TASK 6:
DATABASE DESIGN AND IMPLEMENTATION

GROUP 14

Project: Network QoE Mobile App – Vital signal

Instructor : Dr. Nkemeni Valery

Data Elements – What We Capture

User Inputs:

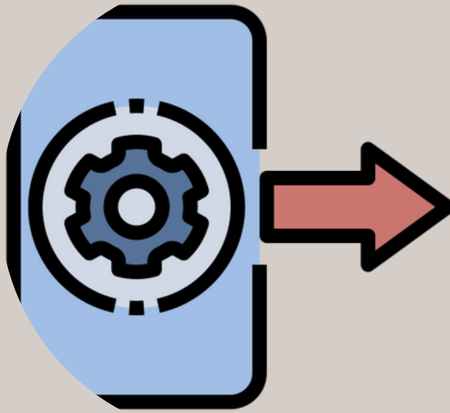
Ratings (1–5), issue types, text comments
Preferences & language settings

Auto-Collected Metrics:

Signal strength, network type (3G/4G/5G), operator
GPS, device model, timestamp

System-Generated:

Unique IDs (user, session, feedback)



Outputs:

Real-time status, feedback history, analytics dashboard

Conceptual Design — How It's Organized

Each module plays a focused role in turning raw data into decision-ready output.

01

Data Collection
Handles all user + system data inputs

02

User Feedback
Collects ratings, manages sessions

03

Analytics
Computes trends, satisfaction levels

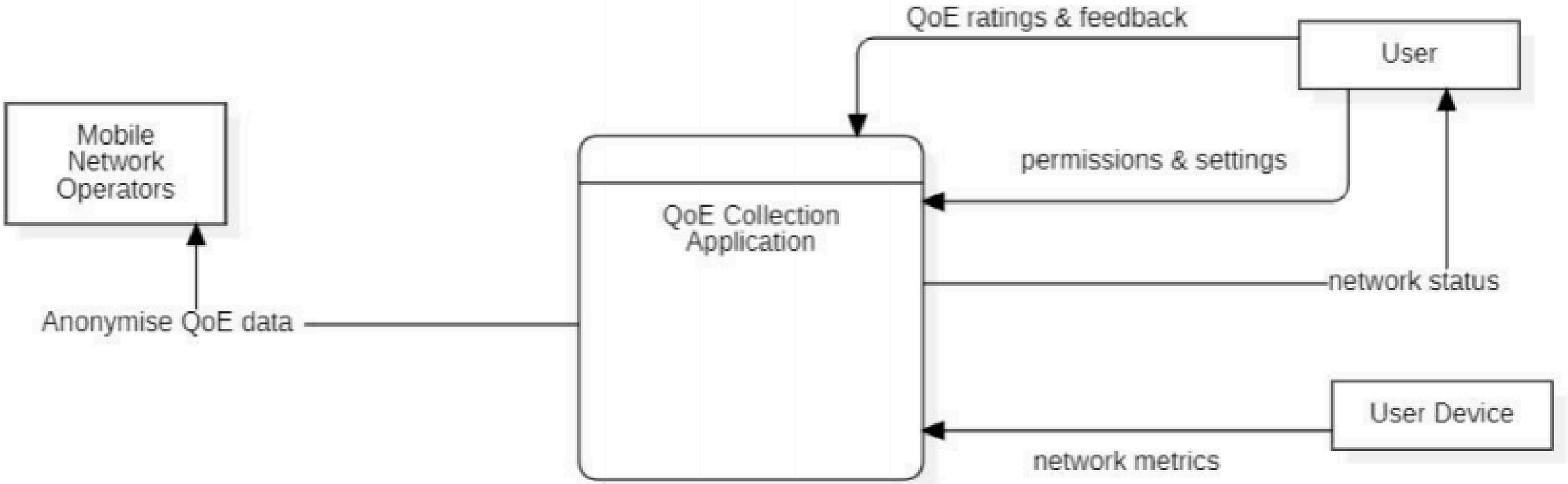
04

Visualization
Turns data into graphs and maps

05

Export & Access
API endpoints for external use

Conceptual Design – Data Flow



ER Diagram — Our Logical Data Model

User - **user_id, preferences**

Session - **session_id, user_id, timestamps**

SignalMetric - **signal strength, network type, location**

Feedback - **rating, issue_type, comment**

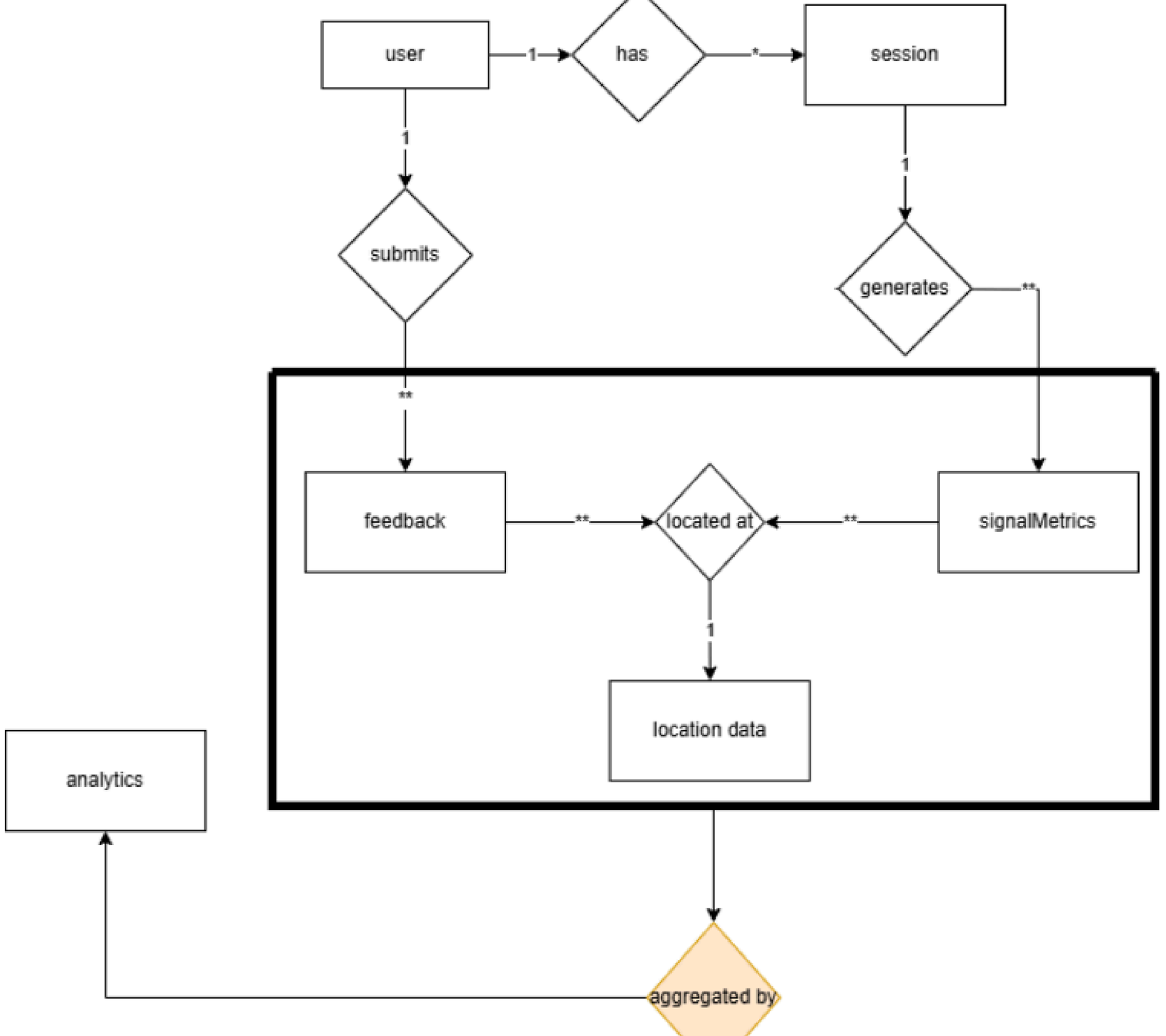
LocationData - **lat, long, accuracy**

Key Relationships:

- One user → many sessions
- One session → many signal metrics
- One user → many feedbacks
- Feedback + signalMetrics → reference locationData

Conceptual Design – Data Flow

ER Diagram for QOE application



Database Implementation
- Using Firebase Firestore

Cloud Firestore

Why Firestore?

- Real-time sync
- Schema-less (but structured)
- Auto-scaling + global availability

Database Implementation

- Using Firebase Firestore

Collection Design:

- /users/{user_id}
- /sessions/{session_id}
- /feedback/{feedback_id}
- /signalMetrics/{metric_id}
- /locationData/{location_id}

Normalization:

- 1NF to 3NF principles applied
- Cross-referencing for relational consistency

Backend Implementation – Core Logic

Built with: Node.js + Express.js

Endpoints:

- POST /api/network-feedback: Accepts feedback
- GET /api/network-feedback/analytics: Returns trends
- GET /api/health: System check

Responsibilities:

- **Input validation**
- **Session tracking**
- **Multi-collection writes**
- **Analytics updates**

Connecting Backend to Firestore

Firebase Admin SDK handles:

- Authenticated DB access
- Connection pooling
- Batch writes + transactions

Secure Environment Setup:

- Environment variables (.env)
- Key-based credential access

Query Patterns

- Range queries (by timestamp)
- Batched writes for feedback + metrics

Conclusion

Given the above parameters,

- We built a feedback-driven mobile QoE tracker
- Cloud-based, real-time, scalable and structured
- Emphasized:
  - Logical modeling
  - Clean backend design
  - Real-world usability

Thank You!

GROUP 14