INTERNET PROGRAMMING AND MOBILE PROGRAMMING

CEF440

GROUP 20

NetPulse is a mobile application built to capture and analyze real-world user experiences with mobile networks. By combining user feedback with live signal performance data, it delivers a clear, location-aware view of network quality. The app bridges the gap between subjective experience and technical performance, empowering users and providers with meaningful insights. With a clean interface and a powerful backend, NetPulse turns everyday network usage into actionable intelligence.

Jesus

COURSE INSTRUCTOR: DR VALAERY NKEMENI

TABLE OF CONTENT

TASK 6:

DATABASE DESIGN AND IMPLEMENTATION

INTRODUCTION:

This details the comprehensive database design and implementation for the Quality of Experience (QoE) Mobile Application. The app aims to empower users to monitor their network performance and provide direct feedback to Internet Service Providers (ISPs). This document outlines the conceptual design, data elements, ER diagram, and the practical implementation using Supabase for the database, backend, and critical Row Level Security (RLS). The goal is to establish a secure, scalable, and efficient data infrastructure that supports real-time network insights and facilitates continuous improvement in mobile network services.

1. **Data Elements:**
   The database will store comprehensive information about users, their network experiences, feedback, and ISP interactions to support the Quality of Experience (QoE) Mobile App. Here are the detailed data elements:

   - **Users:**

     - **User ID:** A unique UUID (e.g., "550e8400-e29b-41d4-a716-446655440000") serving as the primary key for each user.
     - **Email:** The user's email address (e.g., "user@example.com"), stored as a unique text field for login and identification.
     - **Password:** A secure, hashed version of the password, stored as text, ensuring encryption for security.
     - **PhoneNumber:** The user's phone number (e.g., "+237670123456"), stored as text, providing a direct link for ISP recognition and communication.
     - **created_at:** Timestamp when the user account was created (e.g., "2025-05-24 09:00:00+00"), automatically generated as a timestamp with timezone.
     - **Purpose:** Tracks user identities, manages authentication, and links to their submitted metrics and feedback.

   - **Network Metrics:**

     - **Metric ID:** A unique UUID (e.g., "d3d09d59-7b6d-4ba2-b4de-7cbc28a8b12b") acting as the primary key for each measurement set.
     - **User ID:** A UUID foreign key linking to the Users table, identifying the user who generated the data.
     - **ISP ID:** A UUID foreign key linking to the ISPs table, identifying the network provider (e.g., "MTN").

- **Signal Strength:** An integer value representing network signal quality (e.g., -50 dBm, ranging from -120 to -30).
- **Latency:** A float value indicating time delay in network response (e.g., 50.5ms).
- **Packet Loss:** A float percentage of lost data (e.g., 2.3%).
- **Throughput:** A float value for data transfer rate (e.g., 102.4 kbps).
- **Latitude:** A float representing the geographic latitude (e.g., 3.87, in degrees).
- **Longitude:** A float representing the geographic longitude (e.g., 11.52, in degrees).
- **created_at:** Timestamp when the data was collected (e.g., "2025-05-24 09:00:00+00"), automatically generated as a timestamp with timezone.
- **Purpose:** Records real-time network performance data for users and ISPs to monitor and improve QoE.

- **Feedback:**
  - **Feedback ID:** A unique UUID (e.g., "28f69273-7825-4ce5-af65-034dd404ca6c") serving as the primary key for each feedback entry.
  - **User ID:** A UUID foreign key linking to the Users table, identifying the user who provided feedback.
  - **ISP ID:** A UUID foreign key linking to the ISPs table, tying feedback to the correct network provider.
  - **Rating:** An integer score from 1 to 5 stars (e.g., 4) reflecting user satisfaction.
  - **Comment:** A text field for the user's description (e.g., "Slow internet during peak hours"), with a maximum length of 500 characters.
  - **created_at:** Timestamp when feedback was submitted (e.g., "2025-05-24 10:00:00+00"), automatically generated as a timestamp with timezone.
  - **Purpose:** Captures detailed user opinions to assist ISPs in identifying and addressing service issues.
- **ISPs:**
  - **ISP ID:** A unique UUID (e.g., "7d736441-ec49-473c-9539-dbb8740b16aa") acting as the primary key for each provider.
  - **Name:** A unique text field for the provider's name (e.g., "MTN Cameroon").
  - **Created_at:** Timestamp when the ISP record was created (e.g., "2025-01-01 00:00:00+00"), automatically generated as a timestamp with timezone.
  - **Purpose:** Identifies and authorizes network providers, enabling data segregation and ISP-specific analytics.

- **Additional Note:** Excluded "Credentials" from the original design, opting for Supabase authentication roles to manage ISP access securely.

2. **Conceptual Design:**
   The database is structured into four main entities (Users, Network Metrics, Feedback, ISPs) to fulfill the QoE Mobile App's objectives of tracking network performance and user feedback:

   - **Users** manage their accounts, submit network metrics, and provide feedback, serving as the central entity.
   - **Network Metrics** capture real-time performance data, linked to individual users and their respective ISPs for contextual analysis.
   - **Feedback** collects user ratings and comments, tied to users and ISPs to provide actionable insights.
   - **ISPs** act as the network providers, receiving and analyzing data specific to their services for quality improvement.

   Relationships:
   - **Users to Network Metrics and Feedback:** A **one-to-many** relationship, where each user can submit multiple metric sets and feedback entries, enforced via UserID as a foreign key.
   - **Network Metrics and Feedback to ISPs:** A **many-to-one** relationship, where each metric or feedback entry is associated with one ISP, enforced via ISPID as a foreign key.
   - **Purpose:** This relational design supports user-specific data collection while enabling ISPs to access aggregated, ISP-specific insights, ensuring scalability and data integrity.

3. **ER Diagram:**
   The ER diagram, as shown in the figure below, visually represents the relationships:
   Entities:

   - **Users** with attributes UserID, Email, Password, PhoneNumber, and created_at.
   - **NetworkMetrics** with attributes MetricID, UserID, ISPID, SignalStrength, Latency, PacketLoss, Throughput, Latitude, Longitude, and created_at.
   - **Feedback** with attributes FeedbackID, UserID, ISPID, Rating, Comment, and created_at.
   - **ISPs** with attributes ISPID, Name, and Created_at.

   Relationships:
   - **Users** have a **one-to-many** relationship with NetworkMetrics and Feedback, indicating each user can submit multiple records.
   - **NetworkMetrics** and **Feedback** have a **many-to-one** relationship with ISPs, reflecting that multiple records tie to a single ISP.

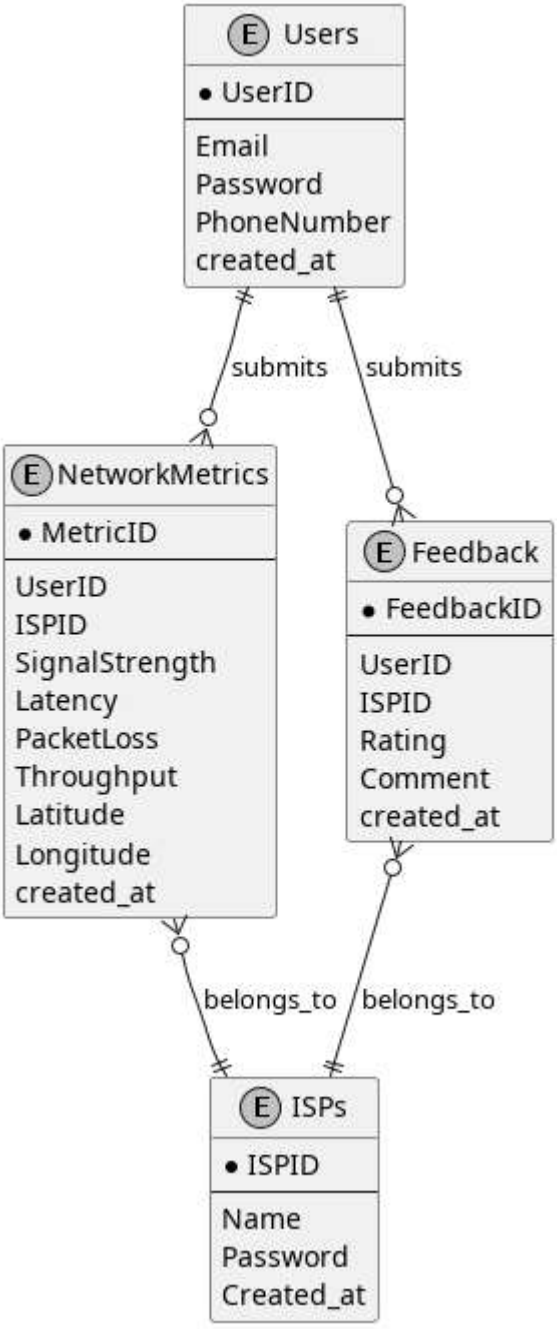

Fig-1: ER Diagram

4. **Database Implementation:**
   The database was implemented using Supabase, a managed PostgreSQL service, with a detailed process:

   - **Setup:** Created a Supabase project via the web console, obtaining the project URL and anon key for client integration.
   - **Table Creation:**
     - **Users:** Created with CREATE TABLE Users (UserID UUID PRIMARY KEY DEFAULT uuid_generate_v4(), Email TEXT UNIQUE NOT NULL, Password TEXT NOT NULL, PhoneNumber TEXT, created_at TIMESTAMPTZ DEFAULT NOW());.
     - **NetworkMetrics:** Defined with CREATE TABLE NetworkMetrics (MetricID UUID PRIMARY KEY DEFAULT uuid_generate_v4(), UserID UUID REFERENCES Users(UserID), ISPID UUID REFERENCES ISPs(ISPID), SignalStrength INTEGER, Latency FLOAT, PacketLoss FLOAT, Throughput FLOAT, Latitude FLOAT, Longitude FLOAT, created_at TIMESTAMPTZ DEFAULT NOW());.
     - **Feedback:** Established with CREATE TABLE Feedback (FeedbackID UUID PRIMARY KEY DEFAULT uuid_generate_v4(), UserID UUID REFERENCES Users(UserID), ISPID UUID REFERENCES ISPs(ISPID), Rating INTEGER CHECK (Rating BETWEEN 1 AND 5), Comment TEXT, created_at TIMESTAMPTZ DEFAULT NOW());.
     - **ISPs:** Set up with CREATE TABLE ISPs (ISPID UUID PRIMARY KEY DEFAULT uuid_generate_v4(), Name TEXT UNIQUE NOT NULL, Created_at TIMESTAMPTZ DEFAULT NOW());.
   - **Indexes:** Added indexes on foreign keys (CREATE INDEX ON NetworkMetrics(UserID, ISPID);, CREATE INDEX ON Feedback(UserID, ISPID);) to optimize query performance.
   - **Data Population:** Seeded ISPs with initial entries (e.g., “MTN”, “CAMTEL”, “ORANGE”) via SQL inserts, and allowed Users data to populate through app registrations.
   - **Validation:** Ensured data types and constraints (e.g., CHECK for Rating) were enforced at the database level.

5. **Backend Implementation:**
   The backend was developed using Supabase’s managed services, with a structured approach:

   - **Authentication:** Configured Supabase Auth with email/password login, enabling user signup (supabase.auth.signUp) and login, and enabled email validation so users are required to validate their emails up0n account creation (supabase.auth.signInWithPassword). Integrated role-based access for ISPs using custom claims if needed.

- **API Endpoints:** Utilized Supabase's REST API for CRUD operations:
  - POST /network_metrics: Inserted new metric data with validation for UserID and ISPID matching the authenticated user's ISP.
  - POST /feedback: Submitted feedback entries with similar validation.
  - GET /network_metrics: Retrieved metrics filtered by UserID and ISPID.
  - GET /feedback: Fetched feedback with the same filters.
- **Real-Time:** Enabled subscriptions on NetworkMetrics using supabase.channel('metrics').on('postgres_changes', ...).
- **Business Logic:** Implemented lightweight logic via Supabase Edge Function and Role Level Security (RLS) but most of this will be handled by client-side validation.
- **Performance:** Optimized by limiting returned fields and using pagination for large datasets.

6. **Connecting Database to Backend:**
   The connection between the database and backend was established through the Supabase client in the Flutter app:
   - **Integration:** Installed supabase_flutter via pubspec.yaml and initialized it with SupabaseClient(supabaseUrl, supabaseAnonKey) in a singleton pattern.
   - **Data Flow:**
     - **Authentication:** Used supabase.auth.signInWithPassword to obtain a session, storing UserID and deriving ISPID from user metadata or a linked table.
     - **CRUD Operations:** Implemented functions like:
       - supabase.from('NetworkMetrics').insert({'UserID': user.id, 'ISPID': ispId, ...}) for inserts.
       - supabase.from('Feedback').select('*', const {'filter': 'eq', 'UserID': user.id, 'ISPID': ispId}) for queries.
     - **Real-Time:** Subscribed to NetworkMetrics changes with supabase.channel('metrics').on('postgres_changes', (payload) => {...}) to update the app UI live.
   - **Error Handling:** Added try-catch blocks (e.g., try { await supabase.from('NetworkMetrics').insert(data); } catch (e) { print(e.message); }) to manage network errors and invalid data, logging to the console.
   - **Security:** Ensured all requests included the auth token via the Supabase client, aligning with RLS policies.

- **Testing:** Validated connections by simulating user actions (e.g., submitting metrics from the app) and verifying data in the Supabase dashboard, testing edge cases like unauthorized access.

7. **RLS Implementation:**
   Row Level Security (RLS) was implemented in Supabase to enforce data access control, tailored to the mobile app's multi-user, multi-ISP environment:

   - **Setup:** Enabled RLS on all tables (ALTER TABLE table_name ENABLE ROW LEVEL SECURITY;) and defined policies using the Supabase SQL editor.
   - **Policies:**
     - **Users:**
       - **Policy:** CREATE POLICY "Users can view and update own data" ON Users FOR ALL TO authenticated USING (auth.uid() = UserID);
       - **Effect:** Allowed authenticated users to SELECT, INSERT, UPDATE, and DELETE only their own records.
     - **NetworkMetrics:**
       - **Policy:** CREATE POLICY "Users can manage own metrics" ON NetworkMetrics FOR ALL TO authenticated USING (auth.uid() = UserID) WITH CHECK (auth.uid() = UserID AND EXISTS (SELECT 1 FROM Users WHERE UserID = auth.uid() AND ISPID IN (SELECT ISPID FROM ISPs WHERE ISPID = NetworkMetrics.ISPID)));
       - **Effect:** Restricted access to metrics where the UserID matches the authenticated user and the ISPID aligns with the user's associated ISP, ensuring ISP-specific data isolation.
     - **Feedback:**
       - **Policy:** CREATE POLICY "Users can manage own feedback" ON Feedback FOR ALL TO authenticated USING (auth.uid() = UserID) WITH CHECK (auth.uid() = UserID AND EXISTS (SELECT 1 FROM Users WHERE UserID = auth.uid() AND ISPID IN (SELECT ISPID FROM ISPs WHERE ISPID = Feedback.ISPID)));
       - **Effect:** Limited feedback access and modifications to the user's own entries, with ISPID validation matching their ISP.
     - **ISPs:**
       - **Policy:** CREATE POLICY "Authenticated users can view ISPs" ON ISPs FOR SELECT TO authenticated USING (true);
       - **Policy:** CREATE POLICY "Admins can manage ISPs" ON ISPs FOR ALL TO authenticated USING (auth.role() = 'admin');

        - **Effect:** Allowed all authenticated users to view ISP names, while restricting INSERT and UPDATE to admin roles, managed via Supabase authentication.

    - **Testing:** Verified RLS by logging in as different users, attempting unauthorized access (e.g., querying another user's metrics), and confirming access was denied with appropriate HTTP 403 responses.

    - **Optimization:** Added indexes on UserID and ISPID to improve RLS query performance.

This detailed process ensured a secure, scalable backend for the QoE Mobile App, with RLS providing fine-grained access control tailored to the ISP-user relationship.

Conclusion:

This outlines a robust and well-structured database design and implementation for the Quality of Experience (QoE) Mobile App. By meticulously defining **data elements** across users, network metrics, feedback, and ISPs, the design ensures comprehensive data capture essential for monitoring and improving service quality. The **conceptual design** establishes clear relational links, creating a scalable foundation that accurately represents the complex interplay between users and network providers. The **ER diagram** visually confirms these relationships, offering clarity and precision.

Leveraging **Supabase** has proven to be a strategic choice, providing a managed PostgreSQL service that simplifies database implementation, backend development, and connectivity with the Flutter front-end. Its integrated authentication, powerful REST APIs, real-time capabilities, and robust **Row Level Security (RLS)** features have been instrumental in building a secure and efficient system. The RLS policies, in particular, ensure granular access control, safeguarding user data and upholding ISP-specific data isolation, which is critical for a multi-tenant application.

GROUP 20 MEMBERS:

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