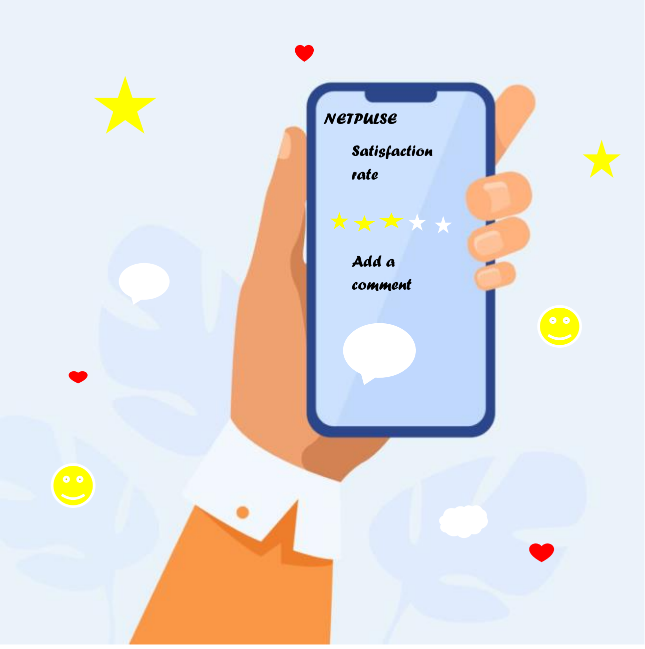


INTERNET PROGRAMMING AND MOBILE PROGRAMMING

CEF440

BY,

Group 20

NETPULSE:

"Capturing What You Feel, Delivering What You Deserve – Real Insights for Real Connectivity."

Course instructor:

Dr. Valery Nkemeni

Table of content.

Report on TASK 4:

SYSTEM MODELLING AND DESIGN

INTRODUCTION:

To ensure a robust and scalable architecture, various system modelling techniques have been applied. These include context diagrams, data flow diagrams, use case diagrams, sequence diagrams, class diagrams, and deployment diagrams. Each diagram provides a different perspective on the system, helping to visualize how components interact, how data flows, how users engage with the system, and how the application is structured and deployed.

The modelling and design tasks documented in this report form a critical foundation for the implementation and testing of the mobile application. They enable stakeholders to understand system functionality, identify potential issues early, and guide the development process in a structured and efficient manner.

A. CONTEXT DIAGRAM:

The context diagram illustrates the high-level interaction between the Mobile App System and its external entities.

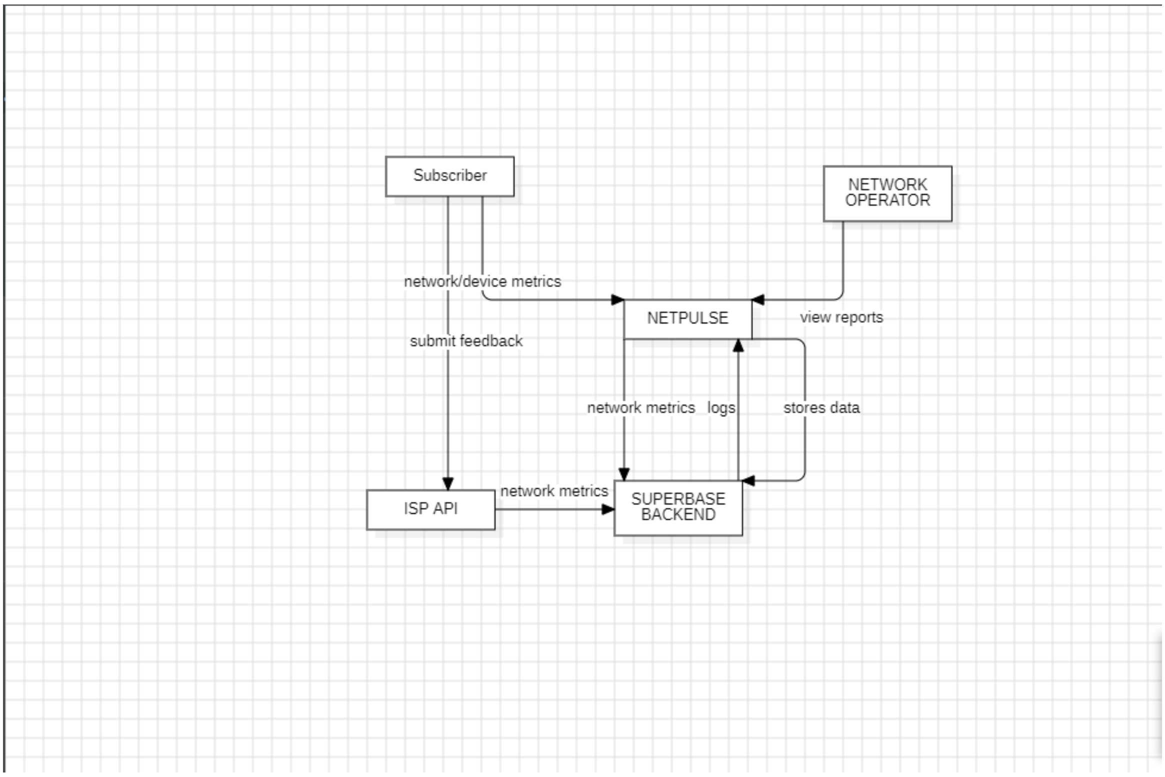
At the centre of the diagram is the QoE App System, which communicates with two main external actors:

the mobile network subscriber (user) and the network operator (admin).

- The subscriber uses the mobile app to submit feedback and allow passive collection of network metrics.
- The admin accesses the backend to view aggregated reports and monitor network performance trends.

The diagram also shows the app's integration with external services, such as the ISP API for real-time network data and the Supabase backend for data storage, authentication, and analytics. This context view helps define system boundaries and the flow of information between users, the system, and external resources.

DIAGRAM:



B. **DATAFLOW DIAGRAM (level 0):**

The Level 0 Data Flow Diagram (DFD) breaks down the QoE system's core processes and how data moves through them.

It includes key processes such as:

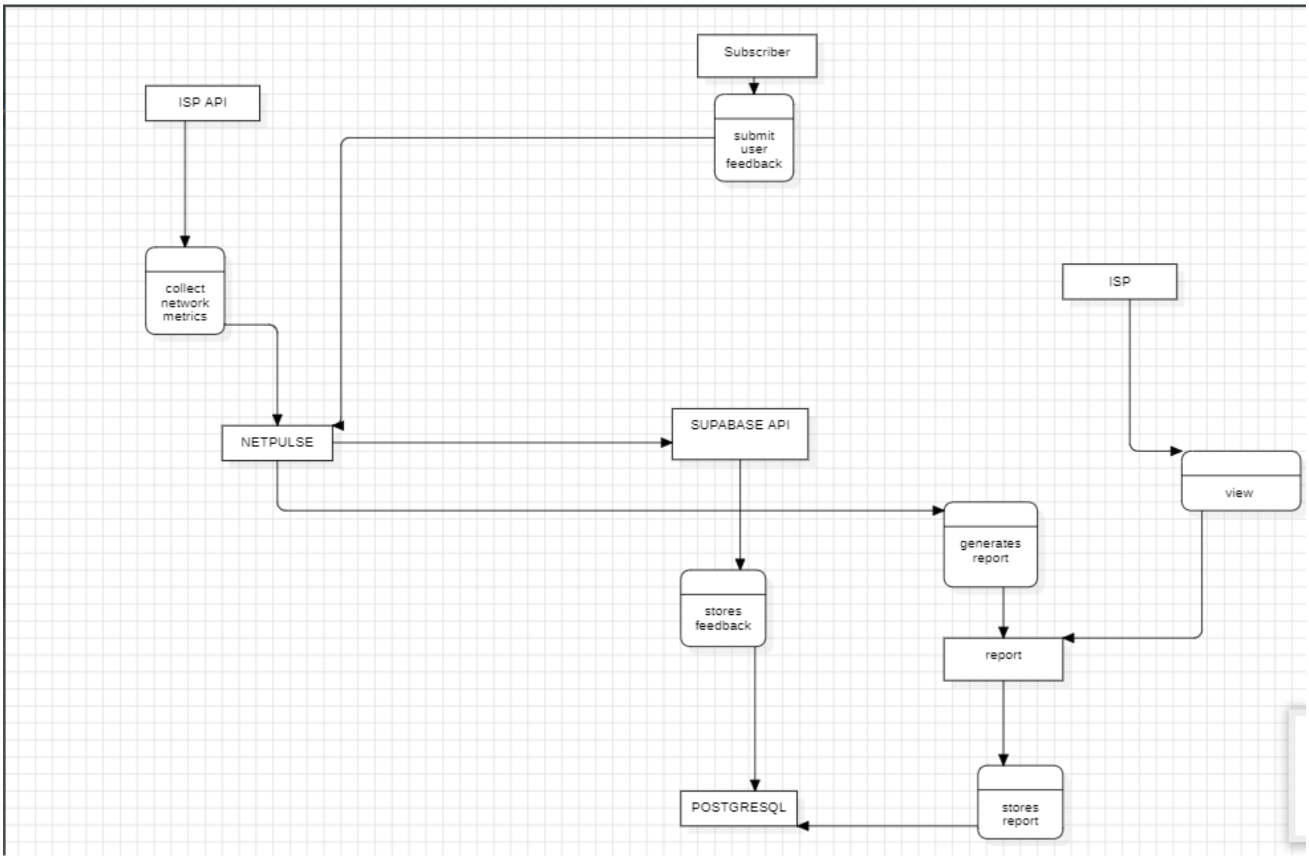
- ✓ submitting user feedback
- ✓ collecting network metrics
- ✓ storing data
- ✓ generating reports.

-Subscribers initiate interactions by submitting feedback, which is sent to the Supabase API and stored in a PostgreSQL database.

- Simultaneously, the app collects network metrics—either passively or via the ISP API—and stores them in the backend. The local monitoring component also logs app performance for later analysis. The ISP accesses this data through a reporting interface that queries the database.

-Each data flow, such as feedback, metrics, and reports, is explicitly shown between processes and data stores, emphasizing how the system handles and transforms information from input to output.

DIAGRAM:



C. **USE CASE DIAGRAM:**
Represents the functional interactions between users (actors) and the system.

□ Main **Components**:

- **Actors**:
  - User
  - Internet service provider

Description:

The User actor, positioned on the left, engages with use cases such as:

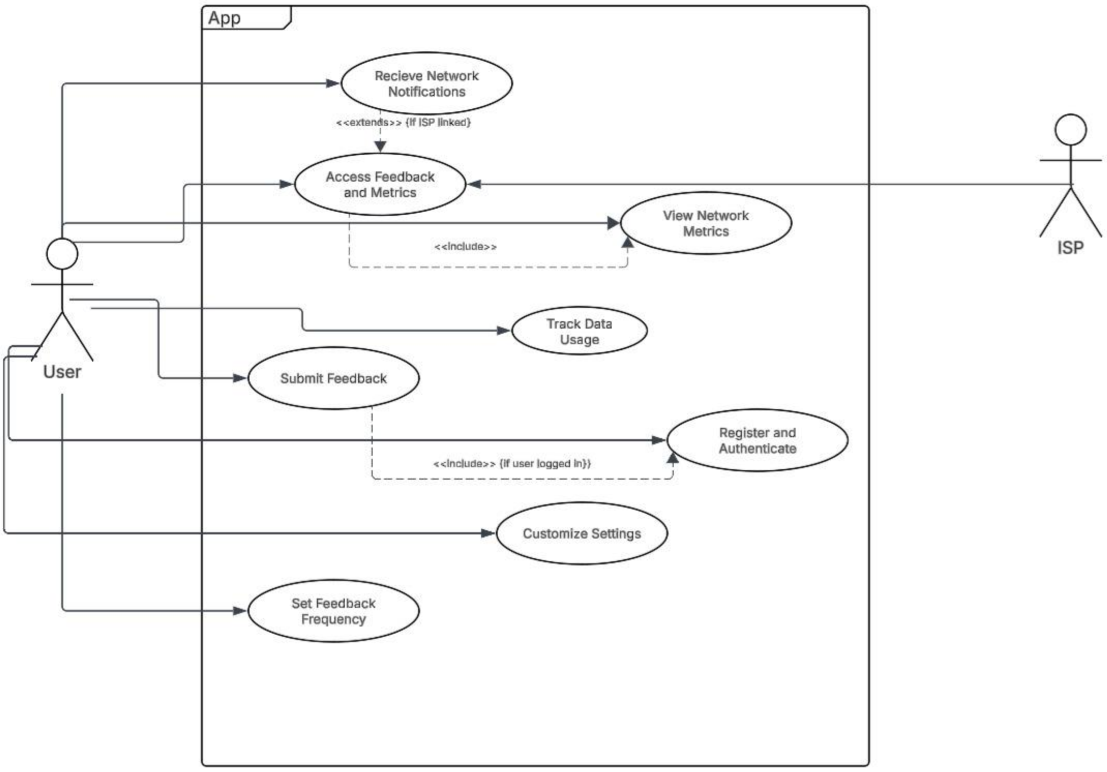
- Register and Authenticate
- View Network Metrics (displaying signal strength, latency, and packet loss)
- Submit Feedback (submitting ratings and comments)
- Set Feedback Frequency (controlling feedback prompts)
- Receive Network Notifications (alerts for network issues)
- Track Data Usage (monitoring consumption)
- Customize Settings (adjusting preferences like notifications and location).

The ISP actor, on the right, interacts with access Feedback and Metrics via a dashboard interface. Relationships are defined as:

- Submit Feedback requires Register and Authenticate (<<include>> with a condition)

- Access Feedback and Metrics includes View Network Metrics (<<include>>)
- Receive Network Notifications extends Access Feedback and Metrics (<<extend>> with a condition).
- The diagram uses a monochrome style with a rectangular package, ensuring a clear, vertically oriented structure.

Diagram:


App
Recieve Network Notifications
<<extends>> {If ISP linked}
Access Feedback and Metrics
View Network Metrics
<<include>>
ISP
User
Track Data Usage
Submit Feedback
Register and Authenticate
<<include>> {If user logged in}}
Customize Settings
Set Feedback Frequency

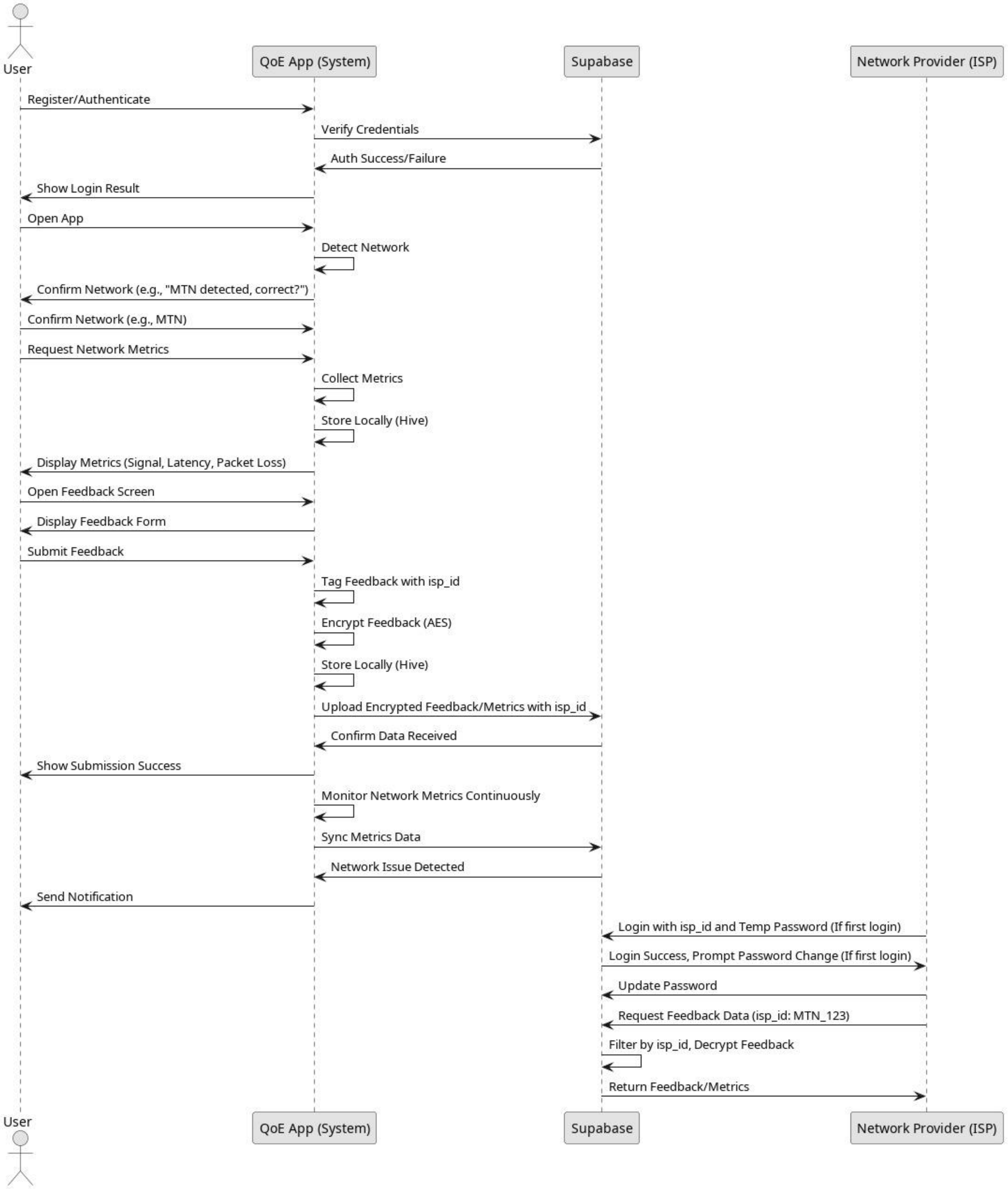
D. SEQUENCE DIAGRAM:

The Sequence Diagram maps the dynamic flow for the “Submit Feedback” use case, detailing the interaction sequence between the User, the mobile app, and the backend database.

- The process begins with the User initiating feedback submission through the app’s interface. The app first verifies the user’s authentication status with the backend.
- If authenticated, the app collects the user’s rating and comments, encrypts the data for security, and stores it locally.
- The app then syncs the feedback to the backend database, which updates the relevant records (linked to user and ISP identifiers).
- The backend confirms the operation, and the app notifies the User of success.
- Simultaneously, the ISP accesses this feedback via a dashboard interface, querying the backend for specific data.

The diagram highlights the step-by-step flow, emphasizing secure data handling and real-time synchronization between actors.

Diagram:


User
QoE App (System)
Supabase
Network Provider (ISP)
Register/Authenticate
Verify Credentials
Auth Success/Failure
Show Login Result
Open App
Detect Network
Confirm Network (e.g., "MTN detected, correct?")
Confirm Network (e.g., MTN)
Request Network Metrics
Collect Metrics
Store Locally (Hive)
Display Metrics (Signal, Latency, Packet Loss)
Open Feedback Screen
Display Feedback Form
Submit Feedback
Tag Feedback with isp_id
Encrypt Feedback (AES)
Store Locally (Hive)
Upload Encrypted Feedback/Metrics with isp_id
Confirm Data Received
Show Submission Success
Monitor Network Metrics Continuously
Sync Metrics Data
Network Issue Detected
Send Notification
Login with isp_id and Temp Password (If first login)
Login Success, Prompt Password Change (If first login)
Update Password
Request Feedback Data (isp_id: MTN_123)
Filter by isp_id, Decrypt Feedback
Return Feedback/Metrics
User
QoE App (System)
Supabase
Network Provider (ISP)

E. CLASS DIAGRAM:

The Class Diagram represents the static structure of the system's data model, focusing on key entities and their relationships. It includes entities such as:

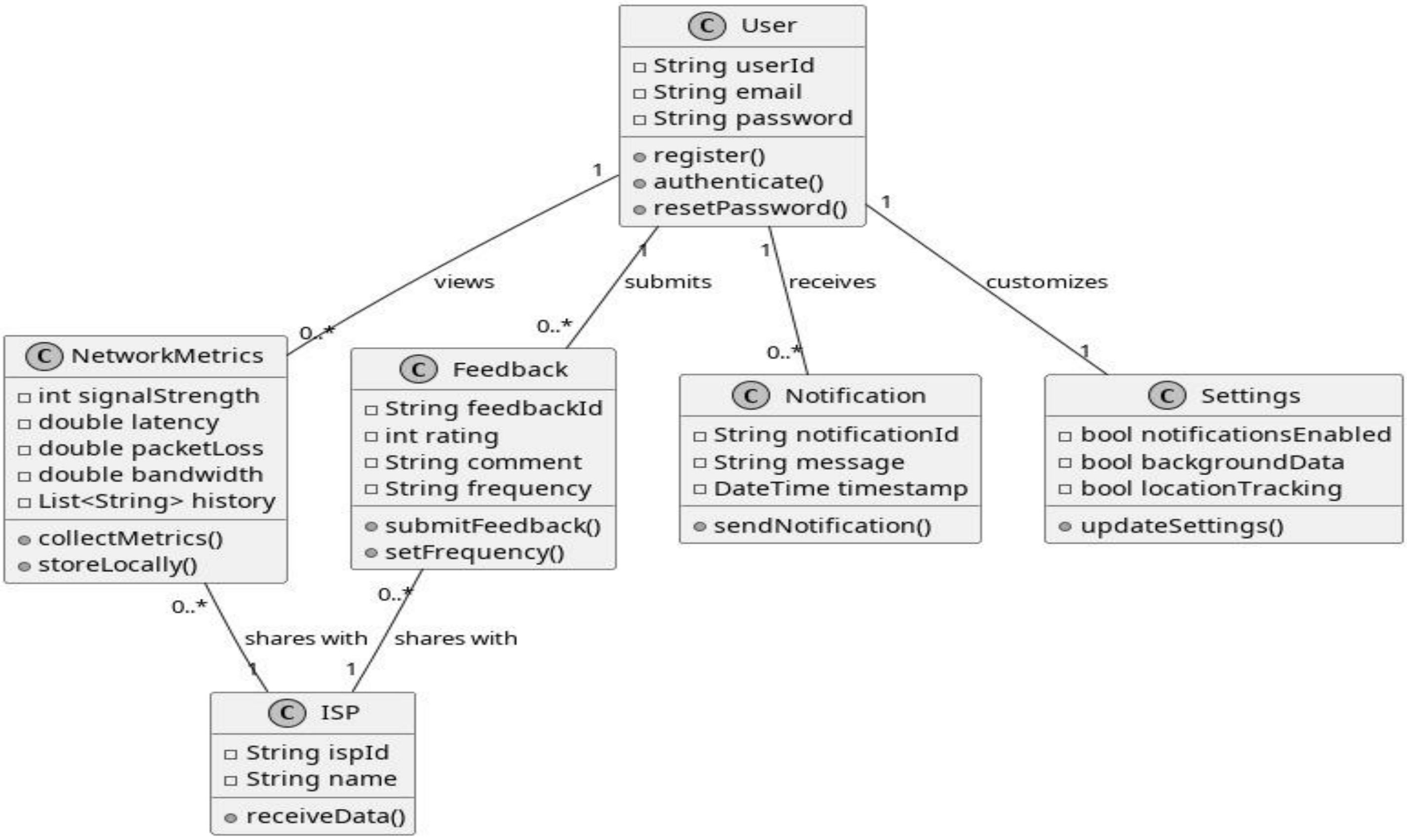
- User (UserID, Email, Password)
- NetworkMetrics (MetricID, UserID, ISPID, SignalStrength, Latency, PacketLoss, Timestamp)
- Feedback (FeedbackID, UserID, ISPID, Rating, Comment, Timestamp)
- ISP (ISPID, Name, Credentials).

Relationships are defined as:

- User has a one-to-many relationship with NetworkMetrics and Feedback (one user submits multiple entries)
- NetworkMetrics and Feedback each have a many-to-one relationship with ISP (many entries belong to one ISP).

The diagram uses a monochrome style, presenting a clear, hierarchical structure of entities and their associations, serving as a blueprint for the system's data architecture.

DIAGRAM:



F. DEPLOYMENT DIAGRAM:

This diagram shows how components are deployed across hardware.

✓ User Device:

- **Description**:
  - Hardware used by the end user (e.g., smartphone or tablet).
- **Subcomponent**: **Flutter App**
  - Cross-platform mobile app built using Flutter (iOS & Android support).
  - Provides UI for:
    - User registration
    - Feedback submission
    - Viewing network metrics
  - Communicates via HTTP/HTTPS with backend services.
- **Role**:
  - Hosts the QOE App for user interaction and data submission.
- **Connections**:
  - Sends HTTP/HTTPS requests to **Supabase API** for:
    - Authentication
    - Feedback submission
    - Metrics retrieval
  - Queries **ISP API** for real-time network metrics.
  - Sends performance logs to **Local Monitoring** on the **Local Server**.

✓ Local Server

- **Description**:
  - A PC or dedicated server within the local network.
  - Hosts the Flutter build/distribution environment and monitoring tools.
- **Subcomponents**:
  - **Frontend Server**:
    - **Flutter Build**:
      - Builds the Flutter App.
      - Generates APKs (Android) or serves for local testing (web/installation).
  - **Local Monitoring**:
    - Logging mechanism for capturing app performance/errors.
    - Stores logs sent by the Flutter App.
- **Role**:
  - Builds and distributes the QOE App.
  - Provides local performance monitoring environment.
- **Connections**:
  - Receives logs from Flutter App.
  - Updated by **CI/CD Pipeline** with latest builds.

✓ **Supabase Cloud:**

Description:

- Managed backend-as-a-service platform supporting the QOE App.
- **Subcomponents**:
  - **Backend Service**:
    - **Supabase API**:
      - REST API for frontend-backend interaction.
      - Handles:
        - User authentication
        - Feedback storage (AES-encrypted)
        - Network metrics retrieval
  - **PostgreSQL DB**:
    - Stores:
      - User data
      - Encrypted feedback
      - Network metrics
    - Uses *isp_id* to tag and relate data to specific providers.
- **Role**:
  - Backend service for secure data handling and user management.
- **Connections**:
  - Receives API requests from Flutter App (auth, feedback, metrics).
  - Interacts with PostgreSQL DB for data storage and retrieval.

✓ **Network Provider:**

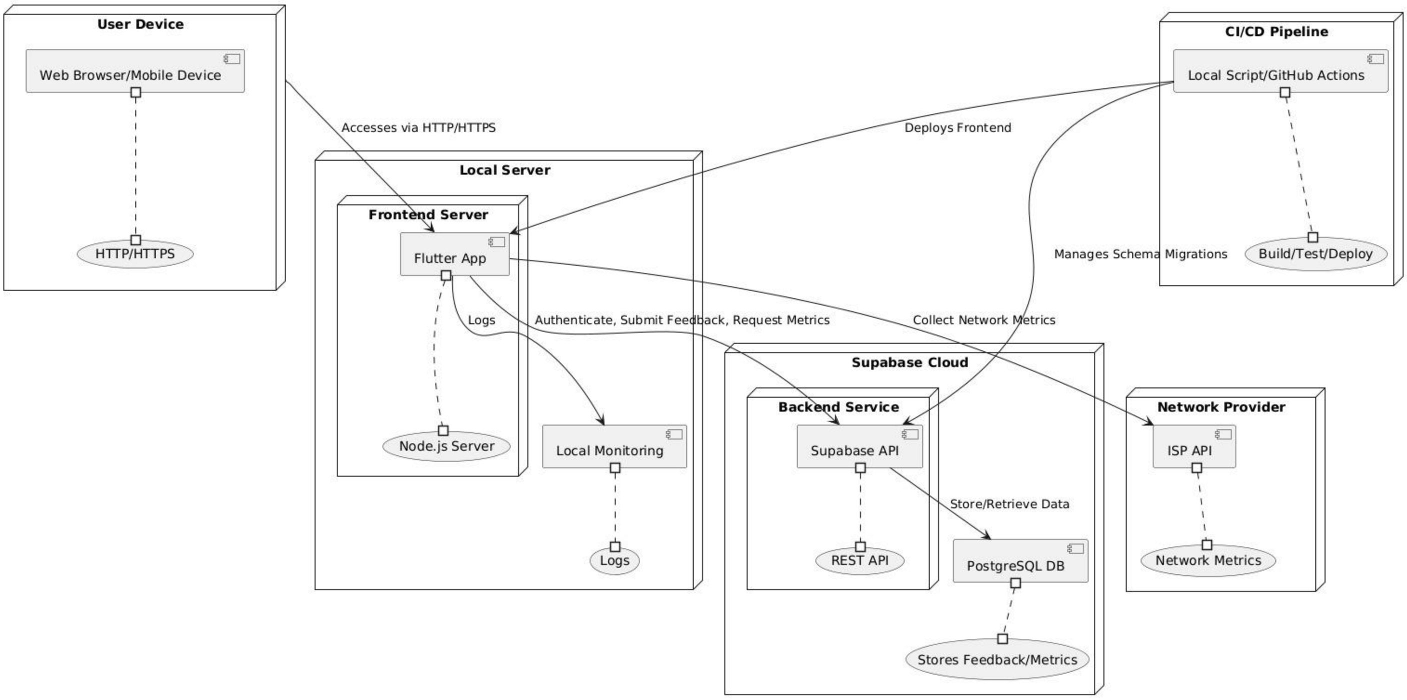
- **Description**:
  - External ISP (Internet Service Provider).
- **Subcomponent**: **ISP API**
  - Provides real-time network metrics:
    - Signal strength
    - Latency
    - Packet loss
- **Role**:
  - Supplies performance data for evaluation.
- **Connection**:
  - Queried directly by the Flutter App for metrics collection.

✓ **CI/CD Pipeline:**

- **Description**:
  - Automates build, test, and deployment of app components.

- **<u>Subcomponent</u>: Local Script / GitHub Actions**
  - Uses a local runner or GitHub Actions to:
    - Build and test the Flutter frontend (generates APKs).
    - Deploy frontend builds to the **Local Server**.
    - Manage backend schema migrations (Supabase DB updates).
- **<u>Role</u>:**
  - Ensures efficient and automated updates of the app.
- **<u>Connections</u>:**
  - Deploys builds to **Flutter Build** on Local Server.
  - Manages schema changes for the **Supabase API**.

DIAGRAM:



TOOLS USED:

- ✓ PlantUML
- ✓ StarUML

G. CONCLUSION:

This task provided a comprehensive foundation for understanding and designing the architecture of the mobile application for collecting user experience data in real time. Through the development of various system modeling diagrams—including the Context Diagram, Data Flow Diagram (DFD), Use Case Diagram, Sequence Diagram, Class Diagram, and

Deployment Diagram—we successfully visualized and documented the functional and structural aspects of the system.

These models collectively enabled us to:

- Define the boundaries and interactions of the system with external entities (Context Diagram),
- Illustrate the flow of data within the system and its transformation (DFD),
- Capture the functional requirements from the user's perspective (Use Case Diagram),
- Detail the interaction between objects over time (Sequence Diagram),
- Describe the static structure and relationships between classes (Class Diagram),
- Depict the physical deployment of software components (Deployment Diagram).

Overall, Task 4 established a solid blueprint for implementation in subsequent phases by identifying key components, processes, and data flows essential to achieving the project's goals. This phase ensured clarity in design, reduced ambiguity, and laid the groundwork for efficient development and integration in Tasks 5 and 6.

GROUP 20 MEMBERS:

Name	Matricule
TIOKENG SAMUEL	**FE19A110**
KUE KOUOKAM GILLES BRYTON	**FE22A235**
LUM BLESSING NFORBE	**FE22A239**
NWETBE NJIWUNG LORDWILL	**FE22A285**
TALLA TIZA AGYNUI	**FE22A304**