Assignment 1 - Programming Fundamentals

Abdulla Albraiki 202220593

Identifying Use Cases

A use case is a step-by-step explanation of what the system does. In this system, the main actions are:

Placing an Order → The customer selects an item and orders it.

Making a Payment → The customer pays for their order.

Tracking the Delivery → The customer checks the delivery status.

Use-Case Diagram

A use-case diagram is a simple drawing that shows who is involved and what they do. It includes:

Actors (Stick Figures)

Customer (places orders and tracks them)

System (manages orders and payments)

Payment System (processes payments)

Ovals for Actions

Place Order

Make Payment

Track Delivery

Arrows to Show Connections

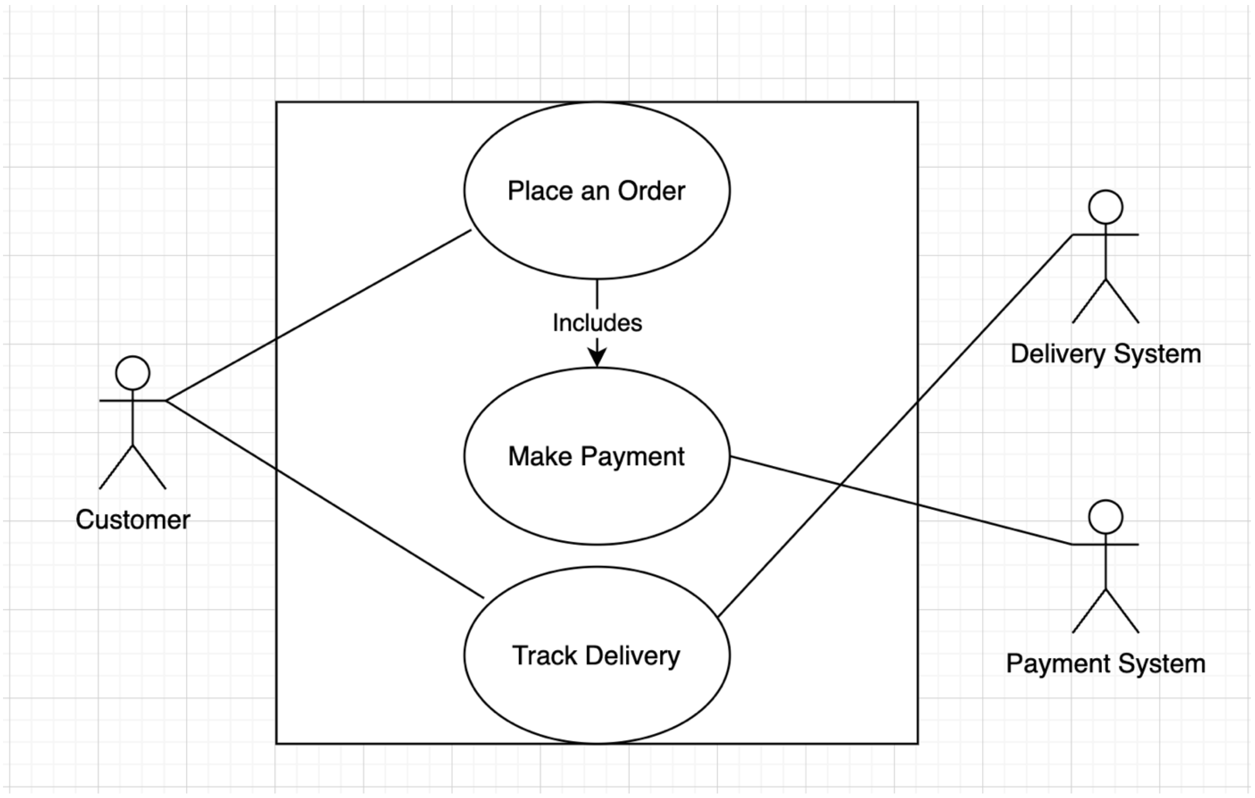
Customer → Place Order

Customer → Track Delivery

Place Order → Includes → Make Payment

Use-Case Table

Use Case	What Happens?
Place Order	The customer selects an item and orders it.
Make Payment	The customer completes payment.
Track Delivery	The customer checks the delivery status.



Identifying Objects and Classes

An object is anything that has details like a customer or an order. A class is a blueprint for creating objects.

The system needs 3 main classes:

Customer → Stores customer details.

Order → Stores order details.

Payment → Stores payment details.

Class Diagram

A class diagram is a simple drawing that shows how different parts of the system connect.

Draw boxes for:

Customer (name, email, address)

Order (order ID, status)

Payment (payment ID, status)

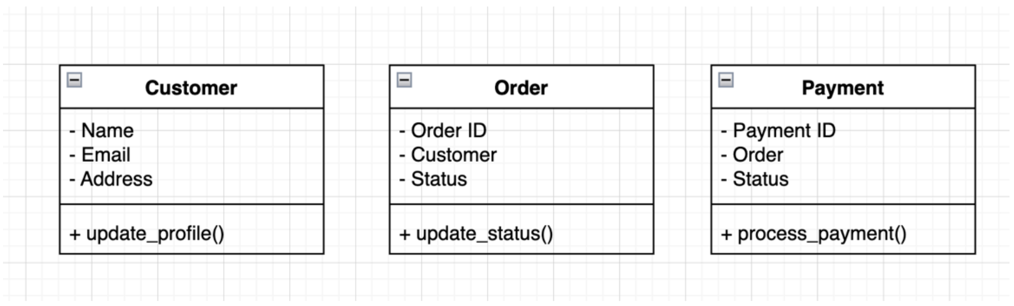
Draw arrows to show relationships:

Customer → Order (places an order)

Order → Payment (is paid for)

Class Table

Class	What it Stores	What it Does
Customer	Name, Email, Address	Update details
Order	Order ID, Status	Change status
Payment	Payment ID, Status	Process payment



Python code

```
# Enums for Order and Payment Status
from enum import Enum


class OrderStatus(Enum):
    PENDING = "Pending"
    SHIPPED = "Shipped"
    DELIVERED = "Delivered"


class PaymentStatus(Enum):
    PENDING = "Pending"
    COMPLETED = "Completed"
    FAILED = "Failed"


# Class: Customer
class Customer:
    def __init__(self, name, email, address):
        self.__name = name
        self.__email = email
        self.__address = address


    def get_name(self): return self.__name
    def get_email(self): return self.__email
    def get_address(self): return self.__address
```

```
# Class: Order
class Order:
    def __init__(self, order_id, customer):
        self.__order_id = order_id
        self.__customer = customer
        self.__status = OrderStatus.PENDING

    def get_order_id(self): return self.__order_id
    def get_status(self): return self.__status
    def update_status(self, new_status): self.__status = new_status

# Class: Payment
class Payment:
    def __init__(self, payment_id, order):
        self.__payment_id = payment_id
        self.__order = order
        self.__status = PaymentStatus.PENDING

    def get_status(self): return self.__status
    def process_payment(self): self.__status = PaymentStatus.COMPLETED

# Creating an order example
customer = Customer("Ahmed Jassim", " Ahmed. Jassim@gmail.com", "Dubai, UAE")
order = Order("Order 11", customer)
payment = Payment("Payment 11", order)
```

```
# Processing payment and updating status

payment.process_payment()

order.update_status(OrderStatus.SHIPPED)


# Printing Delivery Note

print(" Delivery Note")

print("Thank you for shopping with us!")


print("Order Number:", order.get_order_id())

print("Customer Name:", customer.get_name())

print("Customer Email:", customer.get_email())

print("Delivery Address:", customer.get_address())

print("Order Status:", order.get_status().value)

print("Payment Status:", payment.get_status().value)
```

The output:

```
Delivery Note Thank you for shopping with us! Order Number: Order 11 Customer Name:
Ahmed Jassim Customer Email: Ahmed.Jassim@gmail.com Delivery Address: Dubai, UAE Order
Status: Shipped Payment Status: Completed
```

Reflection

Through this assignment, I learned how to plan a system using UML diagrams, write Python classes, and connect different parts of a program. I practiced object-oriented programming by creating classes for customers, orders, and payments. I also understood how to use enums for different statuses and how to update and manage data in a structured way. Writing code for a real-world scenario like a delivery system helped me see how programming is used in everyday applications. This assignment also improved my ability to write clean and organized code, making it easier to understand and use.

Github link

https://github.com/Abdullaalbraiki/Abdullaalbraiki.git