Data Structures and Algorithms

Project: Warehouse & Delivery Management System

1. Overview

This project simulates the operation of a **Warehouse & Delivery Management System**. The system receives customer orders, manages warehouse inventory, schedules vehicles/drivers for deliveries, and tracks shipments until completion. The simulation is event-driven over discrete timesteps and emphasizes careful use of data structures to achieve correct behavior and efficient scheduling.

2. Simulation Concept

Time advances in **discrete timesteps** starting from 1. At each timestep, one or more events may occur (e.g., a new order arrives, a truck is dispatched, inventory is restocked, a shipment is delivered, a route is re-planned). All actions scheduled for the current timestep are executed, then the timestep counter is incremented.

3. System Entities

3.1 Inventory Item

- **ItemID** (unique)
- **Name**
- **Quantity** (in stock at a warehouse)
- **Perishability** (boolean or TTL in timesteps)
- **UnitVolume / UnitWeight** (optional for vehicle capacity constraints)

3.2 Customer Order (Shipment Request)

- **OrderID** (unique)

- **RT** (request timestep)
- **DueBy** (deadline timestep; optional)
- **Priority Class**: VIP / Standard
- **Destination Node** (city/zone ID)
- **Demand**: list of `(ItemID, Quantity)`
- **OrderValue** (total monetary value)
- **Status**: Waiting / Assigned / In-Transit / Delivered / Canceled / Partially-Fulfilled

3.3 Warehouse

- **WID** (identifier)
- **Inventory Map**: `ItemID -> Quantity`
- **Dispatch Queue** of assigned shipments
- **Location Node** (for travel-time matrix)

3.4 Vehicle (or Driver + Vehicle)

- **VID**
- **Type**: Standard / Refrigerated (for perishable goods)
- **Capacity**: by weight/volume/slots
- **Speed**: distance units per timestep
- **Status**: Available / Outbound / Returning / Maintenance
- **Assigned Shipment(s)** (single or batched by capacity policy)

3.5 Road Network (optional abstraction)

- **Travel-Time Matrix** `T[i][j]` in timesteps between nodes (warehouses/destinations).

4. Time Definitions

- **RT (Request Time)**: order arrival time.
- **AT (Assignment Time)**: time when an order is assigned to a vehicle/warehouse.
- **DT (Dispatch Time)**: time when a vehicle departs with the shipment.

- **FT (Finish/Delivery Time)**: delivery completion time.
- **WT (Waiting Time)**: `AT − RT`.
- **Transit Time**: derived from vehicle speed and travel distance (`T[i][j]` or Euclidean approximation).

5. Scheduling & Assignment Rules

5.1 Order Prioritization

Orders are placed into **two logical queues**:

- **VIP Orders**: stored in a **priority queue** (higher priority served earlier).
- **Standard Orders**: stored in **FCFS** queue per destination or global.

A reasonable **VIP priority score**:

```
Priority = α * OrderValue / (β * (CurrentTime − RT + 1)) + γ * Deadlin
```

Where `DeadlineUrgency = max(0, 1 / (DueBy − CurrentTime + 1))`, and `SizePenalty` grows with total quantity/volume. Choose α, β, γ, δ to balance profit vs urgency.

5.2 Warehouse Selection

- Prefer the **nearest warehouse** with sufficient inventory to fulfill the order.
- If no single warehouse can fulfill, **split shipments** across multiple warehouses (optional extension).
- For perishable items, prefer warehouses with **refrigerated vehicles** and shorter routes.

5.3 Vehicle Assignment

- Match vehicle **type** and **capacity** to the shipment.
- Prefer vehicles with **earlier availability** and **shortest travel time** to destination.
- If no vehicle can be assigned now, the order remains **Waiting**.

5.4 Delivery & Return

- When a vehicle arrives at destination, the order becomes **Delivered** at time **FT**.
- The vehicle may return directly or continue to the next planned stop (if batched). Status becomes **Available** once back to its warehouse.

5.5 Cancellations & Stock-outs

- If an order is canceled while Waiting, remove it from the queue.
- If inventory is insufficient upon assignment (race), re-queue the order or split it.

6. Events

- **R** — New order arrival.
- **S** — Inventory restock at a warehouse.
- **D** — Dispatch (explicit scheduling) or an internal action when assignment completes.
- **C** — Order cancellation.
- **M** — Maintenance event for a vehicle (temporarily unavailable).
- **U** — Route update / re-route (e.g., traffic disruption).

All events are loaded from the input file and executed when **TS = current timestep**.

7. Input / Output File Formats

7.1 Input File

```
W N V                     # warehouses, item types, vehicles
<Travel-Time Matrix WxW>  # integer times between nodes (or 0 for same

# Vehicles (V lines)
VID Type Speed Capacity RefrigeratedFlag HomeWID

# Initial Inventory (W blocks, each N integers)
WID
```

```
q1 q2 ... qN              # quantities per ItemID (1..N)

M                         # number of events
<event lines>
```

Event Lines

- **Order Arrival**
  `R TS OrderID DestWID DueBy PriorityClass K`
  followed by **K lines**: `ItemID Quantity`
  PriorityClass: VIP or STD.
- **Restock**
  `S TS WID K` followed by **K lines**: `ItemID QuantityDelta`
- **Cancel**
  `C TS OrderID`
- **Maintenance**
  `M TS VID Duration`
- **Reroute**
  `U TS i j NewTime` (update travel time between nodes `i`, `j`).

7.2 Output File

For each delivered order (sorted by **FT ascending**):

```
FT OrderID RT WT TransitTime AssignedWID AssignedVID Filled(Yes/No) Va
```

Then aggregated statistics:

- Total orders; counts by class (VIP/STD) and fulfillment status (Full/Partial/Unfilled)
- Average waiting time; average transit time
- Total delivered value; on-time delivery rate (relative to DueBy)
- Vehicle utilization (% busy time / total simulation time)
- Inventory turns per item (optional)

8. Program Modes

- **Interactive Mode**: At each timestep, print:
  - First N upcoming events (`[Type, TS, ...]`)
  - Waiting VIP/STD orders (IDs, priority/RT, destination)
  - Available / Outbound / Returning vehicles (IDs, types)
  - Warehouse stock snapshots (top K items with low stock)
  - Delivered orders since last step
  - Prompt to proceed to next timestep
- **Silent Mode**: Run simulation and produce output file only.

9. Suggested Class Design

- **Simulator**: global time, event loop, statistics, I/O.
- **Warehouse**: inventory map; picking/restocking operations; dispatch interface.
- **InventoryItem**: metadata (perishability, unit volume/weight).
- **Order**: demand vector, priority score, destination, status.
- **Vehicle**: attributes, current route/ETA, status.
- **Event (abstract)**: `TS` , `Execute()` ; derived: `ArrivalEvent` , `RestockEvent` , `CancelEvent` , `MaintenanceEvent` , `RerouteEvent` .
- **UI**: console printing for Interactive Mode.

10. Recommended Data Structures

Purpose	DS
Events in chronological order	Queue
VIP orders	Priority Queue
Standard orders	Queue (global or per-destination)
Warehouse inventory	Hash Map (ItemID → Quantity)
Vehicles by status	Queues / Lists per status
Travel times	2D array / adjacency matrix

Purpose	DS
Delivered orders	List (sort by FT)

11. Implementation Notes

- Keep **all lists as pointers** to avoid object copying (share/move).
- Carefully validate capacity constraints before assignment.
- For perishables, enforce refrigerated vehicles and TTL checks.
- Decouple **order selection** (which to serve) from **resource assignment** (which warehouse/vehicle).
- Stop when: event queue empty **and** all vehicles idle **and** all waiting orders resolved (delivered/canceled/unfilled).

12. Sample Scenario

At TS=30:

- Two VIP orders arrive for DestWID=3 and 5; they enter the VIP PQ.
- A restock event raises Item #7 at WID=1 by 250 units.
- Vehicle V12 (refrigerated) becomes available; the top VIP order demands perishable items → assigned to V12 from WID=1 and dispatched.
- At TS=42 the shipment to DestWID=3 is delivered (FT=42).

13. Learning Outcomes

Students will:

- Build **event-driven simulations** with multiple interacting resources.
- Apply **priority queues**, **queues**, and **hash maps** for operational data.
- Implement **resource-constrained scheduling** and **routing** logic.
- Manage **file I/O** for structured inputs and reproducible outputs.
- Design modular **OOP components** and clean interfaces.

- Compute **utilization**, **waiting time**, **service levels**, and other KPIs.
- Reason about **trade-offs** between urgency, profitability, capacity, and route time.