Well MEing

Practical Development Overview

Matteo Bettiati
Lorenzo Bianchi
Alessio Caggiano
Francesco Ostidich
Denis Sanduleanu

April 8, 2025

Version: 1.0

Contents

1 Introduction

1.1 Purpose of the document

This document outlines the practical development overview for Well MEing, a highly customizable wellness tracking application. The goal is to deliver a mobile-first, engaging, and AI-assisted experience that empowers users to track the aspects of well-being that matter most to them, from fitness and nutrition to sleep and stress management.

Built upon the insights gathered in our Product Research Report (PRR), this document presents a comprehensive view of the application's envisioned use cases, technical architecture, and implementation roadmap. Through thoughtful design and modern development practices, we aim to address user pain points identified during the research by bringing a compelling and effective solution to the market.

1.2 Definitions

- **Habit/group:** a specific behavior or activity that a user wants to track and improve, such as nutrition or sport.
- **Metric:** a quantifiable measure of a habit, such as steps taken, hours slept, or glasses of water consumed.
- **Submission/log/record:** the recording of an activity that is added to the history of that habit; for that specific timestamp, a value for each of its metric is stored.
- **Goal/target/objective:** a specific value or range that a user aims to achieve for a habit.
- **Assistant:** the AI-driven set of features that provides personalized insights and voice commands functions.
- **Progress/statistics/charts:** visual representations of user data over time, helping users understand their habits and track their progress toward goals.

2 User interaction

2.1 Features

The table below displays the prioritized features that we plan to develop.

Rank	Feature
1	Custom habit creation
2	Minimalistic UI
3	Quick habit logging
4	AI-generated progress reports
5	Voice-based interaction
6	Targets
7	Configurable notifications
8	Adaptable data visualization

2.2 Scenarios

This section presents practical use case scenarios that demonstrate how the application will serve its diverse user base in real-life situations. These scenarios directly resembles the feature list defined in the user research, focussing on the themes and needs that user mentionned the most. This scenarios are more focussed on make a detailed description of the application fundamental functionalities, in order to allow the reader to well understand their behaviour.

2.2.1 Habit creation

The user, from the main dashboard page, can click a "+" button to create a new habit. It can insert some text fields for name and description, and then select, one by one, a list of metrics. Each metric has also its specific name and description, and requires the user to choose an input format: this are well defined in a drop down menu, and depending on which one the user selects, a set of input-format-specific configuration fields appear. For example, if the input format chosen is a slider, the user will be asked to set its minimum and maximum range, and furthermore choose if the values are integers or floats. After selecting an arbitrary number of metrics, (e.g. a slider for kilometers run and a time duration selector for the running time) the user can save this new habit, which is then shown on the dashboard page.

2.2.2 Habit logging

From the dashboard main page, the user can select an habit from the ones he has created in the past. A modal appears from within which the user can select a value for each matric of the habit, using the input format interactive elements he has decided at the time of creation. For example, if the habit is "Running", he can select a number of kilometers run from the slider (e.g. 10), and select a time duration from the corresponding selector (e.g. 01:30:00). The user can also insert some text in an optional "Notes" field, and can also change the submission time from the default current one to a past one. After inserting all the metrics for that habit, the "Submit" button is enabled and the user can log the submission.

If for any reason the user thinks it has erroneously recorded a submission, that submission can still be shown within a list of past submission, and by clicking on one, he can delete it. The habit history can be seen from the caledar in the progress page, where, by tapping a day, all the submission of that day are shown. Note that it can always re-submit it by selecting a different timestamp from the habit loggin page.

2.2.3 Voice commands

From the main dashboard page, the user can tap a ”Voice commands” button. He can then start recording its speech. He can propose a list of actions to take, which must either be habit creations or habit logging ones. For example, he can say he wants to start tracking its nutrition, specifying the metrics he want to find in the habit (e.g. calorie intake and protein grams); then, within the same recording process, he can also ask to log the habits for the day, which are ”Running” and ”Water Drunk”, telling he ran 10 km in one and a half hour, and that he drank 8 glasses of water. He may also decide to start tracking the number of LeetCode problems he is doing, saying he desire just a simple slider for counting the number of exercises he did in a session.

After he finishes asking for the actions he wants to take, it stops the recording and the recognized text is sent to the AI agent for interpreting it. The AI organizes all this actions in a well pre-defined JSON format, which the mobile app receives, and automatically uses to send the requests to the backend, after previewing them and asking for acceptance and edits to the user.

2.2.4 Reports

Once a week, from the report page, the user can press a button to ask for a user-specific report to the LLM. The user is then presented with a multi box view for selecting the habits for which he is interested in receving a report for. After selecting an arbitrary number of habits he wants to include in the report, the last-week history of those habits is retrieved and sent to the LLM, which is under the hood prompted for producing a medium-sized, natural language text which recaps the submissions it receives, listing some insights it finds and presenting some suggestions the user can exploit to reach faster its goals.

Note that the report is a single one which comprehends information for all the selected habits all-together. This is useful since if the habits selected are ”Running” and ”Nutrition”, the LLM can cross-use their histories to better understand the user situation and better provide suggestions. This is also nice for finding correlations (e.g. the report could understand and tell a user that its running activities have gone better since it started to drink more water).

Moreover, from the report page the user can tell its name and a description of itself which is fed to the LLM for better create a user-specific report, declined on the user charateristics and objectives he told in said description text field.

From the same page, the user can also see all past reports to read again. He can also see a timer telling how much time is to be waited before requesting a new report.

2.2.5 Progress visualization

In the progress page the user can look at its submission history in two ways.

In the first place, by clicking on a specific day in the calendar all the submissions of that day are presented. Secondly, a set of charts are listed below the calendar, and they show progress over time with the possibily of choosing different granularities.

There is a chart for each metric, organized by habit, and they work as follows.

Firstly, not all metrics can be shown on a graph, based on their input format; for instance, a multi-box metric does not have a simple way to be shown on a chart, whereas an integer insertion made with a slider is pretty easy to be shown. Therefore, in the charts, only the presentable metrics are considered. The chart shown are only tower/bar charts, as they are sufficient for showing whichever value the app supports.

Then, the user can select a specific granularity he desires (e.g. day, month, year). If for example he selects a ”Month” granularity, the chart will have the 31 days on the x-axis, whereas if he selects ”Day”, the x-axis will show the 24 hours.

Let’s consider a metric showable in a chart, for example, the calorie intake, which is an integer. If the user wants to see the calorie intake throughout the month, then the chart will show the total calories for each day. This is done by grouping all submissions of each day by summing the integer values. Therefore, e.g. the 26th of march will show 2200 kcal, which is the sum of two submissions, one of 900 and one of 1300 kcal. Instead, if the user selects the ”Year” granularity, then all the submissions of a month will be summed for showing that month’s total.

A final note: each input format has its ”grouping function”. Integers, floats and time durations (sliders and time selectors) have the sum, star ratings have the average.

3 Architectural design

3.1 Deployment overview

The application will run with a client-server schema, with the client on a mobile app interacting with the Firebase server, handling functionalities like authentication, data storage, and initial API routing. Firebase hosts also cloud functions for some endpoints used for security checks and abuse control logic. A dedicated AI server, also triggered via Firebase, processes requests and mediates communication with the LLM, enforcing per-user token limits.

3.2 Components overview

The mobile application can be broken down into its main components, that are helpful to identify in order to ensure a clean, modular, and well-organized codebase.

Each component handles its specific interactions with Firebase, offering a clear interface for the View layer to access the required functionality.

Below are the main components, already defined in terms of their concrete usage within the codebase.

- **Habit Manager:** the habit manager collects all the functions that are used for creating, editing and deleting the habits that the user is tracking, each with its internal metrics. Furthermore, it contains the call needed for logging an habit submission.
- **History Manager:** old submission must be able to be retrieved in order to be shown, for instance, from the calendar view or in the charts. This component allow for the retrieval of past data, e.g. per-day.
- **Voice Commands**: here are organized all the functionalities that allow the user to send its speech to the AI assistant to be processed, for then receving the actions the LLM has interpreted (habit creations and submissions), that by the way the user has to confirm.
- **Report Service:** at a set cadence, the submissions history of the last period must be received and sent to the AI assistant, for it to generate a user-specific report. This component makes the call to the LLM for receving the report text, allowing the View to also retrieve and show past reports. It also manages the user personal information DB insertions.

3.3 Patterns

The architectural design of the application is structured according to the principles of various well-known patters, ensuring modularity, scalability, and maintainability. Additionally, the system is organized into a multi-tier architecture, with a dedicated tier for AI services, to support the features requiring and intelligent agent.

3.3.1 Distributed MVC

The Model-View-Controller (MVC) design pattern is used to separate the presentation layer from the business logic and data definitions. In particular, the model contains all the elements representing the data used by the application, the view is the user interface component, and the controller comprehends all the other services and managers.

In this project, the view resides on the client device, the model reflects the database structure (which is a separate application accessed through defined interfaces), and the controller logic is handled in a backend server. Additionally, the backend relies on another server for AI operations. As a result, the MVC pattern can be considered “distributed.”

3.3.2 Four-tier architecture

The application adopts a four-tier architecture, where each layer has a distinct role while communicating through well-defined interfaces.

The Presentation Tier consists of a native Swift application that serves as the primary interface for users. The Application Tier represents the backend server, which provides the various services through REST endpoints. The Data Tier corresponds to the database manager, which lets the backend server access to the main storage. The AI Service Tier runs independently as a dedicated AI backend, handling mainly natural language processing, exposing its capabilities through APIs.

This four-tier approach promotes maintainability allowing each component to evolve independently while ensuring smooth communication between layers.

4 Implementation plan

4.1 Development roadmap

The first version of the application we plan to develop is an MVP able to provide the following functionalities.

1. Habit creation
2. Habit logging
3. Submissions history visualization
4. Charts for metrics visualization at different time granularities
5. Voice commands of a single interaction (STT)
6. Report generation

Eventually, when the first MVP is completed, we plan of adding the following functionalities.

1. Goal setting
2. Appealing UI elements support, like colors or symbols
3. Notifications
4. Voice commands of continuous interactions (STT and TTS)
5. Habit set-frequencies

4.2 Technologies

For development, we will use Swift for the frontend since it ensures a smooth and responsive user experience. The backend will exploit Firebase functionalities, are they are easy to set up, deploy and use. Then, a little Python server will be run, directly in Firebase, in order to allow an ad hoc AI interaction.

4.3 User interface design

This section showcases mock-up designs that visualize the user experience and interface of the application. It also describes the organization of the UI elements across the main app sections, justifying design decisions for content placement and user flow.

The UI is crafted to be minimalist yet engaging, following the principle of simplicity prioritized by users during research. Each mock-up reflects user-centric design, focusing on accessibility, speed of interaction, and aesthetic appeal, offering a preview of the intuitive and motivating environment we aim to create.

4.3.1 UI elements organization

The application will consist of three main pages, structured for an intuitive user experience:

1. **Dashboard:** this is the main page, providing an overview of tracked habits, allowing for habit creation and logging, even with voice commands.
2. **Assistant:** the AI-based assistant page provides a place to collect the reports that the user receives weekly; the user can also set some information about itself that the report can leverage for specific insights.

3. **Progress:** the progress page allows the user to see the history of its submissions for any specific day, by tapping on a calendar; furthermore, a set of charts summarizes the past data of the user with different granularity options.

4.3.2 Mock-ups

5 Data models

5.1 Voice assistant interface

```json
{
  "creation": [
    {
      "name": "Running",
      "description": "Go for a run in your free time",
      "goal": "I want to run 3 times a week in order to train for PolimiRun",
      "metrics": [
        {
          "name": "Distance",
          "description": "Kilometers run",
          "input": "slider",
          "config": {
            "type": "int",
            "min": 0,
            "max": 100
          }
        },
        {
          "name": "Duration",
          "description": "Minutes of running",
          "input": "time",
          "config": {
          }
        }
      ]
    },
    {
      "name": "Food",
      "description": "Log the food you eat",
      "goal": "I want to eat max 2000 kcal per day",
      "metrics": [
        {
          "name": "Calorie intake",
          "description": "How many calories did this food have?",
          "input": "Slider",
          "config": {
            "type": "int",
            "min": 0,
            "max": 5000
          }
        },
        {
          "name": "Satifisaction",
          "description": "Did you enjoy this meal?",
          "input": "rating",
          "config": {}
        },
        {
          "name": "Meal",
          "description": "What did you eat?",
          "input": "text",
          "config": {}
        }
      ]
    }
  ],

  "logging": [
    {
      "timestamp": "2025-03-27T14:30:00",
      "name": "Running",
      "notes": "I felt really good afterwards",
      "metrics": {
        "Distance": 12,
        "Duration": "01:30:00"
      }
    },
    {
      "timestamp": "2025-03-27T14:30:00",
      "name": "Coding",
      "metrics": {
        "LeetCode done": 12
      }
    },
    {
      "timestamp": "2025-03-24T17:30:00",
      "name": "Meditation",
      "metrics": {
        "Duration": "00:30:00",
        "Satisfaction": "3",
        "Feelings": "Happy"
      }
    }
  ]
}
```

5.2 Database structure

```json
{
  "users": {
    "user-id-123456": {
      "name": "Kello",
      "mail": "kello@mail.com",
      "description": "I like going to the gym, and I'd like to hit 100 kg on bench press one day",
      "habits": [
        {
          "name": "Running",
          "description": "Go for a run in your free time",
          "goal": "I want to run 3 times a week in order to train for PolimiRun",
          "metrics": [
            {
              "name": "Distance",
              "description": "Kilometers run",
              "input": "slider",
              "config": {
                "type": "int",
                "min": 0,
                "max": 100
              }
            },
            {
              "name": "Duration",
              "description": "Minutes of running",
              "input": "time",
              "config": {}
            }
          ],
          "history": [
            {
              "timestamp": "2025-03-27T14:30:00",
              "notes": "Today the run was on a 20% street",
              "metrics": {
                "Distance": 12,
                "Duration": "01:30:00"
              }
            },
            {
              "timestamp": "2025-03-24T16:30:00",
              "metrics": {
                "Distance": 10,
                "Duration": "01:10:00"
              }
            }
          ]
        },
        {
          "name": "Food",
          "description": "Log the food you eat",
          "goal": "I want to eat max 2000 kcal per day",
          "metrics": [
            {
              "name": "Calorie intake",
              "description": "How many calories did this food have?",
              "input": "slider",
              "config": {
                "type": "int",
                "min": 0,
                "max": 5000
              }
            },
            {
              "name": "Satifisaction",
              "description": "Did you enjoy this meal?",
              "input": "rating",
              "config": {}
            }
          ],
          "history": [
            {
              "timestamp": "2025-03-26T15:30:00",
              "notes": "Today I ate a lot",
              "metrics": {
                "Calorie intake": 2200
              }
            },
            {
              "timestamp": "2025-03-21T13:30:00",
              "notes": "The pasta was good",
              "metrics": {
                "Calorie intake": 2100
              }
            }
          ]
        }
      ]
    }
  }
}
```

5.3 Input types

```
[
  {
    "input": "slider",
    "config": {
      "type": "int/float",
      "min": 0,
      "max": 100
    }
  },
  {
    "input": "text",
    "config": {}
  },
  {
    "input": "form",
    "config": {
      "box-list": ["first-value-name", "second-value-name", "third-value-name"]
    }
  },
  {
    "input": "time",
    "config": {}
  },
  {
    "input": "rating",
    "config": {}
  }
]
```