Delivery Management System

Osha Albedwawi - 202317982

ICS220- 21383 Program. Fund.

Prof. Leonce

Table of Contents:

Use case Analysis

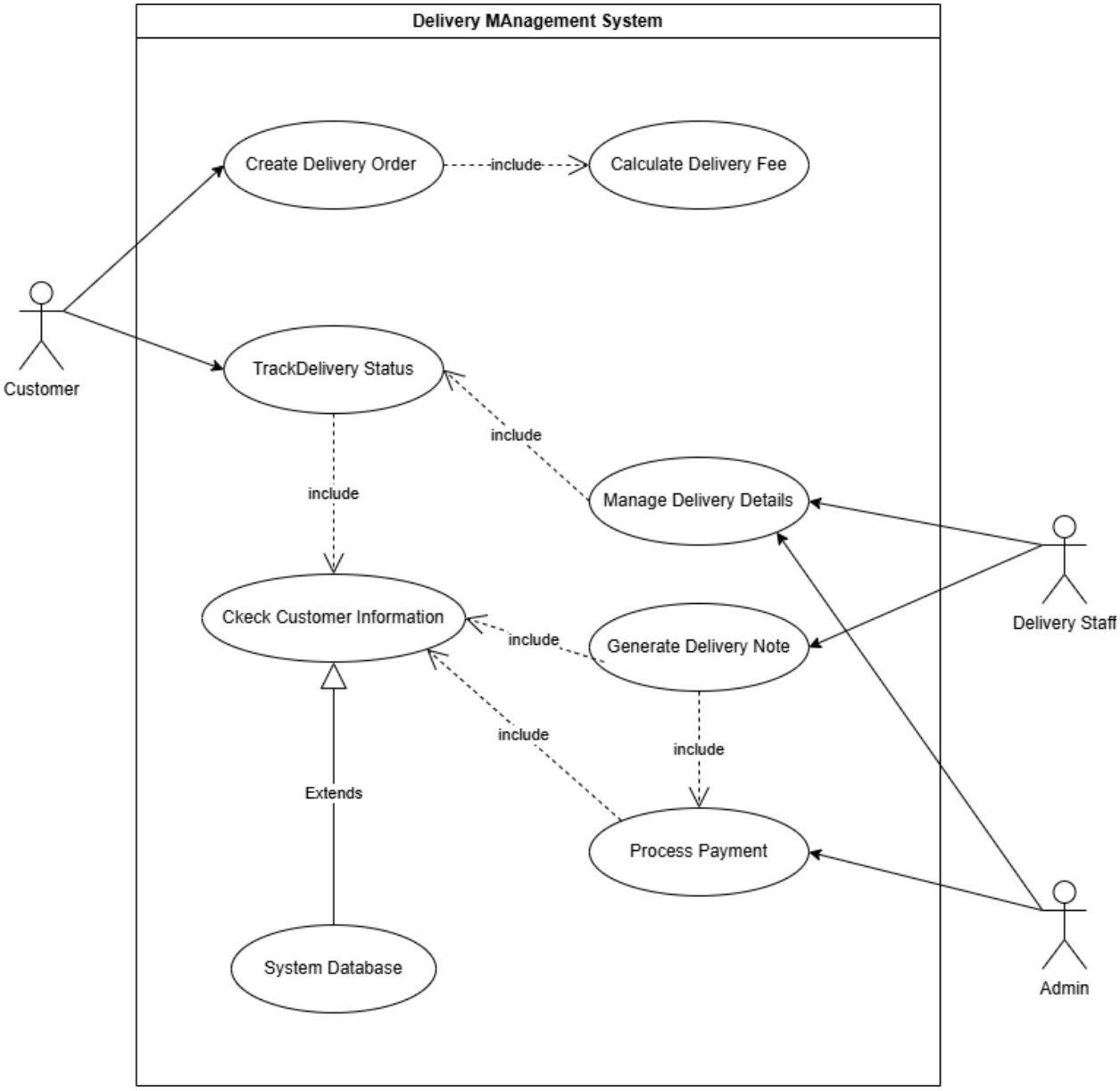
Key Use cases

1. Create Delivery Order
2. Manage Delivery Details
3. Generate Delivery Note
4. Track Delivery Status
5. Process Payment
6. Manage Customer Information
7. Assign Delivery Staff

Actors

1. Customer
2. Admin
3. Delivery staff

Use case diagram:



Use case description

Use case 1: Create Delivery Order

Aspect	Description
Use Case Name	Create Delivery Order
Actor	Customer
Description	Customer creates a new delivery order in the system
Preconditions	Customer must have valid credentials
Main Flow	1. Customer selects "Create New Order" 2. System displays order form 3. Customer enters delivery information 4. System calculates delivery fee 5. Customer confirms order
Alternative Flows	4a. If information is invalid, system prompts correction
Postconditions	New delivery order is created in the system
Include Relationships	Check Customer Information, Calculate Delivery Fee
Extend Relationships	None

Use case 2: Manage Delivery Details

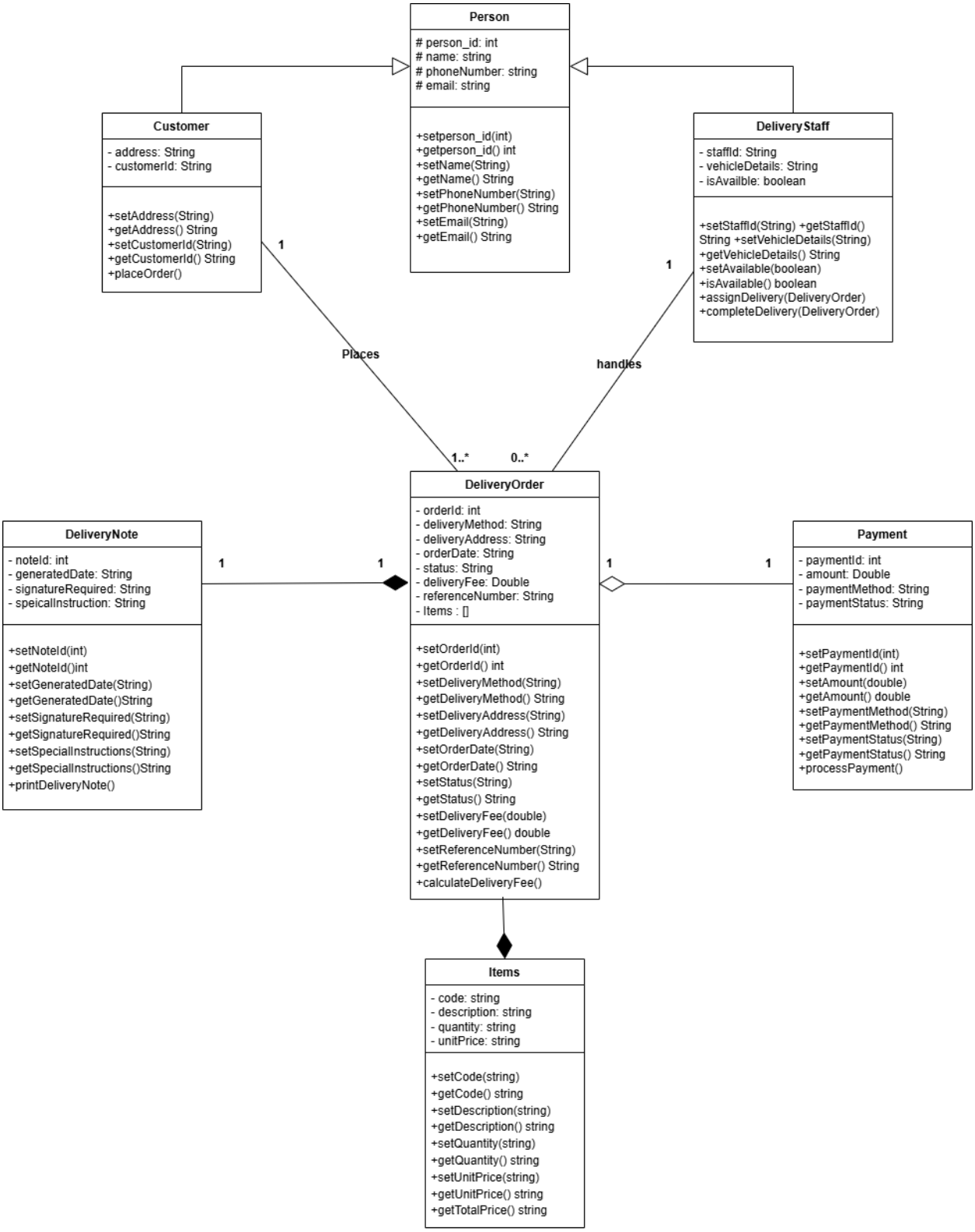
Aspect	Description
Use Case Name	Manage Delivery Details
Actor	Delivery Staff, Admin
Description	Update and manage delivery information
Preconditions	Delivery order must exist in the system
Main Flow	1. Staff/Admin selects delivery order 2. System displays delivery details 3. Staff/Admin updates information 4. System saves changes
Alternative Flows	3a. If updates are invalid, system shows error
Postconditions	Delivery details are updated
Include Relationships	None
Extend Relationships	Track Delivery Status

Use case 3: Generate Delivery Note

Aspect	Description
Use Case Name	Generate Delivery Note
Actor	Delivery Staff
Description	Create a delivery note document for an order
Preconditions	Delivery order must exist with all details
Main Flow	1. Staff selects "Generate Delivery Note" 2. System retrieves delivery details 3. System formats information into note 4. System displays the delivery note
Alternative Flows	2a. If information is missing, system shows error
Postconditions	Delivery note is generated

Include Relationships	Check Customer Information
Extend Relationships	Process Payment

Class Diagram:



Class explanation:

Person:

Abstract parent class containing basic personal information (id, name, contact details). Inherited by Customer and DeliveryStaff classes.

Customer:

Extends Person with address and customerId. Places DeliveryOrders (1-to-many relationship). This is the child class of Person the relation know as inheritance.

DeliveryStaff:

Extends Person with staffId, vehicleDetails, and availability status. Handles multiple DeliveryOrders (1-to-many relationship). This is the child class of Person the relation know as inheritance.

DeliveryOrder:

Central class connecting all entities. Contains order details and has relationships with Customer (placed by), DeliveryStaff (handled by), Payment, DeliveryNote, and Items.

DeliveryNote:

Contains delivery instructions and signature requirements. Has a one-to-one relationship with DeliveryOrder. And has strong relation with it known as composition.

Payment:

Tracks payment information including amount and status. Has a one-to-one relationship with DeliveryOrder. Payment class have relation with DeliveryOrder which is known as aggregation it is weaker relation as compared to composition.

Items:

Contains product details (code, description, quantity, price). Has a composition relationship with DeliveryOrder (DeliveryOrder contains Items). Items are required for Delivery order that is why the relation is strong called composition.

Python Code:

```
# Base class for people in the delivery system
class Person:
    def __init__(self, person_id, name, phoneNumber, email):
        # protected
        self._person_id = person_id
        self._name = name
        self._phoneNumber = phoneNumber
        self._email = email

    def getperson_id(self):
        return self._person_id

    def getName(self):
        return self._name

    def getPhoneNumber(self):
        return self._phoneNumber

    def getEmail(self):
        return self._email

    # Setter methods
    def setperson_id(self, person_id):
        self._person_id = person_id

    def setName(self, name):
        self._name = name

    def setPhoneNumber(self, phoneNumber):
        self._phoneNumber = phoneNumber

    def setEmail(self, email):
        self._email = email

# Class representing a customer who places delivery orders
class Customer(Person):
    def __init__(self, person_id, name, phoneNumber, email, address, customerId):
        super().__init__(person_id, name, phoneNumber, email)
        self.address = address
        self.customerId = customerId

    def getAddress(self):
        return self.address
```

```
    def getCustomerId(self):
        return self.customerId

    # Setter methods
    def setAddress(self, address):
        self.address = address

    def setCustomerId(self, customerId):
        self.customerId = customerId

    def placeOrder(self):
        print("Placing order...")

# Class representing a delivery staff member
class DeliveryStaff(Person):
    def __init__(self, person_id, name, phone, email, staffId, vehicleDetails):
        super().__init__(person_id, name, phone, email)
        self.staffId = staffId
        self.vehicleDetails = vehicleDetails
        self.isAvailble = True

    def getStaffId(self):
        return self.staffId

    def getVehicleDetails(self):
        return self.vehicleDetails

    def isAvailable(self):
        return self.isAvailble

    def setAvailable(self, status):
        self.isAvailble = status

    # Setter methods
    def setStaffId(self, staffId):
        self.staffId = staffId

    def setVehicleDetails(self, vehicleDetails):
        self.vehicleDetails = vehicleDetails

    def setAvailable(self, available):
        self.isAvailable = available

    def assignDelivery(self, order):
        print("Assigning Delivery for order id: ", order.orderId)
```

```
    def completeDelivery(self, order):
        print("Complete Delivery for order id: ", order.orderId)

# Class representing an item in the delivery order
class Item:
    def __init__(self, code, description, quantity, unitPrice):
        self.code = code
        self.description = description
        self.quantity = quantity
        self.unitPrice = unitPrice

    def setCode(self, code):
        self.code = code

    def setDescription(self, description):
        self.description = description

    def setQuantity(self, quantity):
        self.quantity = quantity

    def setUnitPrice(self, unitPrice):
        self.unitPrice = unitPrice

    def getCode(self):
        return self.code

    def getDescription(self):
        return self.description

    def getTotalPrice(self):
        return self.quantity * self.unitPrice


# Class representing a delivery order
class DeliveryOrder:
    def __init__(self, orderId, deliveryMethod, deliveryAddress, orderDate, items, referenceNumber):
        self.orderId = orderId
        self.deliveryMethod = deliveryMethod
        self.deliveryAddress = deliveryAddress
        self.orderDate = orderDate
        self.status = "Pending"
        self.deliveryFee = 0.0
        self.items = items
        self.referenceNumber = referenceNumber

    # Getter methods
    def getOrderId(self):
```

```
    return self.orderId

def getReferenceNumber(self):
    return self.referenceNumber

def getDeliveryMethod(self):
    return self.deliveryMethod

def getDeliveryAddress(self):
    return self.deliveryAddress

def getOrderDate(self):
    return self.orderDate

def getStatus(self):
    return self.status

def getDeliveryFee(self):
    return self.deliveryFee

def getPackageDetails(self):
    return self.packageDetails

# Setter methods
def setOrderId(self, orderId):
    self.orderId = orderId

def setReferenceNumber(self, referenceNumber):
    self.referenceNumber = referenceNumber

def setDeliveryMethod(self, deliveryMethod):
    self.deliveryMethod = deliveryMethod

def setDeliveryAddress(self, deliveryAddress):
    self.deliveryAddress = deliveryAddress

def setOrderDate(self, orderDate):
    self.orderDate = orderDate

def setStatus(self, status):
    self.status = status

def setDeliveryFee(self, deliveryFee):
    self.deliveryFee = deliveryFee

def setPackageDetails(self, packageDetails):
    self.packageDetails = packageDetails
```

```
    def calculateDeliveryFee(self):
        pass


# Class representing a delivery note
def generate_sample_delivery_note():
    customer = Customer(1, "Sarah Johnson", "555-123-4567", "sarah.johnson@example.com", " 45
Knowledge Avenue, Dubai, UAE", "CUST1001")
    items = [
        Item("ITM001", "Wireless Keyboard", 1, 100.00),
        Item("ITM002", "Wireless Mouse & Pad Set", 1, 75.00),
        Item("ITM003", "Laptop Cooling Pad", 1, 120.00),
        Item("ITM004", "Camera Lock", 3, 15.00)
    ]
    order = DeliveryOrder(5001, "Courier", "789 Residential Blvd", "2025-02-26", items, "DN-2025-01")
    order.setDeliveryFee(13.50)
    staff = DeliveryStaff(2, "Michael Johnson", "555-987-6543", "michael.j@deliveryco.com",
"STAFF301", "White Van")

    print("=" * 50)
    print("DELIVERY NOTE")
    print("Thank you for using our delivery service! Please print your delivery receipt and present it upon
receiving your items.")
    print("=" * 50)
    print("Recipient Details:")
    print("Name:", customer.getName())
    print("Contact:", customer.getEmail())
    print("Delivery Address:", order.deliveryAddress)

    print("=" * 50)
    print("Delivery information")
    print("ORDER Number:", order.orderId)
    print("Reference Number:", order.getReferenceNumber())
    print("Delivery Date:", order.orderDate)
    print("Delivery Method:", order.getDeliveryMethod())
    print("DELIVERY STAFF:", staff.getName(), "|", staff.getVehicleDetails())
    print("-" * 50)
    print("SUMMARY OF ITEMS DELIVERED:")
    print("Item Code,", "Description,", "Qty,", "Unit Price (AED),", "Total Price (AED)")

    print("-" * 75)
    subtotal = 0
    for item in order.items:
        total_price = item.getTotalPrice()
        subtotal += total_price
        print(item.code, ",", item.description, ",", item.quantity, ",", round(item.unitPrice, 2), ",",
```

```
            round(total_price, 2))

    print("-" * 75)
    print("Subtotal: AED", round(subtotal, 2))
    print("Taxes and Fees: AED", round(order.deliveryFee, 2))
    print("Total Charges: AED", round(subtotal + order.deliveryFee, 2))

    print("=" * 50)

generate_sample_delivery_note()
```

```
DELIVERY NOTE
Thank you for using our delivery service! Please print your delivery receipt and present it upon receiving your items.
==================================================
Recipient Details:
Name: Sarah Johnson
Contact: sarah.johnson@example.com
Delivery Address: 789 Residential Blvd
==================================================
Delivery information
ORDER Number: 5001
Reference Number: DN-2025-01
Delivery Date: 2025-02-26
Delivery Method: Courier
DELIVERY STAFF: Michael Johnson | White Van
--------------------------------------------------
SUMMARY OF ITEMS DELIVERED:
Item Code, Description, Qty, Unit Price (AED), Total Price (AED)
---------------------------------------------------------------------------
ITM001 , Wireless Keyboard , 1 , 100.0 , 100.0
ITM002 , Wireless Mouse & Pad Set , 1 , 75.0 , 75.0
ITM003 , Laptop Cooling Pad , 1 , 120.0 , 120.0
ITM004 , Camera Lock , 3 , 15.0 , 45.0
---------------------------------------------------------------------------
Subtotal: AED 340.0
Taxes and Fees: AED 13.5
Total Charges: AED 353.5
==================================================

Process finished with exit code 0
```