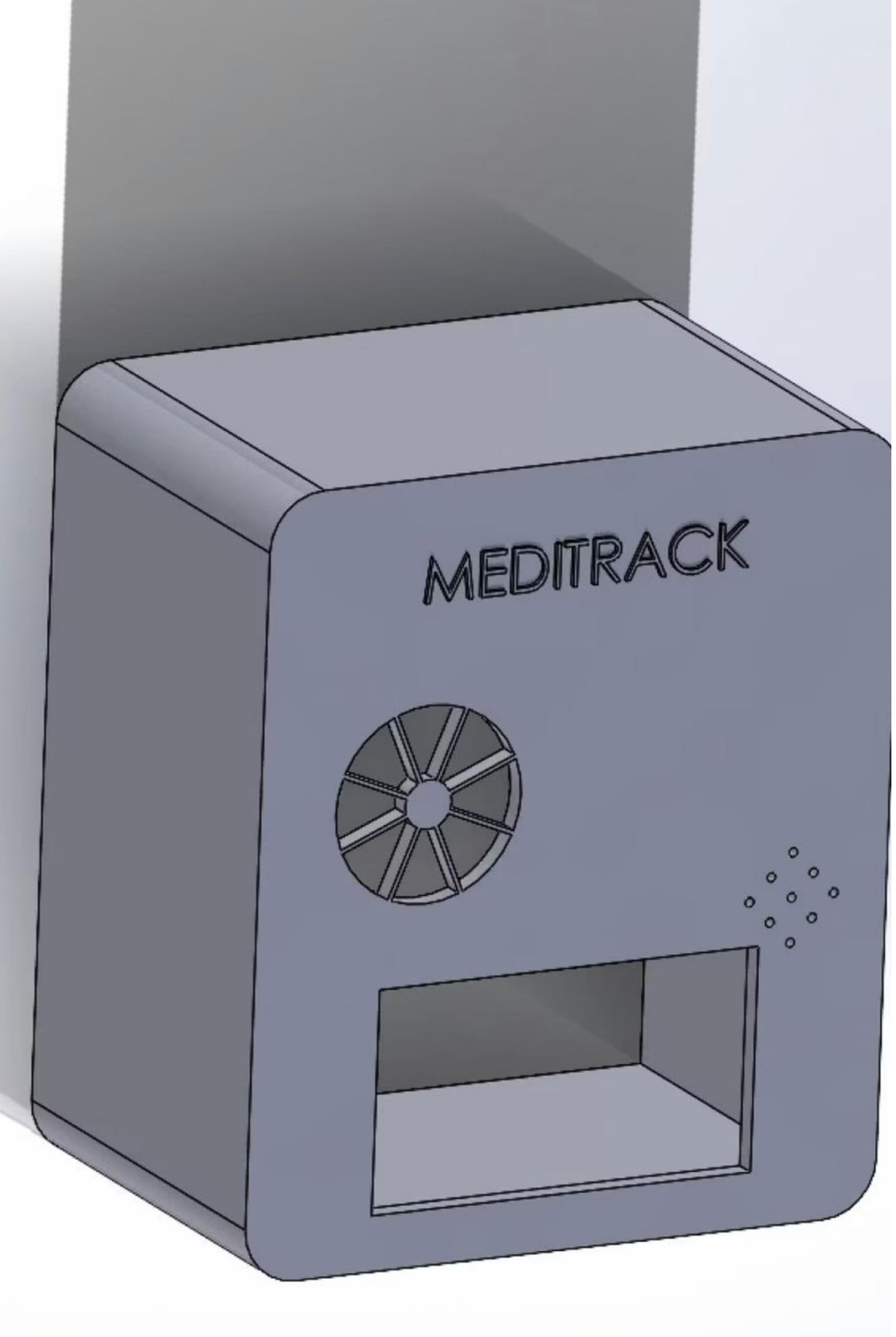


MediTrack: Automatic Medication Dispenser

Group 6 - CEG4912 | University of OttawaFaculty of Engineering - September 2025f

Addressing a Critical Healthcare Challenge

The Problem

50% of chronic patients don't follow prescriptions, leading to severe health issues, preventable hospitalizations, and increased healthcare costs nationwide.

Our Solution

MediTrack delivers automated, connected medication management ensuring the right dose at the right time with secure access and comprehensive caregiver monitoring.

System Requirements

1

Precision Dispensing

Accurate medication delivery with motor-driven mechanisms

Precision sensors to prevent dosing errors.

2

Multi-Modal Alerts

Comprehensive notification system including :

- audio alerts,
- visual indicators,
- haptic feedback for accessibility.

3

Secure Authentication

NFC integration

Bluetooth Low Energy (BLE) integration for tamper-proof access control

User verification.

4

Remote Monitoring

Real-time caregiver notifications

Cloud-based adherence tracking for comprehensive patient oversight.

Non-Functional Requirements

Beyond basic functionality, the MediTrack system must meet specific quality attributes and constraints to ensure its effectiveness and user satisfaction.

1

Accuracy & Precision

More than 90% dispensing precision

Reliable medication delivery

2

Power & Reliability

Backup power supply system

Continuous operation capability

3

Security & Safety

Secure design

Tamper-resistant design

Protected access control

4

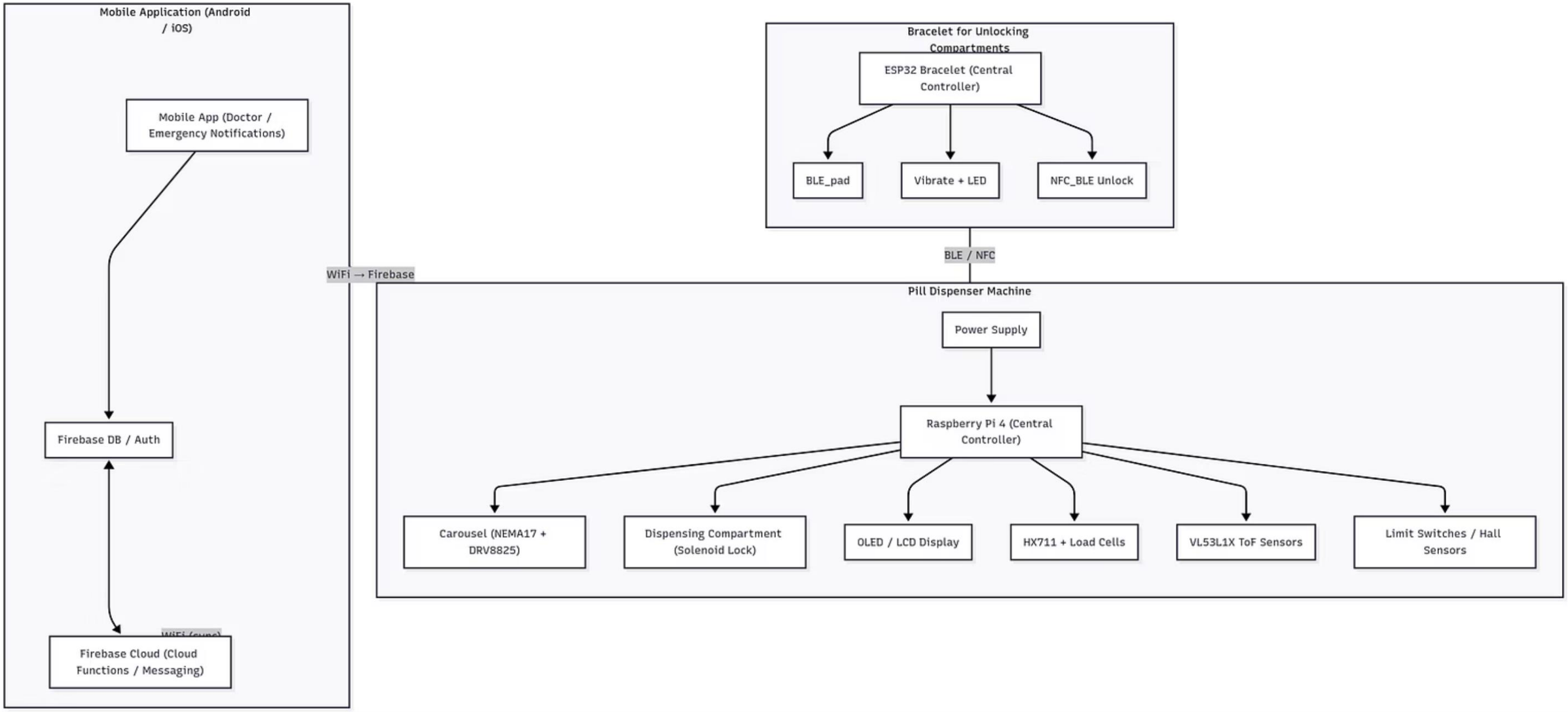
Usability

Simple interface design

Easy operation for elderly users

Intuitive user experience

Hardware Architecture



Hardware Architecture

Raspberry Pi 4

(Main Controller)

ESP32 Dev Board

(Secondary MCU)

Dispensing System

- NEMA17 Stepper Motor + Driver
- Limit Switch / Hall Sensor

Weight & Detection System

- Load Cells (6x) + HX711 Amplifiers (6x)
- ToF Distance Sensors (VL53L1X, 2x)

User Interface & Alerts

- OLED/LCD Display (0.96" I²C)
- RGB LED Module
- Push Button
- Speaker/Buzzer
- Vibration Motor (wristband)

Connectivity & Security

- NFC/BLE (via ESP32)
- Wi-Fi (via Raspberry Pi 4 + ESP32)

Power System

- 5V/3A USB-C Adapter
- Li-ion Battery Pack (7.4V, 5000 mAh) + TP4056
- Li-Po Battery (500 mAh for bracelet)

Software Architecture

Layer 4 - Alerts & Monitoring

- Local Alerts (LED, buzzer, vibration)
- Remote Notifications (Firebase push)
- Event Logging (adherence tracking)

Layer 3 - Database & Cloud

- Firebase Realtime Database/Firestore
- Firebase Authentication
- Cloud Sync APIs

Layer 2 - Backend (System Logic)

- Raspberry Pi Control Logic (Python/C++)
- ESP32 Firmware (BLE/NFC communication)
- Task Scheduler (timing doses, alerts)

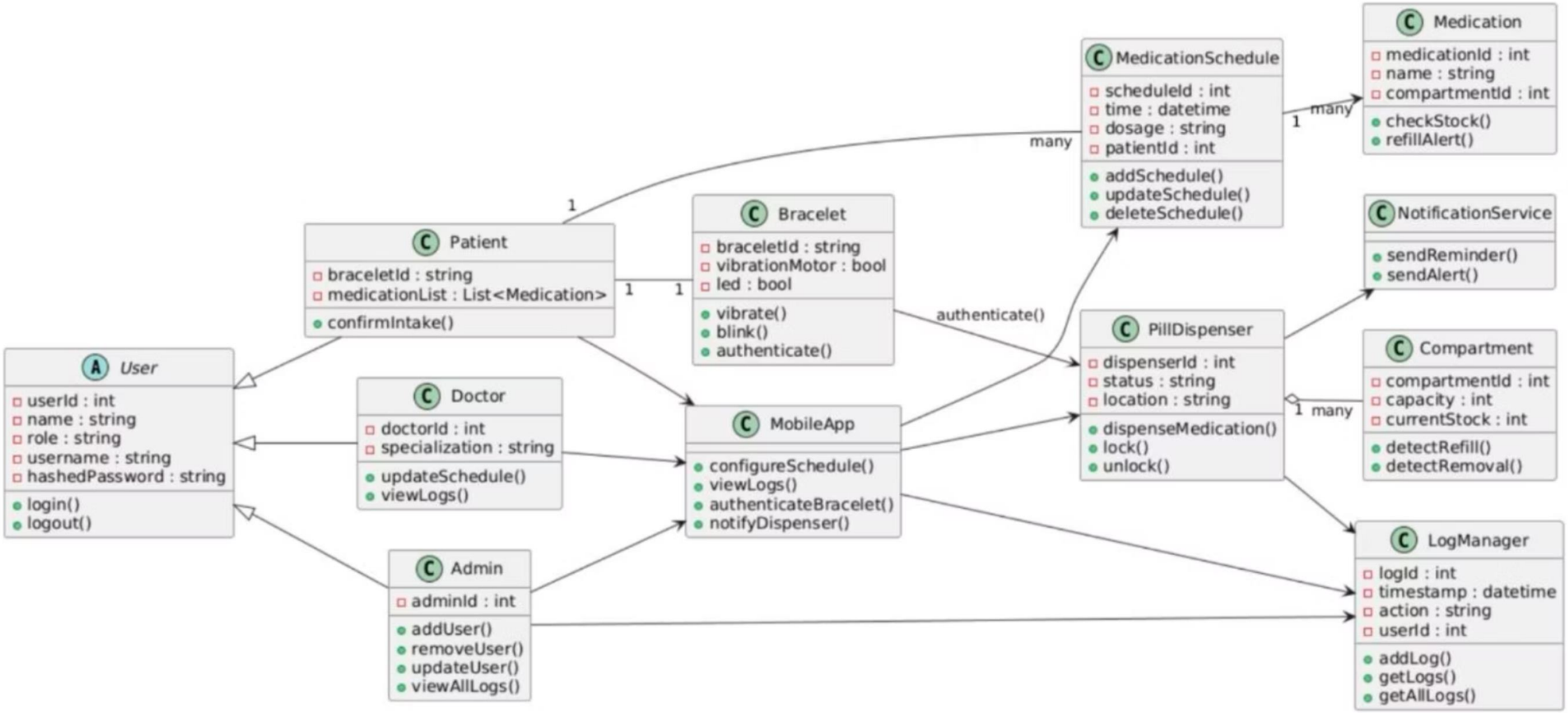
Layer 1 - Frontend (User Interface)

- Mobile App UI (React Native/Flutter)
- OLED/LCD Local Display (I²C)
- Push Button Input

Key Data Flow:

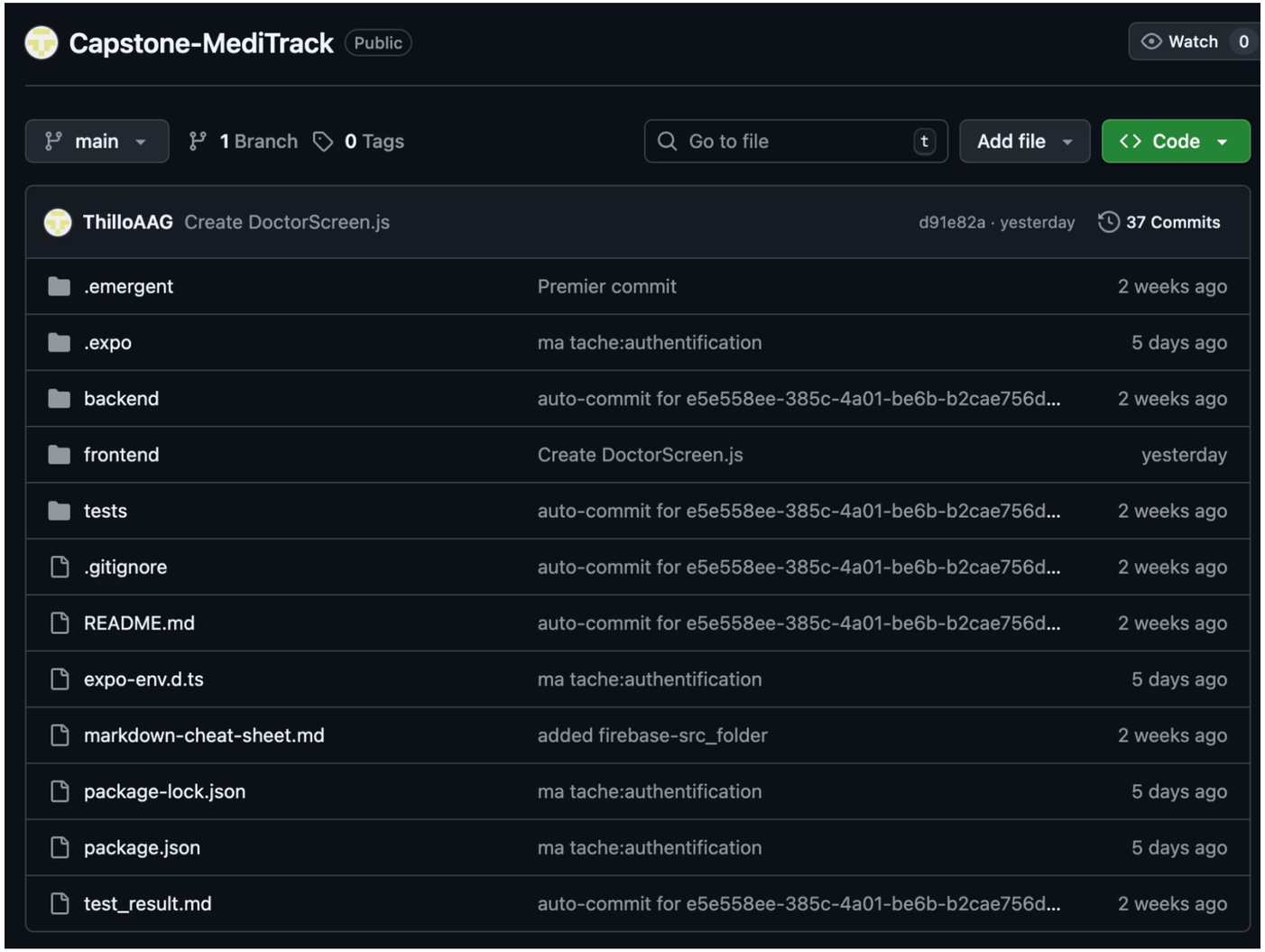
- User (Mobile App) sends schedules & credentials to Firebase.
- Raspberry Pi pulls schedules from Firebase and controls hardware.
- ESP32 communicates with Raspberry Pi via BLE/Wi-Fi for wristband & authentication.
- Firebase sends caregiver notifications.

Software Architecture

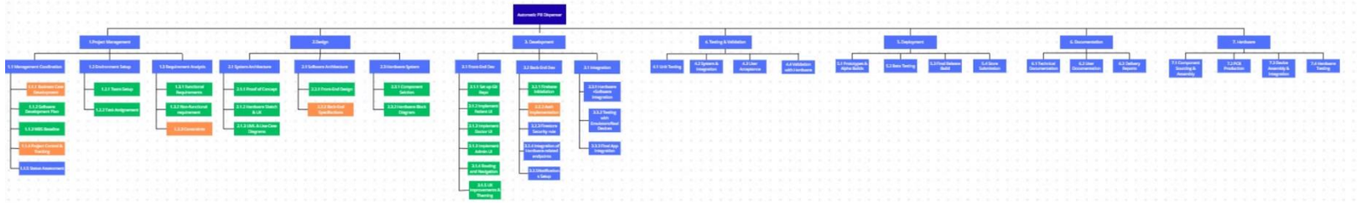


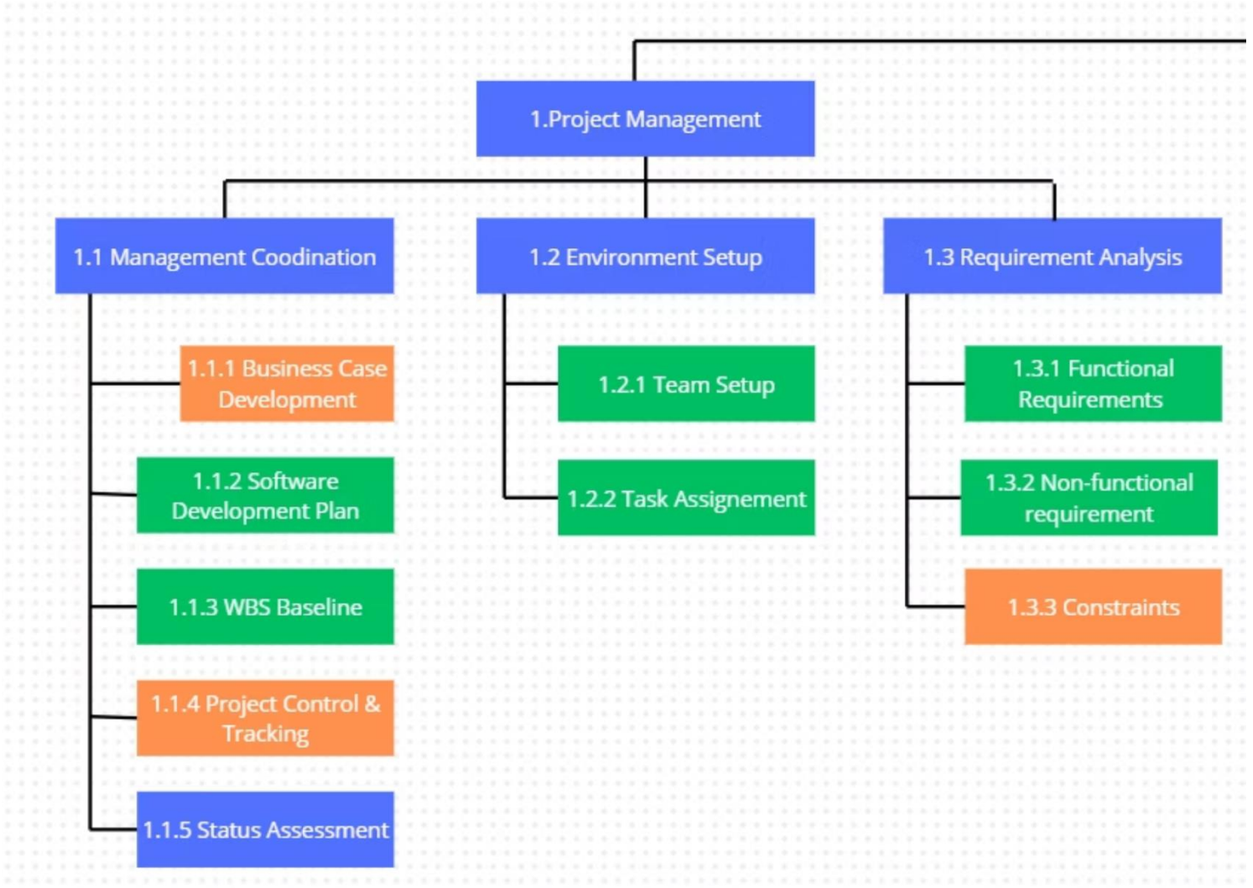
GitHub Repository Structure

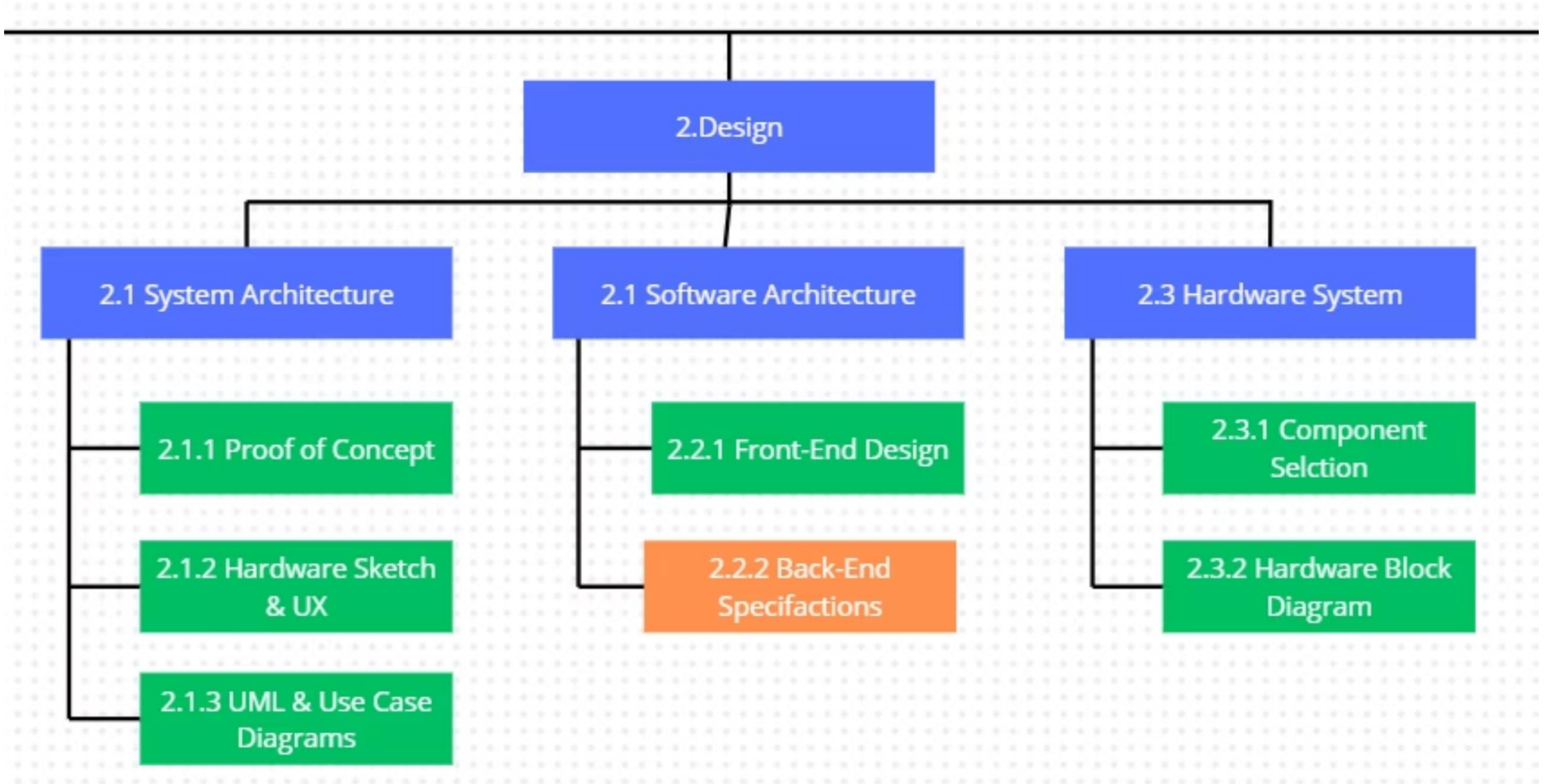
The Capstone-MediTrack GitHub repository features structured folders and recent commit activities.

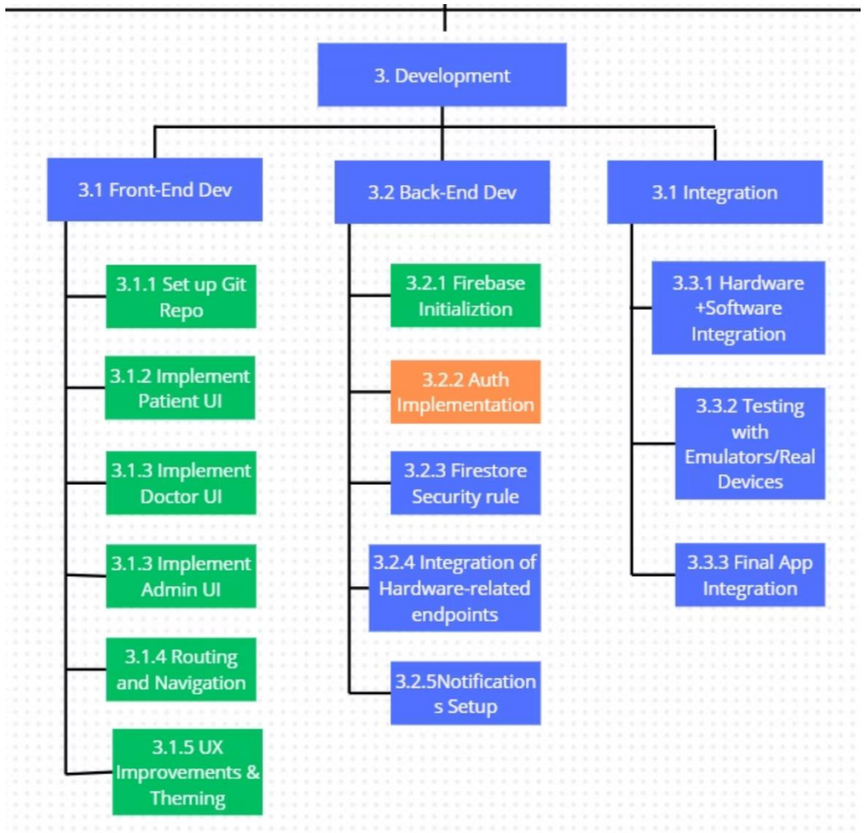


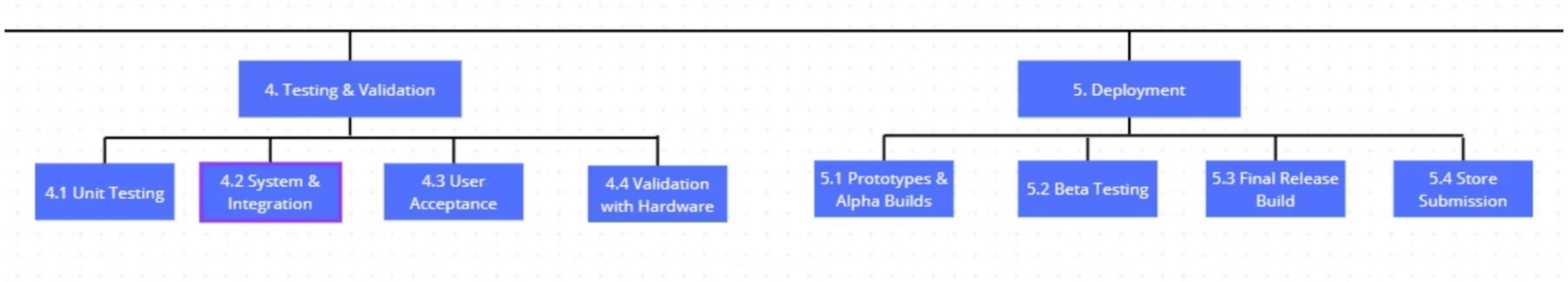
Work Breakdown Structure

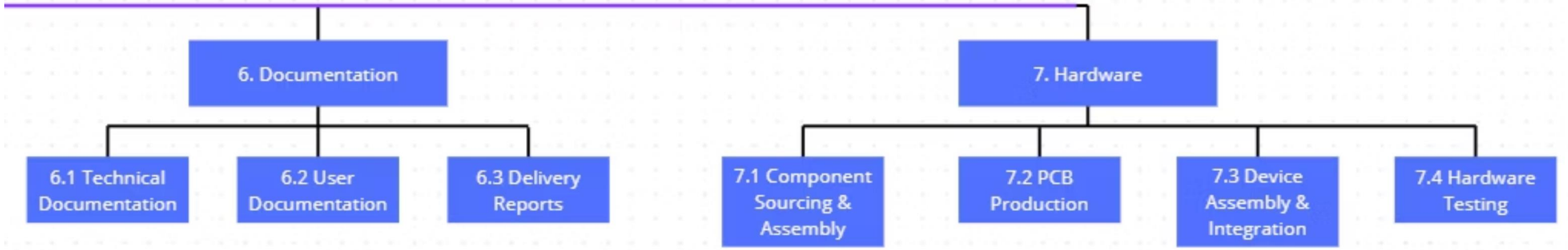




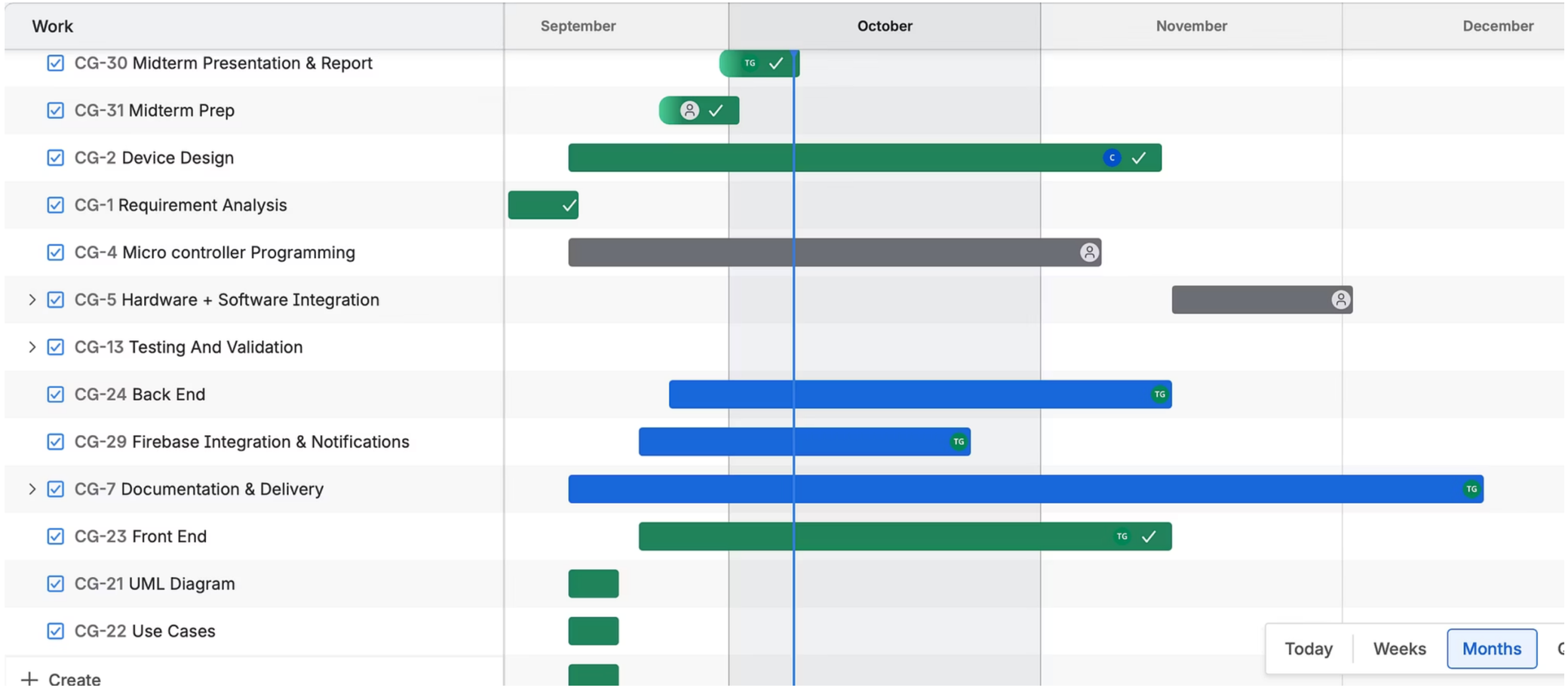

2.Design
2.1 System Architecture
2.1 Software Architecture
2.3 Hardware System
2.1.1 Proof of Concept
2.1.2 Hardware Sketch & UX
2.1.3 UML & Use Case Diagrams
2.2.1 Front-End Design
2.2.2 Back-End Specifactions
2.3.1 Component Selction
2.3.2 Hardware Block Diagram


3. Development
3.1 Front-End Dev
3.2 Back-End Dev
3.1 Integration
3.1.1 Set up Git Repo
3.1.2 Implement Patient UI
3.1.3 Implement Doctor UI
3.1.3 Implement Admin UI
3.1.4 Routing and Navigation
3.1.5 UX Improvements & Theming
3.2.1 Firebase Initializtion
3.2.2 Auth Implementation
3.2.3 Firestore Security rule
3.2.4 Integration of Hardware-related endpoints
3.2.5Notifications Setup
3.3.1 Hardware +Software Integration
3.3.2 Testing with Emulators/Real Devices
3.3.3 Final App Integration

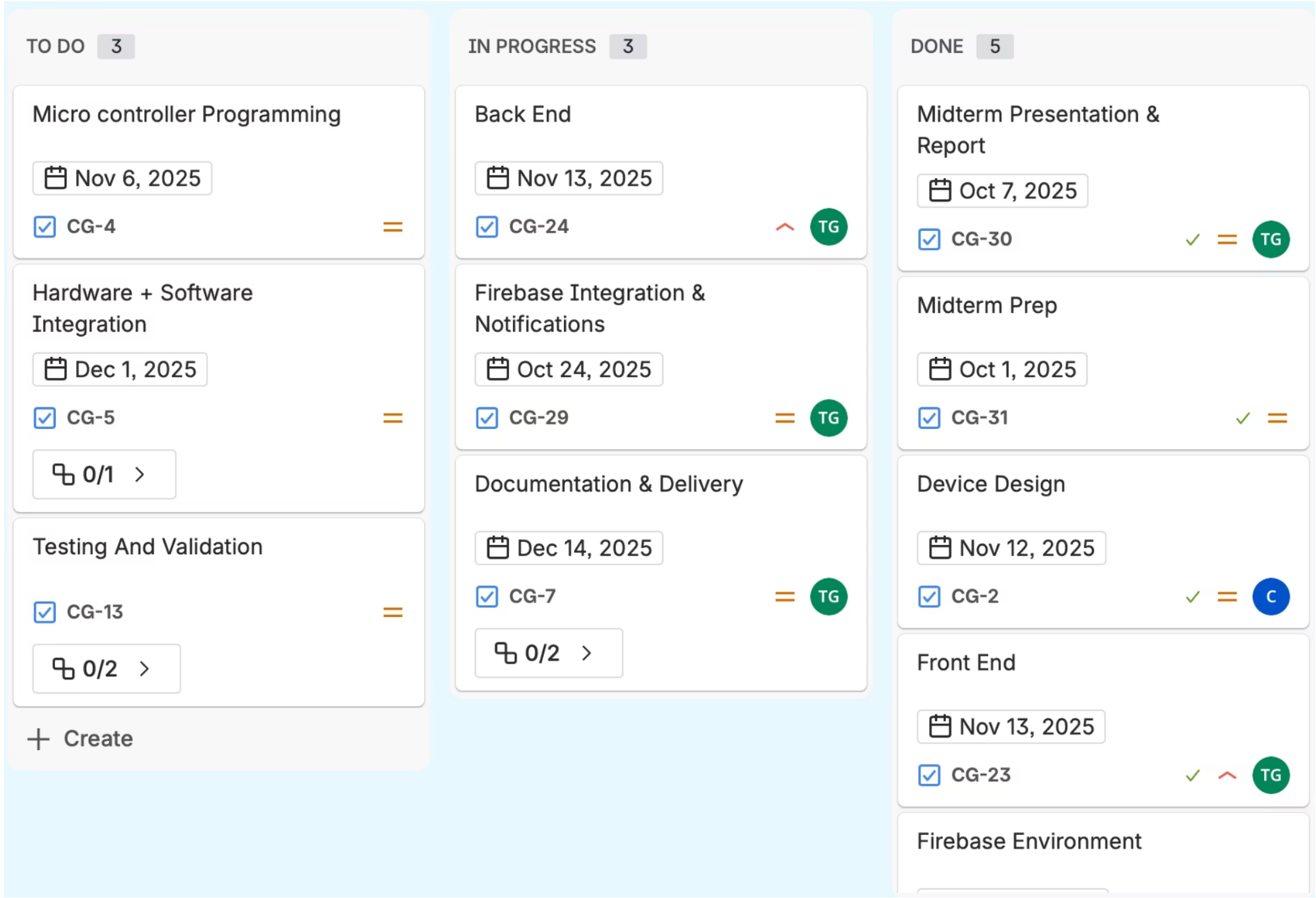

4. Testing & Validation
5. Deployment
4.1 Unit Testing
4.2 System & Integration
4.3 User Acceptance
4.4 Validation with Hardware
5.1 Prototypes & Alpha Builds
5.2 Beta Testing
5.3 Final Release Build
5.4 Store Submission


6. Documentation
7. Hardware
6.1 Technical Documentation
6.2 User Documentation
6.3 Delivery Reports
7.1 Component Sourcing & Assembly
7.2 PCB Production
7.3 Device Assembly & Integration
7.4 Hardware Testing

Project Timeline



Project Management



Current Implementation Status

25%

Overall Progress

Core functionality started with a prototype

6

Team Members

Split across hardware (box design), software (UI + backend), and integration

3

Modules Complete

UI design finished, Firebase DB structure set, authentication working, backend foundation in place

Next Development Phase

1

Hardware Prototype

Start building the medication box with dispensing mechanism and actuators

2

Backend Integration

Full Firebase connection with user authentication + data logging

3

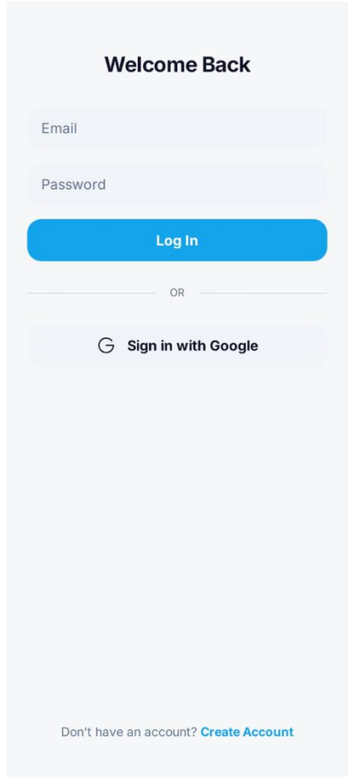
Alerts & Connectivity

Add notifications (LED, buzzer, mobile push alerts)

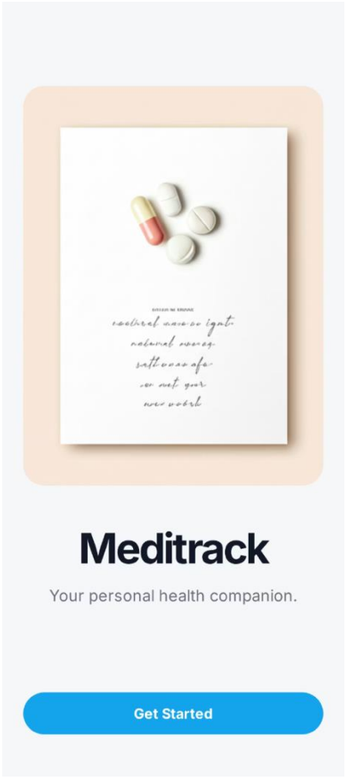
Mobile Application Interface

Onboarding Flow

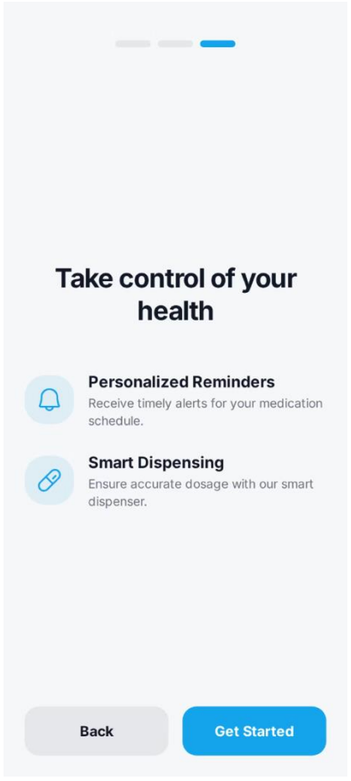
Login Screen



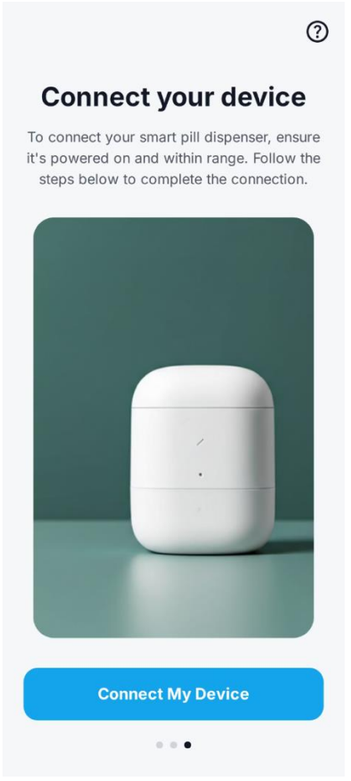
Welcome Screen



Features Overview



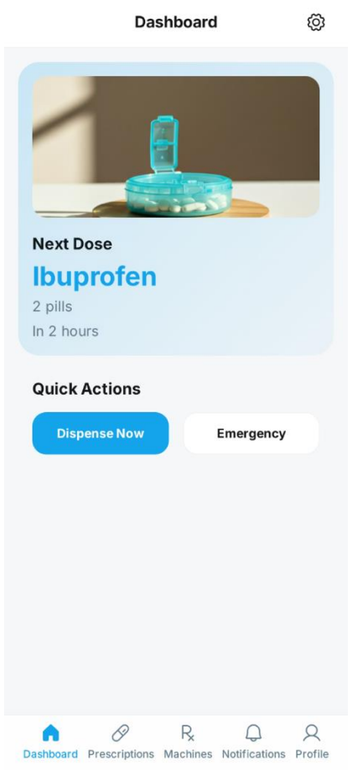
Connect Device



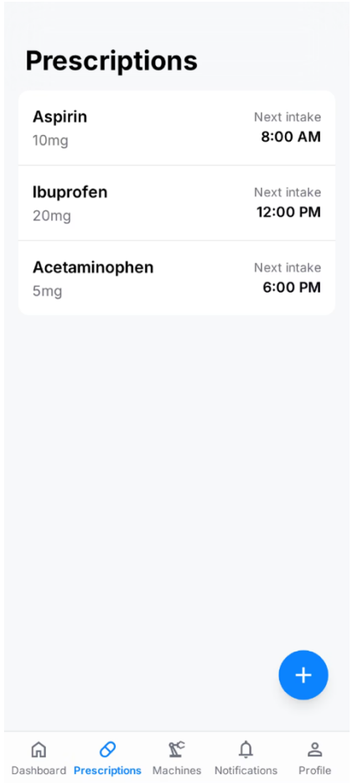
Mobile Application Interface

Main Application

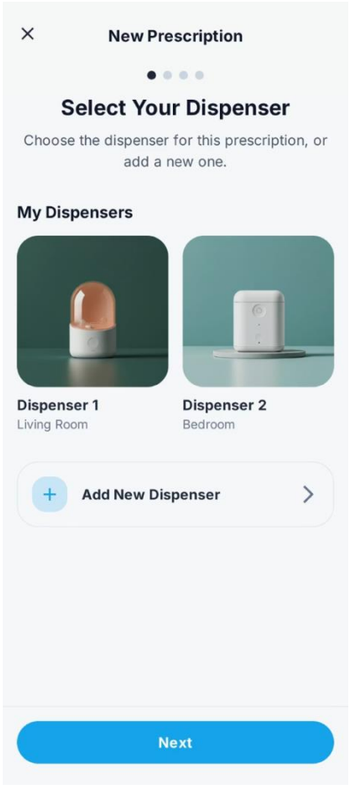
Dashboard



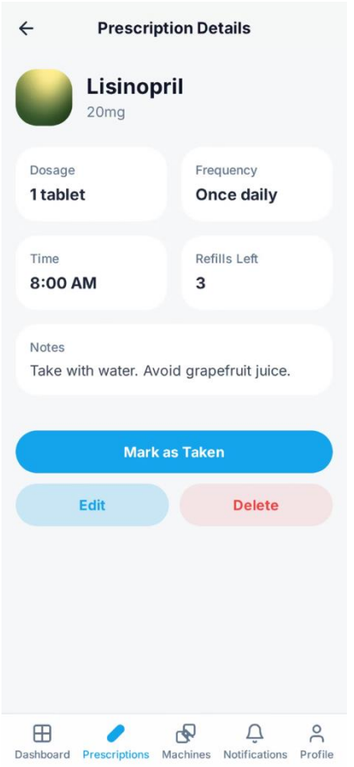
Prescriptions List



New Prescription Setup

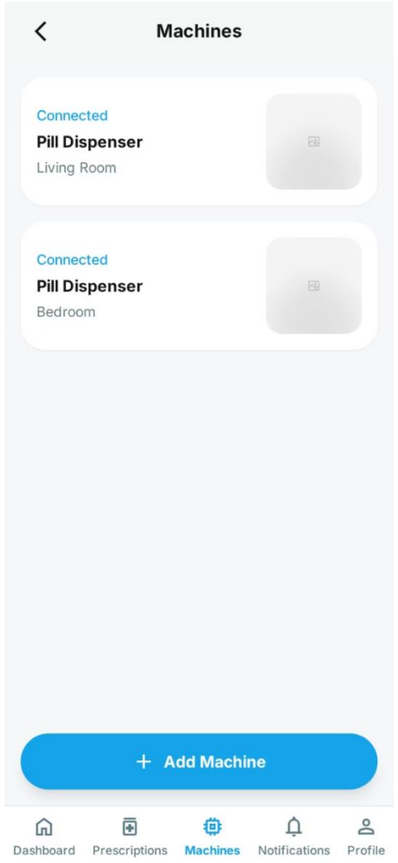


Prescription Details

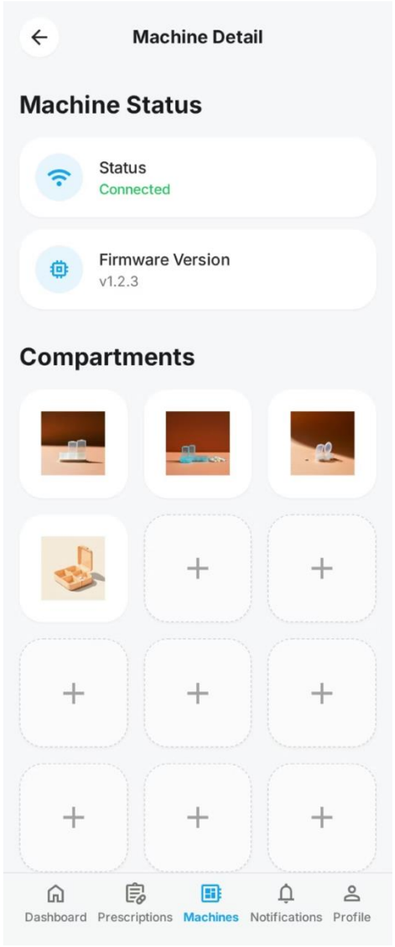


Mobile Application Interface

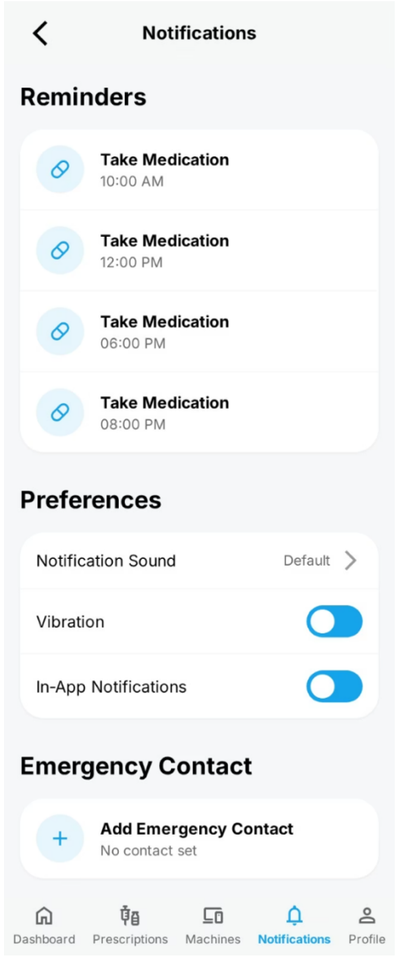
Machine List



Machine Details



Notifications



Settings