Project Decision Report: EHR & Offline Architecture

1. Core Concept & Market Analysis

Core Idea: Connects patient, doctor, and pharmacy. Transfers datas from paper to digitalized version. Time saving, easily accessible.

The Idea: An EHR that connects Patients, Doctors, and Pharmacy and digitalizes the datas that always uses paper (either as prescription or appointment token etc...).

Main Work: Connects patient, doctor, and pharmacy. Transfer datas from paper to digitalized version. Time saving, easy accessible.

Competitor Analysis

Other companies close to us:

- Meddibuddy
- Practo
- Meddiassist

Novelty Factors

- Offline-first architecture
- Comparison of medicine's price across medical delivery platforms

2. Offline Architecture Strategy

Current Constraints

Offline Services (Doctor Side): Any services that can be entered but can't be seen-viewed by other entities until the internet is enough to push it into the cloud.

- Appointment booking
- Prescription Entry

Why not full offline?

- Cloud is the main transportation of data.

- The only way to access cloud is through internet.
- So only Entry services and the data's entered will be stored in an offline available database. When it gets internet, it pushes into the cloud making other updates visible.
- Appointments wouldn't be displayed - updation wouldn't happen across every display.
- API can't be accessed without connection.

Fault Found with Initial Idea

The product prioritizes patients with the most internet over the patient with least internet. To avoid this, an alternative approach is needed.

Issue Found: The data won't be updated for the entity who has no internet. (i.e. the data can be entered by entity who doesn't have internet. But... The updation will be shown only to the entity with internet).

What doesn't work in offline: The Data's that's been edited.

3. The Communication Approach

I don't know how the fuck the sms work but the apps can follow the same methodology and send the msg from patients (no matter what place they are from) to the place (or whatever technical shit that's been called) where the internet will be available.

Proposed Flow (with SMS Notification)

1. **Patient books appointment while offline**
    - App stores booking locally with status: PENDING_SYNC
    - Patient UI shows: Waiting list (Not sent yet)
2. **When patient gets internet (even briefly)**
    - App syncs booking to cloud
    - Cloud creates appointment request for Doctor A
3. **Doctor approves (doctor side online)**
    - Doctor hits Approve
    - Cloud marks appointment as APPROVED
    - Now cloud triggers SMS to patient: *"Appointment confirmed: Dr A, 10:00 AM, Token 52"*

Result: Patient receives SMS even if:

- Mobile data is OFF
- Internet is weak
- App is not opened

As long as they have cellular coverage (signal) and SMS service.

Technical Reality Check

> *"Use cellular network like SMS does, but with app data."*

That’s not exposed to regular apps. Operators keep those signaling channels private. So unless you are building with telecom-grade access (rare), your app can’t send arbitrary data over cellular without internet.

So its basically not possible to build an app which works like sms [unless we do a partnership with a telecom company]. Only notification systems can be embedded.

4. Tech Stack Recommendations

Front End

FLUTTER: Fast, not better for big projects.

REACT NATIVE: Easily deployable nd more work but better control on our side [**RECOMMENDED**].

Back End

FIREBASE: u guys know it.. easy to set.. we hav to pay but it is not much flexible.. hard to do offline mode.

SUPABASE: (Postgres + realtime + auth, free tier like Firebase but SQL).

NodeJS: more work from our side.. better for scalability nd deploying [**RECOMMENDED**].

Database (DB)

Local: PouchDB

Cloud: PostgreSQL

Why not MongoDB: Its harder to sync from offline DB… poor flexibility… PostgreSQL is easily connected to PouchDB cause they are of same base component.

5. Final Decision Summary

- **Frontend:** React Native
- **Backend:** Node.js + Express
- **Sync Architecture:** Custom queue (pending → synced)
- **SMS Solution:** Free Indian gateways
- **Deployment:** Railway (seem easy to deploy backend in this/ it got its own cloud db hosting too)