

Food Delivery Services

Presented by: GROUP 1

RONALD TUSIIME

AGABA ITUNGO

M RACHEL ISOOBA

JOY ABAHO

EMMANUEL NSUBUGA



Project Outline

1 Introduction

2 OOP Pillars

3 Functions

4 Delivery Services Guide

5 Food Delivery System Scenario

5 Contact Us



Introduction

UCU students face issues with the unreliable and costly 60 Second Delivery service. To solve this, we propose developing a new food delivery system using Object-Oriented Programming (OOP) principles. This approach will allow us to build a flexible, efficient, and user-friendly platform, making it easier for students to order food from anywhere on campus.



Object-Oriented Programming - Pillars

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism



```
from abc import ABC, abstractmethod
from datetime import datetime
import random
```

Comment Code

```
class OrderingSystem:
    def __init__(self):
        self.__restaurants = []
        self.__customers = []
        self.__DeliveryPersons = []

    def set_deliver(self, instance):
        self.__DeliveryPersons.append(instance)

    def get_deliver(self):
        return self.__DeliveryPersons

    def get_restaurants(self):
        return self.__restaurants

    def set_restaurants(self, instance):
        self.__restaurants.append(instance)
```

Encapsulation

Encapsulation is demonstrated in the code by using private attributes (e.g., `__restaurants`, `__orders`, `__amount`, etc.), which are only accessible through getter and setter methods or specific functions within each class.



```
class OrderingSystem:
    def __init__(self):
        self.__restaurants = []
        self.__customers = []
        self.__DeliveryPersons = []

    def set_deliver(self, instance):
        self.__DeliveryPersons.append(instance)

    def get_deliver(self):
        return self.__DeliveryPersons
```

In the `OrderingSystem` class, `__restaurants`, `__customers`, and `__DeliveryPersons` are encapsulated, and access to them is managed through the `get_` and `set_` methods.

Abstraction

Abstraction is demonstrated with the Members class, which is an abstract class defining an interface for all types of members without implementation details.

```
class Members(ABC):  
    def __init__(self, name, address, number, password):  
        self.__name = name  
        self.__address = address  
        self.__number = number  
        self.__password = password  
  
    @abstractmethod  
    def display(self):  
        pass
```

The display method is an abstract method in the Members class, forcing derived classes (like Customer, Restaurant, and DeliveryPerson) to implement this method based on specific details.



Inheritance

Inheritance is shown through the relationship between the Members superclass and its subclasses Customer, Restaurant, and DeliveryPerson.

```
class Restaurant(Members):  
    def __init__(self, name, address, number, OH, CH, password):  
        super().__init__(name, address, number, password)  
        self.__openingHours = OH  
        self.__closingHours = CH  
        self.__menu = {}  
        self.__orders = []
```

Here, Restaurant inherits from Members using `super().__init__()` to initialize common attributes, while adding specific attributes such as `__openingHours` and `__closingHours`.



Polymorphism

Polymorphism is illustrated by the `process_payment` method, which is implemented differently across various payment classes (`CashPayment`, `CreditCardPayment`, and `MobileMoneyPayment`), each processing payments with unique methods and fees.



```
class MobileMoneyPayment(CashPayment):
    def __init__(self, phone_number):
        self._phone_number = phone_number # Encapsulated attribute for mobile money transactions
        self._otp = None # Placeholder for OTP

    def process_payment(self, order):
        # Assume a fixed UGX 500 fee for mobile money transactions
        total = order.get_total()
        self._amount = total
        processing_fee = 500
        self._amount += processing_fee
        print(f"Mobile Money processing fee applied. Total amount: UGX {self._amount}")
```

Each `process_payment` method in `CashPayment`, `CreditCardPayment`, and `MobileMoneyPayment` provides different implementations, showcasing polymorphism by allowing `process_payment` to be used across different payment types with varying behavior.



Functions

1. **SettingRestaurants()**

- Gathers details for creating a new restaurant, such as name, address, opening and closing times, and password. It then sets the restaurant's menu and adds the restaurant to the OrderingSystem instance Mukono.

4. **CreatingCustomers()**

- Collects details to create a new customer, such as name, address, and password, and adds the customer to the OrderingSystem instance Mukono.

6. **Ordering(customer)**

- Facilitates the ordering process. The customer selects a restaurant and the number of items to order. Then, the customer chooses a payment method, and the respective payment method function is called. Finally, it displays the updated order information.

8. **card_payment(order)**

- Processes payment through a credit card. Adds a processing fee, marks the payment as "Paid," and returns the payment information.

10. **CreateDeliveryGuy()**

- Creates a new delivery person by collecting their details (name, address, phone number, and password) and adds them to the OrderingSystem instance Mukono.

7. **process_payment(order)**

- Processes payment using Mobile Money. It verifies the payment through an OTP (One Time Password), setting the payment status as "Paid" if verified successfully, otherwise "Failed."

9. **cashpayment(order)**

- Processes cash payment by prompting the user to input the cash amount, then compute change if any, and marks the transaction as complete or pending based on the payment amount.

2. **Orderupdate_Restaurant(restaurant)**

- Allows the restaurant to update the status of an order by prompting the user for the order number and status (approved or canceled).

3. **assigndeliverer()**

- Displays a list of available delivery persons and allows the user to select one. Returns the chosen delivery person.





Functions

1. **menu0()**

- Displays an initial menu asking the user to select their role (Restaurant, Customer, or Delivery) or exit. Returns the user's choice.

1. **menu1()**

- Displays a menu for login or signup options and returns the user's choice.

1. **menu3()**

- Displays a customer menu with options to make an order, track an order, or exit. Returns the user's choice.


1. **menu2()**

- Displays a menu for restaurants with options to display orders, update order status, or exit, returning the user's choice.

1. **menu4()**

- Displays a menu for delivery persons, providing options to view orders, update orders, record cash payments, or exit. Returns the user's choice.

11. **Login(memberlist)**

- Authenticates a user by matching their name and password with entries in the provided memberlist. Returns the member if successful, allows re-entry on failure, or gives the option to exit.
- 



Delivery Services Guide

1

As a Restaurant

2

As a Customer

3

As a Delivery person

Start the App

Guide to Using the Restaurant Delivery Service App

This guide assumes you have already set up the necessary classes and modules (`Attempt1`, `Attempt2`, `Attempt3`) correctly, as these handle different parts of the application's functionality.

Starting the App

1. **Run the Application:** Start the app by running the script. You should see:

```
'''
```

```
Welcome to The Restaurant Delivery Service App
```

```
'''
```

2. **Choose Your Role:** The app will ask you to choose your role:

```
'''
```

```
Which are you?
```

```
- Restaurant
```

```
- Customer
```

```
- Delivery
```

```
enter 'exit' to exit
```

```
'''
```

Enter one of the options to continue or type `exit` to close the app.

Restaurant

Using the App as a Restaurant

1. **Sign Up or Log In:** After selecting `restaurant`, you'll see:

...

Menu

- **Login**
- **Signup**
- **Exit**

...

- **Signup:** Type `signup` to register a new restaurant.
 - Enter details as prompted, like the restaurant's name, address, phone number, opening and closing times, and password.

- Set up the restaurant menu by specifying the number of items, item names, and prices.
- **Login:** Type `login` if you're an existing restaurant.
 - Enter the restaurant name and password to log in.

2. Manage Orders:

After logging in, you'll see:

...

Menu

- **D to display Orders**
- **U to update order status**
- **E to exit**

...

- **Display Orders:** Type `d` to display all orders received by the restaurant.
- **Update Order Status:** Type `u` to change the status of an order.
 - Enter the order number and new status (`Approved`, `Cancelled`, etc.) as prompted.
- **Exit:** Type `e` to exit the restaurant menu.

Customer

Using the App as a Customer

1. Sign Up or Log In: After selecting `customer`, you'll see:

...

Menu

- Login
- Signup
- Exit

...

- **Signup:** Type `signup` to register as a new customer.
 - Enter details like your name, address, phone number, and password.
- **Login:** Type `login` if you're an existing customer.
 - Enter your name and password to log in.

2. Place or Track Orders:

After logging in, you'll see:

...

Menu

- order to Make an order

- track to Track an order
- exit to exit

...

- **Place an Order:** Type `order` to start a new order.
 - Choose a restaurant from the list displayed.
 - Enter the items you want to order and the quantity.
 - Select a payment method (`cash`, `mobile`, or `card`).
 - For **mobile:** Enter your phone number, and follow the OTP verification process.
 - For **card:** Enter your card number and CVV.
 - For **cash:** Confirm that you'll pay upon delivery.
- **Track an Order:** Type `track` to check the status of your most recent order.
- **Exit:** Type `exit` to exit the customer menu.

Delivery Person

Using the App as a Delivery Person

1. **Sign Up or Log In:** After selecting `delivery`, you'll see:

...

Menu

- Login
- Signup
- Exit

...

- **Signup:** Type `signup` to register as a new delivery person.
 - Enter details like your name, address, phone number, and password.
- **Login:** Type `login` if you're an existing delivery person.
 - Enter your name and password to log in.

2. Manage Deliveries:

After logging in, you'll see:

...

Menu

- view to view orders
- update to update orders
- pay to record cash payment
- exit to exit

...

- **View Orders:** Type `view` to display all assigned orders.
- **Update Order Status:** Type `update` to mark an order as delivered or update its status.
- **Record Cash Payment:** Type `pay` to register a cash payment.
 - Enter the order number and confirm payment details as prompted.
- **Exit:** Type `exit` to exit the delivery menu.

Example Scenario

Food Delivery System Scenario

Interaction Flow for Restaurant, Customer, and Delivery Roles

1. Restaurant Registration and Menu Setup

- **Step 1:** Launch the app and choose "restaurant."
 - Options:
 - Restaurant
 - Customer
 - Delivery
 - Type `exit` to exit the app.
- **Step 2:** Select `signup` to register as a new restaurant.
 - **Enter Restaurant Details:**
 - Name: `Touch of Class`
 - Address: `Turker road`
 - Phone Number: `256700123456`
 - Opening Time: `10:00`
 - Closing Time: `22:00`
 - Password: `password123`
 - Set the Menu:
 - Number of items: 3
 - Menu items with prices:
 - `Spaghetti Carbonara` - 15000 UGX
 - `Penne Arrabbiata` - 14000 UGX
 - `Lasagna` - 20000 UGX
- **Step 3:** Log out and return to the main menu.

Example Scenario

2. Customer Registration and Placing an Order

- **Step 1:** Go to the main menu, choose ``customer``.
- **Step 2:** Select ``signup`` to create a new customer profile.
 - Enter Customer Details:
 - Name: ``Mawejje Yokana``
 - Address: ``Nsibambi Hall``
 - Phone Number: ``256700987654``
 - Password: ``password456``
- **Step 3:** Log in as a customer, choose ``order`` to place an order.
 - Choose a restaurant (options displayed):
 - ``Touch of Class``
 - ``Canopy``
 - Enter the number of items: 2
 - Select items from the menu:
 - ``Spaghetti Carbonara``
 - ``Lasagna``
 - Payment Options:
 - Payment method: ``mobile``
 - Phone number: ``256700987654``
 - Confirm payment with OTP:
 - OTP: ``123456``
 - Confirmation: "OTP verified successfully. Order placed successfully!"

Example Scenario

3. Delivery Person Registration

- **Step 1:** Go to the main menu, choose `delivery`.
- **Step 2:** Select `signup` to create a new delivery profile.
 - **Enter Delivery Person Details:**
 - Name: `Geno Joshua`
 - Address: `Food Delivery Bugujju`
 - Phone Number: `256700234567`
 - Password: `delivery789`
- Step 3:** Log out and return to the main menu.

4. Restaurant Approval

- **Step 1:** Go to the main menu, choose `restaurant`.
- Step 2:** Log in, choose `d` to display orders.
- **Step 3:** select the order number and approve it.
- **Step 4:** select an available delivery guy.
- **Step 5:** Log out and return to the main menu.

Example Scenario

5. Delivering the Order

- **Step 1:** Go to the main menu, choose `delivery`.
- **Step 2:** Log in and choose [`view`](#) to see assigned orders.
 - [Order Details:](#)
 - [Order #1](#)
 - [Customer: Mawejje Yokana](#)
 - [Items: `Spaghetti Carbonara`, `Lasagna`](#)
 - [Total: 35000 UGX](#)
 - [Delivery Address: `Nsibambi Hall`](#)
 - [Status: Pending](#)
- **Step 3:** Choose update, select the order number, then change order status to `Delivered`.
- **Step 4:** Record payment if cash was collected, inputting the amount: 35000 UGX.

6. Tracking Order as a Customer

-
- From the customer menu, choose `track` to view order status.
 - Status: `Delivered`

Example Scenario

7. Logging Out

- Typing `exit` logs out of the current role and returns to the main menu.

8. Final Flow Example

1. Restaurant registers and sets up a menu, then logs out.
2. Customer registers, places an order with mobile payment, and confirms it via OTP.
3. Delivery Person registers, views and delivers the order, and records cash payment.
4. Customer tracks the order status as 'Delivered'.

This scenario provides a cycle that facilitates interactions among restaurants, customers, and delivery personnel.



Contact Us



+256 770 348006



HELLO@FOODDELIVERY.COM



WWW.FOODDELIVERY.COM



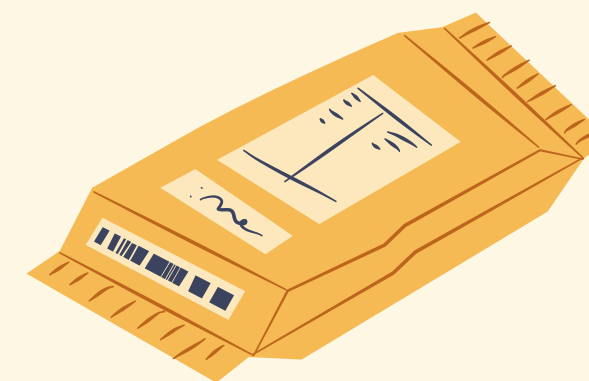
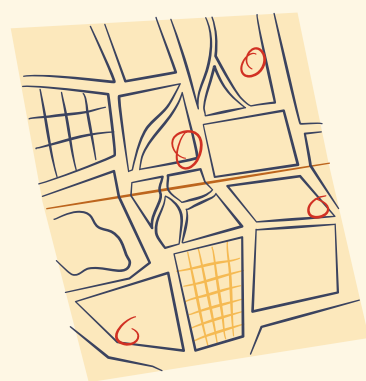
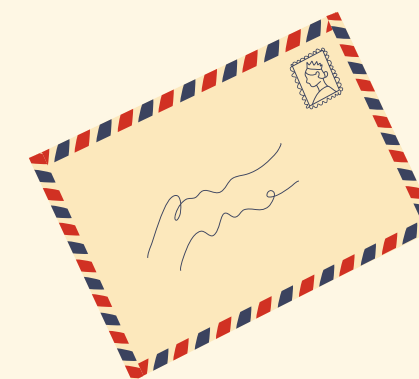
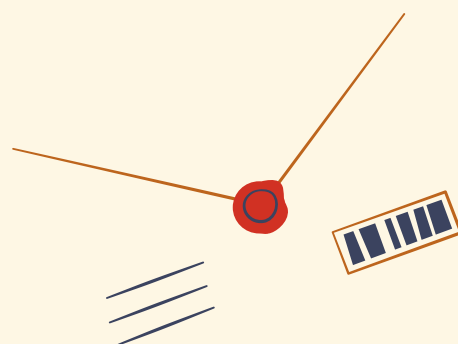
4 BISHOP TURKER., MUKONO CITY

Thank you for listening!

Don't hesitate to ask any questions!



Free Resource Page



Free Resource Page

