DIGITAL ASSIGNMENT - 4

Project Documentation - 4

22BCE0544

APRAJITA NANDKEULIAR

Project Title

SwiftDrop: Ultra-fast Delivery Logistics System

UML Use case Model

A UML Use Case Model represents the interactions between users (actors) and a system by defining its functionalities through use cases. It helps visualize how different stakeholders interact with the system, ensuring that all functional requirements are covered. The model includes:

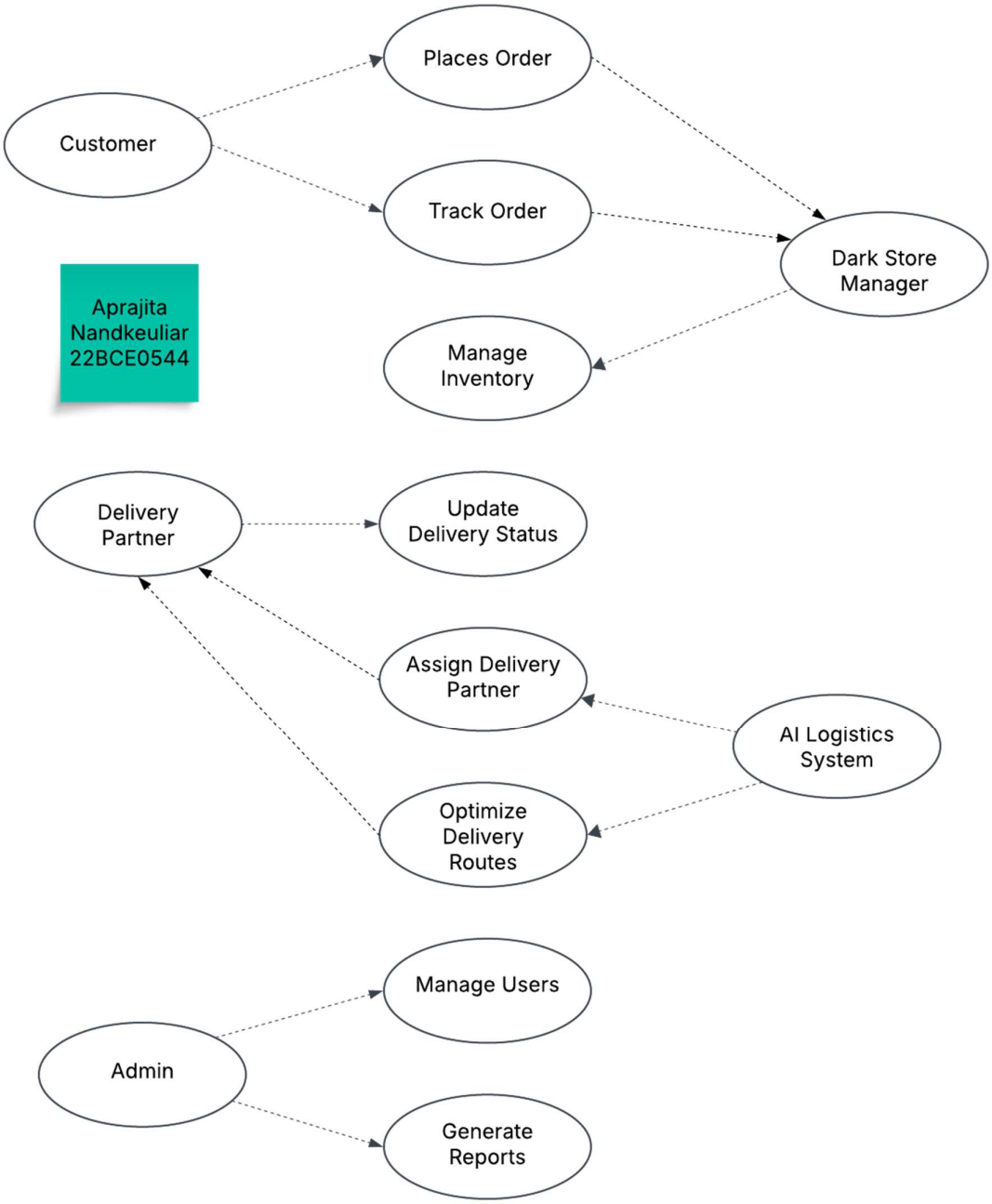
- Actors (users or external systems interacting with the application).
- Use Cases (functionalities the system provides).
- Relationships (associations between actors and use cases).

Actors:

1. **Customer** – Places orders, tracks deliveries.
2. **Dark Store Manager** – Manages inventory, confirms packing.
3. **Delivery Partner** – Receives delivery assignments, updates order status.
4. **AI Logistics System** – Assigns delivery personnel, optimizes routes, tracks inventory.

Use Cases:

1. **Place Order** (Customer)
2. **Track Order** (Customer)
3. **Manage Inventory** (Dark Store Manager)
4. **Assign Delivery Partner** (AI System)
5. **Optimize Delivery Route** (AI System)
6. **Update Delivery Status** (Delivery Partner)

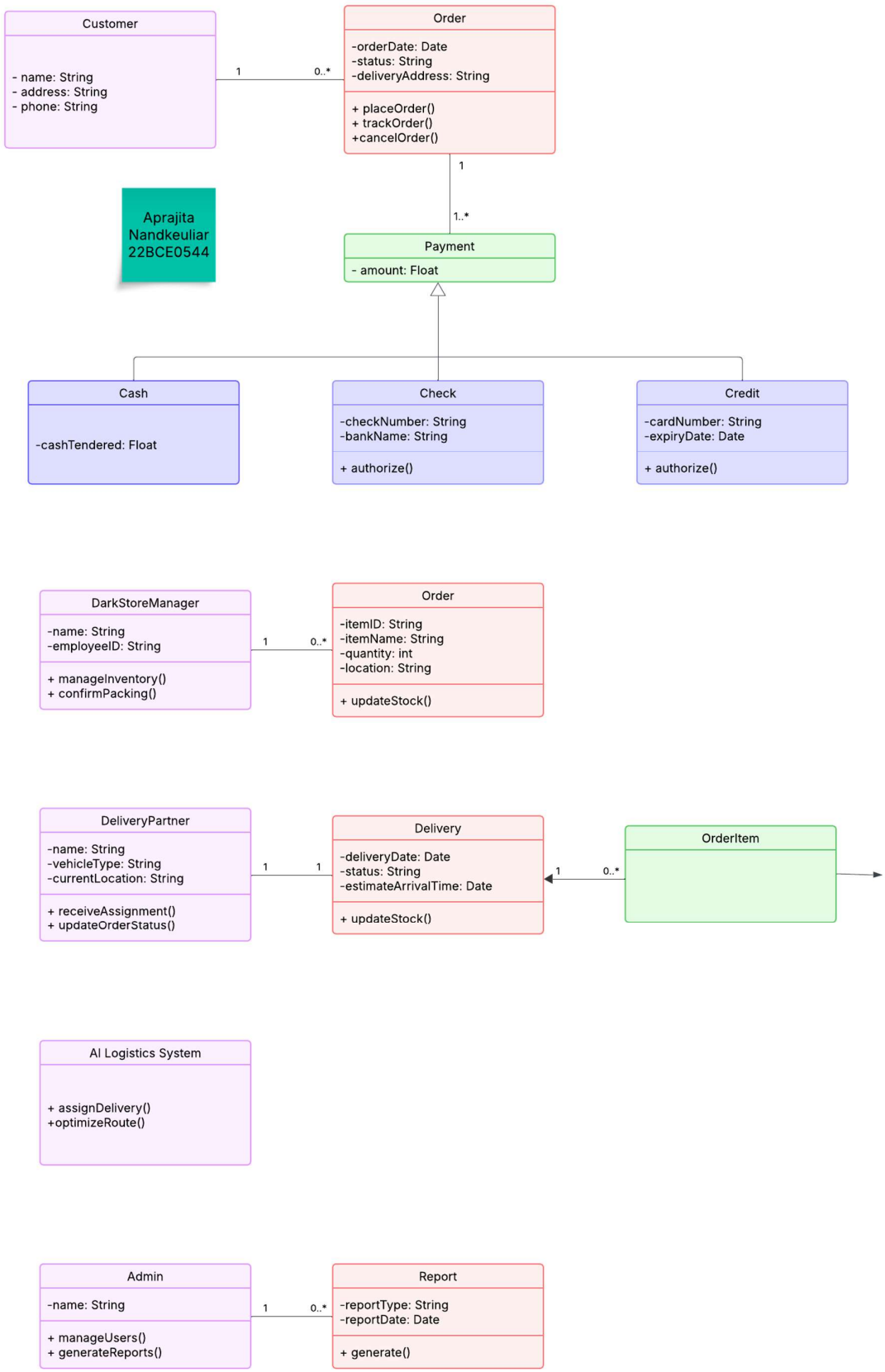

Places Order
Customer
Track Order
Dark Store Manager
Aprajita Nandkeuliar 22BCE0544
Manage Inventory
Delivery Partner
Update Delivery Status
Assign Delivery Partner
AI Logistics System
Optimize Delivery Routes
Manage Users
Admin
Generate Reports

UML Class Model

A **UML Class Model** represents the **structure of a system** by defining its **classes, attributes, methods, and relationships** among objects. It provides a **blueprint** for implementing the system in code.

For the **AI Logistics System (SwiftDrop)**, the UML **Class Diagram** includes key entities such as:

1. **Customer** (places orders, tracks delivery)
2. **Order** (contains order details, status)
3. **DarkStore** (manages inventory, processes orders)
4. **DeliveryPartner** (assigned orders, tracks location)
5. **RouteOptimizer** (calculates best delivery routes)
6. **Payment** (handles transactions)


Customer
- name: String
- address: String
- phone: String
1
0..*
Order
-orderDate: Date
-status: String
-deliveryAddress: String
+ placeOrder()
+ trackOrder()
+cancelOrder()
1
1..*
Aprajita Nandkeuliar 22BCE0544
Payment
- amount: Float
Cash
-cashTendered: Float
Check
-checkNumber: String
-bankName: String
+ authorize()
Credit
-cardNumber: String
-expiryDate: Date
+ authorize()
DarkStoreManager
-name: String
-employeeID: String
+ manageInventory()
+ confirmPacking()
1
0..*
Order
-itemID: String
-itemName: String
-quantity: int
-location: String
+ updateStock()
DeliveryPartner
-name: String
-vehicleType: String
-currentLocation: String
+ receiveAssignment()
+ updateOrderStatus()
1
1
Delivery
-deliveryDate: Date
-status: String
-estimateArrivalTime: Date
+ updateStock()
1
0..*
OrderItem
AI Logistics System
+ assignDelivery()
+optimizeRoute()
Admin
-name: String
+ manageUsers()
+ generateReports()
1
0..*
Report
-reportType: String
-reportDate: Date
+ generate()

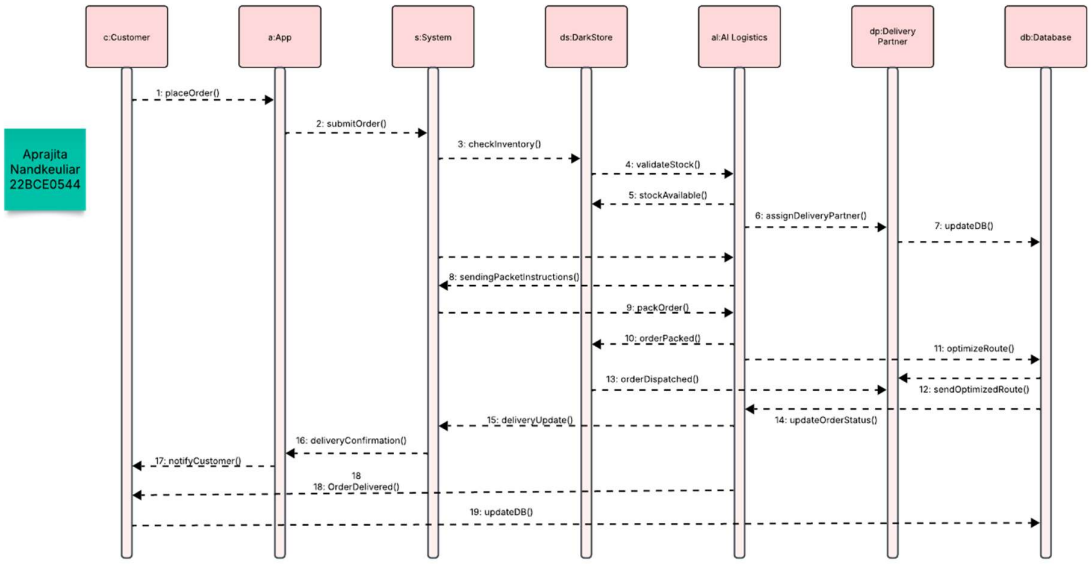
Sequence Diagram

A **UML Sequence Diagram** is used to visualize how objects interact in a system over time. It represents:

- The flow of messages between system components.
- The sequence of interactions between users and the system.
- How the system responds to various events.

The sequence diagram for this project will cover:

1. **Customer places an order** via the app.
2. **System selects the nearest dark store** with available stock.
3. **AI assigns a delivery partner** based on location and availability.
4. **Optimal delivery route is calculated.**
5. **Order is packed and dispatched** to the delivery partner.
6. **Delivery partner completes the delivery.**



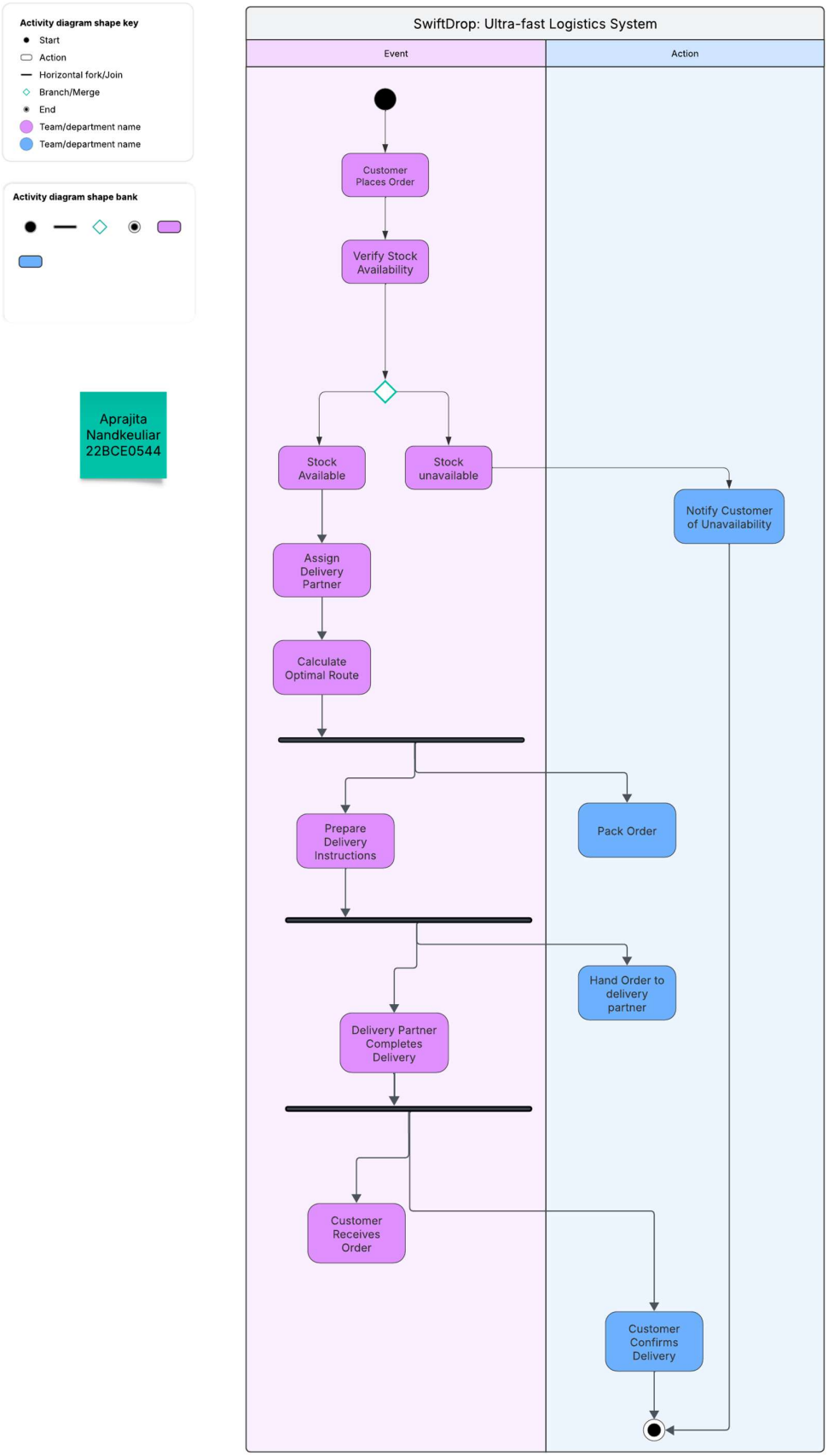
Activity Diagram

A **UML Activity Diagram** represents the workflow of a system, illustrating:

- The sequence of actions and decisions.
- The flow of control from one activity to another.
- Parallel and conditional processes within the system.

The activity diagram for this project will cover:

1. Customer places an order via the app.
2. System verifies stock availability at the nearest dark store.
3. AI assigns a delivery partner based on availability and proximity.
4. Optimal route is calculated.
5. Order is packed and handed to the delivery partner.
6. Delivery partner completes the delivery.
7. Customer receives the order and confirms delivery.


Activity diagram shape key
Start
Action
Horizontal fork/Join
Branch/Merge
End
Team/department name
Team/department name
Activity diagram shape bank
Aprajita Nandkeuliar 22BCE0544
SwiftDrop: Ultra-fast Logistics System
Event
Action
Customer Places Order
Verify Stock Availability
Stock Available
Stock unavailable
Notify Customer of Unavailability
Assign Delivery Partner
Calculate Optimal Route
Prepare Delivery Instructions
Pack Order
Hand Order to delivery partner
Delivery Partner Completes Delivery
Customer Receives Order
Customer Confirms Delivery