

Technical Project Report - Flutter Module

WellTrack



Subject: Universidade de Aveiro - Computação Móvel

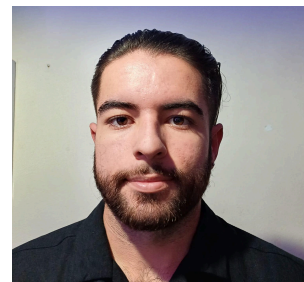
Date: 10 June 2025

Students:

114588: Henrique
Ferreira Teixeira

114614: Martim José
Souto Marques Dos
Santos

114597: Carlos Luis
Fernandes Verenzuela



Project
abstract:

A mobile-first wellness journal that unites mental and physical health tracking in one seamless experience. Users can log their emotional state throughout the day, track daily steps, view weekly activity charts, and record thoughts through text, audio, photos, or videos. The app leverages native mobile functionalities like GPS, pedometer, microphone, and camera to provide accurate and real-time data. Social features allow users to connect with friends, view each other's mood status (if shared), and discover nearby people. Designed with privacy, usability, and motivation in mind, the app acts as a personal companion to encourage a balanced and mindful lifestyle.

Report contents:

1 Application concept.....	3
Overview.....	3
Motivation.....	4
2 Implemented solution.....	4
Architecture overview (technical design).....	4
Architecture Patterns and Design Approach.....	4
Layer Architecture and Module Interaction.....	5
State Management Strategy.....	6
Data Persistence and Content Update Strategies.....	7
Connected Devices and Sensor Integration.....	8
External Service Integration.....	9
Dependency Injection.....	9
Advanced Design Strategies.....	9
Implementation.....	10
Home.....	10
Location.....	11
Statistics.....	11
Bluetooth.....	12
Profile.....	12
Project Limitations.....	13
3 Conclusions and supporting resources.....	13
Lessons learned.....	13
Work distribution within the team.....	14
Reference materials.....	15
Project resources.....	15

1 Application concept

Overview

WellTrack is a physical and mental health wellness tracking application designed to help users monitor and manage their emotional well-being while fostering community support.

- The app helps users track their mental state and mood over time
- It enables users to discover and connect with other WellTrack users in their vicinity
- It provides a platform for community support, especially for those experiencing difficult times
- It shows uses the correlation between their mental and physical well being

Typical Users:

- Mental Health Conscious Individuals
 - People who want to actively monitor their emotional well-being
 - Users who value self-awareness and personal growth
- People Seeking Community Support
 - Individuals who benefit from knowing they're not alone
 - Users who want to connect with others who might be going through similar experiences
- Friend Groups
 - Friend groups who want to know how their friends are doing

Key Benefits:

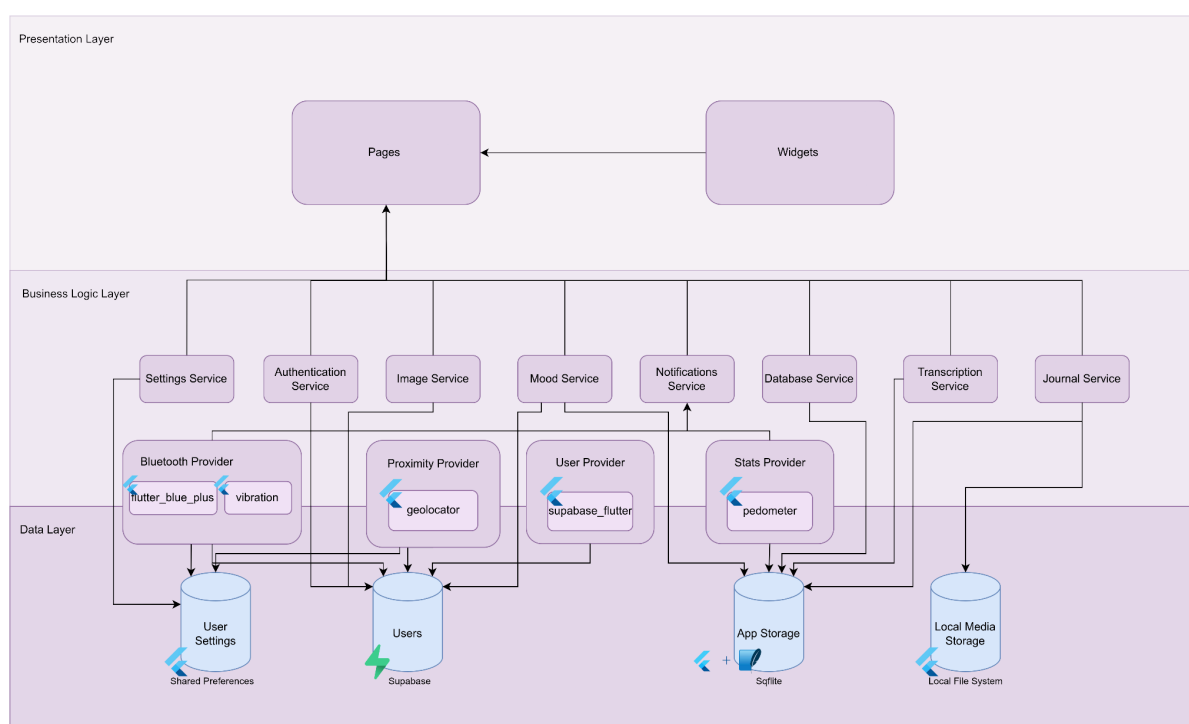
- Personal Wellness Tracking
 - Users can track their mood and mental state over time
 - Helps users identify patterns in their emotional well-being
- Community Connection
 - Real-time discovery of nearby WellTrack users
 - Creates a sense of community and belonging
- Support System
 - Vibration alerts notify users when someone nearby needs support
 - Users can see others' mental states through a color-coded system
- Privacy and Control
 - Mood sharing is optional and can be toggled
 - Vibration alerts can be enabled/disabled based on user preference

Motivation

Each day more and more people struggle with their mental health, and the worst part is that most of them don't realize it, and if they do, they struggle to identify that it is happening and how to fight against it. The motivation behind the app is to allow users to keep track of their mental well being, as well as being able to know how their friends are doing, allowing for faster help when it is needed. The app also incentivizes users to stay active and keep track of their physical well being, showing users the direct correlation between both types of well being.

2 Implemented solution

Architecture overview (technical design)



Architecture Patterns and Design Approach

WellTrack implements a hybrid architecture combining **Model-View-ViewModel (MVVM)** and **Repository** patterns to achieve a clean separation of concerns and maintainable code structure. The architecture follows Flutter best practices and Android Architecture Guidelines while incorporating advanced features for mental health tracking and social connectivity.

MVVM Implementation

The application employs the MVVM pattern to separate presentation logic from business logic:

- **Models:** Located in ``lib/models/``, these data classes define domain entities such as ``MentalState``, ``JournalEntry``, and ``BluetoothUser``. They represent the core data structures used throughout the application.
- **Views:** The UI layer consists of Pages (``lib/pages/``) and Components (``lib/components/``). These consume ViewModels and Providers through Flutter's Provider pattern for reactive UI updates.
- **ViewModels:** The ``MentalStateViewModel`` exemplifies the MVVM approach, managing mental state data and business logic while exposing observable state through `ChangeNotifier` for reactive UI updates.

Repository Pattern

Services in the ``lib/services/`` directory act as repositories, abstracting data source operations from business logic:

- ``DatabaseHelper``: Manages local SQLite database operations
- ``MentalStateService``: Handles mental state data synchronization between local and remote sources
- ``JournalService``: Manages journal entries with support for multiple media types
- ``AuthenticationService``: Abstracts authentication operations through Supabase

Layer Architecture and Module Interaction

Presentation Layer

The presentation layer is organized into three main categories:

1. **Pages:** Full-screen views that represent distinct application screens
2. **Components:** Reusable UI components like ``BottomNavBar``, ``Calendar``, and ``MoodSlider``
3. **Widgets:** Small, focused UI elements such as ``AudioPlayerWidget``

Navigation is centralized through two wrapper components:

- ``MainPageWrapper``: Manages pages in the bottom navigation (Home, Calendar, Stats, Bluetooth, Profile)
- ``NonMainPageWrapper``: Handles standalone pages not in the bottom navigation

Business Logic Layer

The business logic is distributed across several architectural components:

- **Providers (State Management):**
 - `UserProvider`: Manages user authentication state and profile information
 - `ProximityProvider`: Handles location-based features and nearby user detection
 - `BluetoothProvider`: Manages Bluetooth device discovery and mental state sharing
 - `StatsProvider`: Tracks activity statistics including steps, calories, and distance
- **Services**
 - Services in act as repositories, abstracting data source operations from business logic, as discussed above.

Data Layer

The data layer implements a dual-storage strategy:

- **Local Storage:**
 - SQLite database for offline-first functionality
 - SharedPreferences for simple key-value storage
 - File system for media files (photos, audio recordings)
- **Remote Storage:**
 - Supabase for cloud synchronization
 - Real-time database for user profiles and mental states
 - Storage buckets for user avatars

State Management Strategy

The application utilizes the **Provider** pattern as its primary state management solution:

```
MultiProvider(
  providers: [
    ChangeNotifierProvider(create: (_) => UserProvider()),
    ChangeNotifierProxyProvider<UserProvider, BluetoothProvider>(...),
    ChangeNotifierProvider(create: (_) => ProximityProvider()),
    ChangeNotifierProvider(create: (_) => StatsProvider()),
    ChangeNotifierProvider(create: (_) => MentalStateViewModel()),
  ],
  child: MyApp(),
)
```

This approach enables:

- Reactive UI updates through ChangeNotifier
- Dependency injection between providers
- Scoped access to state
- Lifecycle management

Data Persistence and Content Update Strategies

Local Database Schema

The application uses SQLite with the following schema:

```
-- Mental States Table
CREATE TABLE mental_states (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  state REAL NOT NULL,
  date TEXT NOT NULL,
  emotions TEXT,
  factors TEXT
);
```

```
-- Journal Entries Table
CREATE TABLE journal_entries (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  user_id INTEGER NOT NULL,
  date TEXT NOT NULL,
  type TEXT NOT NULL,
  text_content TEXT,
  photo_path TEXT,
  audio_path TEXT,
  caption TEXT,
  transcription TEXT,
  created_at TEXT NOT NULL,
  updated_at TEXT NOT NULL
);
```

Connected Devices and Sensor Integration

Bluetooth Low Energy (BLE)

The application leverages BLE for proximity-based features:

- **Device Discovery:** Scans for nearby WellTrack users
- **Mental State Sharing:** Broadcasts encrypted mental state data
- **Support Alerts:** Notifies by vibrating nearby users when someone needs support
- **Privacy Controls:** Users can toggle visibility and set sharing preferences

Implementation uses *flutter_blue_plus* with a custom protocol for secure data exchange.

Location Services

GPS integration enables several key features:

- **Proximity Detection:** Calculates distance between users
- **Location Sharing:** Optional real-time location sharing for support
- **Privacy:** Granular control over location visibility

The *geolocator* package provides cross-platform location access with background support.

Pedometer and Activity Tracking

Step counting is implemented using the *pedometer* package with custom algorithms:

- Real-time step detection
- Calorie calculation based on user profile
- Distance estimation
- Tracking of time (minutes of walk).
- Daily goal tracking with notifications

Audio and Speech Recognition

The application integrates multiple audio services:

- **Recording:** *record* package for audio journaling
- **Playback:** *audioplayers* for reviewing recordings
- **Transcription:** *speech_to_text* for converting audio to text

The *TranscriptionService* coordinates these features for seamless audio journaling.

External Service Integration

Supabase Backend

The application uses Supabase as part of its backend:

- **Authentication:** Email/password and Google Sign-In
- **Real-time Database:** User profiles and mental states
- **Storage:** Avatar images and media files
- **Custom Functions:** RPC functions for complex queries

Google Services

Integration with Google ecosystem:

- Google Sign-In: OAuth authentication
- Google Maps: Location selection and visualization

Dependency Injection

The application uses GetIt for simple, effective dependency injection:

```
final getIt = GetIt.instance;

void setupInjection() {
  getIt.registerLazySingleton<NavigationService>(() =>
  NavigationService());
  getIt.registerLazySingleton<SettingsService>(() => SettingsService());
  getIt.registerLazySingleton<DatabaseHelper>(() => DatabaseHelper());
}
```

This approach provides:

- Service location without code generation
- Lazy initialization
- Testability through interface injection
- Global access through helper functions

Advanced Design Strategies

Privacy-First Design

The application implements multiple privacy features:

- End-to-end encryption for Bluetooth communication
- Granular privacy controls for location and mental state sharing
- Anonymous mode for sensitive journal entries

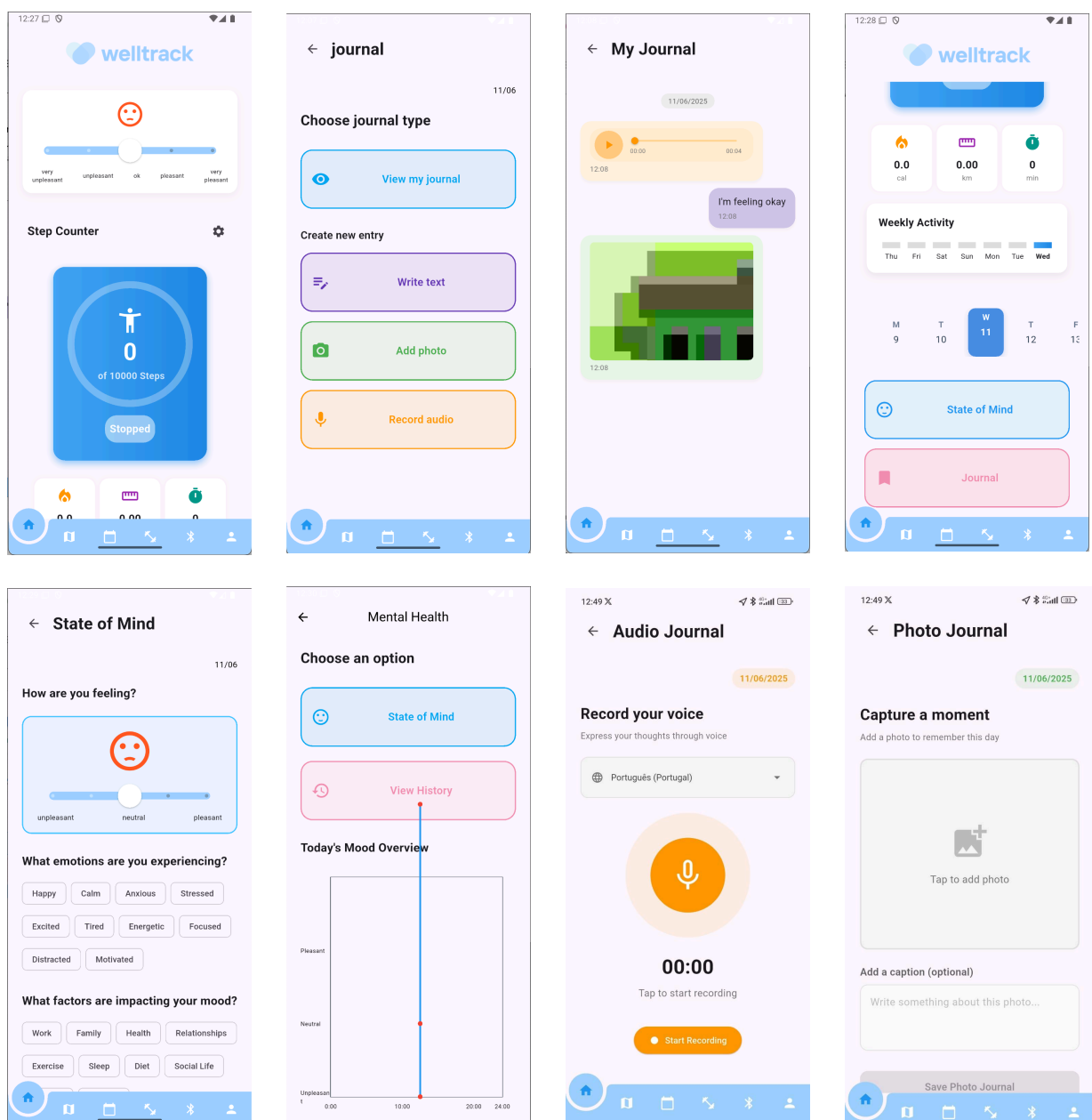
Offline Resilience

The offline-first architecture ensures functionality without connectivity:

- Local SQLite database for core features
- Queue-based synchronization for eventual consistency
- Cached media files with progressive loading
- Graceful degradation of online features

Implementation

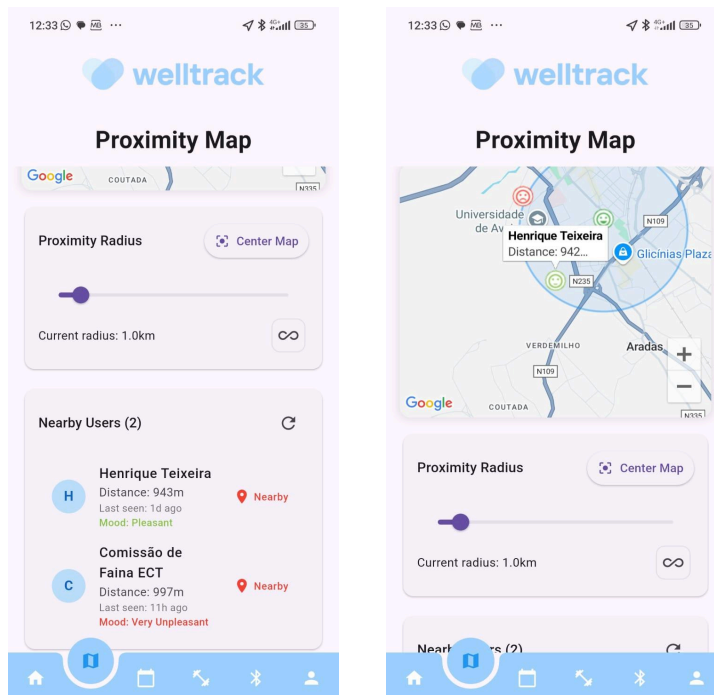
Home



In the home page the user can update his mental state more quickly and see the stats of the pedometer, the goal of steps can be set on the settings icon. Then we have 2 other pages that are accessible in the home page, the journal and the mental state. In the journal page the user can input through photos, text and audio his thoughts in the day. In the mental state page the user can update his mental state and attach emotions.

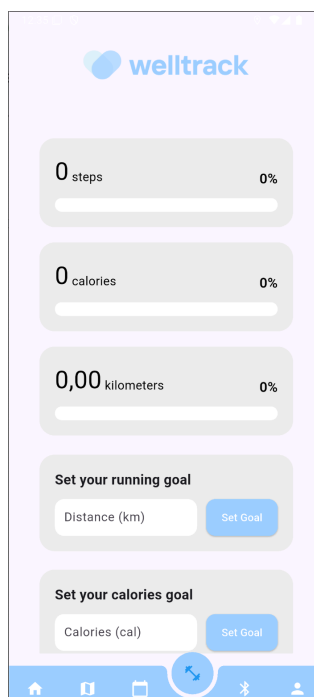
Location

In the location page the user can see his location and see the mood of nearby users, the user has the option to filter by a radius.



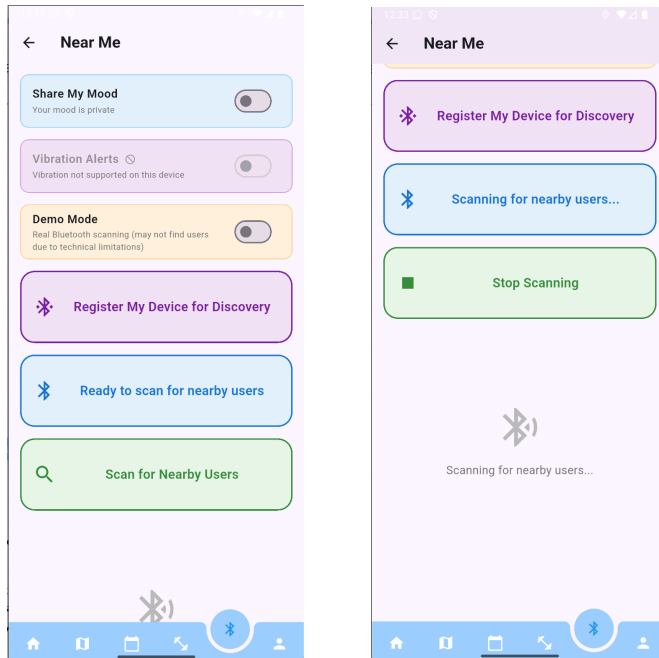
Statistics

In the stats page the user can see his physical stats and set diary goals.



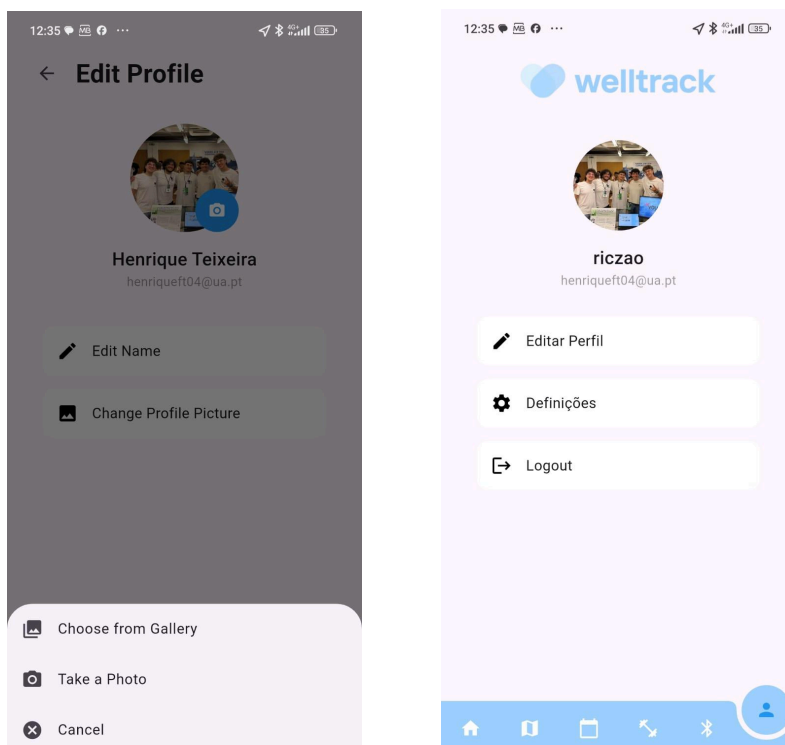
Bluetooth

In the bluetooth page the user can search nearby users and enable the vibration when a user with a bad mood is close to him.



Profile

In the profile page the user can change his profile, user settings in the app and logout.



Project Limitations

Despite the successful implementation of the core functionalities, there are several limitations that should be acknowledged in the current version of the project:

- **Code Optimization and Modularization:** Certain parts of the application, particularly the HomePage, could benefit from further code optimization and modularization. This would improve maintainability, readability, and scalability of the codebase.
- **Weekly Data Visualization Interactions:** The feature that displays weekly historical data through interactive charts on the Statistics page is currently incomplete. While the weekly bars are displayed, tapping on each bar to view detailed information for the selected week has not yet been implemented.
- **Database Improvement:** The database schema and structure can be further enhanced to support more efficient data retrieval and scalability. This includes normalization, better indexing, and the potential integration of more advanced features to support future functionalities.

3 Conclusions and supporting resources

Lessons learned

Throughout the development of this project, the team encountered several challenges that required careful analysis and creative problem-solving.

One of the most significant difficulties was understanding and integrating the various native sensors available on mobile devices, such as the GPS, pedometer, microphone, and camera. Although Flutter provides several plugins to access these features, ensuring their correct configuration, handling permissions dynamically, and integrating them smoothly into the user experience proved to be a complex task. For example, retrieving accurate and real-time step data required experimenting with different packages and approaches to balance performance and reliability.

Another major challenge was designing the user interface in a way that effectively combines mental and physical health tracking in a single, coherent experience. Defining how the interface should look and behave across multiple features—such as mood logging, journal entries, and activity visualization—demanded extensive iteration and user-focused design decisions. Achieving a balance between aesthetics, usability, and functionality was a continuous learning process.

On a more positive note, some aspects of mobile development using Flutter were

surprisingly accessible and powerful. The availability of well-documented packages and the declarative UI paradigm made it easier than expected to prototype complex interfaces. However, integrating platform-specific functionality still required a good understanding of native capabilities and some manual adjustments beyond what Flutter abstracts. We used Figma so all members of the group could be on the same page.

Overall, the experience deepened our understanding of cross-platform development, sensor integration, and user interface design, providing valuable insights for future mobile development projects.

Work distribution within the team

Taking into consideration the overall development of the project, the contribution of each team member was distributed as follows:

All members contributed equally to the project, participating in discussions, planning, implementation, and testing. However, each member focused on specific areas according to their interests and strengths:



- **Henrique Teixeira** was primarily responsible for the implementation of the authentication system (login), the interactive map feature, several code refactors to improve structure and reusability, and the integration of Bluetooth-related functionalities. **(33%)**
- **Martim Santos** focused mainly on designing and managing the database structure, implementing the "State of Mind" feature and the "Journal", and developing the navigation bar, along with various other components throughout the application. **(33%)**
- **Carlos Verenzuela** laid the foundational structure of the app, including the initial setup, main navigation flows, and early page layouts. He also implemented the pedometer functionality, the notification system, and contributed to several other features and design decisions during the development process. **(33%)**

This collaborative and complementary distribution of tasks allowed the team to efficiently cover all aspects of the project while ensuring high-quality results across all features.

Reference materials

<https://pub.dev/>

Project resources

Resource:	Available from:
Code repository:	henriqueft04/WellTrack: ICM 24/25 Flutter Project
Ready-to-deploy APK:	 WellTrack
Demo video:	 WellTrack