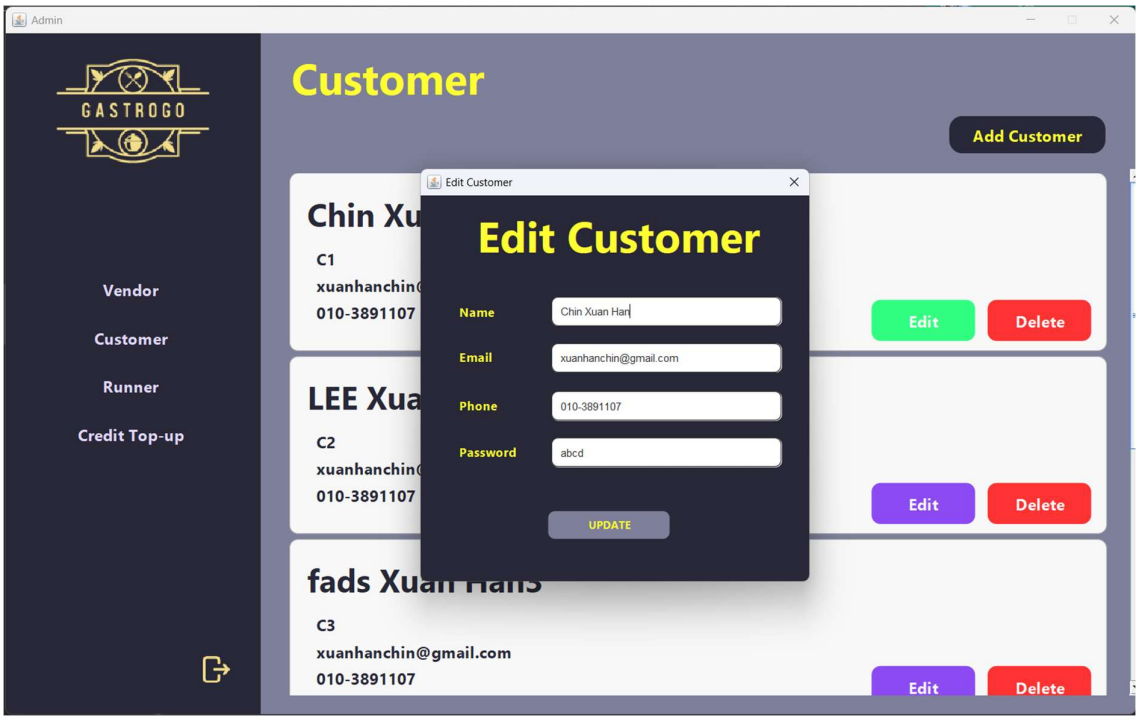


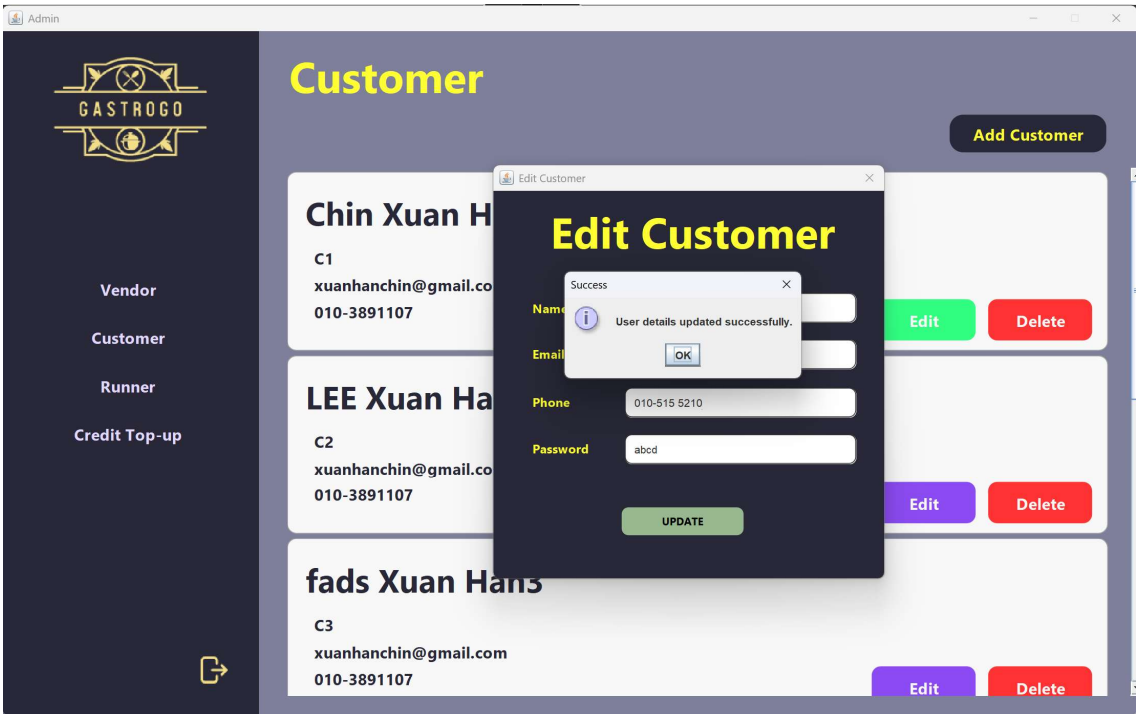
Admin needs to refresh the page to view the new added customer.



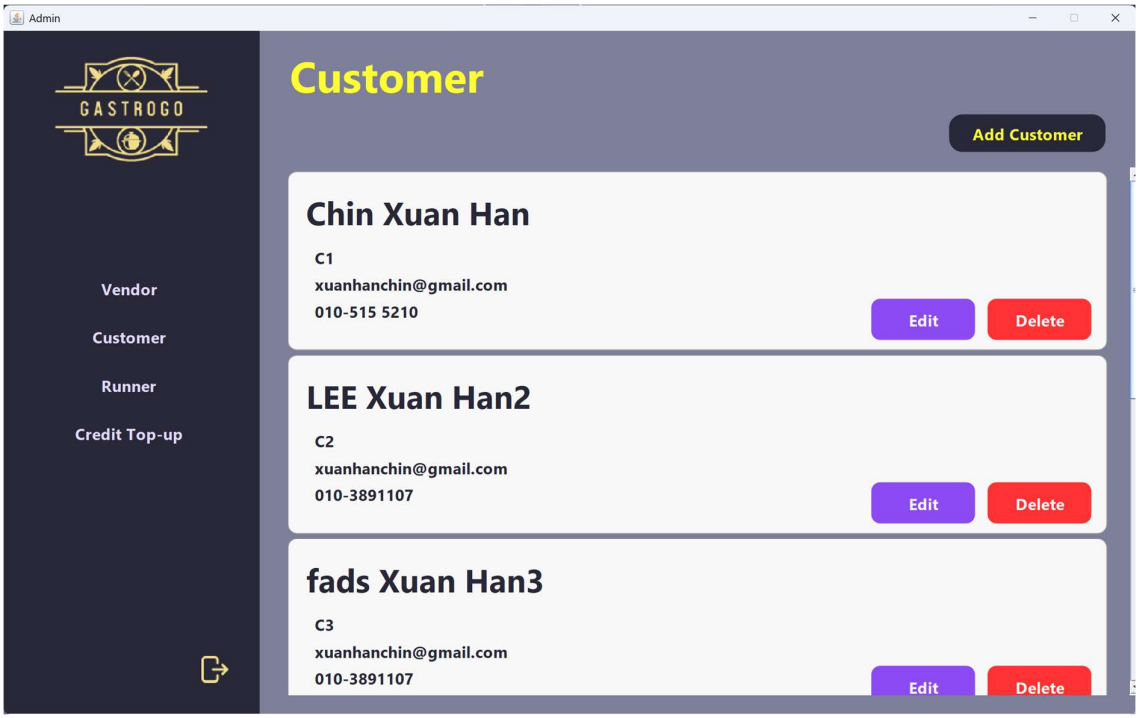
If there is any field haven't been filled, then the system will show a message to ask admin filled in all fields.

2.4.7 Edit Customer





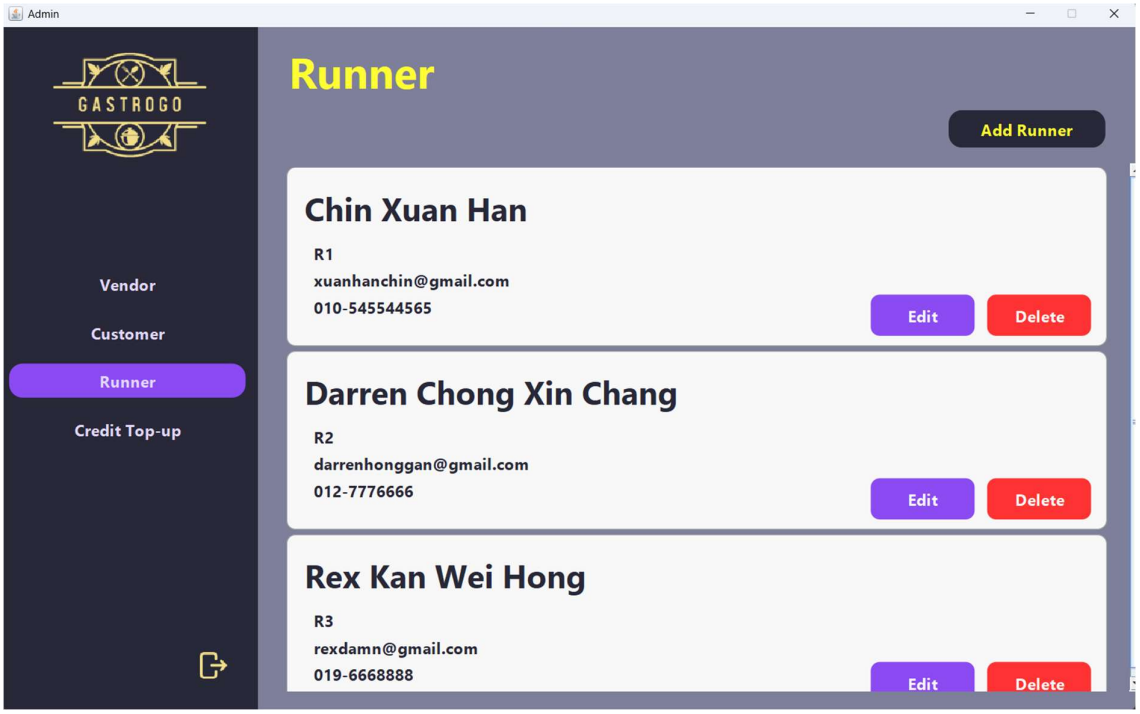
Customer details can also be updated by clicking on Edit button and all the customer details will be showed at the pop-up window. Admin just need to modify the field that need to be updated and click on UPDATED button to update the data to the database.



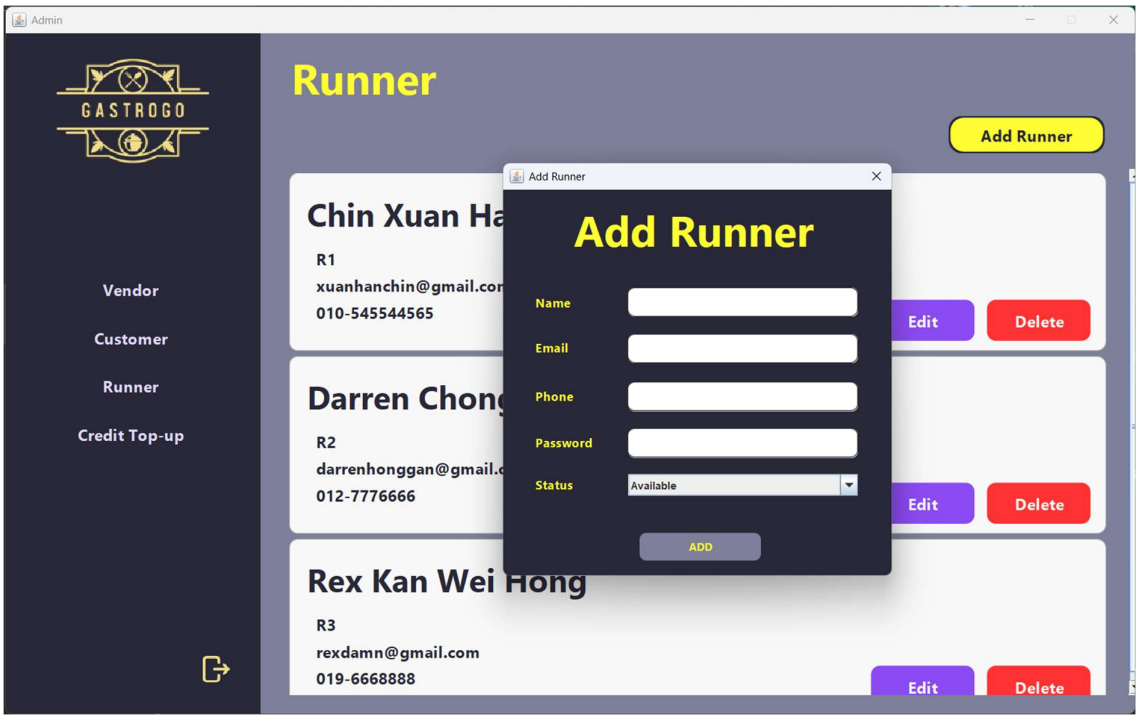
The updated data can be viewed by refresh the Customer List Page.

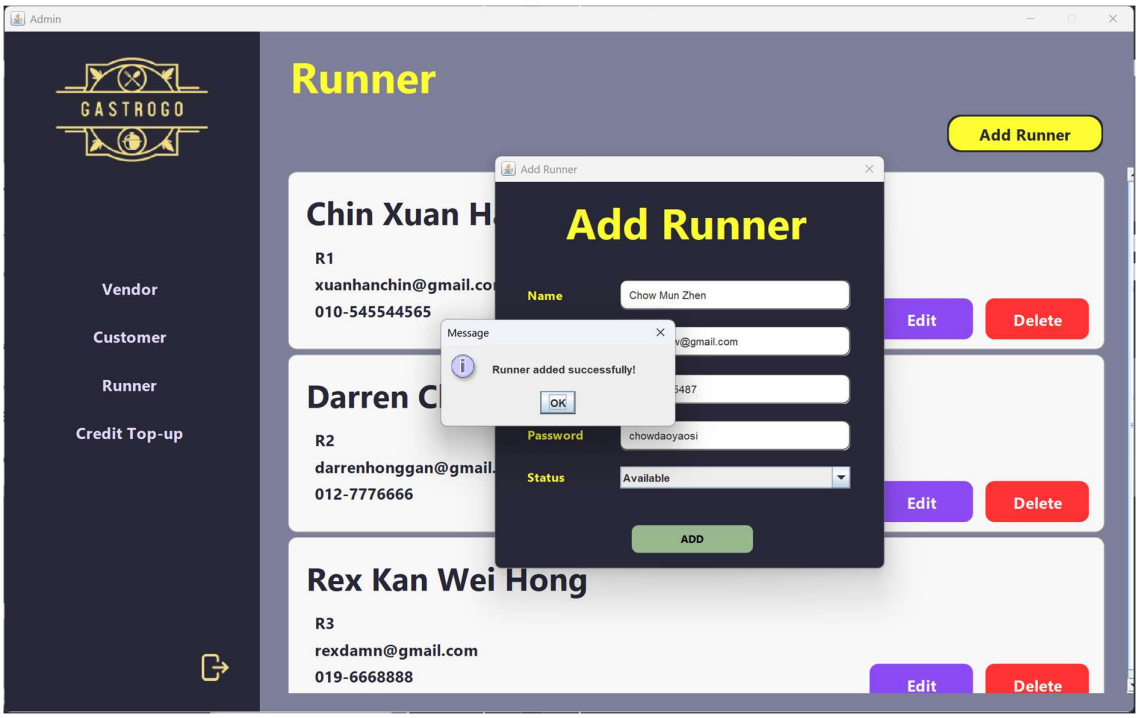
2.4.8 Runner List Page

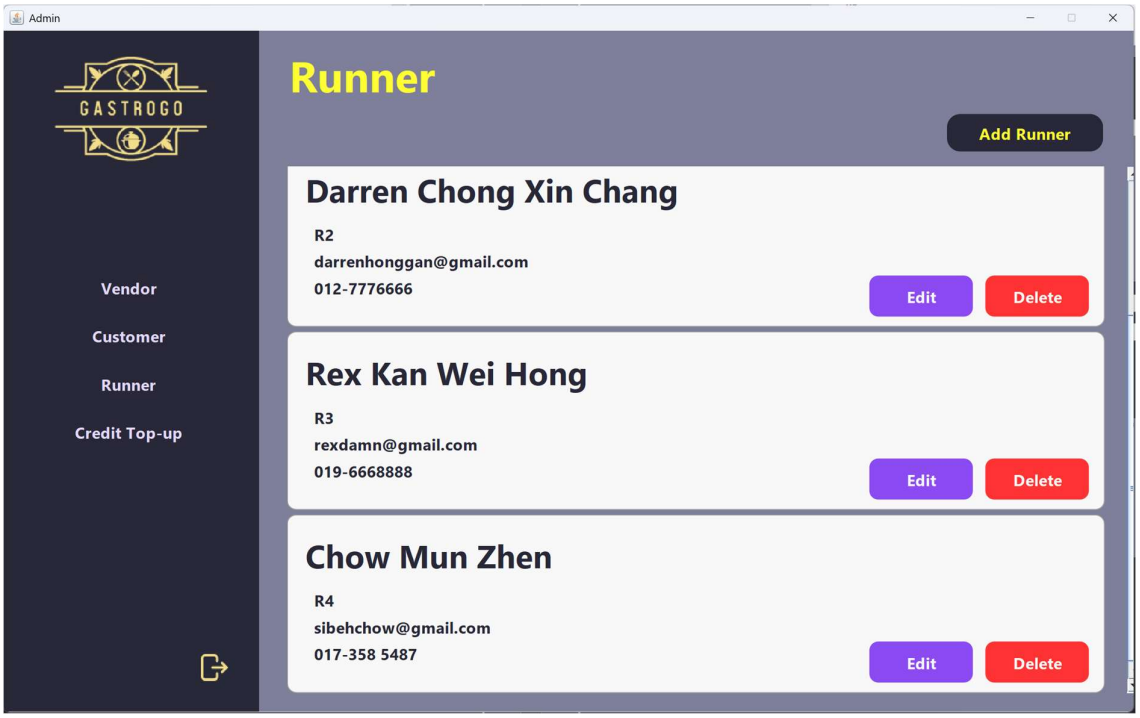
The Runner has the same features with the Customer. The only different is that is one more status selection when add and update runner details. Below are some images of the runner demo.

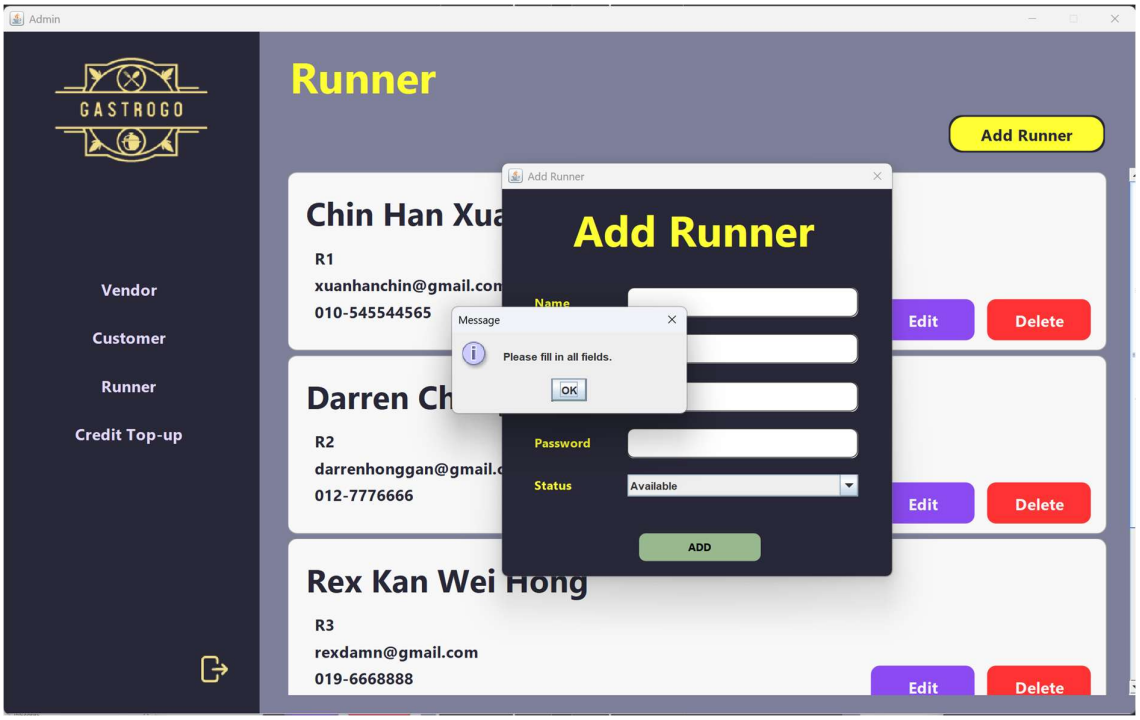


2.4.9 Add Runner

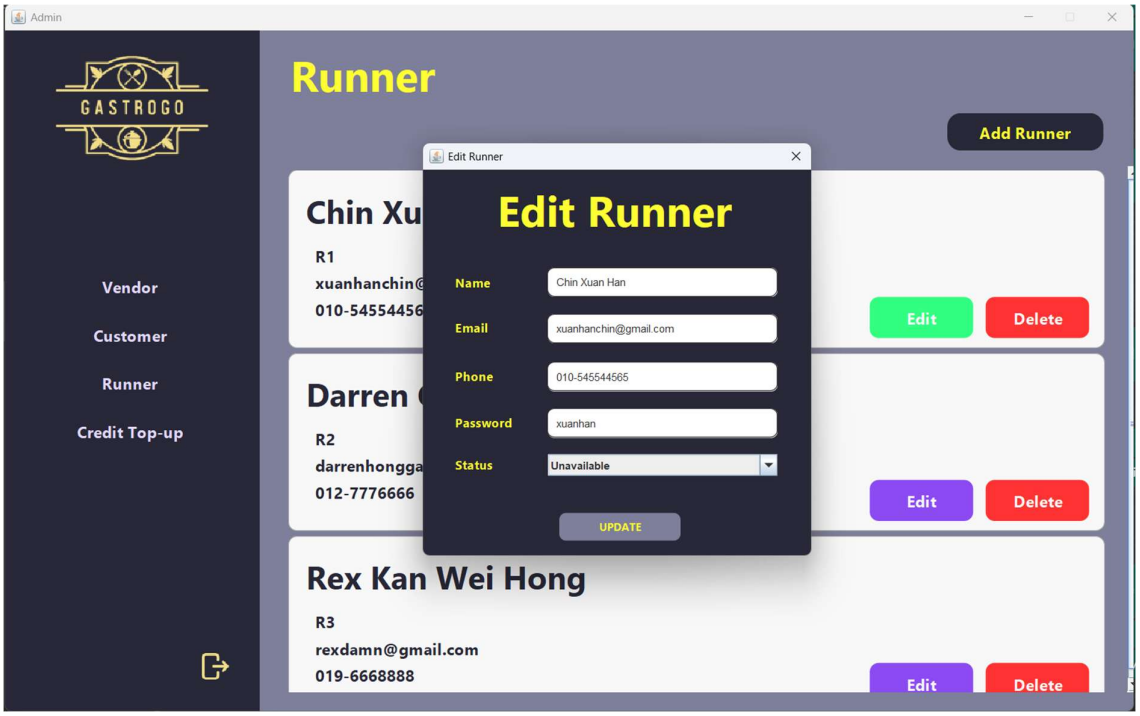


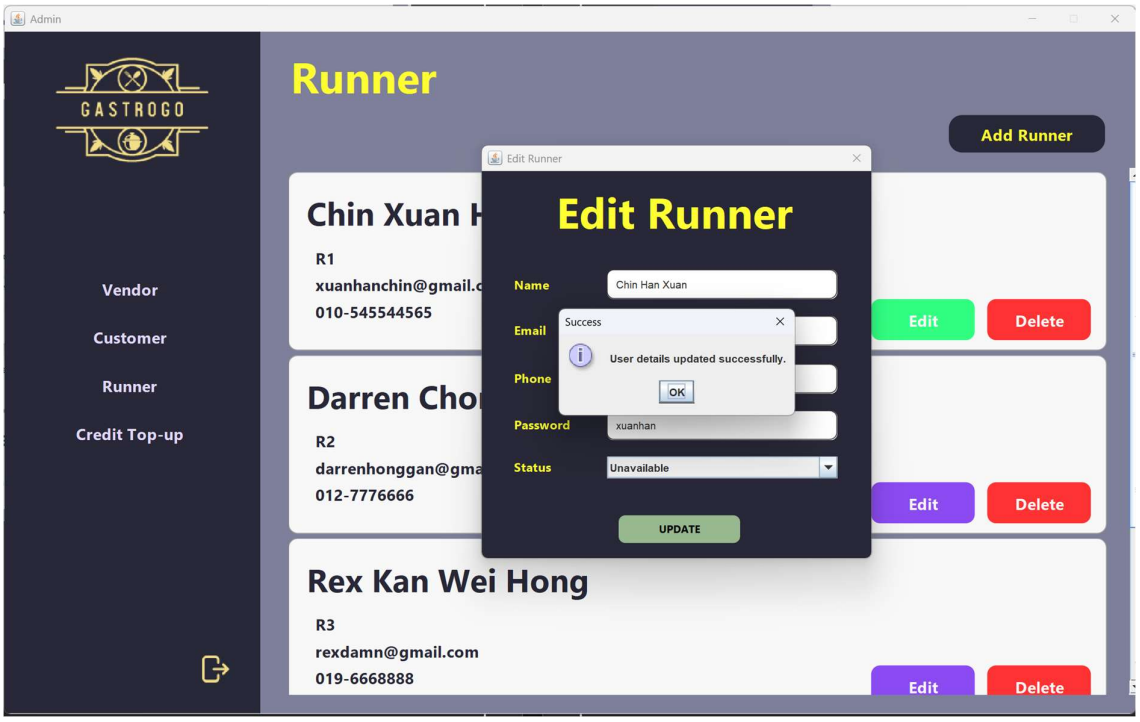

Admin
GASTROGO
Vendor
Customer
Runner
Credit Top-up
Runner
Add Runner
Chin Xuan H
R1
xuanhanchin@gmail.co
010-545544565
Edit
Delete
Darren C
R2
darrenhonggan@gmail.
012-7776666
Edit
Delete
Rex Kan Wei Hong
R3
rexdamn@gmail.com
019-6668888
Edit
Delete
Add Runner
Add Runner
Name
Chow Mun Zhen
@gmail.com
5487
Password
chowdaoyaosi
Status
Available
ADD
Message
Runner added successfully!
OK

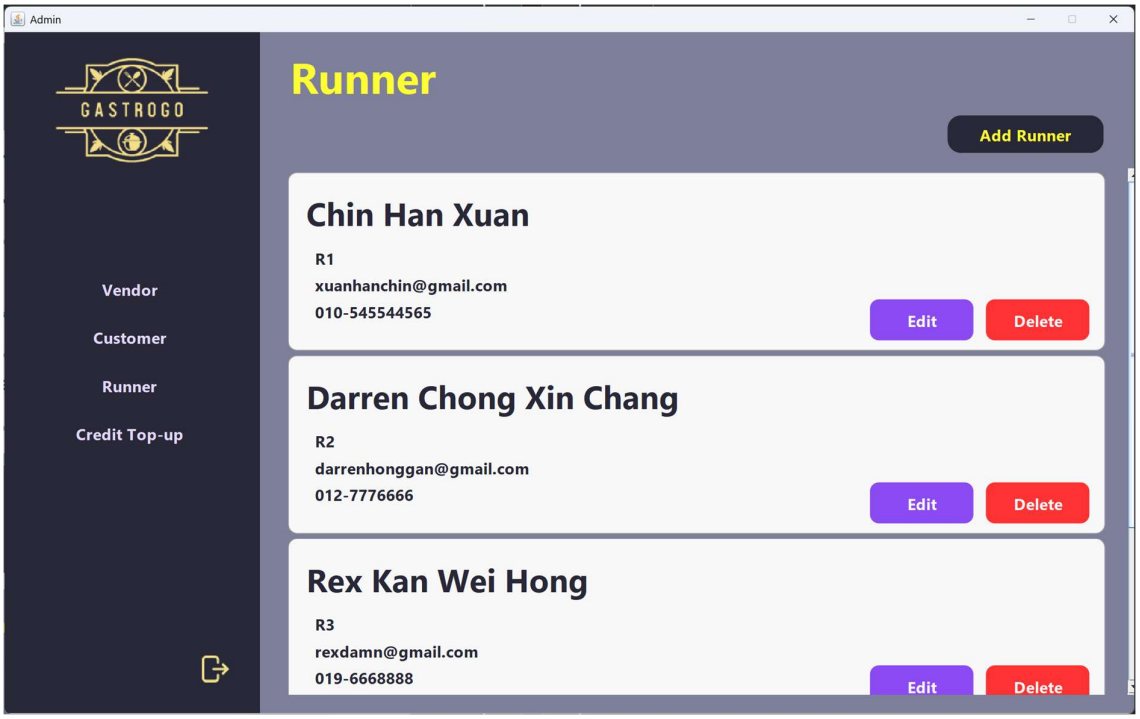

Admin
GASTROGO
Vendor
Customer
Runner
Credit Top-up
Runner
Add Runner
Darren Chong Xin Chang
R2
darrenhonggan@gmail.com
012-7776666
Edit
Delete
Rex Kan Wei Hong
R3
rexdamn@gmail.com
019-6668888
Edit
Delete
Chow Mun Zhen
R4
sibehchow@gmail.com
017-358 5487
Edit
Delete



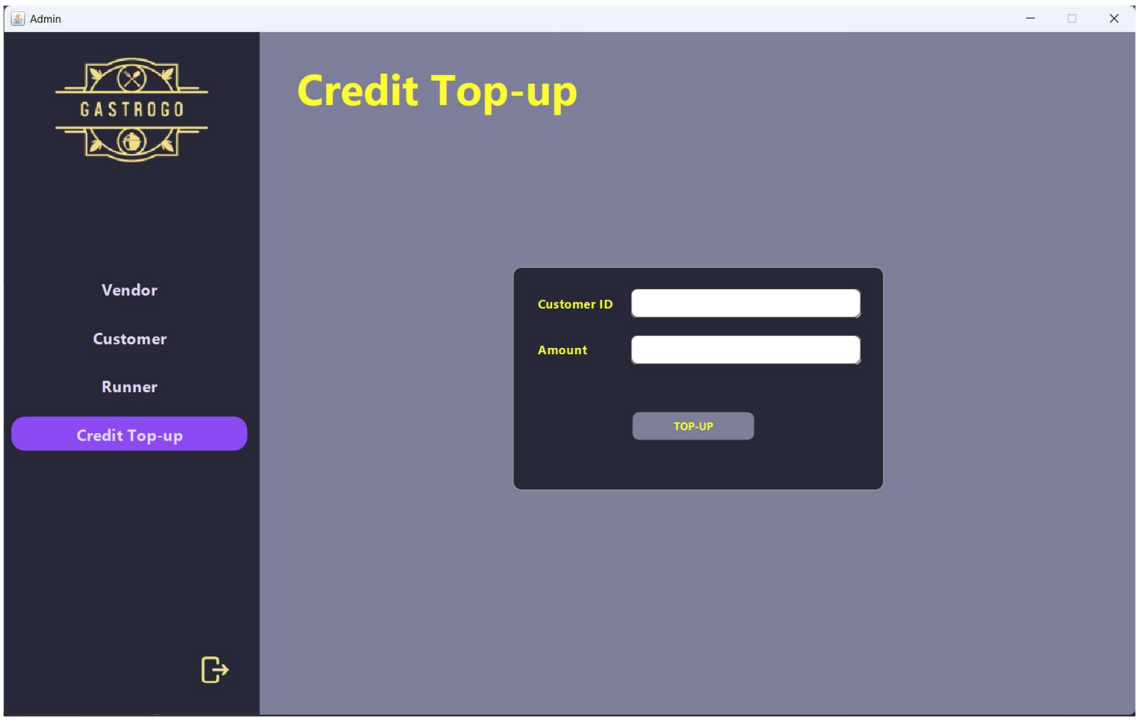
2.4.10 Edit Runner



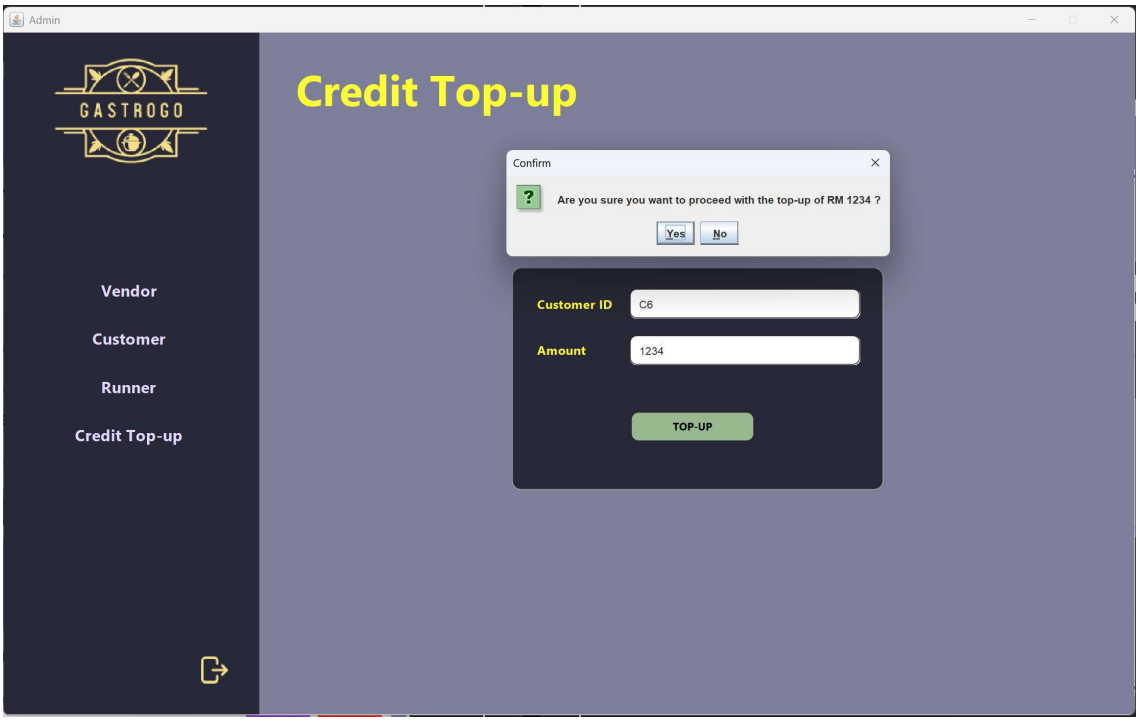

Admin
GASTROGO
Vendor
Customer
Runner
Credit Top-up
Runner
Add Runner
Chin Xuan H
R1
xuanhanchin@gmail.c
010-545544565
Edit
Delete
Darren Cho
R2
darrenhonggan@gma
012-7776666
Edit
Delete
Rex Kan Wei Hong
R3
rexdamn@gmail.com
019-6668888
Edit
Delete
Edit Runner
Edit Runner
Name
Chin Han Xuan
Email
Phone
Password
xuanhan
Status
Unavailable
UPDATE
Success
User details updated successfully.
OK


Admin
GASTROGO
Vendor
Customer
Runner
Credit Top-up
Runner
Add Runner
Chin Han Xuan
R1
xuanhanchin@gmail.com
010-545544565
Edit
Delete
Darren Chong Xin Chang
R2
darrenhonggan@gmail.com
012-7776666
Edit
Delete
Rex Kan Wei Hong
R3
rexdamn@gmail.com
019-6668888
Edit
Delete

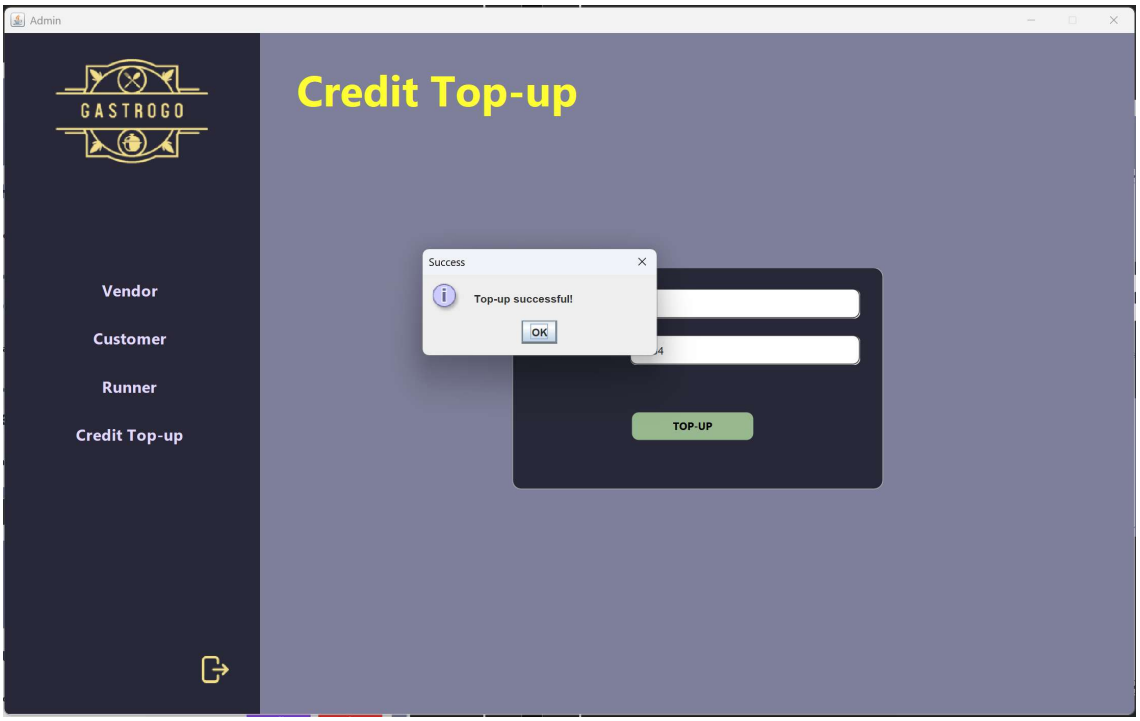
2.4.11 Credit Top-up



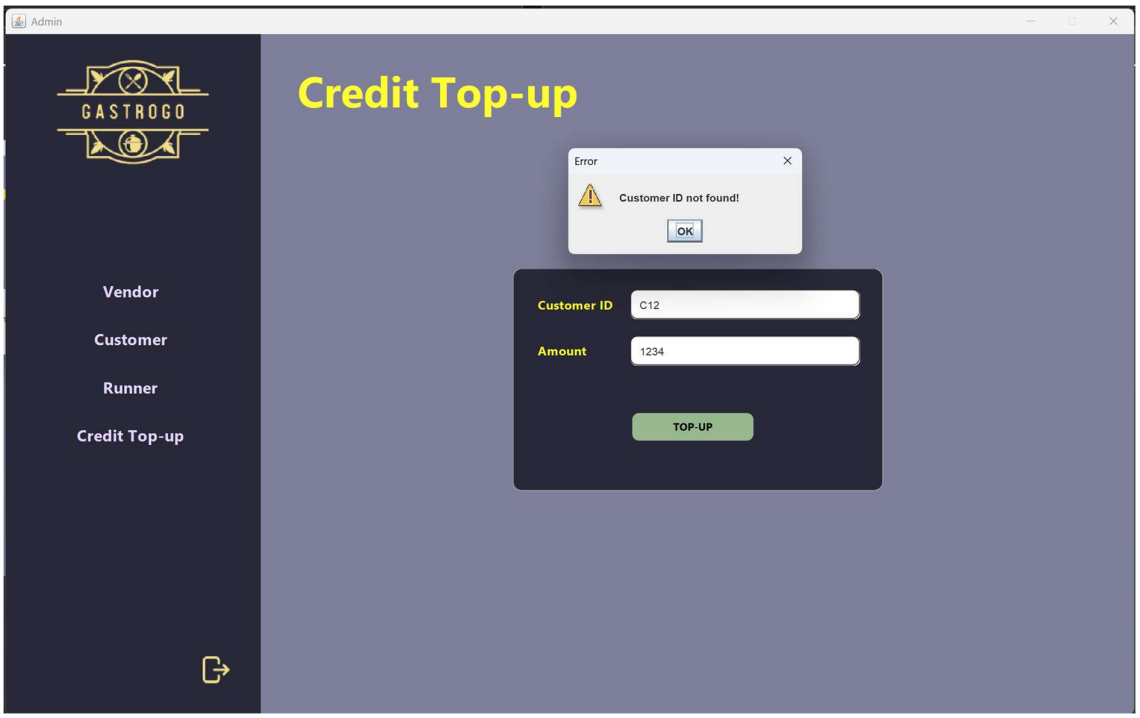
Admin can help customers to top up their accounts credit if the customer want to top up by cash. Admin just needs to enter the existing customer ID and the top up amount, and then click on TOP-UP button to complete the top up process. The minimum top up amount is RM20, if admin enter less than the RM20 amount, there will be a message showed to admin and tell him/her the minimum amount.



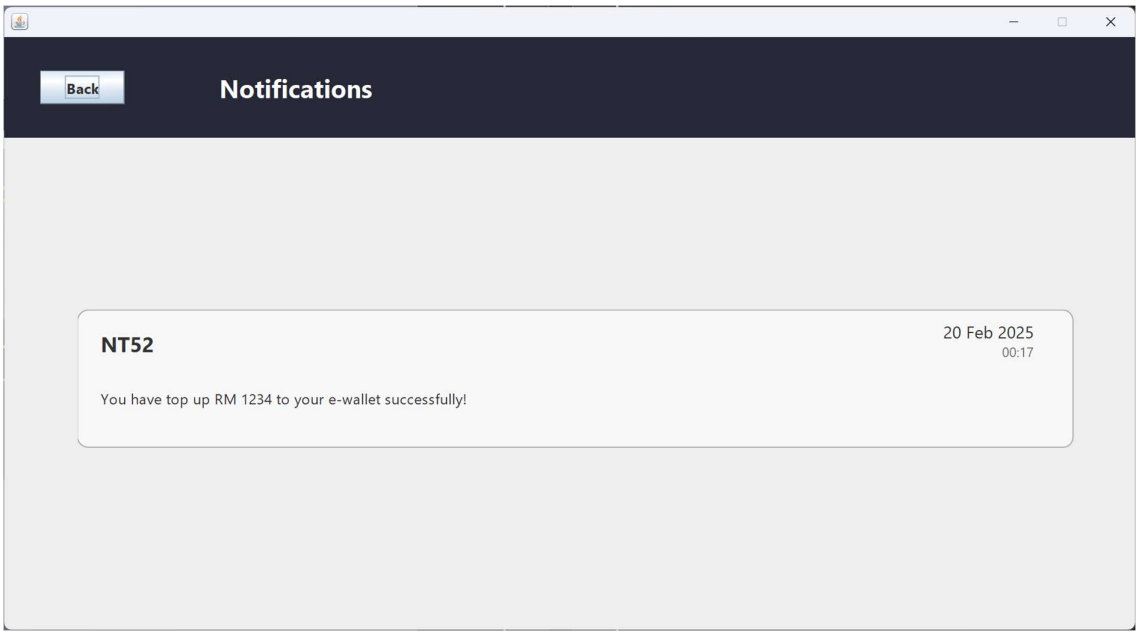
There will be a Double-Confirm message show to admin whether he/she want to top-up the amount for the customer.



If admin clicks on Yes, then the top-up was successful.



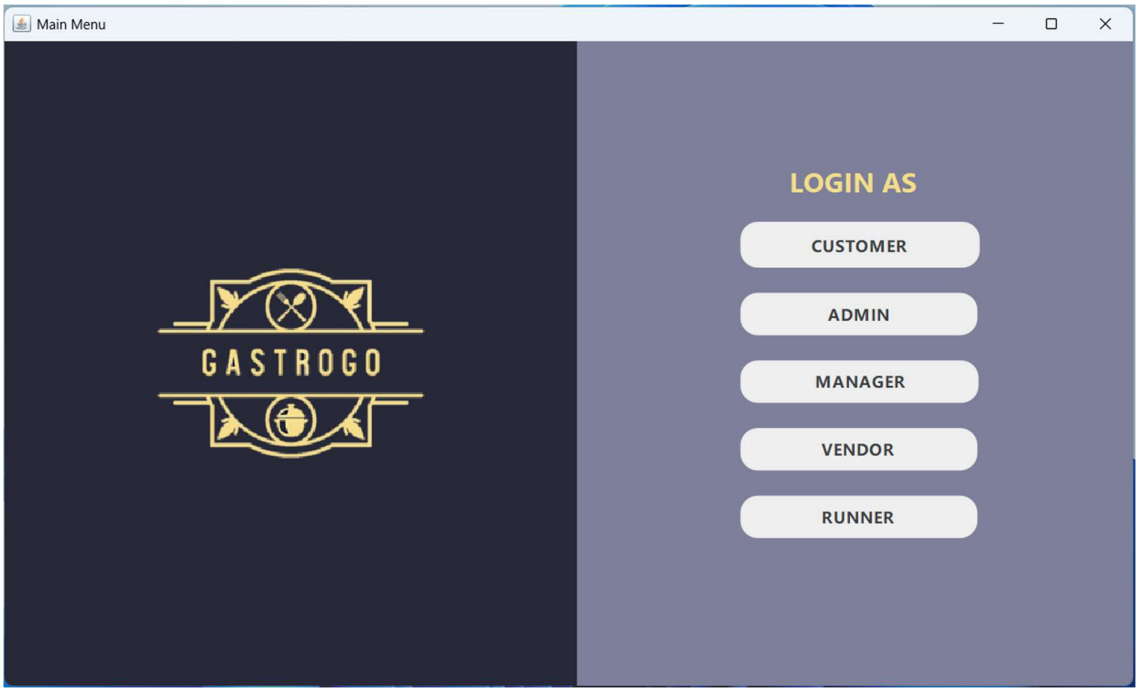
If the customer ID can't be found in database, then an error message will be showed to admin.



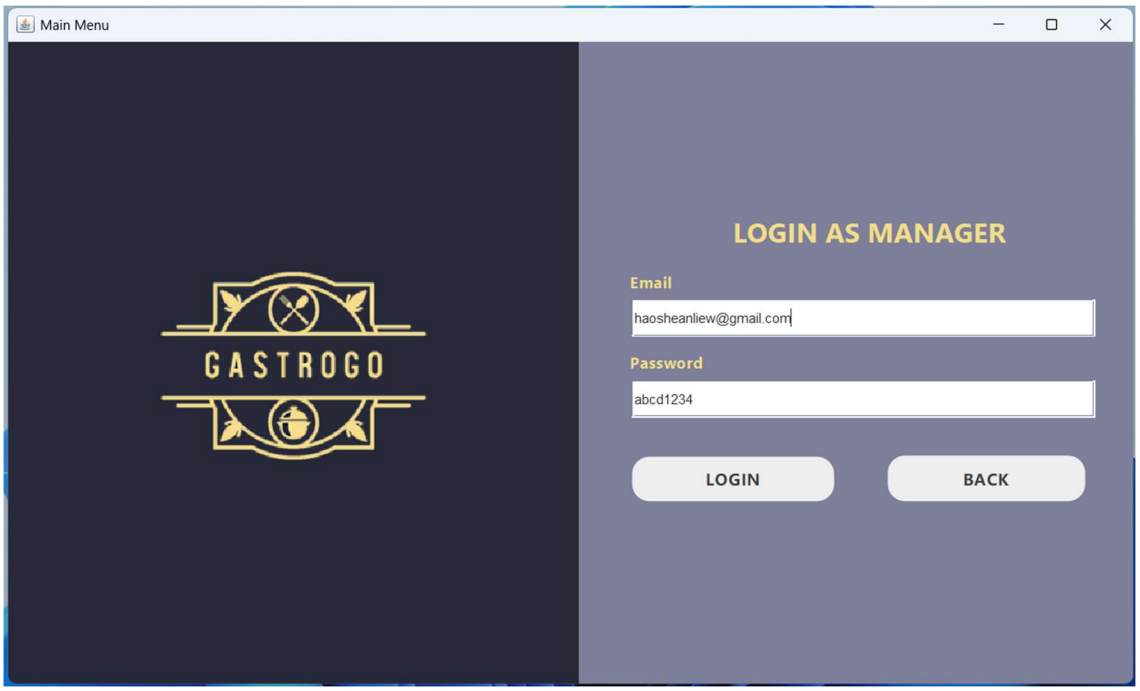
Once the top up process is completed, a notification fill be sent to the customer's account and then can view the transaction later inside the Finance feature.

2.5 Manager

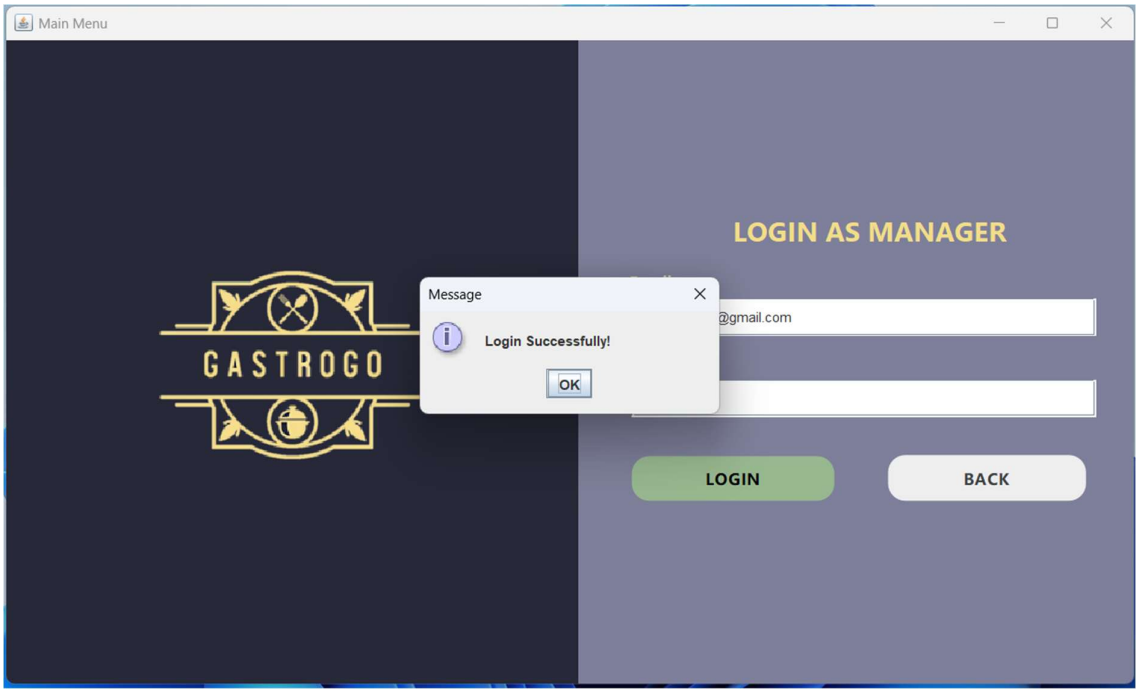
2.5.1 Manager Login



Firstly, the manager will need to choose his/her role by clicking on the manager button.

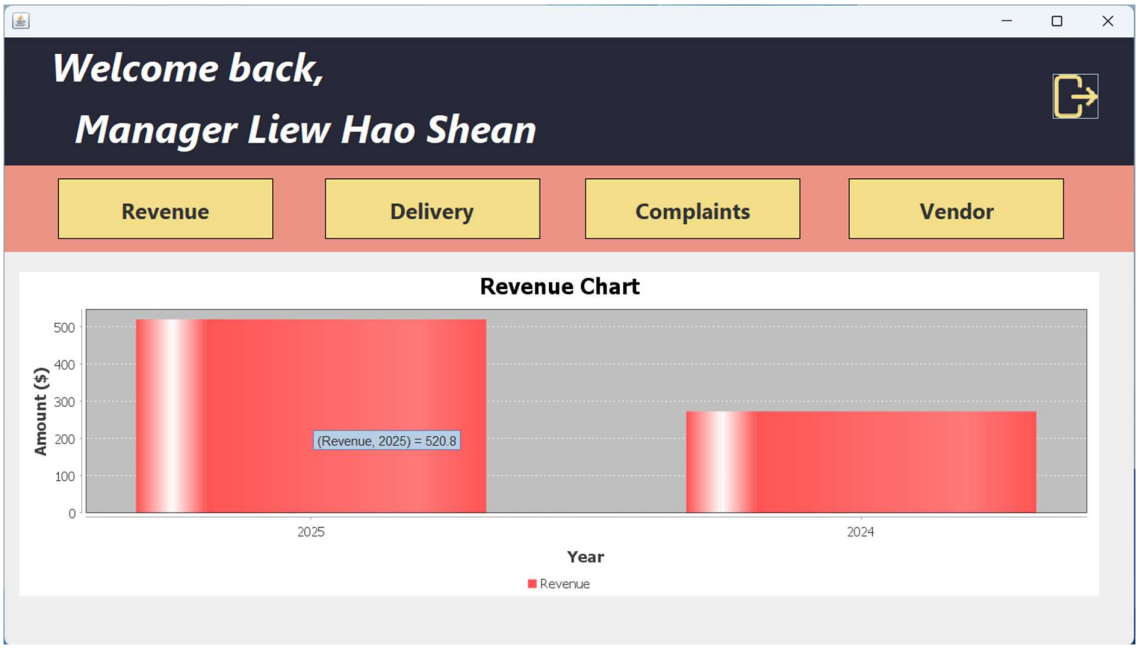


After entering the correct credentials, the manager can log into his/her account by clicking on the login button.

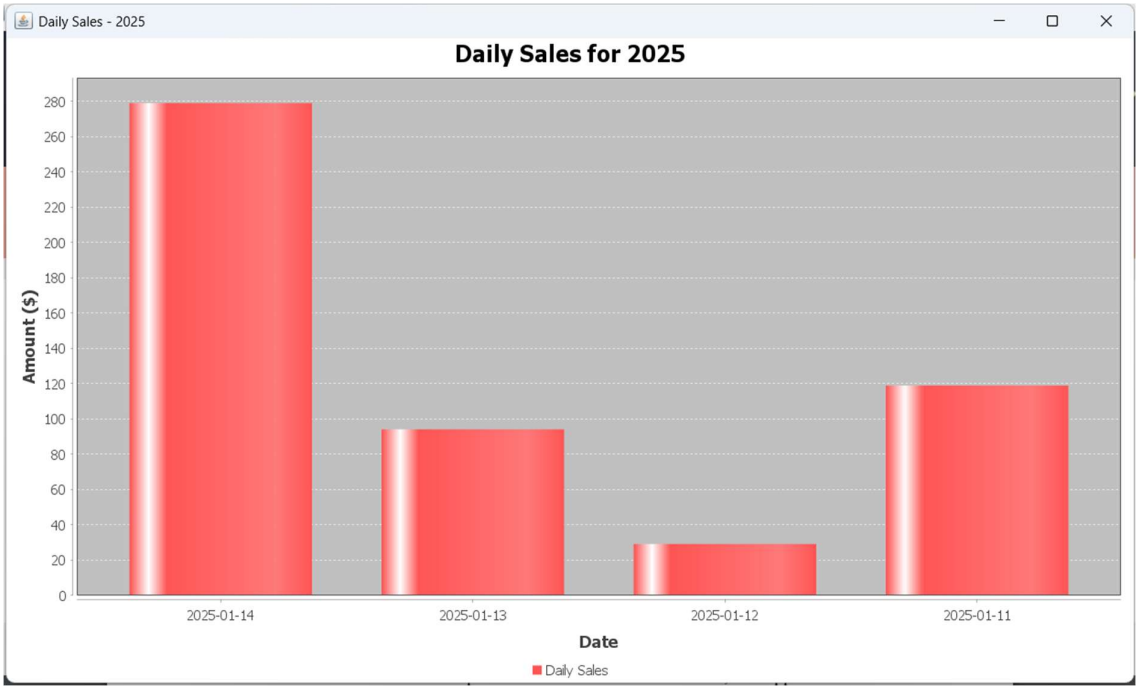


2.5.2 System Revenue Dashboard

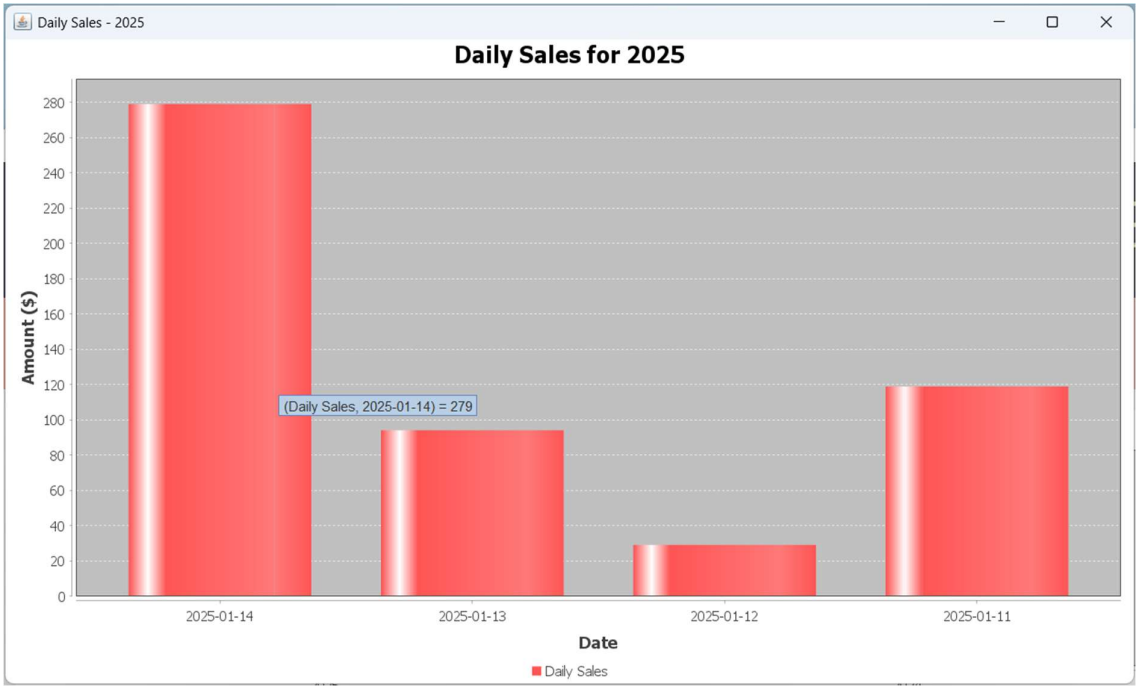
After logging into the manager account, there is a revenue chart showing the yearly revenue of the whole system.



By hovering on the bar chart, it will show the total amount of revenue for each year.

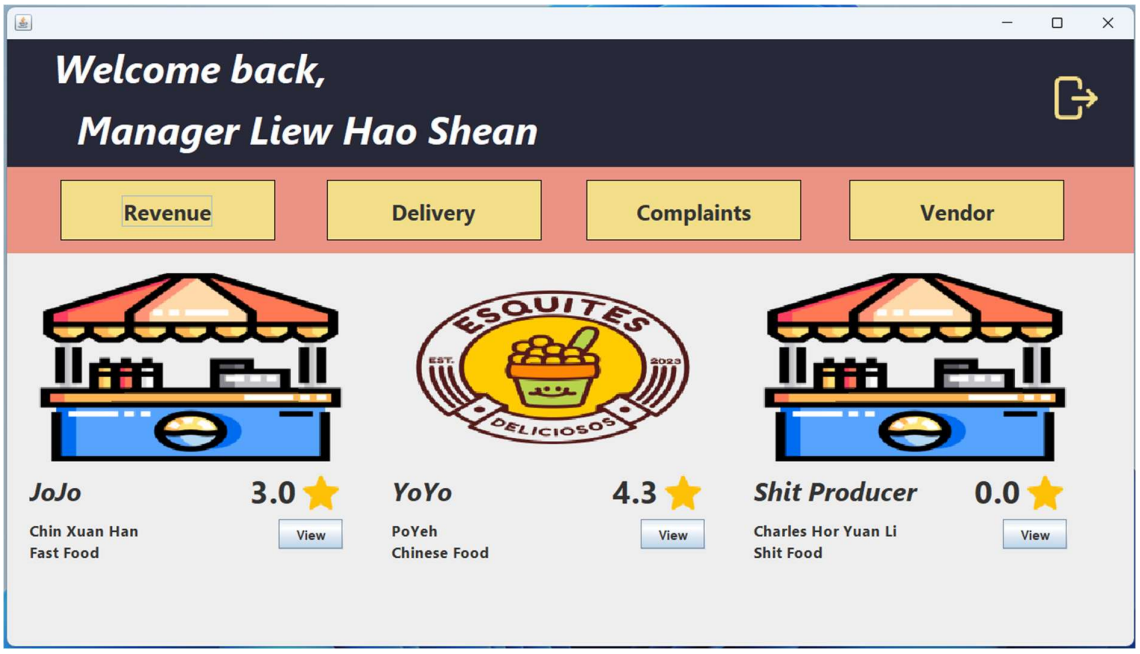


If the manager clicks on the bar chart, our system will show the daily revenue for the selected year.

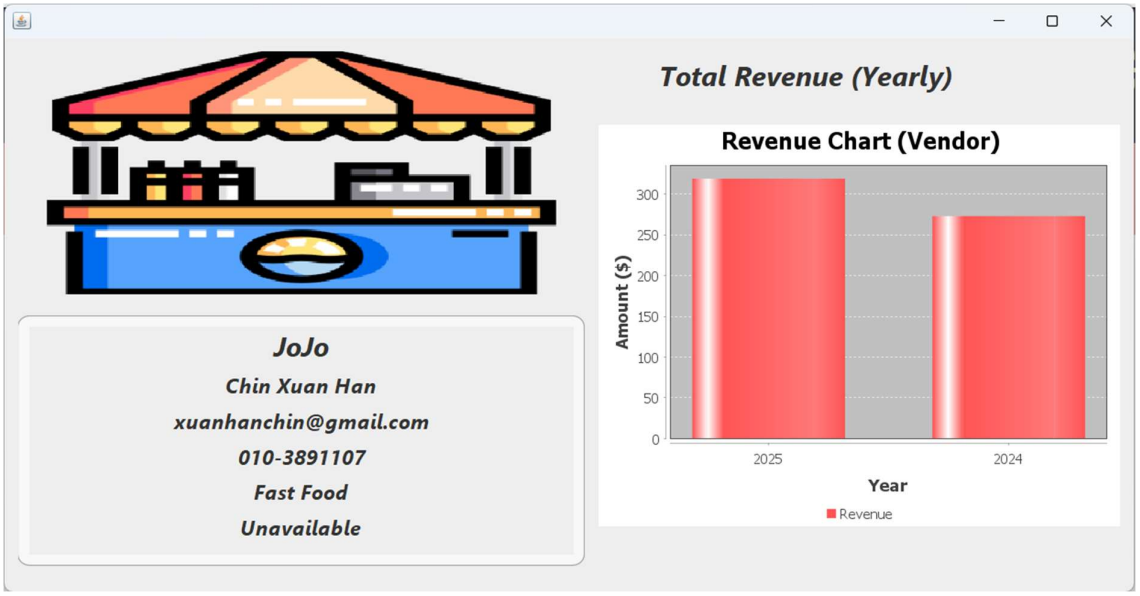


The manager can also view the total amount of daily revenue by hovering on the bar chart.

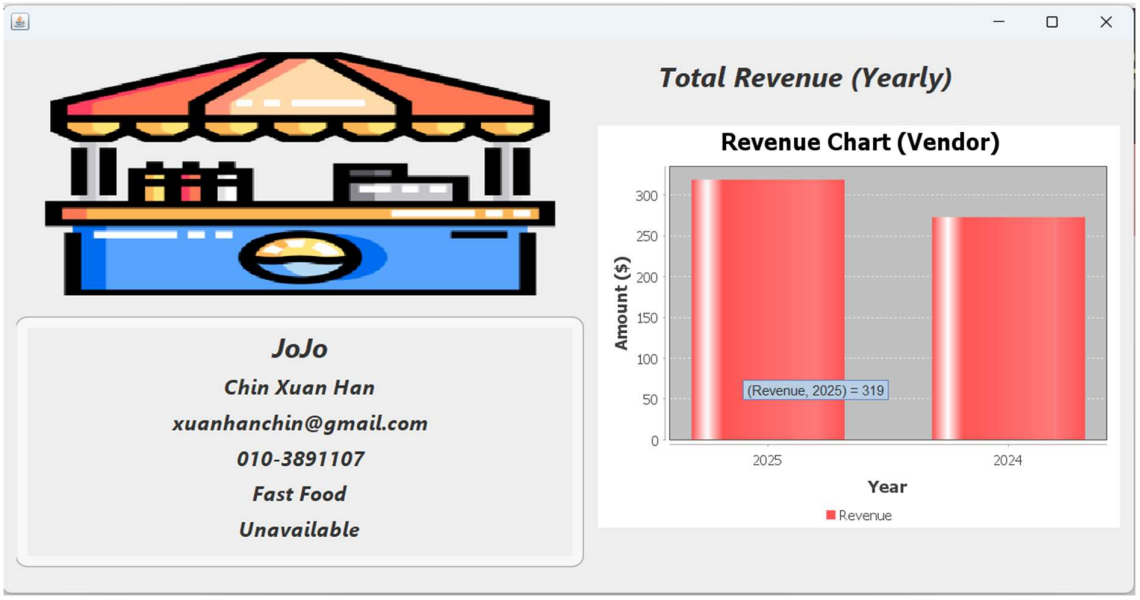
2.5.3 Vendor Revenue Dashboard



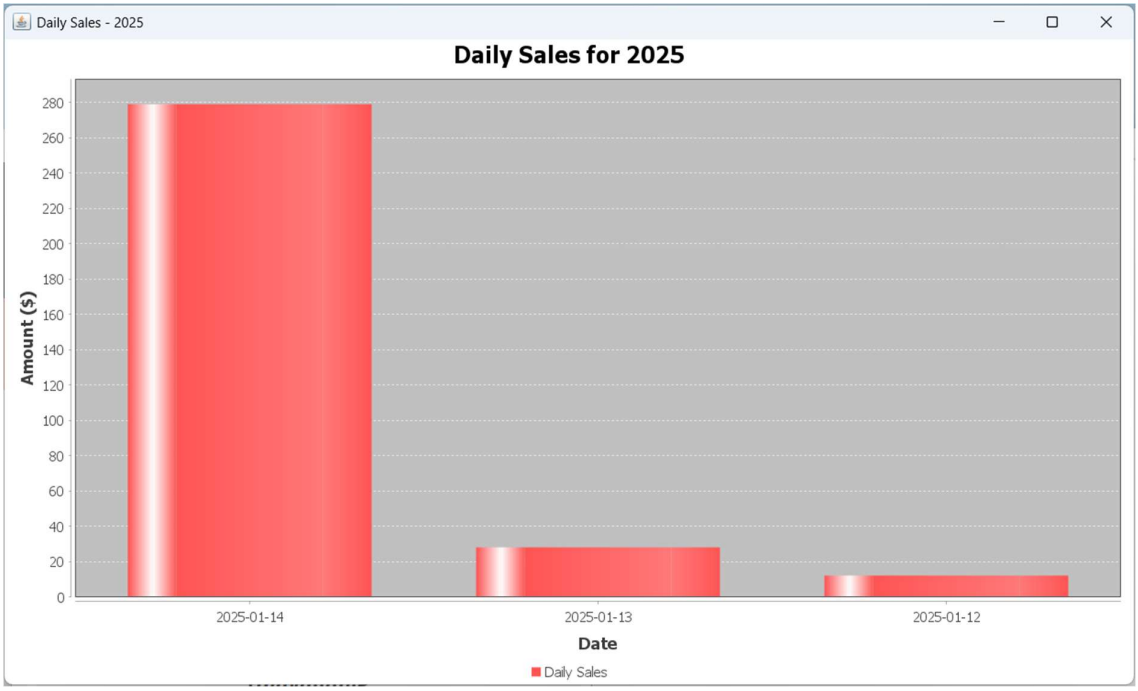
By clicking on the revenue button, the manager can view all of the vendors available in the system, including their information such as the ratings, stall name, vendor name, and food type.



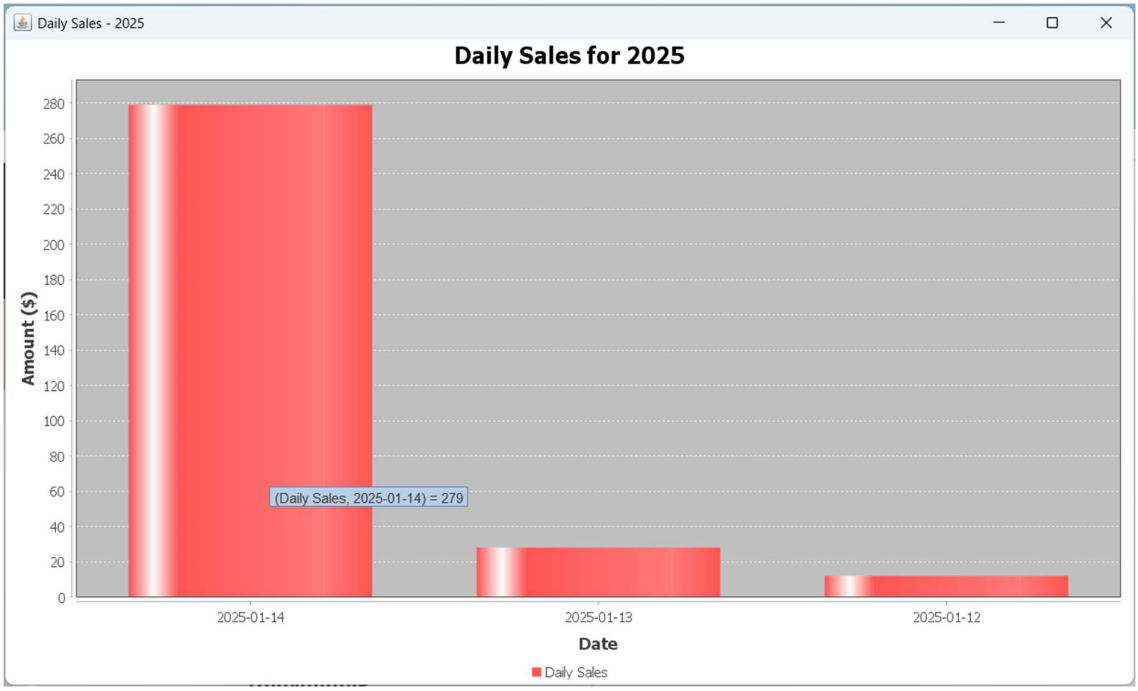
After clicking on the view button, the manager can view all of the details and the revenue of the selected vendor.



By hovering on the bar chart, our system will also show the total amount of revenue for each year.



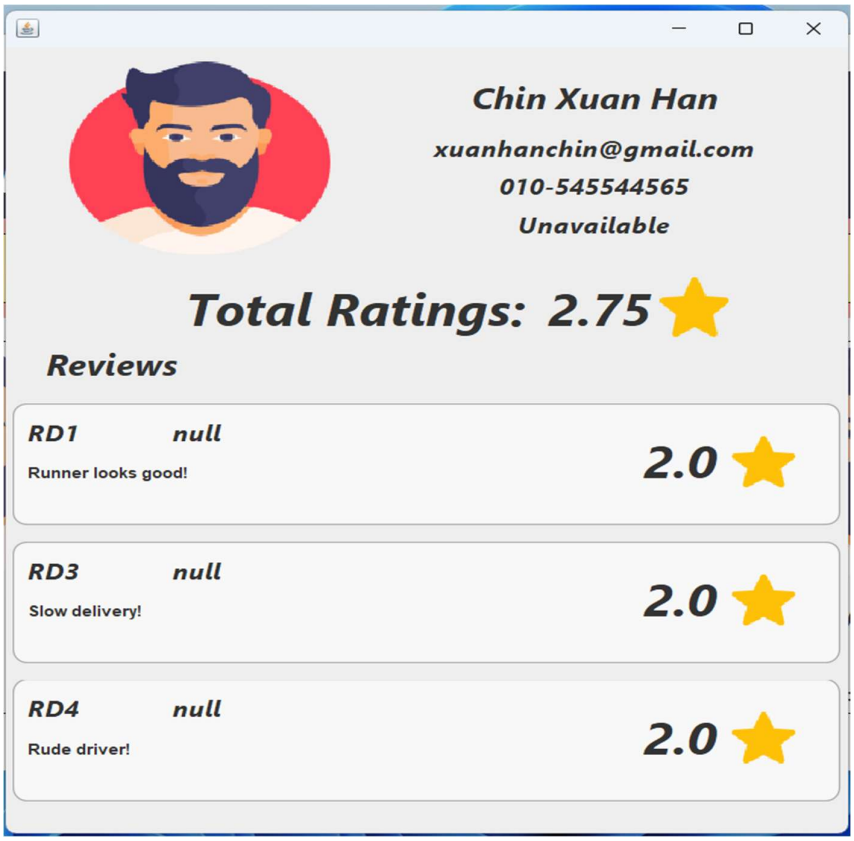
When the manager clicks on the year bar chart, he/she can view the daily revenue of the selected vendor.

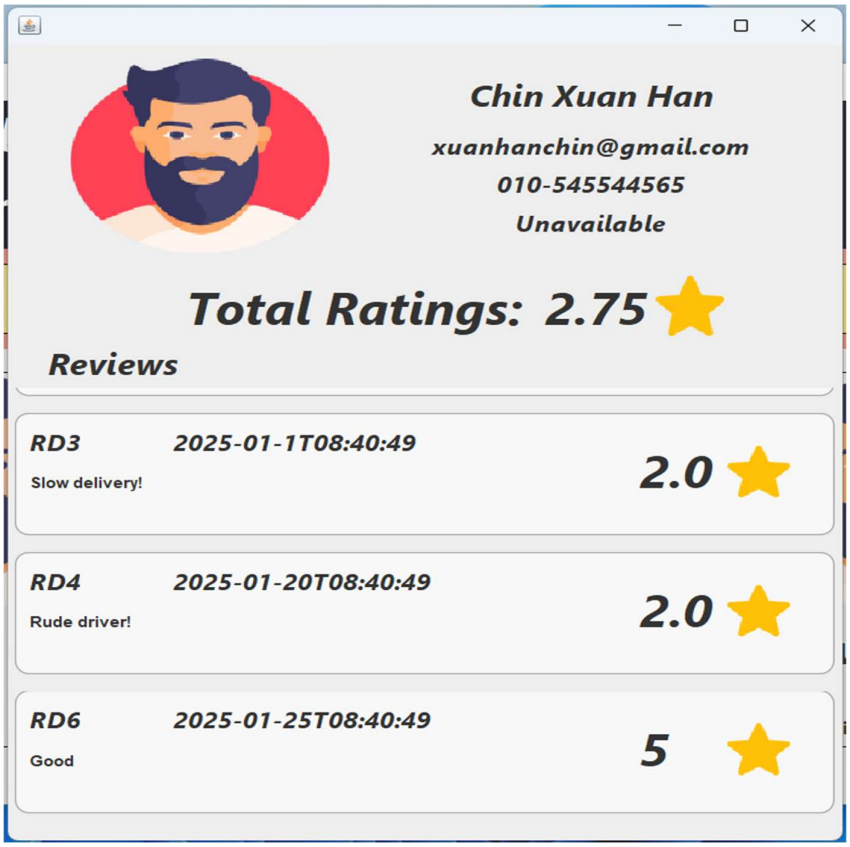


By hovering on the bar chart, the manager to view the total amount of daily revenue.

2.5.4 Runner Dashboard

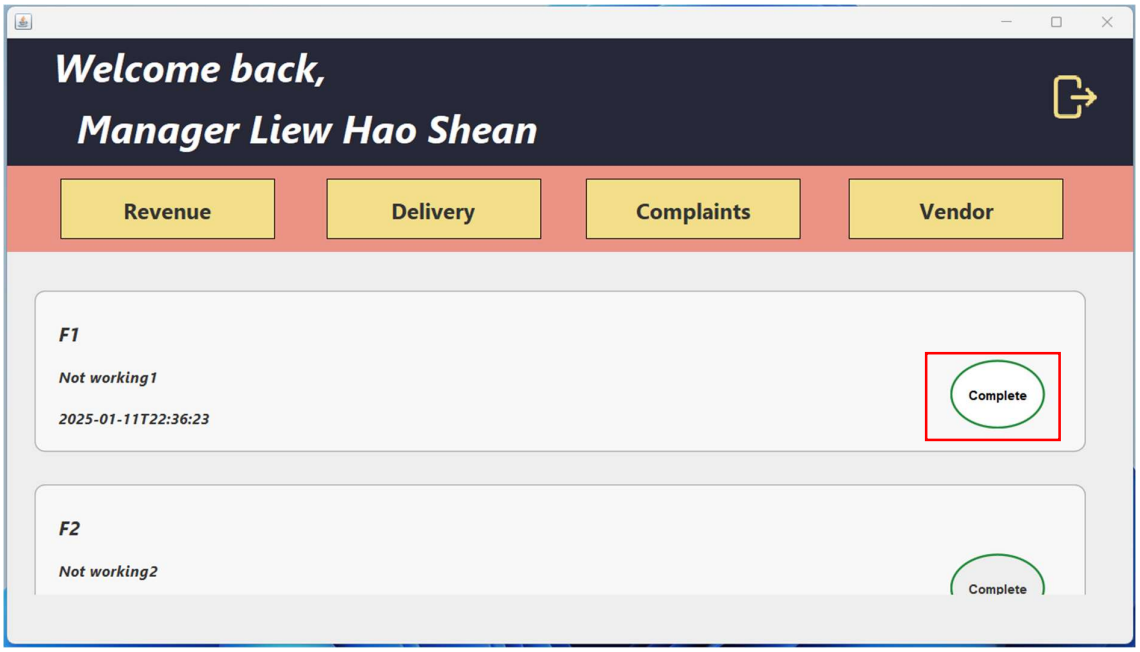
When the manager clicks on the delivery button, he/she can view all of the runners available in the system, including their information such as name, email, phone number and average ratings.



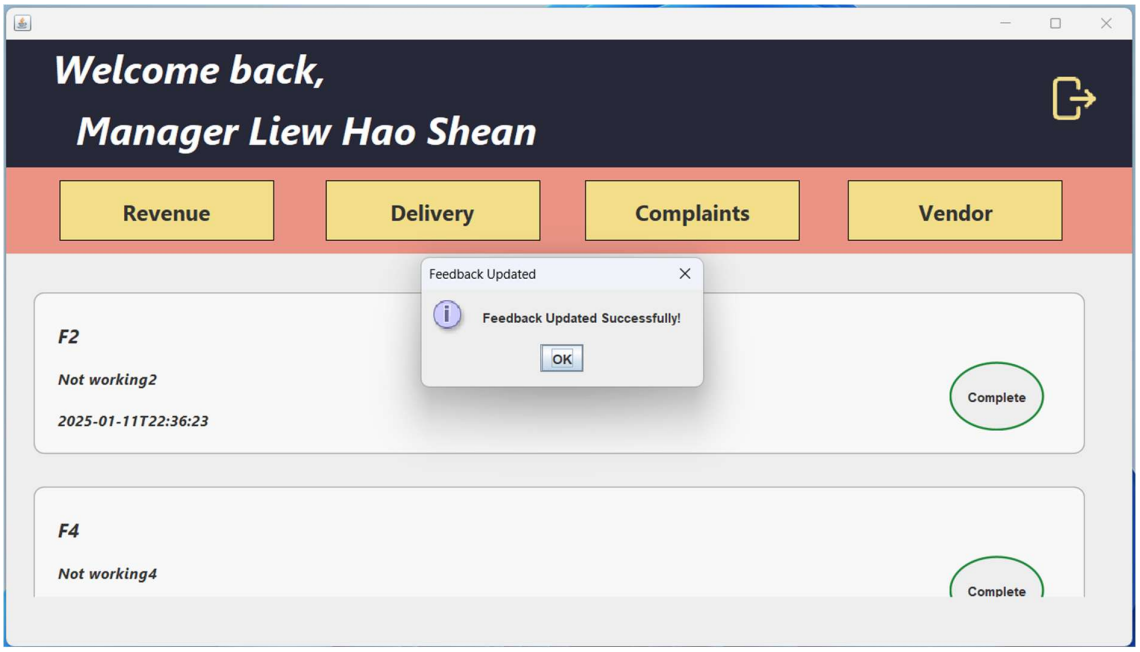


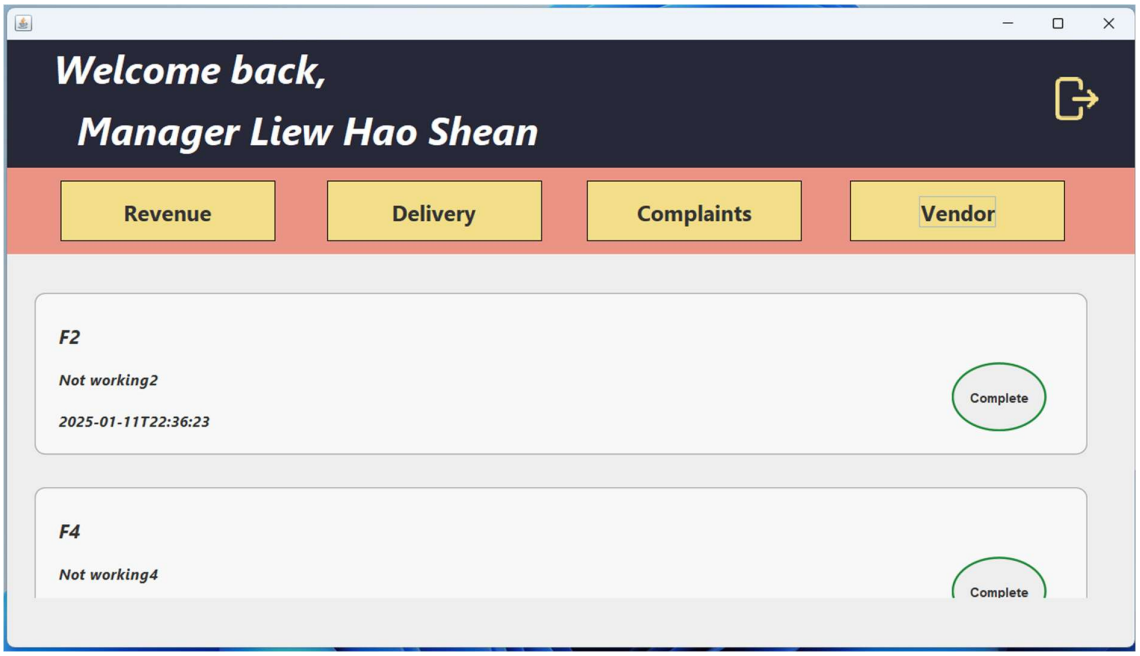
By clicking on the view button, the system will show their detailed personal information and also all of the reviews of the selected runner. Through this feature provided by our system, the vendor can evaluate the performance of the vendor.

2.5.5 Customer Complaints Dashboard



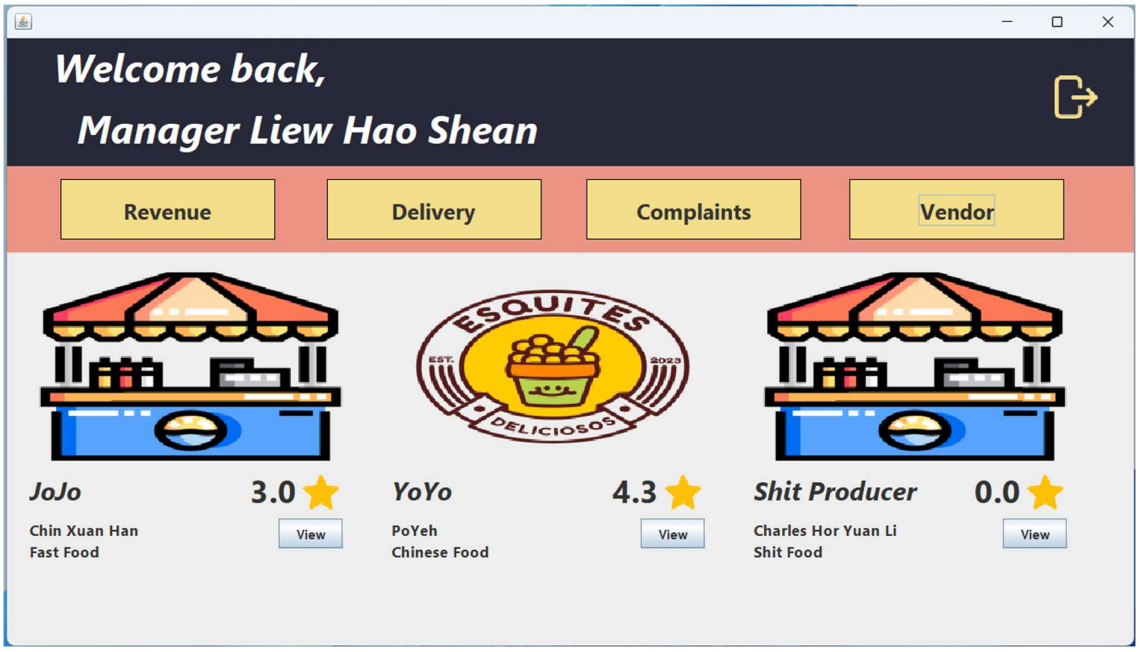
By clicking on the complaints button, the manager can view the complaints/disputes submitted by the user.



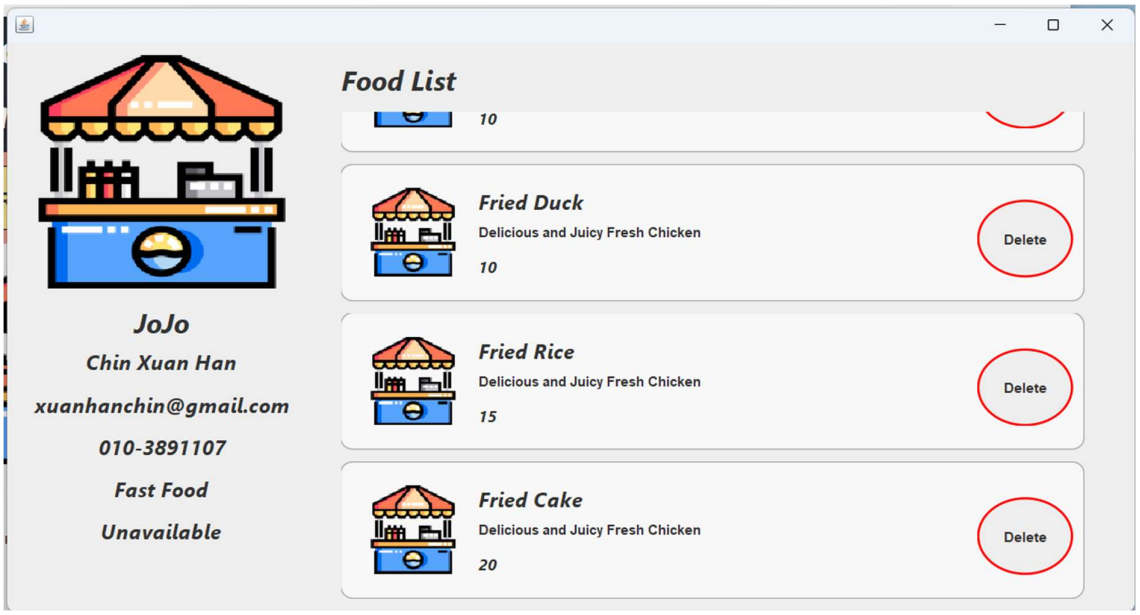


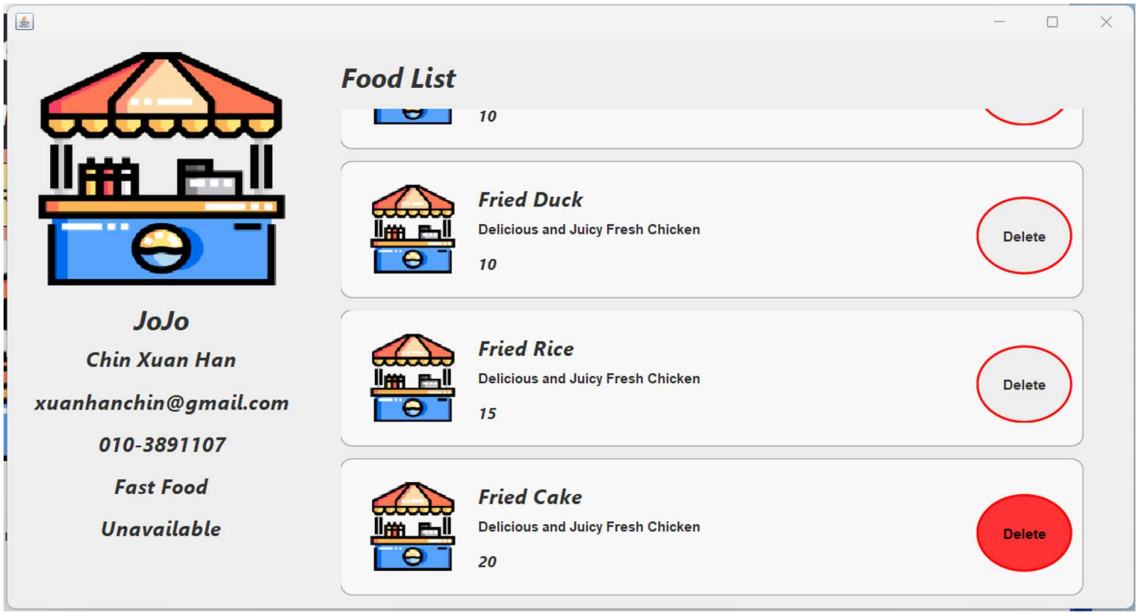
Managers can resolve the complaints by clicking on the complete button, and hence the status of the complaints will be updated.

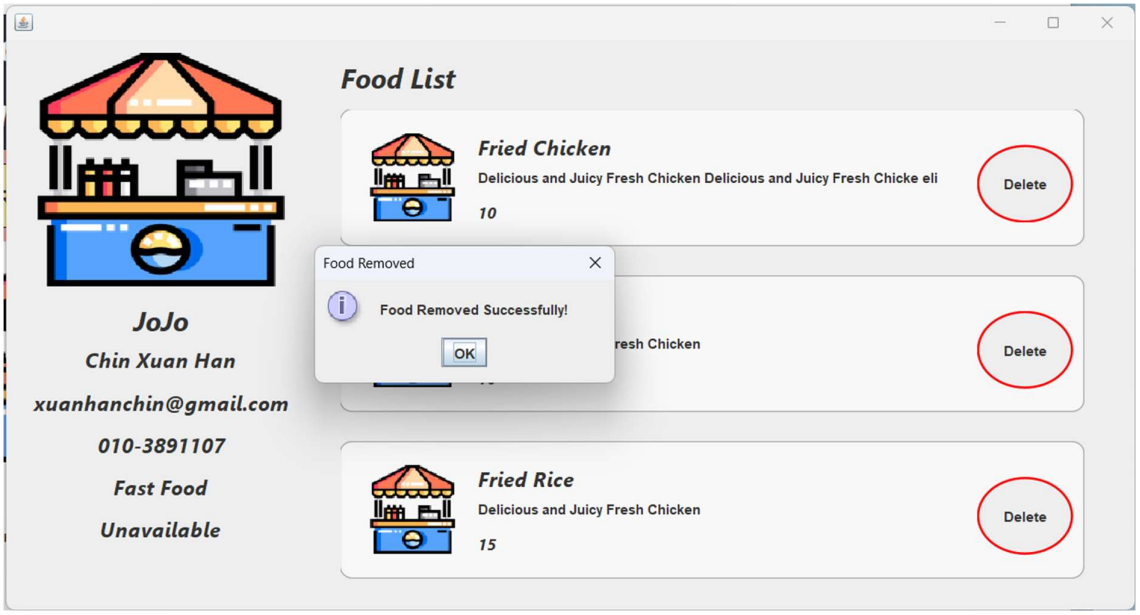
2.5.6 Vendor Item Dashboard



The manager can also remove inappropriate items from the vendor by clicking on the vendor button. All of the vendors in the system are listed out and the manager can choose the vendor by clicking on the view button.







By clicking on the delete button on each listed item, the item will be removed from the system.

3.0 Object-oriented concepts

3.1 Abstraction

3.1.1 Interface

```
public interface interfaceInput {
    List<String> getAllInput();
}
```

This is the interface definition called interfaceInput with getAllInput method, which returns string list.

```
public List<String> getCurrentPanelInput() {
    if (currentPanel instanceof tngPanel) {
        return ((interfaceInput) currentPanel).getAllInput();
    } else if (currentPanel instanceof bankPanel) {
        return ((interfaceInput) currentPanel).getAllInput();
    } else if (currentPanel instanceof cardPanel){
        return ((interfaceInput) currentPanel).getAllInput();
    }
    return null;
}
```

This is the method that polymorphism used to get all the input from the particular panel. When the currentPanel is matched with the instance of specific panel classes such as tngPanel, bankPanel, and cardPanel, it will cast currentPanel to interfaceInput and call the method getAllInput in the interface to get the return list value.

```
public class tngPanel extends javax.swing.JPanel implements interfaceInput{
    customer_backend backend = new customer_backend();
    private String phoneNum;

    /**
     * Creates new form tngPanel
     */

    public tngPanel() {
    }

    public tngPanel(String phoneNum,String option) {
        this.phoneNum = phoneNum;
        initComponents();
        if (option.equals("tng")){
            jLabel1.setIcon(backend.scale.processImage("src\\main\\java\\image_repository\\tng.png", 200, 200));
            jLabel3.setText("Touch'n Go");
        }else{
            jLabel1.setIcon(backend.scale.processImage("src\\main\\java\\image_repository\\grab.png", 240, 200));
            jLabel3.setText("Grab Pay");
        }
        jTextField1.setText(phoneNum);
    }

    public List<String>  getAllInput() {
        List<String> allInput = new ArrayList<>();
        allInput.add(jTextField1.getText());
        allInput.add(jTextField2.getText());
        allInput.add(jLabel3.getText());
        return allInput;
    }
```

```java
public class cardPanel extends javax.swing.JPanel implements interfaceInput{
    customer_backend backend = new customer_backend();
    private String paymentMethod;


    public cardPanel() {
        initComponents();
        jLabel1.setIcon(backend.scale.processImage("src\\main\\java\\image_repository\\credit.png", 200, 200));
        paymentMethod = (String) jComboBox1.getSelectedItem();
    }

    public List<String>  getAllInput() {
        List<String> allInput = new ArrayList<>();
        allInput.add(jTextField1.getText());
        allInput.add(jTextField2.getText());
        allInput.add(paymentMethod);
        return allInput;
    }
```

```java
public class bankPanel extends javax.swing.JPanel implements interfaceInput{
    customer_backend backend = new customer_backend();
    private String paymentMethod;


    public bankPanel() {
        initComponents();
        jLabel1.setIcon(backend.scale.processImage("src\\main\\java\\image_repository\\bank.png", 200, 200));
        paymentMethod = (String) jComboBox1.getSelectedItem();
    }


    public List<String> getAllInput() {
        List<String> allInput = new ArrayList<>();
        allInput.add(jTextField1.getText());
        allInput.add(new String(jPasswordField1.getPassword()));
        allInput.add(paymentMethod);
        return allInput;
    }
```

This is the all the classes that extend the JPanel and implements interfaceInput interface. These classes will get the input from text field that entered by the user, stored into the allInput list, and then returned the list to the classes that called the interface.

3.2 Encapsulation

3.2.1 Classes creation and constructor

```java
public class Order {
    private String orderID;
    private String customerID;
    private String deliveryID;
    private String vendorID;
    private String orderReviewID;
    private String orderType;
    private String orderTypeDetails;
    private String datetime;
    private String totalAmount;
    private String status;
    private String orderFile = "src\\main\\java\\repository\\order.txt";

    public Order(){};

    public Order(String orderID, String customerID, String deliveryID, String vendorID, String orderReviewID, String orderType,
            String orderTypeDetails, String datetime, String totalAmount, String status) {
        this.orderID = orderID;
        this.customerID = customerID;
        this.deliveryID = deliveryID;
        this.vendorID = vendorID;
        this.orderReviewID = orderReviewID;
        this.orderType = orderType;
        this.orderTypeDetails = orderTypeDetails;
        this.datetime = datetime;
        this.totalAmount = totalAmount;
        this.status = status;
    }
```

Z

```java
public String getOrderID() {
    return orderID;
}

public void setOrderID(String orderID) {
    this.orderID = orderID;
}

public String getCustomerID() {
    return customerID;
}

public void setCustomerID(String customerID) {
    this.customerID = customerID;
}

public String getDeliveryID() {
    return deliveryID;
}

public void setDeliveryID(String deliveryID) {
    this.deliveryID = deliveryID;
}

public String getVendorID() {
    return vendorID;
}

public void setVendorID(String vendorID) {
    this.vendorID = vendorID;
}

public String getOrderReviewID() {
    return orderReviewID;
}
```

```java
public String getOrderType() {
    return orderType;
}

public void setOrderType(String orderType) {
    this.orderType = orderType;
}

public String getOrderTypeDetails() {
    return orderTypeDetails;
}

public void setOrderTypeDetails(String orderTypeDetails) {
    this.orderTypeDetails = orderTypeDetails;
}

public String getDatetime() {
    return datetime;
}

public void setDatetime(String datetime) {
    this.datetime = datetime;
}

public String getTotalAmount() {
    return totalAmount;
}

public void setTotalAmount(String totalAmount) {
    this.totalAmount = totalAmount;
}

public String getStatus() {
    return status;
}

public void setStatus(String status) {
    this.status = status;
}
```

This is the order class that has attributes such as order id, customer id, delivery id, vendor id, order review id, order type, order type details, datetime, total amount, status, and order file path. Additionally, all the variables are defined as private and empty constructor and constructor with parameters to initialize all the variables. The class provides getter and setter methods to access and modify the value for each attributes.

3.2.3 Object Creation

```
public List readOrder(String filePath){
    List<Order> orders = new ArrayList<>();
    try{
        FileReader fr = new FileReader(filePath);
        BufferedReader br = new BufferedReader(fr);
        br.readLine();
        String line;
        while ((line = br.readLine()) != null) {
            String[] fields = line.split(",");
            String orderID= fields[0].trim();
            String customerID= fields[1].trim();
            String deliveryID= fields[2].trim();
            String vendorID= fields[3].trim();
            String orderReviewID= fields[4].trim();
            String orderType= fields[5].trim();
            String orderTypeDetails= fields[6].trim();
            String datetime= fields[7].trim();
            String totalAmount= fields[8].trim();
            String status= fields[9].trim();
            orders.add(new Order(orderID,customerID,deliveryID,vendorID,orderReviewID,orderType,orderTypeDetails,datetime,totalAmount,statu
        }
    }catch(IOException e){
        e.printStackTrace();
    }
    return orders;
}
```

This is the readOrder method that reads the order data from the file and returns Order object list. It reads the file using the file reader and wrapped in buffered reader. The first line will be skipped as there is header in the text file. The commas are split into array and extract the order attributes such as orderID, customerID, deliveryID, vendorID, orderReviewID, orderType, orderTypeDetails, datetime, totalAmount, and status. All the attributes will be created an Order object and store into an Order object list.

3.2.3 Getter

```
public Map<Object, Object> getOrderByOrderID(String orderID){
    List<managefile.Order> orders = read.readOrder(orderFile);
    List<managefile.OrderItems> orderItems = read.readOrderItems(orderItemFile);
    List<managefile.Food> foodItems = read.readFood(foodFile);

    List<managefile.Order> allOrders = new ArrayList<>();
    List<managefile.OrderItems> allOrderItems = new ArrayList<>();
    List<managefile.Food> matchingFoodItems = new ArrayList<>();
    for (managefile.Order order :orders){
        for (managefile.OrderItems orderitem:orderItems){
            for (managefile.Food foodItem : foodItems) {
                if(orderID.equals(order.getOrderID())){
                    if (order.getOrderID().equals(orderitem.getOrderID())){
                        if (orderitem.getFoodID().equals(foodItem.getId())){
                            allOrders.add(order);
                            allOrderItems.add(orderitem);
                            matchingFoodItems.add(foodItem);
                        }
                    }
                }
            }
        }
    }
    Map<Object, Object> result = new HashMap<>();
    result.put("orders", allOrders);
    result.put("ordersItems", allOrderItems);
    result.put("foodItems", matchingFoodItems);
    return result;
}
```

This is the method to get order list by filtering order id and returns result object hashmap. It reads the order file, order item file, and food file. By iterating these three object array list, it will return the list into the matched list when comparing the order.getOrderID with orderitem.getOrderID and orderitem.getFoodID with foodItem.getID.

```
private void getDetails(){
    Map<Object, Object> allOrders = backend.getOrderByOrderID(orderID);
    List<managefile.Order> orders = (List<managefile.Order>) allOrders.get("orders");
    List<managefile.OrderItems> orderItems = (List<managefile.OrderItems>) allOrders.get("ordersItems");
    List<managefile.Food> foodItems = (List<managefile.Food>) allOrders.get("foodItems");
    List<managefile.Vendor> vendors = backend.getVendors();
    Vendor vendorDetail = new Vendor();
    for (Vendor vendor : vendors) {
        if (vendor.getId().equals(orders.getFirst().getVendorID())){
            vendorDetail = vendor;
        }
    }
    List<managefile.VendorReview> vendorReviews = backend.getOrderReviewByID(orders.getFirst().getOrderReviewID());

    boolean reviewGiven = false;
    if ((vendorReviews.getFirst().getRating().equals("") || vendorReviews.getFirst().getRating().equals("null")) &&
        (vendorReviews.getFirst().getComments().equals("") || vendorReviews.getFirst().getComments().equals("null"))){
        reviewGiven = true;
    }

    naviButton.setText("View Vendor ("+vendorDetail.getStallName()+")");
    naviButton.addActionListener(e->{
        navigateFood(orders.getFirst());
    });

    JPanel orderPanel = new JPanel(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();

    gbc.insets = new Insets(5, 5, 5, 5);
    gbc.gridx = 0;
    gbc.gridy = 0;

    for (int i = orders.size()-1; i >= 0; i--) {
        managefile.OrderItems innerOrderItem = orderItems.get(i);
        managefile.Food innerFoodItem = foodItems.get(i);
        JPanel panel = addOrderPanel(innerOrderItem,innerFoodItem);
        orderPanel.add(panel, gbc);
        gbc.gridx++;
        if (gbc.gridx == 1) {
            gbc.gridx = 0;
            gbc.gridy++;
        }
    }
    JPanel containerPanel = new JPanel(new BorderLayout());
    containerPanel.add(orderPanel, BorderLayout.NORTH);

    JPanel detailsPanel = new JPanel();
    detailsPanel.setLayout(new BoxLayout(detailsPanel, BoxLayout.Y_AXIS));
    detailsPanel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));

    String description = "";

    orderType = orders.getFirst().getOrderType().toUpperCase();
    String orderTypeD = orders.getFirst().getOrderTypeDetails().toUpperCase();
    detailsPanel.add(homepage.createDetailLabel("Datetime:", orders.getFirst().getDatetime()));
    jLabel1.setText("Order #"+orderID+"      ("+orderType+")");
    switch (orderType) {
        case "DINE IN" -> {
            detailsPanel.add(homepage.createDetailLabel("Table Number:", "Table "+orderTypeD));
            description = String.format("<html><div style='text-align: right;'>" +
                    "<span style='font-size: 12px;'>%s</span></html>",
                    backend.returnOrderDescription(orders.getFirst().getStatus()));
            detailsPanel.add(homepage.createDetailLabel("Total Amount: ", "RM "+String.format("%.2f", Double.parseDouble(orders.getFirst().g
        }
        case "PICKUP" -> {
            detailsPanel.add(homepage.createDetailLabel("Pick up Time:", orderTypeD));
            description = String.format("<html><div style='text-align: right;'>" +
                    "<span style='font-size: 12px;'>%s</span></html>",
                    backend.returnOrderDescription(orders.getFirst().getStatus()));
            detailsPanel.add(homepage.createDetailLabel("Total Amount: ", "RM "+String.format("%.2f", Double.parseDouble(orders.getFirst().g
        }
        case "DELIVERY" -> {
            Map<Object, Object> deliveryDetails = backend.getDeliveryDetails(orderID);
            List<managefile.Delivery> deliverys = (List<managefile.Delivery>) deliveryDetails.get("deliverys");
            List<managefile.Runner> runners = (List<managefile.Runner>) deliveryDetails.get("runners");

            managefile.Delivery delivery = deliverys.getFirst();
            description = String.format("<html><div style='text-align: right;'>" +
```

This is the class that calls the method of getOrderbyOrderID with the parameter of order id in the customer_backend class. The orders object list is extracted from the allOrders hashmap and can get the attributes value from the object by using getter method that defined in the order class.

3.3 Inheritance

3.3.1 Superclass

```
public abstract class GeneralRole {
    private String id;
    private String name;
    private String email;
    private String phone;
    private String password;

    public GeneralRole() {}

    public GeneralRole(String id, String name, String email, String phone, String password){
        this.id = id;
        this.name = name;
        this.email = email;
        this.phone = phone;
        this.password = password;
    }
```

This is the superclass that stores all the same attributes such as id, name, email, phone and password for different users. Additionally, the variables are defined as private which is not accessible by subclasses. However, the subclass can retrieve and use the values through the constructor using the super() that allows the encapsulation for variable access.

3.3.2 Subclass

```
public class Customer extends GeneralRole {
    private String feedbackID;
    private double credit;

    private String filepath = "\\src\\main\\java\\repository\\customer.txt";

    public Customer(){}

    public Customer(String id, String name, String email, String phone, String password, double credit,String feedbackID) {
        super(id, name, email, phone, password);
        this.credit = credit;
        this.feedbackID = feedbackID;
    }
```

```
public class Admin extends GeneralRole{
    private String filepath = "\\src\\main\\java\\repository\\admin.txt";

    public Admin() {}

    public Admin(String id, String name, String email, String phone, String password) {
        super(id, name, email, phone, password);
    }

    public String getFilepath(){ return this.filepath; }
}
```

```
public class Manager extends GeneralRole{
    private String filepath = "\\src\\main\\java\\repository\\manager.txt";

    public Manager() {}

    public Manager(String id, String name, String email, String phone, String password) {
        super(id, name, email, phone, password);
    }

    public String getFilepath(){ return this.filepath; }
}
```

```java
public class Runner extends GeneralRole{
    private String status;
    private String filepath = "src\\main\\java\\repository\\runner.txt";

    public Runner(){}

    public Runner(String id, String name, String email, String phone, String password,String status) {
        super(id, name, email, phone, password);
        this.status = status;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getFilepath(){ return this.filepath; }

}
```

```java
public class Vendor extends GeneralRole {
    private String stallName;
    private String stallType;
    private String imagePath;
    private String status;
    private String filepath = "src\\main\\java\\repository\\vendor.txt";

    public Vendor(){}

    public Vendor(String id, String name, String email, String phone, String password,
            String stallName,String stallType,String imagePath,String status) {
        super(id, name, email, phone, password);
        this.stallName = stallName;
        this.stallType = stallType;
        this.imagePath = imagePath;
        this.status = status;
    }
```

This is all the different users subclasses that extend the general role superclass. In these subclasses, the constructor parameter can include id, name, email, phone, password in the general role superclass. Additionally, each subclass can define its own unique attributes and inherit the attributes from general role which required in the specific subclass. For customers, it inherits all the attributes from the general role and defines feedback id and credit. For admin and manager, it inherits all the attributes and defines the file path. For runners, it inherits all the attributes and defines the status and file path. Lastly, vendor subclass inherits all the attributes and defines the stallName, stallType, imagePath, status, and file path.

3.4 Polymorphism

3.4.1 Overriding

```
@Override
public void actionPerformed(ActionEvent e){
    JButton[] allButtons = {button20, button50, button80, button100, button150, button180};

    for (JButton allButton : allButtons) {
        if(e.getSource()==allButton){
            amountTopUp.setText(allButton.getText().split(" ")[1]);
        }
    }
    if (e.getSource() == tngButton) {
        showPanel("tng");
    } else if (e.getSource() == grabButton) {
        showPanel("grab");
    } else if (e.getSource() == bankButton) {
        showPanel("bank");
    } else if (e.getSource() == cardButton) {
        showPanel("card");
    }
}
```

This is the overriding method that overrides the actionPerformed method which handles and performs the action event based on the user interactions. It defines the array of buttons such as button20, button50, button80, button100, button150, and button180 which indicates the amount to top up. By iterating the button array, it will check the source of clicked button if the button is on clicked. When the button variable is matched with the clicked button source, the amount top up will set the value of button's text by splitting and get the first index. Similarly, the button of tng, grab, bank and card will switch and display the panel based on the source of clicked button.

```
jTextField1.getDocument().addDocumentListener(new DocumentListener() {
    @Override
    public void insertUpdate(DocumentEvent e) {
        jLabel7.setText(jTextField1.getText());
        getVendorFoodDetails();
    }

    @Override
    public void removeUpdate(DocumentEvent e) {
        jLabel7.setText(jTextField1.getText());
        getVendorFoodDetails();
    }

    @Override
    public void changedUpdate(DocumentEvent e) {
        jLabel7.setText(jTextField1.getText());
        getVendorFoodDetails();
    }
});
```

This shows another example of overriding that overrides the method of insertUpdate, removeUpdate, and changedUpdate. The insertUpdate method is triggered when the text is added to the text field, the removeUpdate method is triggered when the text removed, and the changedUpdate method is used for general document changes. By overriding it, the jLabel7 will display the text and the latest vendor food details based on the searched text.

3.4.2 Overloading

```
public Map<Object, Object> getVendorsReviews(){
    List<Vendor> vendors = read.readVendorAccount(vendorFile);
    List<managefile.VendorReview> vendorReviews = read.readVendorReview(orderReviewFile);

    Map<Object, Object> result = new HashMap<>();
    result.put("vendors", vendors);
    result.put("reviews", vendorReviews);
    return result;
}
public Map<Object, Object> getVendorsReviews(String vendorID){
    List<Vendor> vendors = read.readVendorAccount(vendorFile);
    List<managefile.VendorReview> vendorReviews = read.readVendorReview(orderReviewFile);

    List<Vendor> matchedVendors = new ArrayList<>();
    List<managefile.VendorReview> matchedReviews = new ArrayList<>();
    for (Vendor vendor : vendors) {
        for (VendorReview vendorReview : vendorReviews) {
            if (vendor.getId().equals(vendorID) && vendor.getId().equals(vendorReview.getVendorID())){
                matchedVendors.add(vendor);
                matchedReviews.add(vendorReview);
            }
        }
    }

    Map<Object, Object> result = new HashMap<>();
    result.put("vendors", matchedVendors);
    result.put("reviews", matchedReviews);
    return result;
}
```

This shows the overloading method that have the same method name, getVendorsReviews() and getVendorsReviews(String vendorID) but have different functions.

In the first method, getVendorsReviews() reads all the vendors accounts from vendorFile and vendors reviews from the orderReviewFile. It will store the lists with keys vendors and reviews in a hashmap and return the result (hashmap).

In the second method, getVendorsReviews(String vendorID) will get the vendorID and filter the vendors to get the specific vendor and reviews by iterating the vendor and vendorReview object list. When the vendorID is matched with the vendor id in the list, the vendor list and vendor review list will add into the matched lists and put into the hashmap and return the result (hashmap).

3.5 Aggregation

```
public class VendorReview1 {
    private String reviewID;
    private Vendor vendor;
    private String vendorID;
    private String rating;
    private String comments;

    public VendorReview1(String reviewID, Vendor vendor, String vendorID, String rating, String comments) {
        this.reviewID = reviewID;
        this.vendor = vendor;
        this.vendorID = vendorID;
        this.rating = rating;
        this.comments = comments;
    }

    public Vendor getVendor() {
        return vendor;
    }

    public void setVendor(Vendor vendor) {
        this.vendor = vendor;
    }

    public String getReviewID() {
        return reviewID;
    }

    public void setReviewID(String reviewID) {
        this.reviewID = reviewID;
    }

    public String getVendorID() {
        return vendorID;
    }
}
```

This shows the aggregation relationship between the VendorReview1 and Vendor class, which VendorReview1 has included the Vendor object as attributes. This can help to access the particular vendor details when using getter in the vendor review class.

```
public List<VendorReview1> setVendorReviews(String targetVendor){
    List<String> reviewVendor = read.readVendorReview1(orderReviewFile);
    List<String> vendor = read.readVendorAccount1(vendorFile);
    Map<String, Vendor> vendorMap = new HashMap<>();

    for (String vendorLine : vendor) {
        String[] parts = vendorLine.split(",");
        String id = parts[0];
        String name = parts[1];
        String email = parts[2];
        String phone = parts[3];
        String password = parts[4];
        String stallName = parts[5];
        String stallType = parts[6];
        String imagePath = parts[7];
        String status = parts[8];
        Vendor vendors = new Vendor(id, name, email, phone, password, stallName,stallType,imagePath,status);
        vendorMap.put(id, vendors);
    }
    List<VendorReview1> reviews1 = new ArrayList<>();
    for (String reviewLine : reviewVendor) {
        String[] parts = reviewLine.split(",");
        String reviewId = parts[0];
        String vendorId = parts[1];
        String rating = parts[2];
        String comments = parts[3];

        if (vendorId.equals(targetVendor)) {
            Vendor vendors = vendorMap.get(vendorId);
            if (vendors != null) {
                VendorReview1 review = new VendorReview1(reviewId, vendors, vendorId, rating, comments);
                reviews1.add(review);
            }
        }
    }
    return reviews1;
}
```

This is the method that setVendorReviews by using the targetVendor (vendorID) that return the VendorReview1 object list called reviews1. It will get the lists of review vendor and vendor. The vendor list will be iterated and stored into vendorMap (hashmap). After retrieving the vendor data, the reviewVendor list will be iterated and compare the vendorId with targetVendor. This is to get the vendorId from the vendorMap in order to add the vendor object and create a VendorReview1 object.

```
private void getReviewDetails(){
    List<managefile.VendorReview1> reviewData = backend.setVendorReviews(vendorID);

    Vendor vendor = reviewData.getFirst().getVendor();
    File imageFile = new File(vendor.getImagePath());
    if (imageFile.exists()){
        jLabel2.setIcon(backend.scale.processImage(vendor.getImagePath(), 50, 50));
    }else{
        jLabel2.setIcon(backend.scale.processImage("src\\main\\java\\image_repository\\food-stall.png", 50, 50));
    }
    jLabel2.setText(vendor.getStallName()+"'s Stall - "+vendor.getStallType());
    jLabel2.setHorizontalTextPosition(SwingConstants.LEFT);

    JPanel orderPanel = new JPanel(new GridBagLayout());
    JPanel reviewPanel = new JPanel(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();

    gbc.insets = new Insets(5, 5, 5, 5);
    gbc.gridx = 0;
    gbc.gridy = 0;

    ImageIcon icon = backend.scale.processImage("src\\main\\java\\image_repository\\star.png", 20, 20);
    int totalRate = 0;
    List<String> ratingList = new ArrayList<>();
    for (VendorReview1 vendorReview : reviewData) {
        String rating = vendorReview.getRating();
        String comments = vendorReview.getComments();
        if (!rating.equals("null") || !comments.equals("null")){
            totalCount += 1;
            int rate = Integer.parseInt(rating);
            totalRate += rate;
            switch (rate) {
                case 5 -> star5 +=1;
                case 4 -> star4 +=1;
                case 3 -> star3 +=1;
                case 2 -> star2 +=1;
                case 1 -> star1 +=1;
                default -> {
```

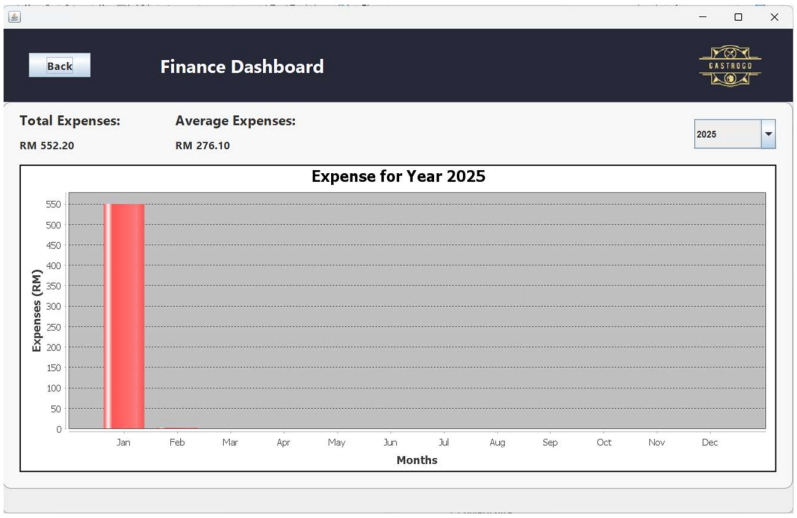
This is the getReviewDetails method that is called the setVendorReviews method with vendorID in the customer_backend class, which the returned list stored as reviewData under VendorReview1 object. As it retrieved the data by vendorID, therefore the vendor will be duplicate and it is safe to use the getFirst() method from the reviewData list and store the vendor details into the Vendor object called vendor. By getting the vendor object, it can be called the get method to get the value of the attribute.

4.0 Additional Features

Online Top Up

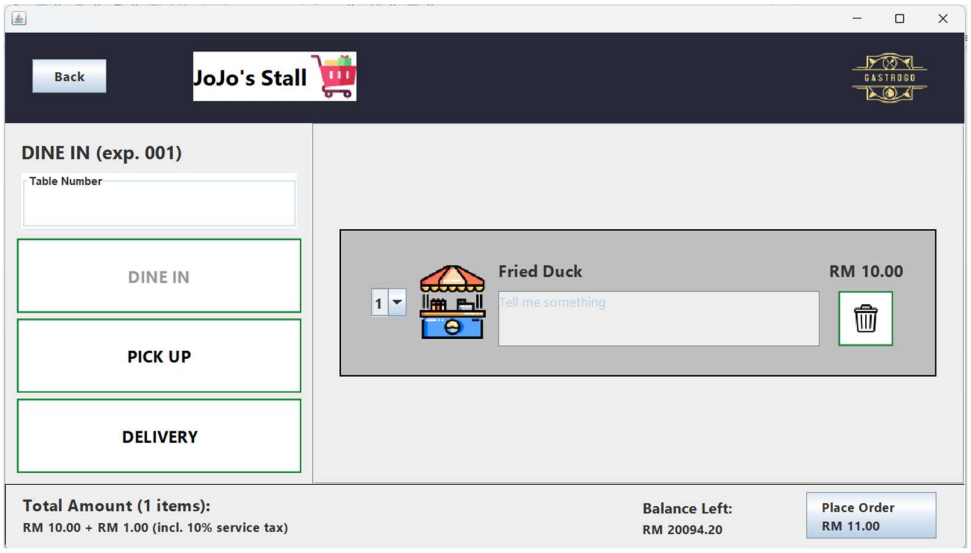
This is the online top up interface that the customer can top up their balance by choosing the amount via e-wallet, bank and card. This allows the customer to top up online instead of manually top up via administrator.

View Finance Dashboard



This is the customer finance dashboard that shows the bar chart of expenses and the total and average expenses for each year. Additionally, the average expense is calculated based on the current month in a certain year.

View Cart



This is the add cart interface that the user can add their food items into the cart and store them until the food is placed in order or removed.

5.0 Limitations

Lack of GPS Functionality for Address Search and Runner Tracking

Currently, our system don't have the GPS functionality to automatically get the customer address and tracking delivery runners in real time. This will affects the overall efficiency of the delivery process, customers must manually enter their address instead of selecting from a GPS-based location service. Besides, they can't monitor the exact location of the delivery runner, which may lead to uncertainty about the delivery status. Implementing GPS functionality would enhance the user experience by allowing seamless address selection, reducing errors in manual input, and providing real-time delivery tracking. This feature could also benefit runners by optimizing their routes and improving delivery efficiency, ensuring timely order fulfillment.

Two-Factor Authentication for Enhanced Security

The current system lacks two-factor authentication (2FA) for account security, meaning that users rely solely on a username and password for access. Without an additional layer of verification, user accounts are more vulnerable to unauthorized access, especially in cases where weak passwords are used or credentials are compromised. Implementing 2FA would significantly enhance security by requiring customers to verify their identity through an email or SMS-based code when changing their password or logging in from an unrecognized device. This would prevent unauthorized access and protect sensitive user data, such as order history and stored credits.

Reliance on Text Files Instead of an Online Database

The system currently relies on text files for storing user data, orders, and transaction history, which limits scalability and data integrity. Text files are not optimized for handling large amounts of data, making retrieval and updates inefficient, especially as the system grows. Additionally, text files do not support concurrent access, increasing the risk of data corruption when multiple users interact with the system simultaneously. Transitioning to an online relational database such as MySQL and MongoDB would provide better data management, faster queries, and improved security. A database-driven approach would allow for real-time

updates, backup capabilities, and better data organization, ensuring a more robust and scalable food ordering system.

6.0 Conclusion

In conclusion, we have designed and implemented a food ordering system that streamlined the order processing process in a food court environment. The system was designed to ensure efficient and user-friendly experience. By incorporating object-oriented programming principles, we successful to built the application to be modular, maintainable, and extensible, which can provide seamless interaction between different system components.

Furthermore, we have learned a lot of valuable knowledge and skills through the development of this project. There is a big improvement on our problem-solving skills, programming abilities, and understanding of real-world software development. The implementation of the system required careful consideration of data management, user interaction, and process automation, which emphasized the importance of software engineering best practices. Not only that, we also learned how to work as a team, including effective communication between teammates, task allocation, and how to combine our respective code.

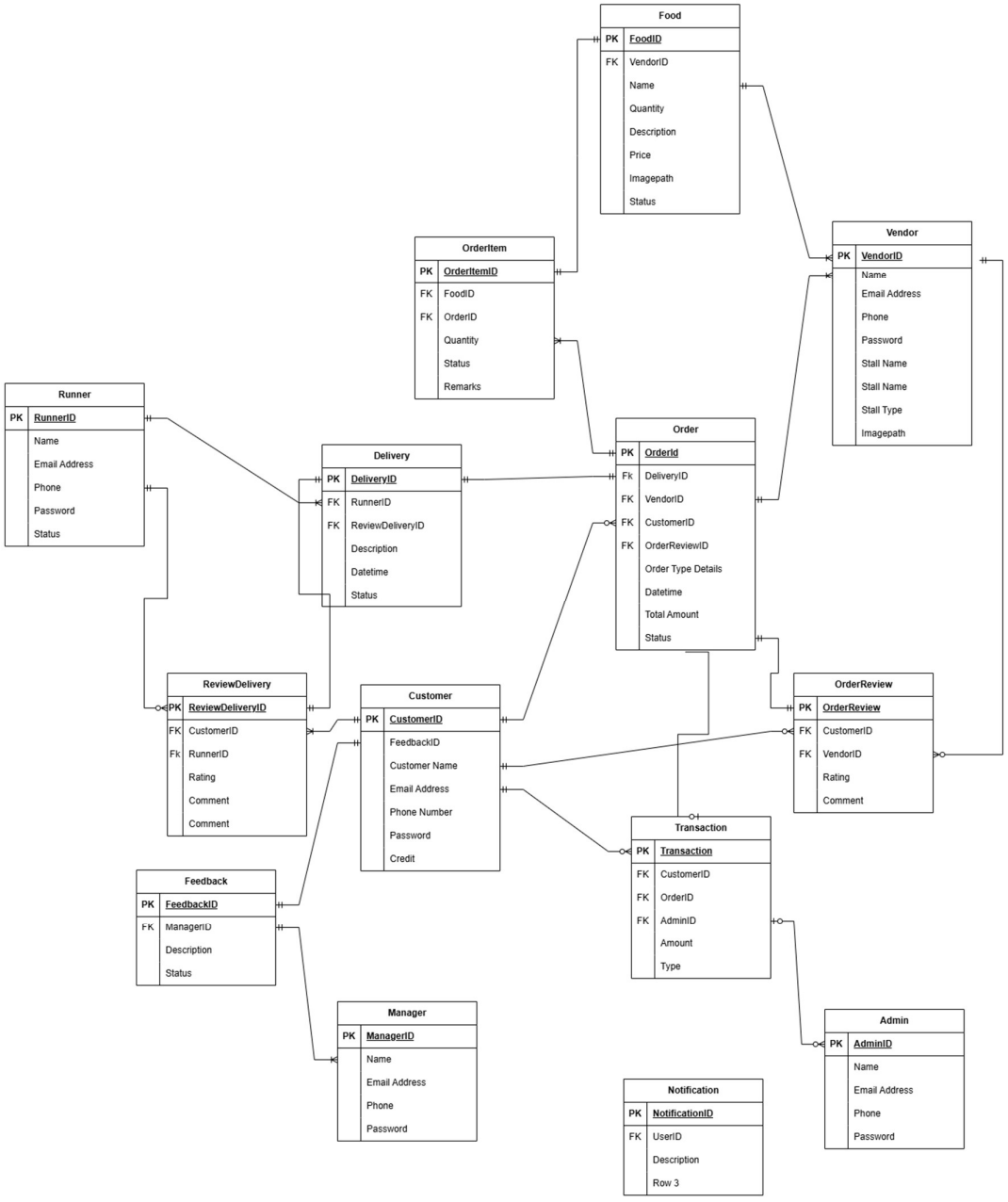
Overall, this assignment provided us with valuable hands-on experience in object-oriented programming, system design, and collaborative software development, equipping us with essential skills for future projects.

7.0 References

There are no sources in the current document.

8.0 Appendix

ERD Diagram (all text files how to store the data)



Runner.txt

```
RunnerID,Name,Email,Phone,Password,Status
R1,Chin Xuan Han,xuanhanchin@gmail.com,010-545544565,xuanhan,Available
R2,Darren Chong Xin Chang,darrenhonggan@gmail.com,012-7776666,dcxc,Available
R3,Rex Kan Wei Hong,rexdamn@gmail.com,019-6668888,rex,Unavailable
```

Admin.txt

```
ID,Name,Email,Phone,Password
A1,Chong Jia Jun,jiajunchong00@gmail.com,018-7783486,cjj6693
```

Vendor.txt

```
ID,Name,Email,Phone,Password,Stall Name, Stall Type, Image path,Status
V1,Chin Xuan Han,xuanhanchin@gmail.com,010-3891107,1234,JoJo,Fast Food,null,Available
V2,PoYeh,a,010-3891107,1234,YoYo,Chinese Food,src/main/java/image_repository/esquites.png,Available
```

Customer.txt

```
CustomerID,Name,Email,Phone,Password,Credit,Feedback
C1,Chin Xuan Han,xuanhanchin1234@gmail.com,010-454546468,abcd,20104.000000000004,FB1
```

Manager.txt

```
ID,Name,Email,Phone,Password
M1,Liew Hao Shean,haoshean@gmail.com,012-2342344,abcd1234
```

Delivery.txt

```
DeliveryID,DeliveryReviewID,OrderID,RunnerID,TotalAmount,Description,Datetime,Status
D1,RD1,O6,R1,8,null,2025-01-13T09:27:18,Cancel
D2,RD2,O9,R1,5,null,2025-01-14T10:32:40,Completed
D3,RD5,O10,R1,5.0,null,2025-01-17T22:59:51,Pending
D4,RD6,O12,R1,5.0,null,Completed,Pending
D5,RD7,O13,R1,5.0,null,Cancel,Pending
D6,RD8,O14,R1,5.0,null,Cancel,Pending
```

DeliveryReview.txt

```
ReviewID,CustomerId,RunnerId,Rating,Comments,Datetime
RD1,C1,R1,2.0,Runner looks ugly!,2024-11-02
RD2,C1,R2,5,,2025-01-16T09:49:35
RD3,C1,R1,2.0,Runner looks ugly!,2024-11-02
RD4,C1,R1,2.0,Runner looks ugly!,2024-11-02
RD5,D3,R1,null,null
RD6,D4,R1,5,null,2025-02-03T09:59:38
RD7,D5,R1,null,null
RD8,D6,R1,null,null
```

Feedback.txt

```
FeedbackID,CustomerID,ManagerID,Description,Datetime
F1,C1,null,Not working1,2025-01-11T22:36:23
F2,C1,,Not working2,2025-01-11T22:36:23
F3,C5,M1,Not working3,2025-01-11T22:36:23
F4,C2,,Not working4,2025-01-11T22:36:23
F5,C3,M1,Not working5,2025-01-11T22:36:23
```

Food.txt

```
FoodID, Name, Status, Description, Price, Imagepath, Category, VendorID
F1,Fried Chicken,Unavailable,Delicious and Juicy Fresh Chicken,10,null, Rice,V1
F2,Fried Duck,Available,Delicious and Juicy Fresh Chicken,10,null,Rice, V1
F3,Fried Rice,Available,Delicious and Juicy Fresh Chicken,15,null,Rice,V1
F4,Fried Cake,Available,Delicious and Juicy Fresh Chicken,20,null,Rice,V1
F5,Fried Noodle,Available,Fried to XXXX,15.00,src/main/java/image_repository/F5.png,Rice,V2
F6,Fried Chicken,Available,Ice & Lemon Tea,21,src/main/java/image_repository/default-food.png,Rice,V2
F7,Fried Chicken,Available,Ice & Lemon Tea,10,src/main/java/image_repository/default-food.png,Desert,V2
F8,Fried Chicken,Available,Ice & Lemon Tea,15,src/main/java/image_repository/default-food.png,Beverage,V2
F9,Fried Chicken,Available,Ice & Lemon Tea,21,src/main/java/image_repository/default-food.png,Japanese,V2
F10,Fried Chicken,Available,Ice & Lemon Tea,10,src/main/java/image_repository/default-food.png,Desert,V2
F11,Fried Chicken,Available,Ice & Lemon Tea,10,src/main/java/image_repository/default-food.png,Desert,V2
F12,Fried Chicken,Available,Ice & Lemon Tea,10,src/main/java/image_repository/default-food.png,Desert,V2
F13,Fried Chicken,Available,Ice & Lemon Tea,10,src/main/java/image_repository/default-food.png,Desert,V2
F14,Fried Chicken,Available,,10,src/main/java/image_repository/default-food.png,Desert,V2
```

Cart.txt

```
CartID,CustomerID,FoodID,VendorID,Quantity,Remarks,Datetime
CR1,C1,F2,V1,1,,2025-02-03T09:41:59
```

Notifications.txt

```
NotificationID,Description,Datetime,CustomerID
NT1,Your Order #O1 is placed order!,2025-01-11T22:36:23,C1
NT2,You received Order #O1|x1 Fried Chicken is placed order!|x6 Fried Cake is placed order!|,2025-01-11T22:36:23,V1
NT3,Your Order #O2 is placed order!,2025-01-11T22:38:07,C1
NT4,You received Order #O2|x3 Fried Noodle is placed order!|x3 Fried Chicken is placed order!|,2025-01-11T22:38:07,V2
NT5,Your Order #O3 is placed order!,2025-01-12T16:18:28,C1
NT6,You received Order #O3|x1 Fried Duck is placed order!|,2025-01-12T16:18:28,V1
NT7,Your Order #O4 is placed order!,2025-01-12T16:22:55,C1
NT8,You received Order #O4|x1 Fried Noodle is placed order!|,2025-01-12T16:22:55,V2
NT9,Your Order #O5 is placed order!,2025-01-13T08:40:49,C1
NT10,You received Order #O5|x4 Fried Noodle is placed order!|,2025-01-13T08:40:49,V2
NT11,Your Order #O6 is placed order!,2025-01-13T09:27:18,C1
NT12,You received Order #O6|x2 Fried Chicken is placed order!|,2025-01-13T09:27:18,C1
NT13,RM 22.40 is refunded.,2025-01-13T09:29:07,C1
NT14,Your delivery order is cancel.|The total of RM 22.40 is refunded.,2025-01-13T22:55:32,C1
NT15,Your Order #O7 is placed order!,2025-01-14T10:00:39,C1
NT16,You received Order #O7|x5 Fried Cake is placed order!|,2025-01-14T10:00:39,C1
NT17,Your Order #O8 is placed order!,2025-01-14T10:04:40,C1
NT18,You received Order #O8|x1 Fried Chicken is placed order!|x5 Fried Cake is placed order!|,2025-01-14T10:04:40,C1
NT19,Your order is cancel.|The total of RM 96.80 is refunded.,2025-01-14T10:05:08,C1
```

Order.txt

```
OrderID,CustomerID,DeliveryID,VendorID,VendorReviewID,Type,TypeDetails,Datetime,TotalAmount,Status
O1,C1,null,V1,VR1,dine in,198,2025-01-11T22:36:23,143.0,Completed
O2,C1,null,V2,VR2,dine in,83,2025-01-11T22:38:07,118.80000000000001,Completed
O3,C1,null,V1,VR3,pickup,16:20,2025-01-12T16:18:28,12.0,Completed
O4,C1,null,V2,VR4,pickup,09:00,2025-01-12T16:22:55,17.0,Completed
O5,C1,null,V2,VR5,dine in,200,2025-01-13T08:40:49,66.0,Cancel
O6,C1,D1,V1,VR6,delivery,Bukit Jalil,2025-01-13T09:27:18,28.0,Cancel
O7,C1,null,V1,VR7,dine in,115,2025-01-14T10:00:39,110.00000000000001,Cancel
O8,C1,null,V1,VR8,dine in,100,2025-01-14T10:04:40,121,Cancel
O9,C1,D2,V1,VR9,delivery,bukit jalil,2025-01-14T10:32:40,48.0,Completed
O10,C1,D3,V1,VR10,delivery,bukit jalil,2025-01-17T22:59:51,130.0,Pending
O11,C1,null,V1,VR11,dine in,188,2025-01-18T17:25:09,11.0,Pending
O12,C1,D4,V1,VR12,delivery,bukit jalil,2025-01-18T17:48:02,10.0,Completed
O13,C1,D5,V2,VR13,delivery,bukit jalil,2025-01-20T12:07:15,15.0,Cancel
O14,C1,D6,V1,VR14,delivery,bukit jalil,2025-01-20T12:08:21,25.0,Cancel
O15,C1,null,V1,VR15,dine in,101,2025-02-03T09:32:51,11.0,Completed
```

OrderItems.txt

```
OrderItemID,OrderID,FoodID,Quantity,TotalAmount,Status,Remarks
OI1,O1,F1,1,10.0,Accept,Tell me something
OI2,O1,F4,6,120.0,Accept,Tell me something
OI3,O2,F5,3,45.0,Accept,Dont want noodle
OI4,O2,F6,3,63.0,Accept,No spicy
OI5,O3,F2,1,10.0,Pending,Tell me something
OI6,O4,F5,1,15.0,Pending,Tell me something
OI7,O5,F5,4,60.0,Cancel,Tell me something
OI8,O6,F1,2,20.0,Cancel,Tell me something
OI9,O7,F4,5,100.0,Cancel,Tell me something
OI10,O8,F1,1,10.0,Cancel,Tell me something
OI11,O8,F4,5,100.0,Cancel,null
OI12,O9,F1,4,40.0,Completed,GG dot com
OI13,O10,F4,6,120.0,Pending,Tell me something
OI14,O10,F2,1,10.0,Pending,Tell me something
OI15,O11,F2,1,10.0,Pending,Tell me something
OI16,O12,F2,1,10.0,Completed,Tell me something
OI17,O13,F5,1,15.0,Cancel,Tell me something
OI18,O14,F2,1,10.0,Cancel,Tell me something
OI19,O14,F3,1,15.0,Cancel,Tell me something
OI20,O15,F2,1,10.0,Completed,Tell me something
```

OrderReview.txt

```
VendorReviewID,VendorID,Rating,Review
VR1,V1,4,Good Taste
VR2,V2,5,null
VR3,V1,2,null
VR4,V2,null,null
VR5,V5,null,null
VR6,V6,null,null
VR7,V7,null,null
VR8,V8,null,null
VR9,V9,5,null
VR10,V1,null,null
VR11,V1,null,null
VR12,V1,null,null
VR13,V2,null,null
VR14,V1,5,null
VR15,V1,5,null
```

Transaction.txt

```
TransactionID,CustomerID,OrderID,Datetime,Type,Payment Method
TR1,C1,O1,2025-01-11T22:36:23,143.0,Debit,null
TR2,C1,O2,2025-01-11T22:38:07,118.80000000000001,Debit,null
TR3,C1,O3,2025-01-12T16:18:28,12.0,Debit,null
TR4,C1,O4,2025-01-12T16:22:55,17.0,Debit,null
TR5,C1,O5,2025-01-13T08:40:49,66.0,Debit,null
TR6,C1,O6,2025-01-13T09:27:18,28.0,Debit,E-Wallet
TR7,C1,O6,2025-01-13T09:29:07,22.400000000000002,Refund,E-Wallet
TR8,C1,O6,2025-01-13T22:55:32,22.400000000000002,Refund,E-Wallet
TR9,C1,O7,2025-01-14T10:00:39,110.00000000000001,Debit,E-Wallet
TR10,C1,O8,2025-01-14T10:04:40,121.00000000000001,Debit,E-Wallet
TR11,C1,O8,2025-01-14T10:05:08,96.80000000000001,Refund,E-Wallet
TR12,C1,O7,2025-01-14T10:14:24,88.00000000000001,Refund,E-Wallet
TR13,C1,O6,2025-01-14T10:21:44,22.400000000000002,Refund,E-Wallet
TR14,C1,O5,2025-01-14T10:26:45,52.800000000000004,Refund,E-Wallet
TR15,C1,O9,2025-01-14T10:32:40,48.0,Debit,E-Wallet
TR16,C1,O9,2025-01-14T10:42:42,38.400000000000006,Debit,E-Wallet
TR17,C1,O9,2025-01-15T20:05:47,38.400000000000006,Refund,E-Wallet
TR18,C1,A1,2025-01-15T20:05:47,100,Credit,Cash
```

9.0 Video Presentation Link

https://cloudmails.sharepoint.com/:v:/s/java404/EQvkJ3JbPW5MgrtAW6WZcdcBzh9hzTI41poDAAPi6ObYww?e=r4fNKp