Delivery Management System

Abdulrahman Alzaabi

ICS220 Program. Fund.

Areej Abdulfattah

1. UML Use-Case Diagram and Description

Use-Cases Identified:

Create Delivery Order: User enters delivery details (recipient info, order number, items, etc.).

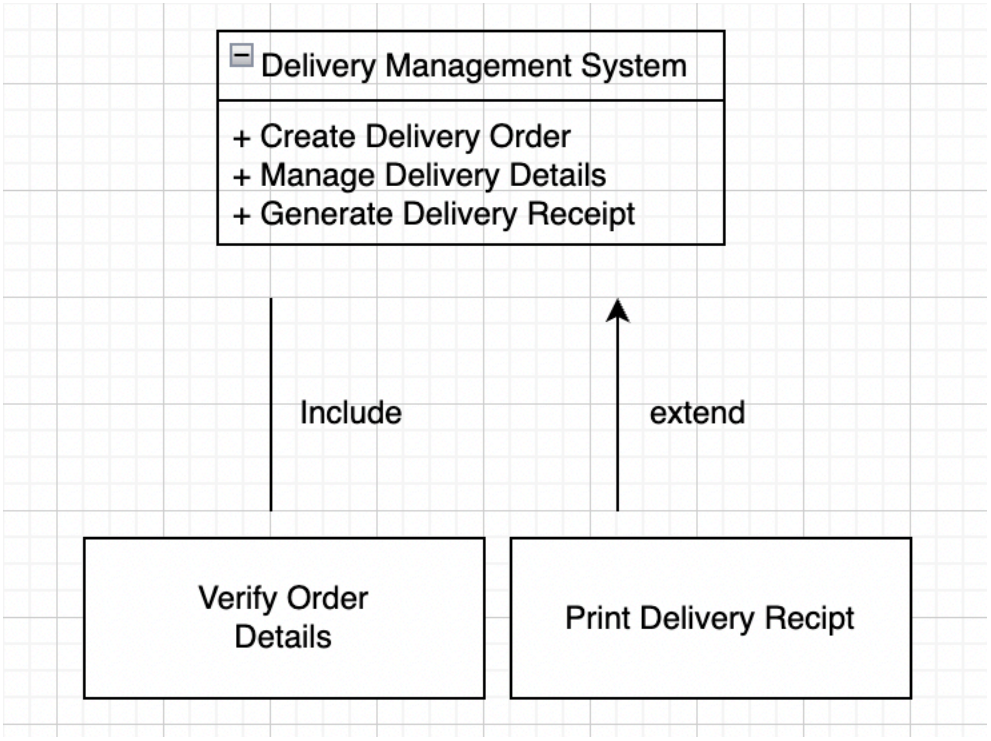
Manage Delivery Details: System saves and updates delivery information (address, dates, package details).

Generate Delivery Receipt: The system produces a delivery note for printing.

Additional scenarios:

Include: "Verify Order Details" is always performed before an order is processed.

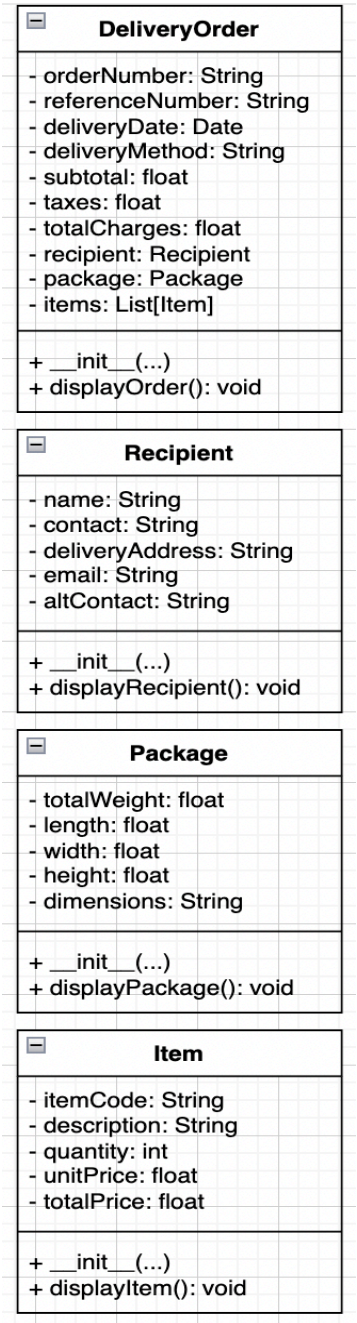
Extend: "Print Delivery Receipt" is an optional action when the user requests a hard copy.



UML Class Diagram and Description

Based on the delivery note sample, we identified these classes:

DeliveryOrder – Holds order number, reference number, delivery date, delivery method, charges, and is linked with Recipient, Package, and Items.
Recipient – Holds recipient name, contact, delivery address, email, and possibly an alternative contact.
Package – Contains package details like total weight and dimensions.
Item – Each item has an item code, description, quantity, unit price, and total price.



Explanation:

- DeliveryOrder is the main class that holds overall delivery details.
- Recipient contains the customer’s personal delivery information.
- Package stores details about the shipment, such as weight and dimensions.
- Item stores each delivered product's details.

Python code:

```
from datetime import date

class DeliveryOrder:
    def __init__(self, orderNumber, referenceNumber, deliveryDate, deliveryMethod, subtotal, taxes, totalCharges, recipient, package, items):
        self._orderNumber = orderNumber
        self._referenceNumber = referenceNumber
        self._deliveryDate = deliveryDate
        self._deliveryMethod = deliveryMethod
        self._subtotal = subtotal
        self._taxes = taxes
        self._totalCharges = totalCharges
        self._recipient = recipient
        self._package = package
        self._items = items
    def displayOrder(self):
        print("Order Number:", self._orderNumber)
        print("Reference Number:", self._referenceNumber)
        print("Delivery Date:", self._deliveryDate)
        print("Delivery Method:", self._deliveryMethod)
        print("Subtotal:", self._subtotal)
        print("Taxes:", self._taxes)
        print("Total Charges:", self._totalCharges)
        self._recipient.displayRecipient()
        self._package.displayPackage()
        for item in self._items:
            item.displayItem()

class Recipient:
    def __init__(self, name, contact, deliveryAddress, email, altContact=""):
        self._name = name
        self._contact = contact
        self._deliveryAddress = deliveryAddress
        self._email = email
        self._altContact = altContact
    def displayRecipient(self):
        print("Recipient Name:", self._name)
        print("Contact:", self._contact)
        print("Delivery Address:", self._deliveryAddress)
```

```
        print("Email:", self._email)

class Package:
    def __init__(self, totalWeight, length, width, height):
        self._totalWeight = totalWeight
        self._length = length
        self._width = width
        self._height = height
        self._dimensions = f"{length}x{width}x{height}"
    def displayPackage(self):
        print("Total Weight:", self._totalWeight, "kg")
        print("Dimensions:", self._dimensions)

class Item:
    def __init__(self, itemCode, description, quantity, unitPrice, totalPrice):
        self._itemCode = itemCode
        self._description = description
        self._quantity = quantity
        self._unitPrice = unitPrice
        self._totalPrice = totalPrice
    def displayItem(self):
        print("Item Code:", self._itemCode)
        print("Description:", self._description)
        print("Quantity:", self._quantity)
        print("Unit Price:", self._unitPrice)
        print("Total Price:", self._totalPrice)

recipient = Recipient("Sarah Johnson", "sarah.johnson@example.com", "45 Knowledge Avenue,
Dubai, UAE", "sarah.johnson@example.com")
package = Package(7, 30, 20, 15)
item1 = Item("ITM001", "Wireless Keyboard", 1, 100.00, 100.00)
item2 = Item("ITM002", "Wireless Mouse & Pad Set", 1, 75.00, 75.00)
item3 = Item("ITM003", "Laptop Cooling Pad", 1, 120.00, 120.00)
item4 = Item("ITM004", "Camera Lock", 3, 15.00, 45.00)
items = [item1, item2, item3, item4]
order = DeliveryOrder("DEL123456789", "DN-2025-001", date(2025,1,25), "Courier", 270.00,
13.50, 283.50, recipient, package, items)
order.displayOrder()
```

Output:

```
Quantity: 1
Unit Price: 120.0
Total Price: 120.0
Item Code: ITM004
Description: Camera Lock
Quantity: 3
Unit Price: 15.0
Total Price: 45.0
```

Summary of Learnings:

In this assignment I have learned how to analyze a real-world problem and design both use-case and class diagrams. I also practiced writing clear Python code and implementing object-oriented principles and this has helped me understand how to map real-world entities to software design.