Candidate Technical Assessment R2

Submit completed repo within **24 hours**.

Objective

Build a mini platform that:

- Simulates sensor devices publishing telemetry,
- Ingests and stores the data on a backend server,
- Generates alerts based on rules,
- Displays live telemetry and alerts on a simple dashboard.

This exercise assesses backend design, API implementation, event handling, and coding discipline.

Deliverables

Your submission must contain **all code, documentation, and configuration files** required to:

1. Run a simulated device that publishes telemetry via MQTT.
2. Run a backend service that subscribes, stores, and serves telemetry + alerts.
3. Expose REST APIs and optionally a WebSocket stream.
4. Display a minimal dashboard showing data and alerts.

Requirements

1 Simulator (device side)

- Publishes JSON telemetry to topic:
  `devices/{deviceId}/telemetry`
  every **1 second**.

JSON payload example:

```
{
  "deviceId": "dev-001",
  "ts": 1714041600000,
  "temperature": 26.3,
  "humidity": 47.9
}
```

-
- Randomly vary values and drop 1 in every 30 messages.

2 Backend Service

- Subscribe to all telemetry topics.
- Store data in SQLite or PostgreSQL.
- REST endpoints:
  - `GET /devices` — list of known devices with last seen time and status.
  - `GET /telemetry?deviceId=...&limit=100` — recent telemetry.
  - `GET /stats/avg?deviceId=...&window=5m` — rolling averages.
  - `GET /alerts` — last 50 alerts.

- `GET /health` — health check.

Alert rules:

- `HIGH_TEMP`: temperature > 30°C
- `OFFLINE`: no data received in 10 seconds
- `ONLINE`: device recovered after being offline

Store alerts in DB and expose via API.

Dashboard

- Display:
  - Device list + ONLINE/OFFLINE status.
  - Line chart for latest temperatures.
  - Table of alerts (type, timestamp, deviceId).
- Auto-refresh (poll every 3 s or via WebSocket).

Optional Bonuses

- Docker Compose setup (backend, MQTT broker, web).
- WebSocket live updates.
- GitHub Actions CI to run tests.
- Grafana or similar visualization.

Evaluation Criteria (100 points)

Area	Points
MQTT Ingestion + DB Persistence	20
REST APIs	20
Alerts (HIGH_TEMP, OFFLINE)	20
Code Quality & Architecture	15
Documentation (README, comments)	10
Testing (unit/integration)	10
Docker / CI / Bonus	5

Passing score: 70 +
Strong candidate: 85 +

Submission Process

1. Create Your Repo

- **Repo name:** `mini-iot-telemetry-{firstname-lastname}`
- **Default branch:** `main`
- **Make it Private** and add our reviewers as collaborators.

2. Branching Rules

Use feature branches:

```
feature/simulator
feature/server
feature/alerts
feature/dashboard
```

Open pull requests into `main` for each feature.

3. Commit Format

Use clear commit messages:

```
feat(server): implement telemetry API
fix(simulator): handle mqtt reconnect
docs(readme): update setup guide
```

4. Final Merge & Tag

When complete:

```
git checkout main
git merge --no-ff feature/*
git tag -a v0.1.0 -m "Initial submission"
git push origin main --tags
```

5. Submit

Send the **private GitHub link** to birame@qat-gps.com
Do **not** send code via ZIP or email.