**Algorithm for “Drinking Water Points & Pollution-Free Waste Management Zones”**

0) System Overview (Two Services)

- Service A — Simulation + Dashboard (app.py)

Simulates bin fill levels, assigns nearest vehicles, animates routes on a campus map, and shows air/water gauges (demo mode).

- Service B — Real-Time IoT Dashboard (blynk\_dashboard.py)

Pulls live sensor values from Blynk Cloud, shows air/water gauges, displays waste panel, and provides QR/form complaint intake.

1) Data Models (Shared Ideas)

- Bins: { id, lat, lng, fill, status } where status ∈ {OK, FULL} and fill ∈ [0..100].

- Vehicles: { id, lat, lng, status, target\_bin? } where status ∈ {IDLE, BUSY}.

- Assignments/Comparisons (Simulation): Records each dispatch decision and distance comparison.

- Complaints (IoT): { time, phone, message } in-memory list.

2) Core Geospatial Utility

- Haversine Distance: Input (lat1, lon1), (lat2, lon2). Output: distance (m).

Uses Earth radius R = 6,371,000 m with haversine formula.

3) Service A — Simulation & Smart Dispatch (app.py)

3.1 Bin Fill Update

- Select random bin b.

- Increase b.fill += Δ, Δ ∈ [20..40] (manual) or [10..25] (auto).

- Clamp at 100. If b.fill == 100, mark status = FULL.

3.2 Nearest-Vehicle Assignment

- Identify full bins without assigned vehicles.

- Identify idle vehicles.

- For each idle vehicle, compute distance to bin using haversine.

- Choose nearest vehicle, mark BUSY, assign target\_bin.

- Log assignment and comparison stats.

3.3 Trip Completion

- On trip completion: reset vehicle status to IDLE and bin fill to 0 (status=OK).

3.4 Auto Simulation Loop

- Every 6s: fill random bin, possibly mark full, attempt vehicle assignment.

3.5 UI Refresh & Animation

- Update bins: marker colors (green/orange/red).

- Update vehicles: marker icons and status text.

- Animate BUSY vehicles along routes with MovingMarker.

- On animation end: trigger trip completion API.

3.6 Dashboard Gauges (Demo Mode)

- Every 3s: generate random values for air (NH3, Smoke, Alcohol, Benzene, CO2, NOx) and water (TDS, EC).

- Update Chart.js doughnut gauges with green/orange/red status.

3.7 Control APIs

- /fill\_random, /assign\_nearest\_full, /reset, /reset\_vehicles, /start\_auto, /stop\_auto.

4) Service B — Real-Time IoT Dashboard (blynk\_dashboard.py)

4.1 Blynk Data Acquisition

- Call Blynk Cloud API using token and virtual pins.

- Air: NH3(v3), Smoke(v4), Alcohol(v5), Benzene(v6), CO2(v7), NOx(v8).

- Water: TDS(v0), EC(v1).

- Return JSON.

4.2 Dashboard Rendering & Update

- Every 5s: fetch /air, /water, /bins, /vehicles, /complaints.

- Update gauges, bins, vehicles, and alerts panel dynamically.

4.3 Complaint Intake

- /qrcode page shows QR linking to complaint form.

- /complaint\_form allows phone/message submission.

- Logs complaint with timestamp.

5) End-to-End UX Flow

- Simulation UI: Fill bins, assign vehicles, observe animation, switch tabs to dashboard, show water tap marker.

- Real-Time Dashboard: Show live gauges, bins, vehicles, submit complaint via QR/form, watch alerts update.

6) Correctness & Complexity

- Dispatch: O(V) or O(V log V) per assignment (V=vehicles).

- Auto loop: constant-time updates.

- UI refresh: efficient for small number of bins/vehicles.

7) Resets, Fault Handling, Edge Cases

- /reset clears state and stops auto loop.

- /reset\_vehicles resets only vehicles.

- Network failures for Blynk values → None (shown as 0).

- No FULL bins or IDLE vehicles: safe fallback.

8) Security/Privacy

- Blynk tokens stored server-side only.

- Complaints held in memory (extendable to DB).

9) Extensibility (Future Work)

- Implement advanced routing (VRP).

- Persist telemetry and complaints in database.

- Integrate real GPS feeds and geofencing.

- Apply WHO/CPCB thresholds for air/water metrics.