**Leadership Close to You System**

**System Architecture**

**1. Client Layer (Multiplatform + SMS)**

| **Platform** | **Framework/Tech** | **Key Functions** |
| --- | --- | --- |
| **Android/iOS** | Kotlin Multiplatform + Jetpack Compose/SwiftUI | Offline interaction, rich UI, sync when online |
| **Web / PWA** | Compose for Web / React with Kotlin/JS | Lightweight version, runs in browser, installable |
| **SMS** | Twilio or Africa’s Talking | Command-style access to core services |

**2. Backend Layer (APIs and Sync)**

| **Service** | **Technology** | **Notes** |
| --- | --- | --- |
| **Core API** | Kotlin + Ktor / Spring Boot | REST API for users, leaders, locations, reports |
| **Authentication** | Firebase Auth or Keycloak | OAuth + SMS login possible |
| **Data Sync API** | Ktor with GraphQL or REST + offline delta sync | Clients sync only changed data |
| **Admin Portal** | React.js / Compose Web | Admin views, approval, data entry |
| **PWA Gateway** | NGINX + Service Workers | Caches data, supports offline mode |

**3. Database Layer**

| **Type** | **Technology** | **Purpose** |
| --- | --- | --- |
| **Relational DB** | PostgreSQL | Primary structured data (users, regions, logs) |
| **Offline Local DB** | SQLite via SQLDelight (KMP) | Local storage per device |
| **Cache/Sync** | Redis or Local Caches | For performance + syncing |

**4. Messaging Layer (SMS)**

| **Component** | **Description** |
| --- | --- |
| **SMS Gateway** | Twilio, Africa's Talking, or Rwanda Telcos |
| **Inbound Processor** | Parses SMS to match commands like REPORT, INFO |
| **Outbound Service** | Sends confirmations, data summaries via SMS |

**5. Deployment/Hosting**

| **Component** | **Platform** | **Purpose** |
| --- | --- | --- |
| **Backend API** | Google Cloud Run / AWS ECS / Railway | Scalable deployment |
| **Database** | Supabase / GCP Cloud SQL | Reliable managed DB |
| **PWA / Web frontend** | Vercel / Firebase Hosting | Fast deploy and CDN |
| **CI/CD** | GitHub Actions or GitLab CI/CD | Auto-build mobile + web + backend |

**🔄 Sync & Offline Architecture**

| **Feature** | **Mechanism** |
| --- | --- |
| Offline storage | SQLDelight or Realm for mobile |
| Background sync | WorkManager (Android) / BackgroundTask (iOS) |
| Web sync | Service Worker + LocalStorage/IndexedDB |
| SMS command store | Queue messages to send when connection is restored |

**🔐 Security**

* JWT-based Auth (API)
* SMS OTP login for low-end devices
* Role-based access: Citizen, Leader, Admin
* Data encryption (at-rest & in-transit)

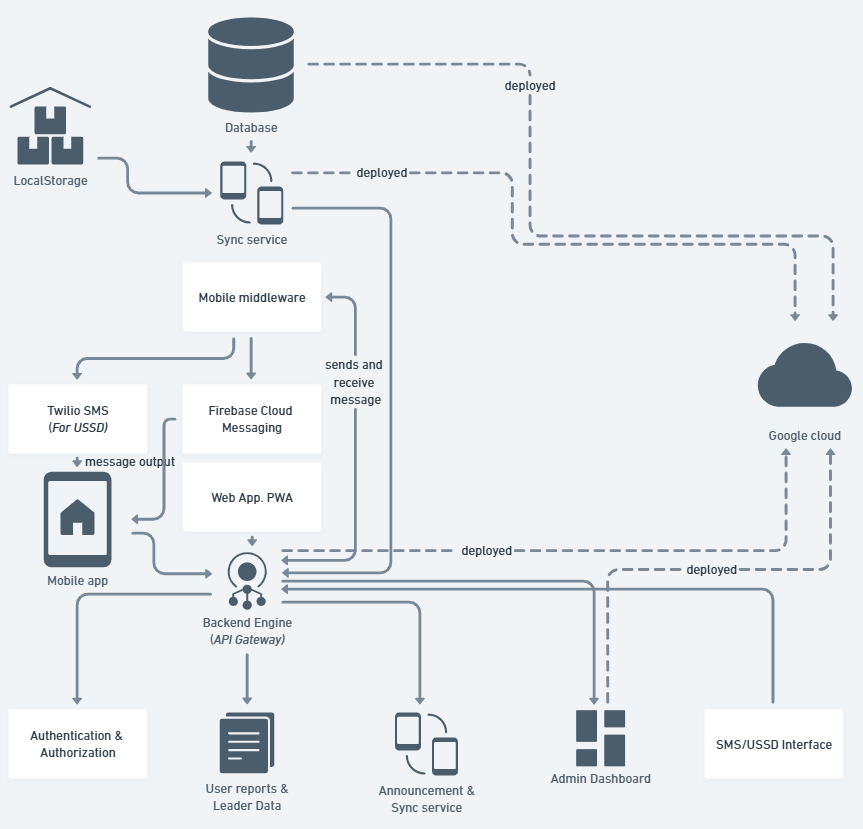
**🧩 Example High-Level Features**

| **Feature** | **How It Works** |
| --- | --- |
| **Report Issue** | Via app or SMS → synced to backend → geo-tagged for local leaders |
| **Find Local Leader** | Based on user district; cache on device and update via sync |
| **Receive Announcements** | Broadcast messages via API → pushed to apps/PWA → sent via SMS if needed |
| **Admin Approval Panel** | Admin web app built with Compose Web or React.js |

**📱 UI & UX Suggestions**

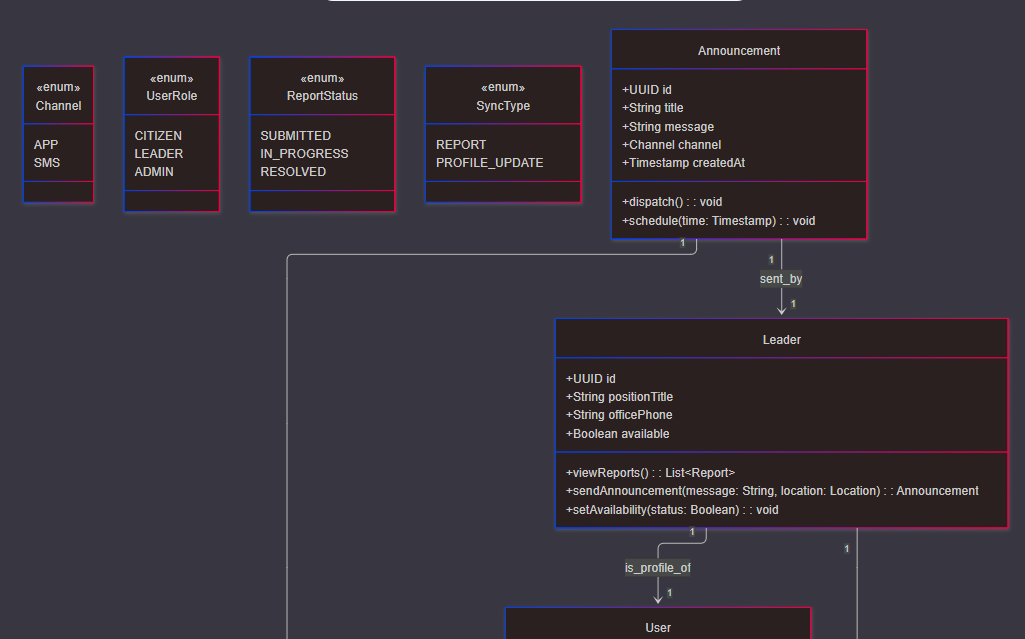
* Use Jetpack Compose (multiplatform UI)
* Provide multi-language support (Kinyarwanda, English, French)
* Offline "queued reports" with status (sent/pending/failed)
* Simple SMS syntax: INFO [Location], REPORT [Issue], etc.

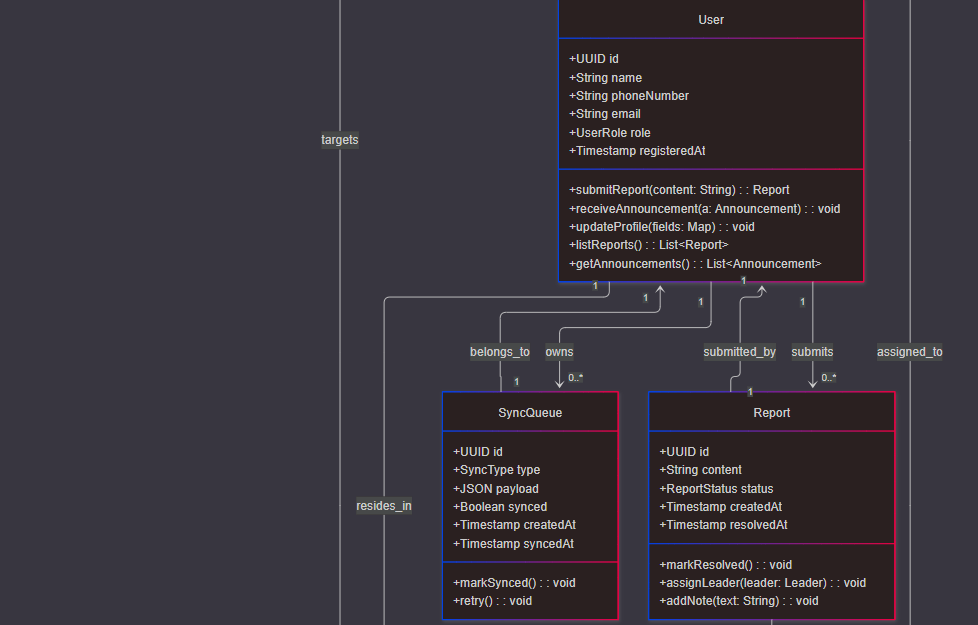
**High level architecture**

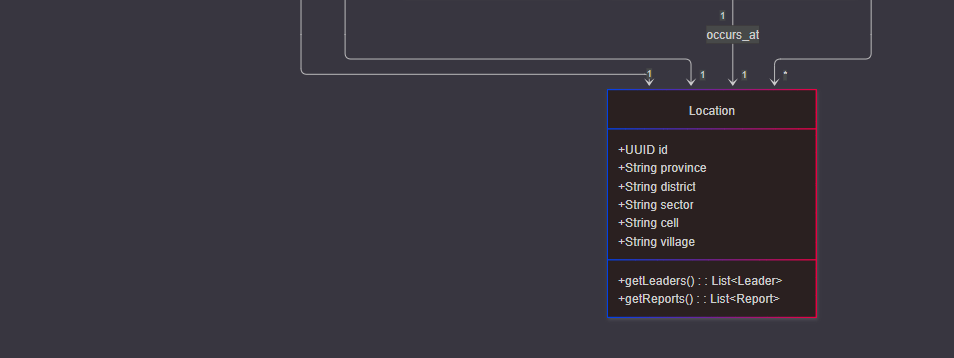
****

[**https://whimsical.com/leadership-close-to-you-Ps4bFZDD2uc2zuvzQpTyBs**](https://whimsical.com/leadership-close-to-you-Ps4bFZDD2uc2zuvzQpTyBs)

**Class diagram**

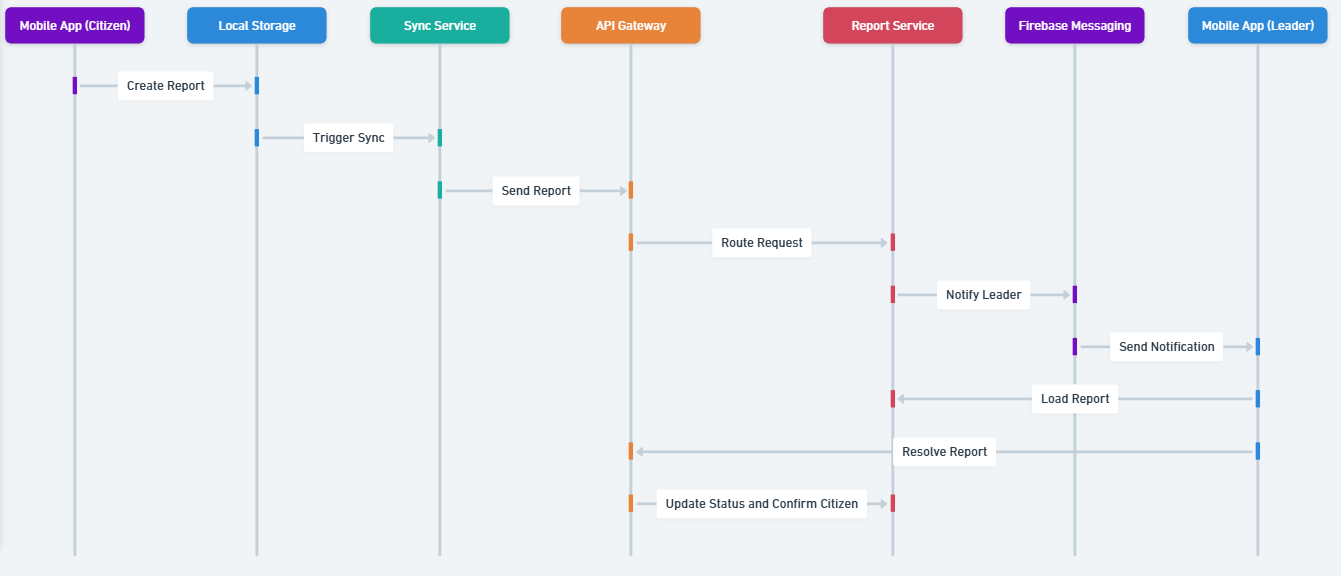
****

****

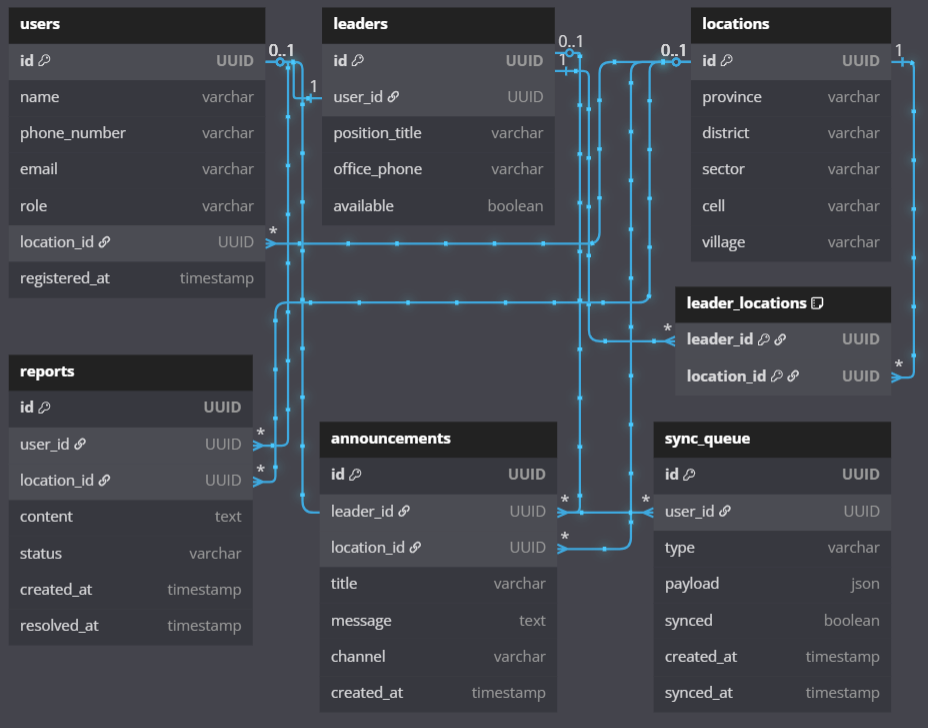
****

[**https://www.mermaidchart.com/raw/e7005374-f5d8-4681-b2ec-7059d343ad9a?theme=dark&version=v0.1&format=svg**](https://www.mermaidchart.com/raw/e7005374-f5d8-4681-b2ec-7059d343ad9a?theme=dark&version=v0.1&format=svg)

**Sequence diagram**

****

**ERD**

****

**Project structure**

leadership-close-to-you/

│

├── backend/ # Spring Boot or Ktor backend

│ ├── src/main/kotlin/

│ │ ├── com/leadership/

│ │ │ ├── controller/ # REST API endpoints

│ │ │ ├── service/ # Business logic

│ │ │ ├── repository/ # JPA/SQLExposed repos

│ │ │ ├── model/ # Data classes / entities

│ │ │ └── config/ # Security, CORS, DB config

│ ├── resources/

│ │ ├── application.yml # Spring configs

│ │ └── db/migration/ # Flyway migrations (PostgreSQL)

│ └── build.gradle.kts

│

├── mobileApp/ # Kotlin Multiplatform Mobile (Android/iOS)

│ ├── shared/ # Shared business logic (KMP)

│ │ ├── data/ # API clients, Sync logic

│ │ ├── model/ # Shared models (User, Report, etc.)

│ │ ├── storage/ # SQLDelight DB setup

│ │ └── util/ # Helper functions

│ ├── androidApp/ # Android-specific UI (Jetpack Compose)

│ ├── iosApp/ # iOS wrapper (UIKit/SwiftUI)

│ └── build.gradle.kts

│

├── webApp/ # Compose Web or React frontend

│ ├── src/

│ │ ├── components/

│ │ ├── pages/

│ │ ├── services/

│ │ └── main.kt # Entry point

│ └── build.gradle.kts

│

├── smsService/ # Twilio/Africa’s Talking integration

│ ├── parser/ # SMS parser (REPORT, INFO, etc.)

│ ├── handler/ # Logic to handle commands

│ └── scheduler/ # Retry/queue handler

│

├── database/ # DBML schema + migration scripts

│ ├── schema.dbml

│ ├── seed/

│ └── migrations/

│

├── deployment/ # DevOps (Docker, CI/CD, Terraform if needed)

│ ├── Dockerfile

│ ├── docker-compose.yml

│ └── github-actions/

│

├── docs/ # Architecture, README, diagrams

│ ├── system\_overview.png

│ ├── class\_diagram.mmd

│ └── README.md

│

└── settings.gradle.kts

**Development Phases**

**Phase 1: Backend API**

* Scaffold services: user, leader, report, announcement
* Implement PostgreSQL + JPA or SQLExposed
* Secure endpoints (JWT, roles)

**Phase 2: Mobile App (KMP)**

* Build shared logic: models, API client, offline DB (SQLDelight)
* Android UI first (Jetpack Compose)
* iOS wrapper with SwiftUI

**Phase 3: Web Frontend / PWA**

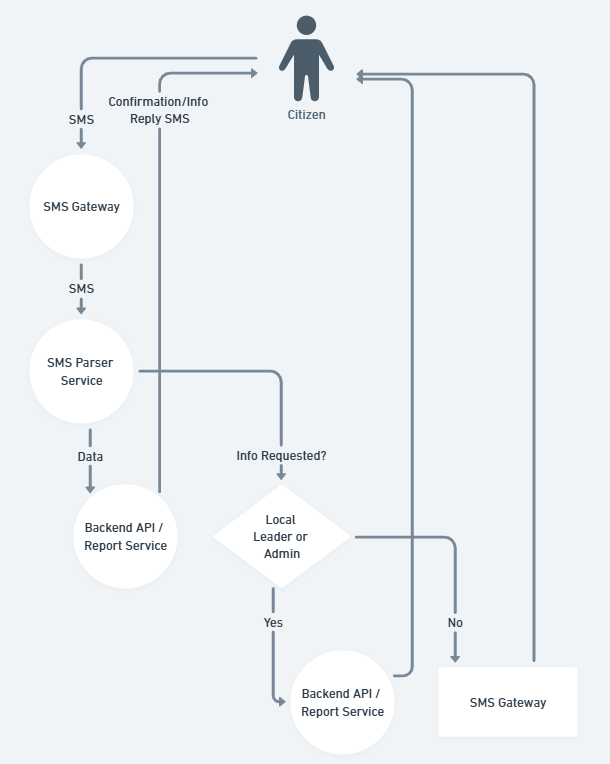
* Use Kotlin/JS + Compose for Web (or React if preferred)
* Public info + Admin dashboard

**Phase 4: SMS/USSD Service**

* Build SMS parser
* Integrate with Twilio/Africa’s Talking
* Route parsed commands to backend

**Phase 5: Deployment + Sync**

* Use GCP or Railway for deployment
* Sync service: retry offline data
* CI/CD for backend and frontend

****