📘 GHMC SMART GVP ROUTING & MONITORING SYSTEM

(Internal Development Document – Team Split & Responsibilities)

1️⃣ PROJECT OVERVIEW (ONE-PAGE UNDERSTANDING)

This system ensures:

- Optimized garbage vehicle routes
- Proof-based verification of GVP clearance
- Real-time visibility for GHMC
- Transparency for civilians
- Accountability for drivers

The system consists of:

- OR-Tools–based routing engine
- Central backend + database
- GHMC Web Dashboard
- Driver Mobile App
- Civilian Mobile App
- File storage (MinIO)

2️⃣ CORE DATASETS PROVIDED

📌 Input Datasets

1. GVP Dataset
    - gvp_id
    - latitude
    - longitude
    - waste_generated
2. SCTP Dataset
    - sctp_id
    - latitude
    - longitude
3. Truck Dataset
    - truck_id
    - capacity
    - count

3️⃣ GLOBAL SYSTEM RULES (VERY IMPORTANT)

- Routes are prepared **before the shift**
- Routing uses **OpenStreetMap + static traffic**
- No dynamic rerouting unless:
    - Truck breakdown
    - Fuel low
- Proof must:
    - Be captured via live camera
    - Contain geo + timestamp
    - Be taken **before 10:30 AM**
- Penalties are **points-based**
- Offline-first for drivers

👥 TEAM SPLITTING (VERY DETAILED)

🧠 TEAMMATE 1 (T1)

ROUTING & OPTIMIZATION ENGINE

Role: Brain of the system
Tech: Python, OR-Tools, OpenStreetMap

🔹 Responsibilities

1. Static Traffic Data Preparation

- Use **OpenStreetMap road network**
- Assign:
    - Average speed per road type
    - Known congestion zones in Hyderabad
- Build **static travel time matrix**

No live traffic APIs are used.

---

2. OR-Tools Routing Logic

Routing constraints:

- Truck capacity
- GVP waste volume
- 20–25 minutes service time per GVP
- Start from SCTP
- End at SCTP if capacity reached

---

3. Route Output (CRITICAL)

For each truck:

- Output route as **JSON**
- Store **sequence of GVPs**

Example:

```
{
 "truck_id": "TRK_12",
 "route": [
  {"type": "SCTP", "id": "SCTP_01"},
  {"type": "GVP", "id": "GVP_23"},
  {"type": "GVP", "id": "GVP_09"}
 ],
 "estimated_time": 215,
 "version": 1,
 "date": "YYYY-MM-DD"
}
```

---

4. Partial Rerouting

Triggered only if:

- Truck failure
- Fuel low

Inputs:

- Remaining unvisited GVPs
- Active trucks
- Current locations

Output:

- New JSON routes
- Version incremented

- **Deliverables from T1**
  - Routing script
  - Time matrix generator
  - Route JSONs
  - Reroute logic

🧱 TEAMMATE 2 (T2)

BACKEND + GHMC DASHBOARD INTEGRATION

Role: System backbone
Tech: FastAPI, PostgreSQL, MinIO, WebSockets

🔹 Responsibilities

A. Route Storage & Retrieval

1. Store routing JSON in DB:

routes:

- route_id

- truck_id

- route_json

- date

- version

2. When driver logs in with truck_id:

- Fetch route from DB
- Send to Driver App

B. GHMC DASHBOARD MAP INTEGRATION

Two Map Views:

1. Routes Map
    - Hyderabad base map
    - Polylines showing truck routes
2. GVP Status Map
    - Green → Cleared
    - Yellow → Partially cleared
    - Red → Not cleared

Routes shown come **directly from DB JSON**.

C. Proof Upload & Verification

Proof Submission Flow:

1. Driver uploads photo
2. Backend stores photo in **MinIO**
3. Extract:

- GPS location
- Timestamp

Verification Rules:

- Distance from GVP < allowed radius (e.g., 50m)
- Timestamp ≤ 10:30 AM

If valid:

- Update GVP status
- Update dashboard maps & list

If invalid:

- Mark as violation

D. GVP Status Updates

Each GVP:

status:

- Cleared

- Partially Cleared (High/Medium)

- Not Cleared

Updates reflected in:

- GHMC Map page
- GHMC GVP List page
- Driver route progress

E. Penalty Logic (Backend-Controlled)

Penalty applied if:

- GVP in assigned route not visited
- No proof submitted
- Proof invalid
- No issue reported same day

Penalty stored as:

penalties:

- driver_id

- gvp_id

- reason

- points

- date

Deliverables from T2

- REST APIs
- DB schema
- MinIO integration
- Dashboard sync logic

🚚 TEAMMATE 3 (T3)

DRIVER MOBILE APP

Role: Field execution
Tech: Flutter, Google Maps SDK, SQLite

🔹 Responsibilities

A. Driver Login & Route Fetch

- Driver logs in using mobile number
- Enters truck_id
- App fetches route JSON from backend

B. Route Display Logic

- Start: SCTP
- Destination: first GVP
- Once reached:
    - Current GVP → source
    - Next GVP → destination

This continues step-by-step.

C. Real-Time Navigation

- Use Google Maps SDK
- Show live navigation
- No route recalculation in app

D. Offline Cache (CRITICAL)

After route fetch:

- Save route JSON in SQLite
- Always show cached route on home screen
- If network lost:
    - Use cached route
    - Queue updates locally

E. Proof Capture

- Camera-only capture
- Auto GPS & timestamp
- Upload when network available

Deliverables from T3

- Driver app
- Offline mode demo
- Proof submission flow

🎨 TEAMMATE 4 (T4)

ALL FRONTEND + CIVILIAN APP + PENALTY VISIBILITY

Role: Visual impact & public trust
Tech: React, Flutter, Firebase

A. GHMC WEB FRONTEND

Pages:

1. Dashboard summary
2. Live GVP Map
3. Route visualization
4. GVP List with status
5. Penalty reports

B. CIVILIAN MOBILE APP

Features:

- Login via mobile
- Location detect or manual pin
- Show nearby GVPs
- Push notifications:
    - Truck arriving in 15 mins

- Delay reported

C. Penalty Visibility

- Drivers see penalty points
- GHMC sees penalty reports
- Civilian sees delay transparency

D. Salary Deduction Flag (Logic Indicator)

If:

- Penalty unresolved same day

Then:

- Mark salary deduction flag (no payroll handling)

Deliverables from T4

- Web dashboard UI
- Civilian app
- Notification UI
- Penalty visual reports

🔁 END-TO-END FLOW (SIMPLE STORY)

1. T1 generates routes → JSON
2. T2 stores routes → DB
3. Driver logs in → route fetched
4. Route cached → offline-safe
5. Driver reaches GVP → uploads proof
6. Backend verifies → updates status
7. Dashboard updates in real-time
8. Civilians notified
9. Missed GVP → penalty applied