Logo, company name

Description automatically generated  
GROUP ASSIGNMENT

CT038-8-2-OODJ

OBJECT ORIENTED DEVELOPMENT WITH JAVA

APD2F2309CS(CYB)

|  |  |  |
| --- | --- | --- |
| HAND OUT DATE | : | WEEK 3 |
| HAND IN DATE | : | WEEK 13 |
| LECTURER | : | Dr. Kadhar Batcha Nowshath |

|  |  |
| --- | --- |
| Name | TP Number |
| SAIF WALEED ABDELAHMAN SAYED ABDELSALAM | TP068702 |
| FOUAD MOHAMED FOUAD ISMAIL | TP069613 |
| ABDULRAHMAN GAMIL MOHAMMED AHMED | TP071012 |
| IBRAHEEM MOHAMMED IMADELDIN AWAD | TP070765 |

**Table of Contents**

[**1.0 Introduction 3**](#_Toc153228177)

[**2.0 OOP Concepts implementation in code and how it was applied. 5**](#_Toc153228178)

[**2.1 Encapsulation 5**](#_Toc153228179)

[**2.2 Classes and Objects 6**](#_Toc153228180)

[**2.3 Inheritance 6**](#_Toc153228181)

[**3.0 UML Diagrams. (2 Diagrams) 8**](#_Toc153228182)

[**3.1 Class Diagram. 8**](#_Toc153228183)

[**3.2 Use case Diagram. 10**](#_Toc153228184)

[**4.0 Program output Screenshots. 12**](#_Toc153228185)

[**4.1Admin Role. 12**](#_Toc153228186)

[**4.2 Vendor Role. 30**](#_Toc153228187)

[**4.3 Delivery Runner Role. 38**](#_Toc153228188)

[**4.4 Customer Role. 46**](#_Toc153228189)

[**5.0 Conclusion 60**](#_Toc153228190)

[**6.0 References 62**](#_Toc153228191)

# Introduction

As the needs of Tech University's dynamic campus change, the administration is developing a state-of-the-art application that will revolutionize the ordering procedure for meals. This creative concept seeks to skillfully connect all parties involved on campus, from food providers and discriminating customers to diligent administrators and dedicated delivery runners. Apart from streamlining the meal ordering process, the proposed application fosters a networked environment that enhances customer satisfaction and operational efficiency.

The goal of this project is to design and implement a cutting-edge program for Tech University that will enhance the efficiency of the campus meal ordering process. The proposed application will connect various users, including administrators, delivery runners, customers, and cafeteria suppliers, as a holistic solution. The major goal is to make ordering easier for customers by providing a seamless manner for them to peruse menus, place orders, and complete transactions. Additionally, the program offers a credit-based payment system that allows users to add credit through administrators. The program has a number of essential features, each designed to meet the unique requirements of various user roles:

**Login to Access:**

Vendor: Monitor orders and their status; review past orders and income; add, edit, delete, and read client comments.

Customer: browse the menu, make and cancel orders, keep track of the status and history of transactions, and provide comments.

Delivery Runner: See, edit, accept, reject tasks; examine task history and revenues.

Creating and distributing transaction receipts, overseeing runners, suppliers, and customers, and replenishing customer credit are among the duties of an administrator.

**Creating an Account:**

Suppliers and consumers can only be registered by administrators. The user's details are contained in a text file.

**Menu:**

displays menus from several suppliers, simplifying the ordering process for clients.

**Putting in a Food Order:**

allows customers to purchase from many vendors and choose delivery, takeaway, or eating in. Orders are accepted at the vendor's discretion; orders that are rejected will be promptly reimbursed.

**Money Transfer:**

has an automated digital wallet feature that deducts the appropriate amount from the user's credit. In addition to changing customer credit, administrators can provide digital receipts.

**Notification:**

Administrators can use an in-app notification system to provide digital receipts. Informs suppliers and customers when an order has been accepted, rejected, or is in progress.

**Order Document:**

lets suppliers and customers keep an eye on and review orders, including placement dates, amounts, and acceptance/decline status. Customers can place orders from the past again.

**Delivery system:**

allows consumers to choose a delivery supplier by emulating the process of delivery. automatically assigns runners so they may accept or decline tasks and keep track of their pay. Informs customers with the progress of their orders and offers substitutes if runners are unavailable.

# 2.0 OOP Concepts implementation in code and how it was applied.

## 2.1 Encapsulation

By using encapsulation, users may prevent direct access to certain object components, preventing them from accessing the state values of all the variables in that object. Data members and data functions or methods connected to an instantiated class or object can both be hidden through encapsulation (Sumo Logic, Inc., 2023).

A computer code with text

Description automatically generated with medium confidence

Figure 1

Figure 1 shows how private variables may be used to encapsulate and restrict access to class data. While the “RunnerID” and username of static members are internally controlled yet shared across the class, immutability is guaranteed by the final Scanner.



Figure 2

Figure 2 shows how the “handleAcceptOrder” function is included within the class and designated as private and static. This ensures controlled usage and modification of the order-handling process because it is only available within the class itself and does not need the calling of an instance of the class.

## 2.2 Classes and Objects

Java classes are essentially object creation blueprints. They may have methods (functions or processes) and fields (variables) that specify how the objects made from the class will behave. A class is not an object; rather, it is a specification of the data and behavior expected of objects of that class. An instance of the class must be created using the new keyword to construct an object, and this instance is then referenced by a reference variable (Java Class and Objects (With Example), n.d.).



Figure 3



Figure 4

The Java public class Admin declaration, an example of object-oriented programming (OOP), is shown in both photos. The class is reachable from other classes across packages thanks to the public modifier. Admin is a blueprint that functions as a class. It can use OOP concepts like inheritance (if it extends to another class), polymorphism (via interfaces and method overloading), and encapsulation (private variables and methods within the class) to contain data and methods. The base for building objects and putting admin-specific actions into practice in a system is this class declaration.

## 2.3 Inheritance

In Java, inheritance is the process of taking traits from other classes to establish a hierarchy across classes. Because Java inheritance is transitive, Sedan will also inherit from the Vehicle class if it extends Car and Car extends Vehicle (Pankaj, 2022).



Figure 5

The Java class declaration in the image clearly illustrates inheritance in the language by utilizing the extends keyword. It is specified that the “DeliveryRunner” class is a subclass of the Runner class. “DeliveryRunner” may reuse common features without having to develop them from scratch since it inherits all public and protected fields and methods from Runner. One of the fundamental tenets of object-oriented programming (OOP) is inheritance, which creates a hierarchical connection between classes and encourages code reuse.

# 3.0 UML Diagrams. (2 Diagrams)

## 3.1 Class Diagram.

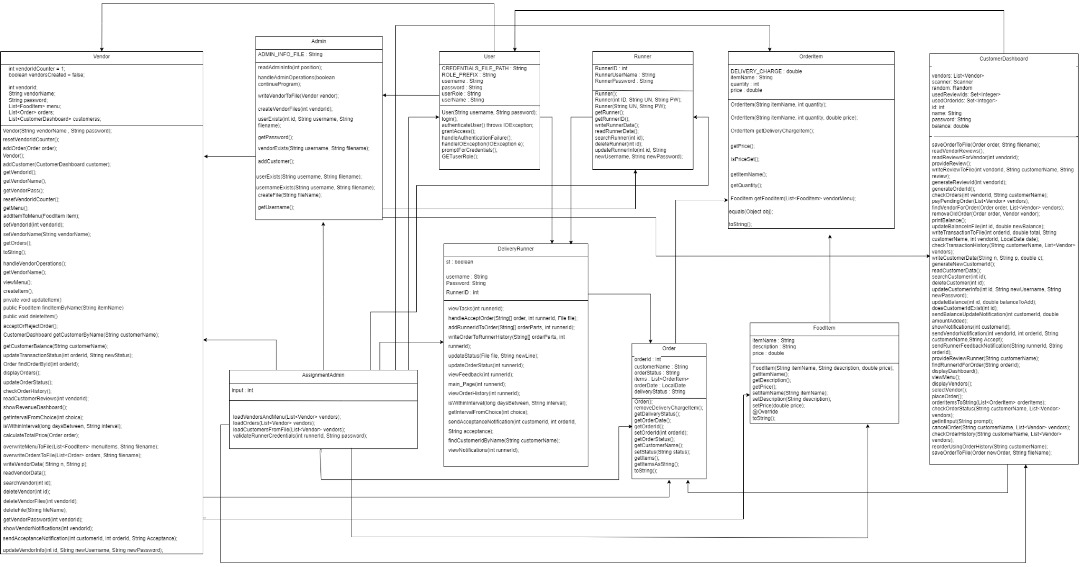


Figure 6

Vendor: It looks that orders, menu items, and dashboard interactions are managed by this class. Additionally, it has attributes like vendorName, vendorId, and password. Based on actions such as createVendor, getMenu, and handleVendorOperations, it seems to be used for managing vendor-related functions.

CustomerDashboard: This class probably shows what a client would see on their user interface. It has functions for ordering takeaway, monitoring orders and handling payments.

Order: This class represents an order placed by a client. It has attributes like orderId, customerName, goods, and orderStatus, among others. The procedures setDeliveryCharge, toString, and getOrderDetails suggest that this class handles the order processing logic.

Certain products inside an order appear to be represented by this class, OrderItem. In addition to an itemName, quantity, and price, it includes ways to manage the item's details.

FoodItem: This class represents a food item on the menu and has characteristics like itemName, description, and price. It can retrieve and change item information using its instructions.

Runner: This refers to the delivery personnel. It has attributes like runnerId, runnerPassword, and runnerUsername. It seems to carry out order delivery and runner maintenance based on handleKickOutOrder, updateRunnerData, and viewOrderHistory functions.

User: This class seems to be general for users; it has authentication and user management methods, along with attributes like username and password.

Admin: This class most likely refers to an administrator user who can read and write data connected to vendors in addition to carrying out administrative duties.

DeliveryRunner: This class extends the Runner class and suggests that it has specific functionality for runners that manage delivery with methods like handleKickOutOrder and updateOrderStatus.

AssignmentAdmin: It is like a controller class or utility that manages the assignment of orders to runners, including methods for loading and saving data.

The composition connections represented by the arrows with diamond ends show that a class owns and manages the lifespan of an object belonging to another class. Links between examples of the two classes are represented by affiliations, which are regular arrows with a plain terminus.

## 3.2 Use case Diagram.

A diagram of a diagram

Description automatically generated

Figure 7

Figure 7 shows the use case diagram, a kind of UML (Unified Modeling Language) diagram used to illustrate how users, also known as actors, interact with various system components. The vendor, customer, and administrator are the actors in this scenario. A use case, or capability or activity that the actor can carry out within the system, is represented by each oval.

Use cases such as ViewMenu, CreateItem, UpdateItem, DeleteItem, and numerous more imply that the vendor is capable of managing orders and menu items. Customers may engage with the system to control their orders, as seen by use cases like ViewMenu, PlaceOrder, and CancelOrder. Customer\_options, Vendor\_options, and Runner\_options are just a few of the administrative operations that the Admin actor is linked to, demonstrating the admin's ability to control and modify many system components.

# 4.0 Program output Screenshots.

## 4.1Admin Role.

The Figures under the “Admin Role” section will focus on all the functionalities of the admin. The admin is required to Add, Update, Search for, Delete, and finally add Credit to customers while the program generates a receipt for the customer.

A screenshot of a computer screen

Description automatically generated

Figure 8

Figure 8 Shows the Main menu where the user is asked to choose their role. Under each roll several individuals are registered with id, usernames, passwords, and in the case of the customer, there is a column for their credit as well.

A screenshot of a computer program

Description automatically generated

Figure 9

Figure 9 shows the “Admin Operations Menu” which appears after the program compares the entered credentials by the user to the credentials present in the “admin\_info.txt” text file. The format of the text file is “id,Username,Password” where the columns are separated by a “,”. Figure 9 also shows the Admin selecting “1. Add Customer” option where they are then prompted to enter/ register the new customer using their Username, Password, and the Credit they want in their account.

A screenshot of a computer program

Description automatically generated

Figure 10

A screenshot of a computer program

Description automatically generated

Figure 11

Figures 10 and 11 show how the information stored about all the customers are read, and how the information of a specific customer is retrieved based on their id respectively.

A screenshot of a computer

Description automatically generated

Figure 12

Figure 12 shows the updating process if the customer wishes to change their information which in this case the customer would like to change both their username and password. But if the customer would like to change only their password that option is also there. When the Admin enters the ID of the Customer, the program identifies which rows to update in the text file causing only the row with that specific ID number to be changed.

A screenshot of a computer program

Description automatically generated

Figure 13

As shown previously in Figure 13 the customer with ID number 4 previously named “Ali” had a credit value of 500. When the option for adding credit is chosen and the information of the customers is read again by entering the number 2 when prompted under the “Customer Options Menu”, the value of credit is observed to have increased by the amount entered previously by the Admin which was 200. At the end of the Figure the customer with ID number 4 is observed to have a credit value of 700 meaning the credit was added successfully to the account of the Customer.

A screenshot of a computer program

Description automatically generated

Figure 14

Figure 14 shows the Admin choosing the 6th option in the menu by enter the number 6 in the “Enter your choice” field, after which the admin is prompted to enter the ID number of the Customer that the Admin wishes to delete the information of from the text file. As shown by the Figure, the Admin enters the user ID number 4 to delete the user “Usama”. Using the ID number the program identifies the line with user ID and deletes that line from the text file.

A screenshot of a computer program

Description automatically generated

Figure 15

Figure 15 shows the admin exiting the “Customer Options Menu” which returns him to the “Admin Operations Menu” where the Admin can access other options for either “Runners” or “Vendors”.

A screenshot of a computer program

Description automatically generated

Figure 16

A screenshot of a computer

Description automatically generated

Figure 17

A screenshot of a computer

Description automatically generated

Figure 18

A screenshot of a computer

Description automatically generated

Figure 19

A screenshot of a computer

Description automatically generated

Figure 20

A screenshot of a computer program

Description automatically generated

Figure 21

A screen shot of a computer

Description automatically generated

Figure 22

Figure 22 shows the Admin exiting the “Vendor\_Operations” menu where a similar set of operations to that of the Customer is done for the Vendor. All the options work the same way as those of the Customer except for the Vendor not having an “Add Credit” option as shown by Figures 16 to 21.

A screenshot of a computer program

Description automatically generated

Figure 23

A screenshot of a computer program

Description automatically generated

Figure 24

A screenshot of a computer program

Description automatically generated

Figure 25

A screenshot of a computer program

Description automatically generated

Figure 26

A screenshot of a computer program

Description automatically generated

Figure 27

A screenshot of a computer program

Description automatically generated

Figure 28

Figure 28 shows the exiting the “Runner Operations Menu” which is option number 6 and is activated when the program receives and integer 6 input from the Admin to return to the “Admin Operations Menu”. All the options work the same way as those of the Vendor as shown by Figures 23 to 27.

A screenshot of a computer

Description automatically generated

Figure 29

Figure 29 shows the customer receiving the receipt in the form of a notification under their account that the admin has updated their account’s credit.

## 4.2 Vendor Role.

This part is all about the vendor's role. Vendors log in with their ID and password to access tasks meant just for them. These functions are related to managing menus, processing orders, tracking revenue, and handling customer interactions.

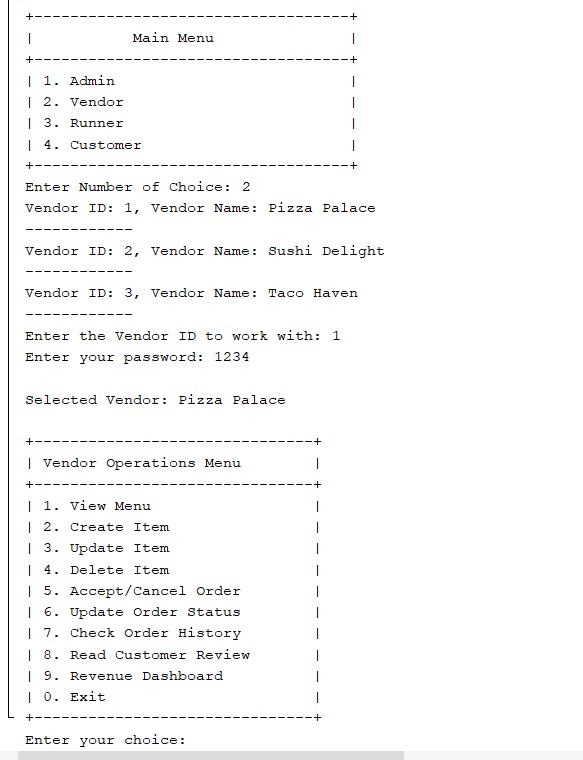


Figure 30

In the figure above, the vendor, using the ID '1' for Pizza Palace, enters the password 'pass1.' Once validated by the program, the vendor gains access to the vendor menu to begin. If the validation fails, the vendor will be redirected back to the menu to choose the role they want to log in with.



Figure 31

In this illustration, the vendor selects option number 1, which allows them to view the menu. Upon choosing this option, the vendor gains access to their specific menu items.

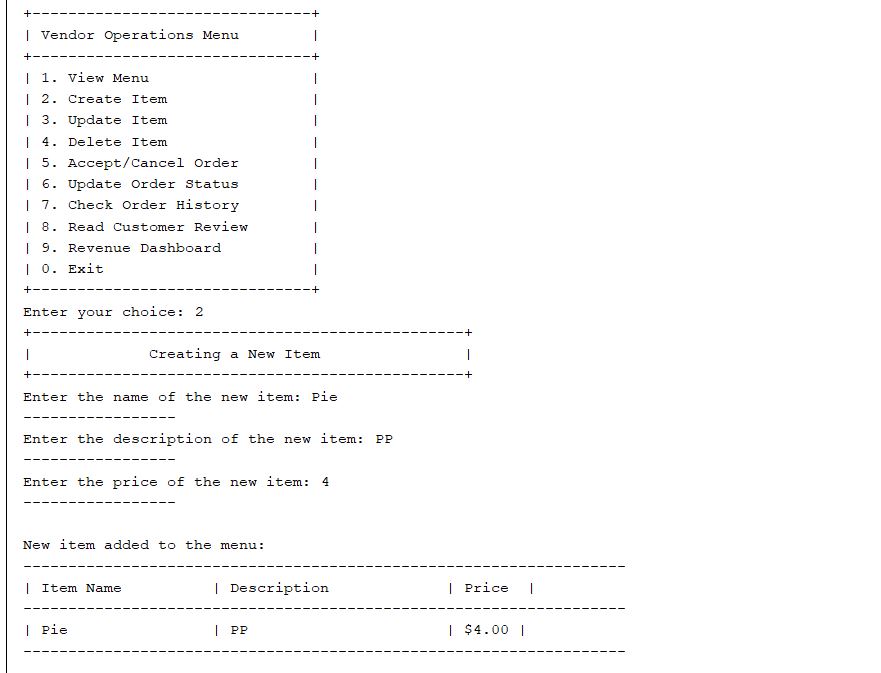


Figure 32

In this scenario, the vendor selects option 2, intending to create a new item in the menu. Following this choice, the vendor is prompted to enter the name of the new item. In this instance, 'Pie' is entered. Next, the vendor provides a brief description for the item, noted as 'PP.' Subsequently, the vendor specifies the price as 4. Once completed, a confirmation displaying the newly added item with all its details appears for the vendor, indicating the successful addition to the menu.

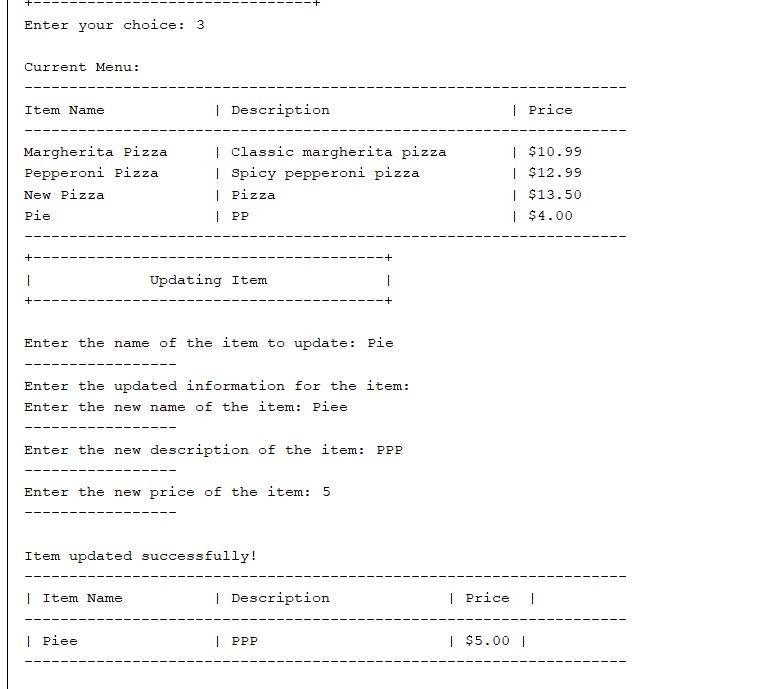


Figure 33

In this instance, the vendor selects option 3, designed for updating items on the menu. Initially, the vendor enters the name of the item to be updated, which is 'Pie.' Following this, the vendor inputs the new name for that item as 'Piee,' a new description labeled as 'PPP,' and a revised price of 5. As a result, the program displays confirmation of the successfully updated item, presenting the latest information regarding the modified menu item.

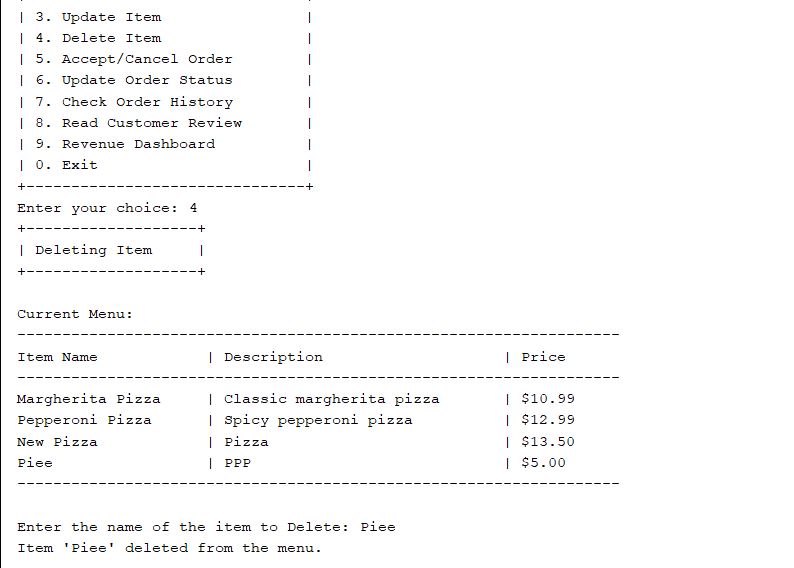


Figure34

When choosing option 4, which involves deleting an item from the menu, the vendor is presented with the current menu. Subsequently, the vendor is prompted to enter the name of the item they wish to remove. In this case, the vendor enters 'Piee.' As a result, the program deletes 'Piee' from the menu and notifies the vendor that the item has been successfully removed.



Figure 35

In this scenario, the vendor opts for option 5, enabling them to either accept or reject orders. Upon selecting this option, the vendor gains visibility into all available orders eligible for acceptance or rejection. The vendor is then asked to input the order ID they wish to act upon. In this case, the vendor selects order 5977 and proceeds to accept it by choosing 'A' for accept. A notification confirming the acceptance of the order appears, indicating a status change to Preparing,' signifying that the order is now being prepared in the vendor's kitchen.

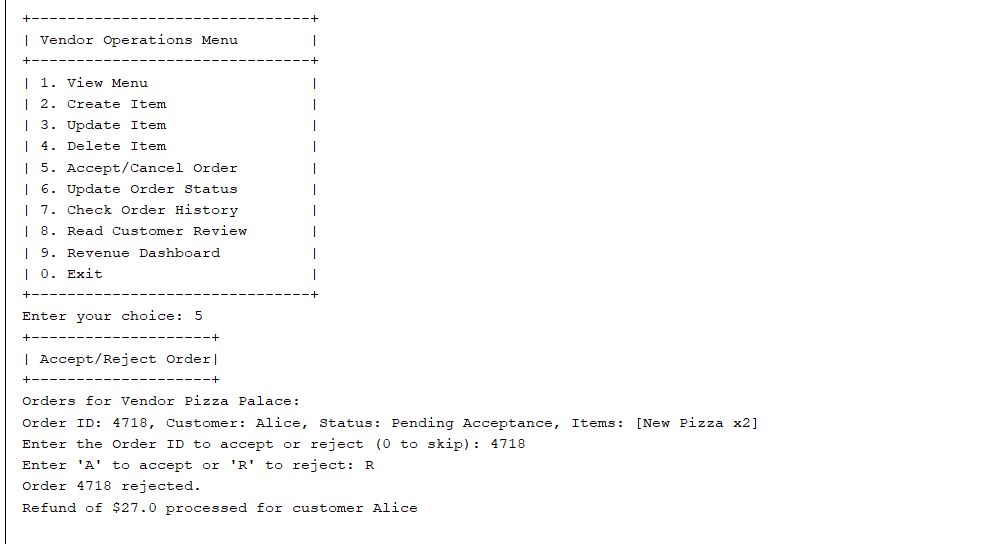


Figure 36

In this instance, the vendor selects order 4710 and opts to reject it by choosing 'R' for reject. Subsequently, a notification confirms the rejection of the order. Additionally, a refund process for the customer named Alice is initiated, ensuring the return of the money.

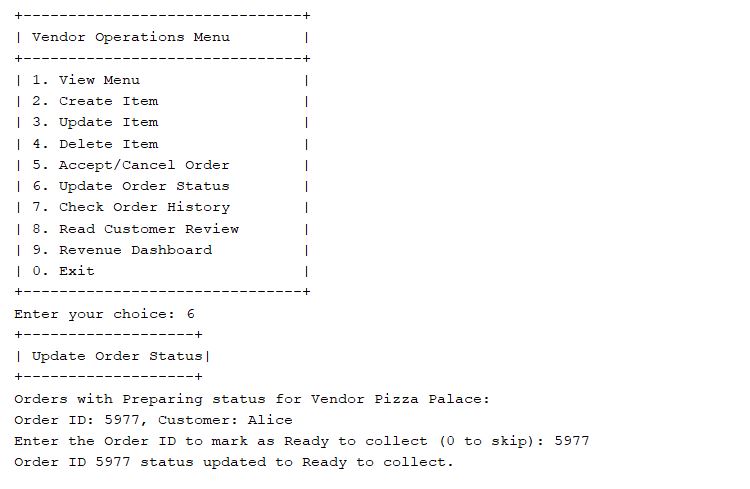


Figure 37

When selecting option 6, designated for updating order status, the vendor transitions orders from 'Preparing' to 'Ready to collect.' Initially, the program displays all orders currently in the 'Preparing' status and prompts the vendor to input the order ID they wish to update to 'Ready to collect.' Upon selecting order ID 5977, the program confirms the status update, indicating that order 5977 is now 'Ready to collect.' This status signifies that the order is available for takeaway, dining in, or delivery.

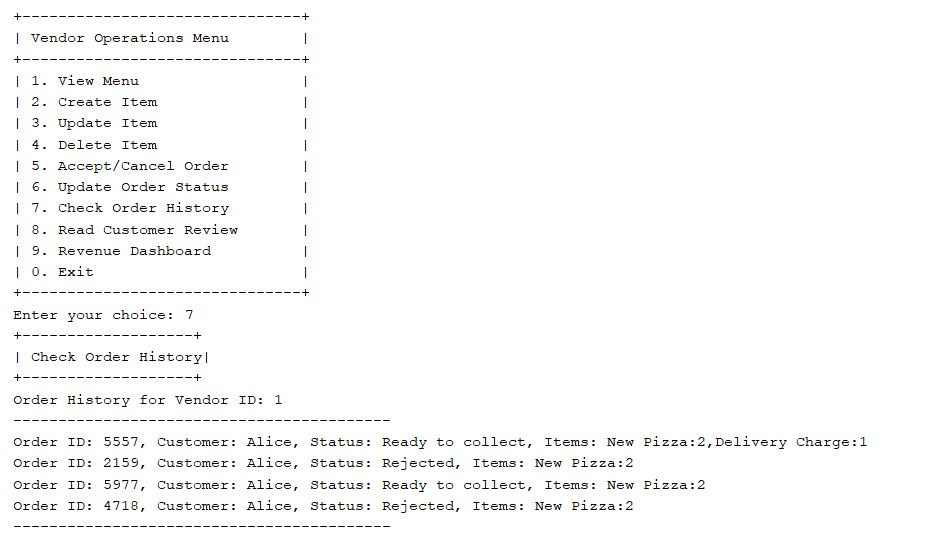


Figure 38

When the vendor chooses option 7, labeled as "Check Order History," a display showcases a comprehensive list of all orders—both accepted and rejected—providing an overview of the complete order history for the vendor to review.

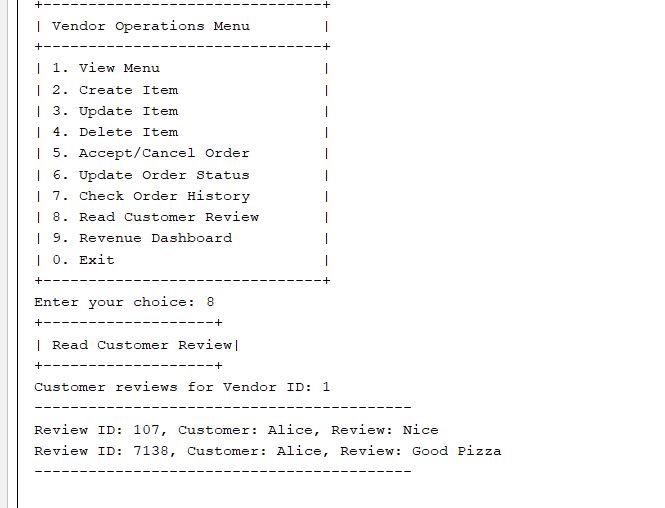


Figure 39

Upon selecting option 8, which is dedicated to "Reading Customer Reviews," the vendor gains access to a compilation of all reviews submitted by customers specifically for them. This feature allows the vendor to peruse and review the feedback provided by their customers.

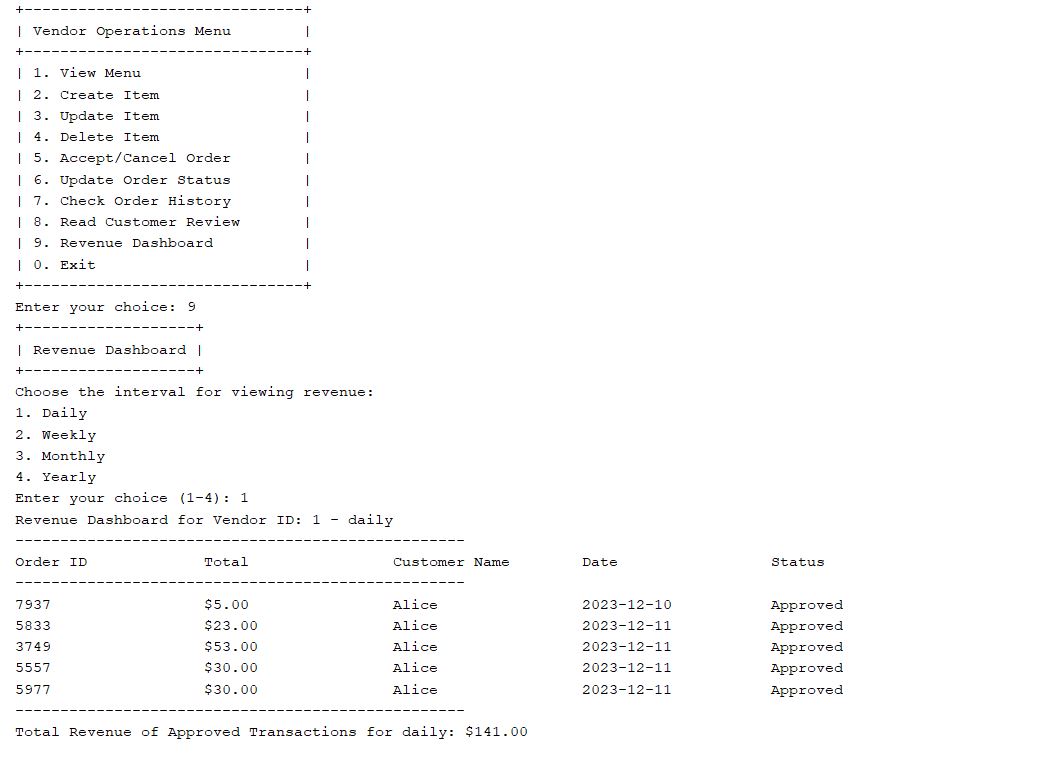


Figure 40

When the vendor selects option 9, labeled as the "Revenue Dashboard," a prompt appears, allowing the vendor to specify their preference to view revenues categorized by daily, weekly, monthly, or yearly transactions. Upon choosing 'daily', the dashboard reveals a comprehensive display presenting all vendor transactions completed on a daily basis, each tagged with the 'Approved' status, alongside the respective date and total for every order. Finally, the dashboard summarizes the total revenue generated from these daily transactions.

## 4.3 Delivery Runner Role.

A screenshot of a computer

Description automatically generated

Figure 41.

It would want an administrator, delivery runner, vendor, or customer to identify their position in the above-presented snapshot of the programmed output. Upon selecting "Delivery Runner," the user is prompted to provide their login credentials, which comprise the username "1" and password "1234." After a successful login, the user may see the Delivery Runner dashboard, if the credentials are not right, the software will show the selecting role page again.

A screenshot of a computer program

Description automatically generated

Figure 42.

Delivery The interface above acts as their menu, where runners may look over new tasks, update delivery status, review delivery history, get runner feedback, and check notifications. Additionally, there is a "Exit" option. A notification stating that there are no outstanding tasks shows when you click "View Available Tasks," indicating that there are now any jobs available.

A screenshot of a computer

Description automatically generated

Figure 43.

Delivery Once a customer places and pays for an order, presuming the vendor has accepted it, runners browse open tasks on the home page. The order ID "527" is displayed as an example. Entering the ID, the Runner chooses "a" for accept and "r" for reeject. The system excludes the order from that Runner's perspective if it is rejected, ensuring efficient work distribution depending on Runner availability and preferences. This method optimises the delivery process by dynamically managing and allocating work to available Runners.

A screenshot of a computer program

Description automatically generated

Figure 44.

When a Runner accepts an order, signifying vendor pickup, delivery to the customer starts. After the Runner arrives at their door, the customer may change the order status by selecting "2" under "Update Delivery Status." Upon entering the order number, the Runner designates the delivery status as "Delivered." This step completes the order completion process. When Runners get to the customer's location, the programme provides them with an easy means to signify that the delivery was successful. It also makes updating delivery status straightforward.

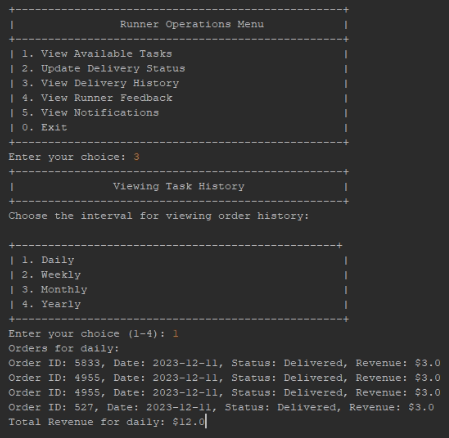


Figure 45.

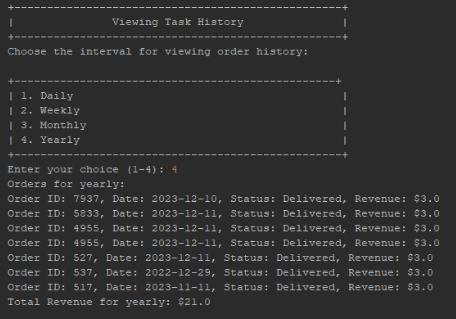


Figure 46.

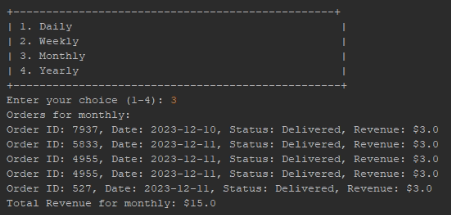


Figure 47.

A screenshot of a computer

Description automatically generated

Figure 48.

The Runner Operation Menu's option '3' allows Runners to 'View Delivery History.' The application instructs the Runner to choose the number (1-4) that represents the selected period: '1' denotes "daily," '2' denotes "weekly," '3' denotes "monthly," and '4' denotes "yearly." The flexibility this structured approach provides allows runners to access specific historical data, enhancing their ability to evaluate performance and make educated decisions. The system then shows the revenue generated during the chosen period and shows statistics based on order dates. This feature-rich tool allows Runners to assess performance, track revenue, and make educated decisions to effectively manage their delivery operations.

A screenshot of a computer program

Description automatically generated

Figure 49.

Following delivery, customers can provide feedback on the Delivery Runner. The Runner receives a notification about a feedback message received for a certain order ID. To view feedback, the Runner selects option "4" from the Runner Operation Menu. The software then shows all the comments pertaining to orders that the Runner accepted. This platform allows Runners to review and respond to customer feedback, promoting communication and continuous service quality improvement.

A screenshot of a computer screen

Description automatically generated

Figure 50.

When a Runner selects '5' from the Runner Operation Menu (View notification), all received notifications will be shown. Runner ID, order ID, customer name, and feedback date are all contained in these alerts. Runners may more easily monitor and manage client interactions with this functionality, which provides a comprehensive overview of feedback-related data. By receiving this notification summary, which gives them information about customer feedback, runners may enhance their ability to address customer concerns and make sure that their delivery service is prompt and customer focused.

A screenshot of a computer

Description automatically generated

Figure 51.

If the user selects '0' in the Runner Operation Menu to exit the program, it initiates a logout process and switches the user back to the login page. This ensures a secure and controlled logout procedure by allowing the user to terminate their session and requiring reauthentication for additional access. The Runner Operation Menu's '0' option offers a standardized and user-friendly method for users to terminate their session, promoting system security and user privacy.

## 4.4 Customer Role.

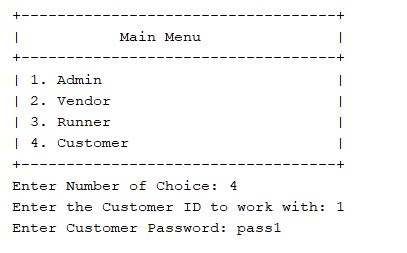


Figure 52.

Here in the figure above is showing the log in part, where you choose your role and then input the valid ID and Password. As shown above I chose customer which is #4 then ID 1 and my Password = pass1 as shown above; which granted me access to customer part.

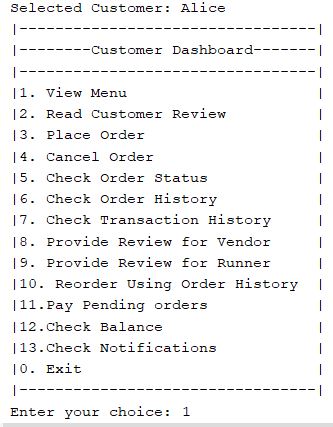


Figure 53.

Here in the figure above is the customer menu and options he can choose in the interface like view menu, read review, place order, cancel order, check order status, provide reviews for vendor and runner, making payments, and checking balance. Where above I input (1) which is the view menu and, in the figure, below it will show the menus available.

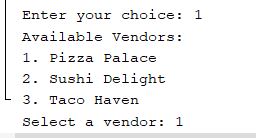


Figure 54.

Firstly, the program shows the available vendors where above gave the choices of :1. **Pizza palace**, 2. **Sushi Delight**, and 3. **Taco Haven**. Based on customer choice of vendor the program will display the chosen vendor menu where I chose **(1) Pizza Palace**; as shown below:

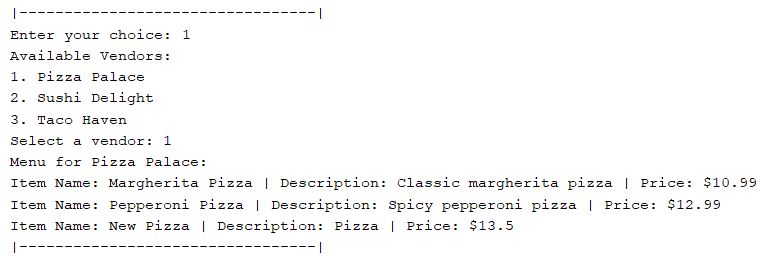


Figure 55.

In the figure above, after choosing vendor (1) the program displayed the menu for **pizza palace** showing the available item, Description, and cost of every item in menu as shown above.

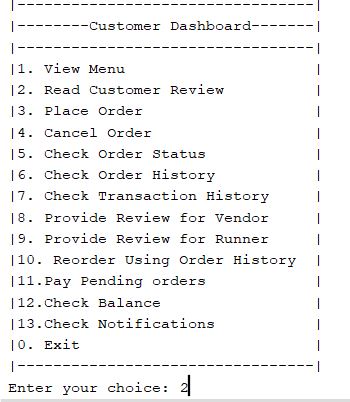


Figure 56.

In this section of program, it shows if customers want to read reviews from customers to choose the place where they would like to eat based on the reviews.

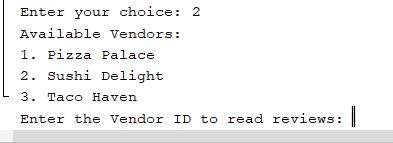


Figure 57.

Here the program asks the customer to input the vendor that they want to add review to by showing the available vendors then asking you to enter the vendor ID as shown above.

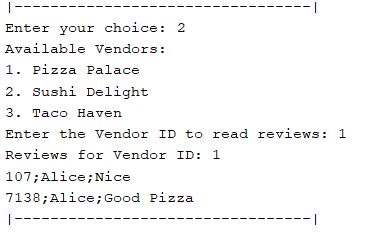


Figure 58.

Here the program displays the reviews of the chosen vendor showing the id then the customer's name and finally the review.

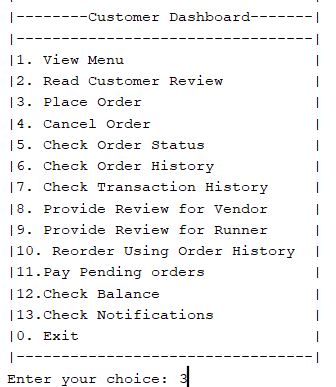


Figure 59.

Here will attempt to place order by choosing input (3) also as shown every time customer is done with operation the program automatically returns to main page the **(Customer Dashboard)** as show above.

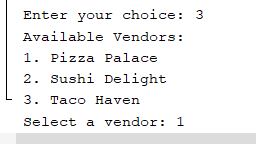


Figure 60.

Here showing the available vendors that customer can place order from then asking customer to select vendor as shown above.



Figure 61.

Here in this step the customer chose **Vendor (1)** then system displayed the menu automatically with items, descriptions, and price; asking customer how many items they will order where customer input 2 then the name of item and quantity, where customer order **1 pepperoni pizza and 2 Margherita** and then it closed automatically since customer chose in beginning, he will need **2 items.**

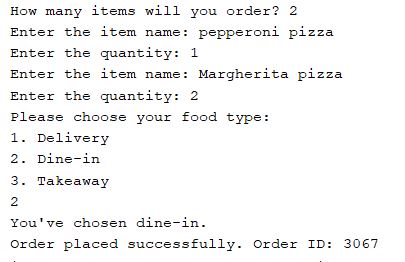


Figure 62.

Finally, the system asks the customer if they would like the food through **Delivery** adding extra fees, or **Dine-in**, or **take-away** and lastly printing the **order ID** where in that case it is: **3067**.

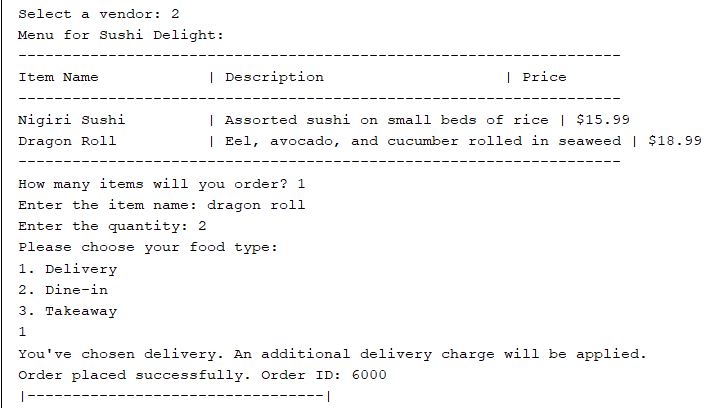


Figure 63.

Here, since the customer chose delivery system confirms order number and delivery charge will be applied as shown above **difference** between **delivery** and **dine-in** the previous order.

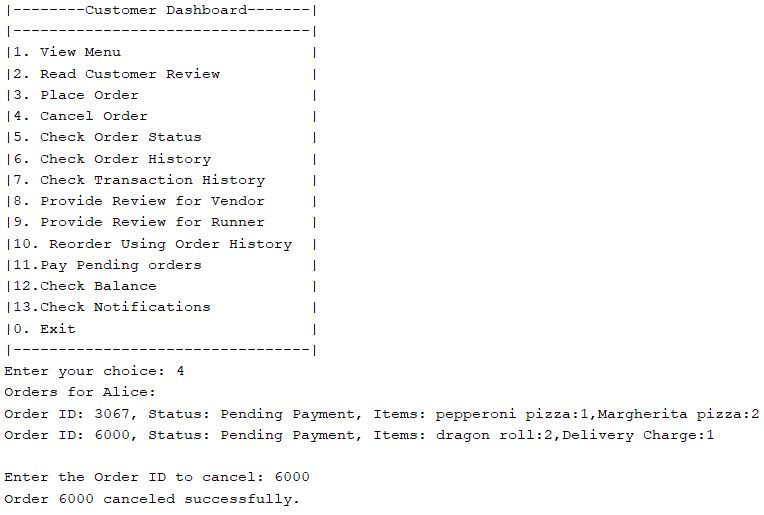


Figure 64.

Returning to main page showing the cancel order operation where user will input **(4)** then system will display the orders made with their Order ID and ask the customer which order would like to cancel where customer input **Order Id: 6000** and system confirmed it has been **successfully canceled** as shown above.

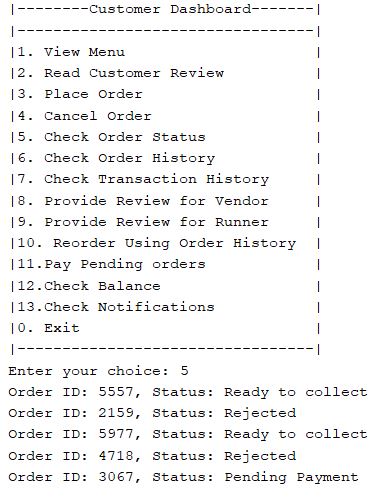


Figure 65.

In this step customer will check his/her order status inputing **(5)** then system will display list of orders made and their status weather it is **Ready to collect** ,**Rejected** ,or **pending payment** with the **Order ID** as shown above.

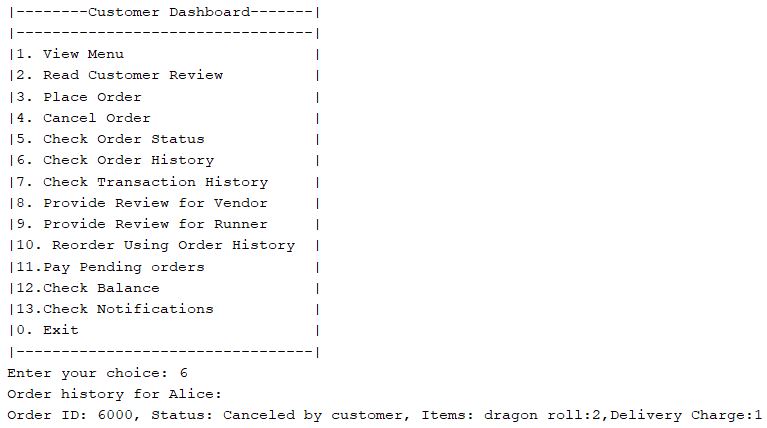


Figure 66.

Here the system offers order history function in case the customer wants to reorder the same things order previously showing the Order ID and the items ordered by the customer.

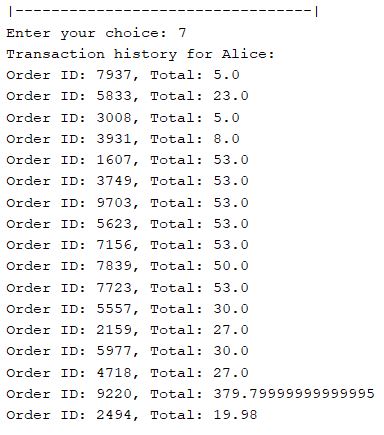


Figure 67.

Here when customer inputs in main menu (7) system automatically displays transaction history with Order ID and amount of money paid as shown in figure above.

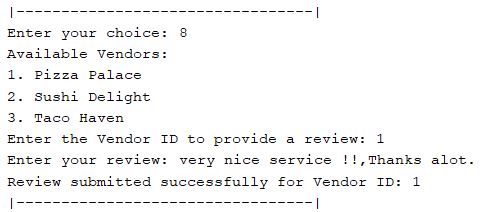


Figure 68.

Program also offers the customers to put reviews for any vendor when choosing (8)in main menu then choosing the Vendor ID that they would like to add review for and asking customer to provide the review where here the customer chose **Vendor 1** and wrote **“very nice service !!, Thanks a lot"** as shown above.

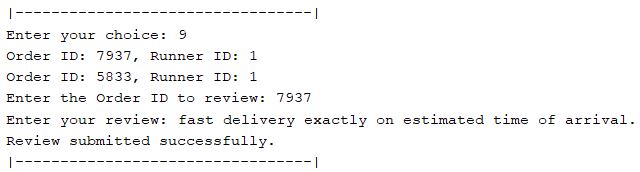


Figure 69.

Same as for vendor review system also offers reviews for delivery runners when customer inputs **(9)** asking customer to write the **Order ID** to review then putting the review where above customer said, “**fast delivery exactly on estimated time of arrival**.” and confirming the review has been submitted successfully as shown above.

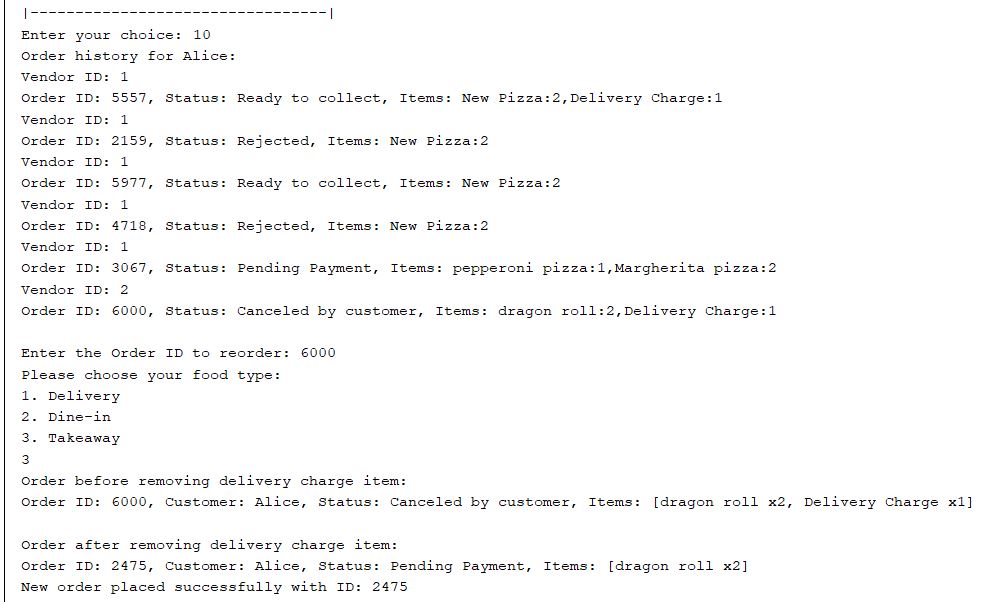


Figure 70.

Here system offers advance function where users can **reorder using history of previous orders** where it would display all previous orders IDs then asking user which Order ID, they want to re-order again then if it would be Delivery, Dine-in, or Takeaway finally confirming the new order details with **new** **Order ID.**

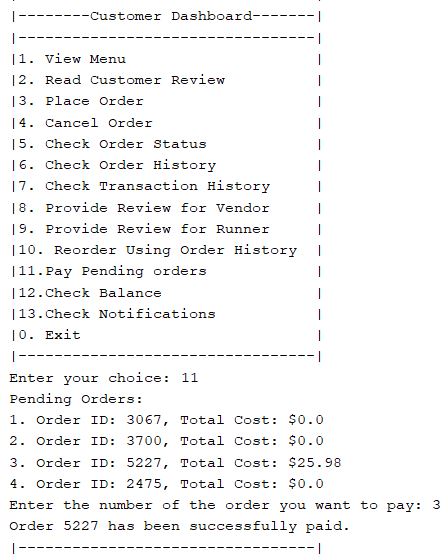


Figure 71.

Lastly, customer can check and pay the pending orders when they input **(11)** the system will automatically show the previous orders and if any **pending money** that needs to be paid , then would ask you to put the number of order you want to pay where above the customer put 3 and **system updated** customer that it **has been paid successfully**.

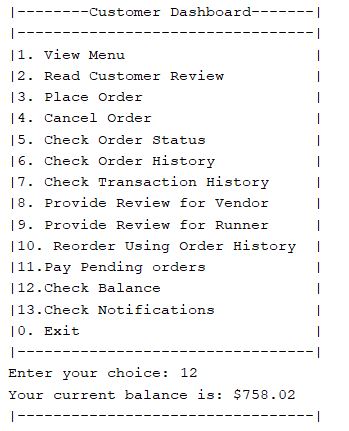


Figure 72.

Finally, for the customer part all users can check their balance from system when inputting (12) the system will return the user the current balance which is automatically calculated from the previous orders and payments customer made before as shown in figure above.

# 5.0 Conclusion

We pause as we complete this project to reflect on the concept, design, and implementation strategy that were applied to create an innovative and user-centered application for Tech University's dynamic meal ordering system. The application's broad features go above and beyond expectations in meeting the various needs of cafeteria providers, finicky customers, loyal delivery runners, and vigilant administrators, in addition to meeting the requirements.

Well-defined roles for each position in the user access hierarchy provide the basis of a secure and efficient system. The administrator-only user registration approach ensures a controlled environment, and the digital storage of user information in a text file enhances data management processes.

The program's menu presentation function makes ordering easier and offers consumers a seamless browsing experience across a variety of cuisines. The implementation of a credit-based payment system enhances financial transactions by providing customers with an easy and secure means of making payments. As a result, managers get the ability to supervise and uphold customer credit, ensuring a stable economic climate.

An essential means of communication for administrators is the in-app notification system, which lets them inform customers and vendors about order statuses and provide digital receipts with ease. By enabling consumers to trace their orders and suppliers to examine revenue patterns and customer comments, the order history tool improves accountability and transparency.

Apart from providing consumers with choices in delivery services, the intricate delivery system, designed to emulate actual occurrences, ensures the automatic assignment of runners and the prompt dissemination of order status updates. An additional level of operational sophistication is provided by the integration of components such as task management, earnings monitoring, and daily, monthly, and annual performance indicators for delivery runners.

In conclusion, the completion of this work demonstrates how well object-oriented programming can handle complex real-world issues. The project offers Tech University a comprehensive application that unifies interactions amongst varied users, improving academic quality and operational efficiency. When the cafeteria suppliers, customers, delivery drivers, and administrators work together, they can build a technologically advanced ecosystem that aligns with the university's objective of innovation and advancement, potentially totally changing the on-campus dining experience.

# 6.0 References

* Dodsworth, D. (2023, April 12). *6 Types of Methods in Java and When to Use Each*. History-Computer. https://history-computer.com/6-types-of-methods-in-java-and-when-to-use-each/
* *What is Java? | IBM*. (2023). Ibm.com. https://www.ibm.com/topics/java
* ‌ *Java OOP (Object-Oriented Programming)*. (2023). W3schools.com. https://www.w3schools.com/java/java\_oop.asp
* *Types of Inheritance in Java - Javatpoint*. (2021). Www.javatpoint.com. https://www.javatpoint.com/types-of-inheritance-in-java
* *Object in Java | Class in Java - javatpoint*. (2021). Www.javatpoint.com. https://www.javatpoint.com/object-and-class-in-java
* Great Learning Team. (2022, September 23). *What is File Handling in Java?* Great Learning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/what-is-file-handling-in-java/
* https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html
* ‌ *Inheritance in Java - Javatpoint*. (2021). Www.javatpoint.com. https://www.javatpoint.com/inheritance-in-java
* ‌ S, R. A. (2020, August 6). *What is Encapsulation in Java and How to Implement It*. Simplilearn.com; Simplilearn. https://www.simplilearn.com/tutorials/java-tutorial/java-encapsulation
* Sumo Logic, Inc. (2023, August 11). *Encapsulation - definition & overview*. Sumo Logic. https://www.sumologic.com/glossary/encapsulation/
* Pankaj. (2022, August 3). *Inheritance in Java example*. DigitalOcean. https://www.digitalocean.com/community/tutorials/inheritance-java-example
* *Java class and Objects (With example)*. (n.d.). https://www.programiz.com/java-programming/class-objects
* Awati, R. (2023, June 19). *polymorphism*. WhatIs. https://www.techtarget.com/whatis/definition/polymorphism