UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Alter Ego App: Philosophical frameworks for mental health therapy

Author:
Yevhenii ORENCHUK

Supervisor:
Serhii MISKIV

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Yevhenii ORENCHUK, declare that this thesis titled, “Alter Ego App: Philosophical frameworks for mental health therapy ” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Alter Ego App: Philosophical frameworks for mental health therapy

by Yevhenii Orenchuk

Abstract

The purpose of this bachelor's thesis is to collect all the knowledge I possess to make a tool that people will want to use because it makes their lives happier.

Code implementation is in repository: github.com/orenchuk/alter-ego

Acknowledgements

I want to thank all the UCU community for such great years and people. I want to thank myself that didn’t give up. And thank all the Alter Ego team. . . .

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1
- 1.1 Motivation 1
- 1.2 Goals 1
- 1.3 Solution 1

2 Background 2
- 2.1 Instruments 2

3 Database 3
- 3.1 Why Firebase? 3
- 3.2 Realtime Database vs Cloud Firestore 3
- 3.3 Models Structure 3
  - 3.3.1 Documents and Collections 3
  - 3.3.2 Storing User's inputs and progress 6

4 App Structure 7
- 4.1 General Structure 7
- 4.2 User Flows 7
- 4.3 App Screens 9
  - 4.3.1 Onboarding 9
  - 4.3.2 Authentication 9
  - 4.3.3 Pay Wall 9
  - 4.3.4 Mood Check 9
  - 4.3.5 Recommended 9
  - 4.3.6 Journeys 9
  - 4.3.7 Journey 9
  - 4.3.8 Practice 10
  - 4.3.9 Rate Practice 10
  - 4.3.10 Entries 10
  - 4.3.11 Profile 10
- 4.4 App Architecture 10

5 Future Improvements 11

6 Conclusions 12

Bibliography 13

List of Figures

3.1 Firestore Story Exercise's Document 4
3.2 Firestore Input Exercise's Document 4
3.3 Firestore Practice's Document 4
3.4 Firestore Journey's Document 5
3.5 Firestore Collection Example of Journeys 5
3.6 Firestore Collection Example of Profiles 6

4.1 Project Infrastructure with Firebase services 7
4.2 User Flows Structure 8
4.3 MVVM 10

List of Tables

List of Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
CRUD	**C**reate **R**ead **U**pdate **D**elete
UI	**U**ser **I**nterface
UX	**U**ser E**x**perience
SDK	**S**oftware **D**evelopment **K**it
SQL	**S**tructured **Q**uery **L**anguage
NoSQL	**No**n-**S**tructured **Q**uery **L**anguage
MVVM	**M**odel-**V**iew-**V**iew**M**odel

Dedicated to my mom and grandma…

Chapter 1

Introduction

1.1 Motivation

Nowadays, with that isolation, uncertainty, and lifestyle changes, people started feeling more vulnerable. As a result, they started looking for ways of dealing with that. And we are those people as well. We started looking for answers in Stoicism. We were surprised how just a simple but right question can have an impact on us. We've found a way that helps us, and now we want to share it with others.

1.2 Goals

Make at least a bit happier as many people as possible.

1.3 Solution

An application with metal frameworks from applied philosophy and psychiatry with engaging practices and self-reflection tools.

Chapter 2

Background

2.1 Instruments

For the purposes of iOS development, I will use the latest Apple frameworks, it's a SwiftUI, precisely the SwiftUI 2, that is compatible with iOS 14 and above, Combine and Firebase as a backend.

SwiftUI is a declarative way of implementing interfaces that uses a reactive binding principle to update its content.

Combine is a reactive framework for processing values over time.

Firebase is an SDK from Google that makes development faster.

Chapter 3

Database

3.1 Why Firebase?

Firebase is an SDK made by Google for mobile application development. Shortly it is a bunch of pre-made cloud services for development that usually developers have to build themselves. Such as backend, database, storage, authentication, push notifications, analytics, configurations, and so on.

So, the answer is straightforward. It is much faster (in terms of time for development), well scalable, and allows you to focus on the app experience itself. Currently, the project does not have any specific needs that cannot be implemented with Firebase. But it doesn't mean it will not have them in the future, so we should keep in mind that fact and don't rely on service-specific implementations and make a good level of abstractions to hold it loosely coupled.

For the project's purpose, I'll use Authentication for signing in users, Cloud Firestore as a database, Cloud Storage for storing bigger files, like images, videos, and audio files, Crashlytics for crash analytics, Cloud Messaging for remote push notifications.

3.2 Realtime Database vs Cloud Firestore

The answer to this question you can find in the official documentation.

Realtime Database is a SQL (relational) database and Cloud Firestore is a NoSQL (non-relational). More about the difference you can check on this site.

But that's the reasons why I chose Cloud Firestore over Realtime Database:

- It doesn't load all the data from nested collections.
- The structure is more flexible because of its schemaless nature.
- More effortless scalability as it scales horizontally, which means more servers instead of larger ones.
- Pricing is less because of billing for reads and writes, not the amount of data transferred.

3.3 Models Structure

3.3.1 Documents and Collections

"Cloud Firestore is a document-model database, which means that all of your data is stored in objects called documents that consist of key-value pairs – and these values can contain any number of things, from strings to floats to binary data to JSON-y looking objects the team likes to call maps. These documents, in turn, are grouped into collections." - *The Firebase Blog: Cloud Firestore vs the Realtime Database: Which one do I use?*

Below I'll introduce some examples of models that are stored in the database.

Exercise can be of two types: story and input. **Story** is an introduction or an explanation of a specific problem, which you may experience sometimes. **Input** is a particular question that a user asks himself and an input field to write down an answer, an actual self-reflection.

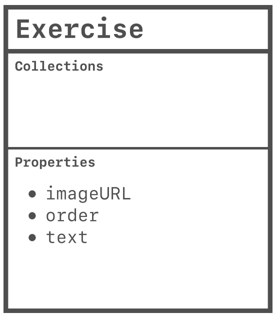

FIGURE 3.1: Firestore Story Exercise's Document

Story exercise contains an image's destination URL, order field stands for an ordinal number, and text. The order field is essential for sorting exercises, as they work sequentially like Instagram stories, and the content has to follow a strict order.

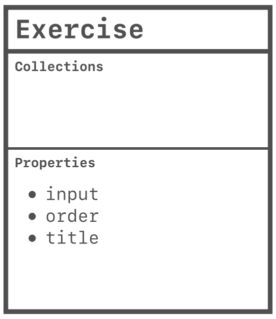

FIGURE 3.2: Firestore Input Exercise's Document

Input exercise contains same order field, title, which usually is a question, and input field. Important to understand that the input field is not the exact place where the actual user's reflection will be stored. It works somewhat like a placeholder's text that may help to begin answering.

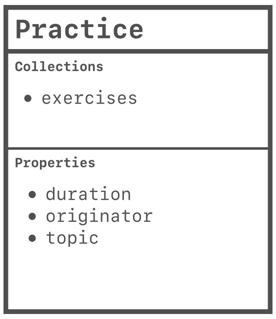

FIGURE 3.3: Firestore Practice's Document

Practice is a collection of exercises with the same topic. It contains a subcollection of exercises, approximated duration of the sequence of exercises, and topic.

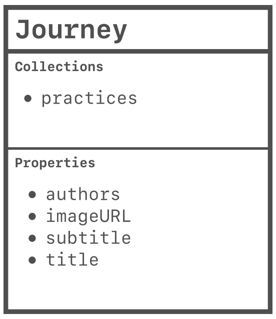

FIGURE 3.4: Firestore Journey's Document

Journey is a collection of practices that refer to the correlated subjects, for example: "Dealing with anxiety" or "Dichotomy of Control". In the figure above, you can see the model of Firestore Journey's Document entity. It contains a subcollection of practices, authors' property, which stores the list of philosophers, scientists (authors of works that were reframed into mental exercises), URL to the image's destination in Cloud Storage, title, and subtitle.

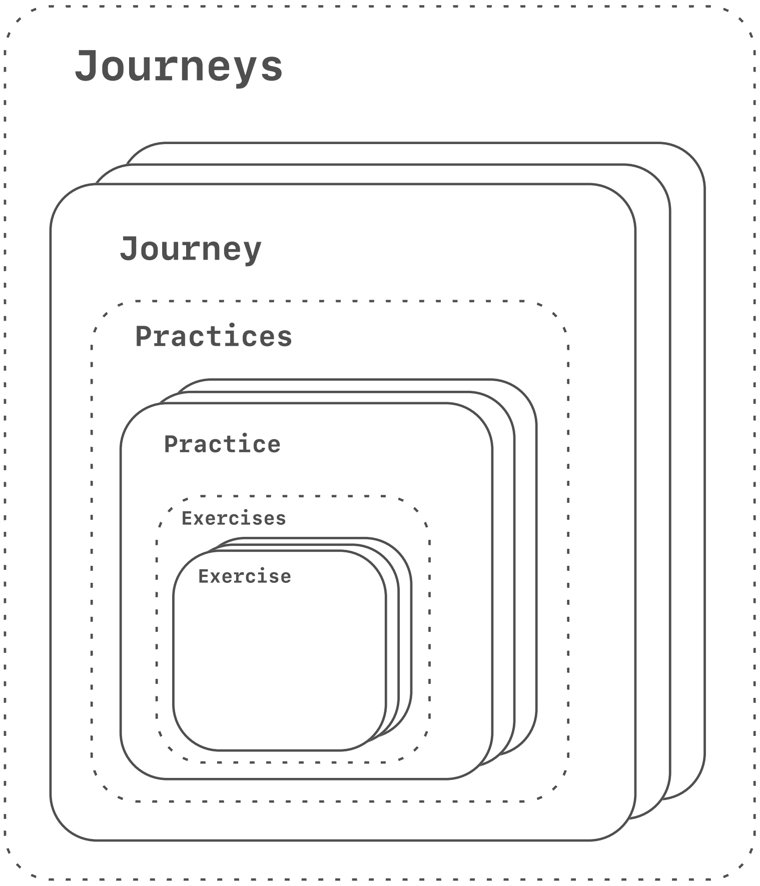

FIGURE 3.5: Firestore Collection Example of Journeys

This is an example of Journeys collection, which illustrates how documents and collections work together.

The path to an exercise would look the following way:
`"journeys/<journey-id>/practices/<practice-id>/exercises/<exercise-id>"`

3.3.2 Storing User's inputs and progress

We want to store user's inputs, practice ratings, and mood points after specific finished practice. Also, we need to know what practices the user completed. For this purpose, we need a separate table in the database, and the system is aware of the user's current progress from the derived data.

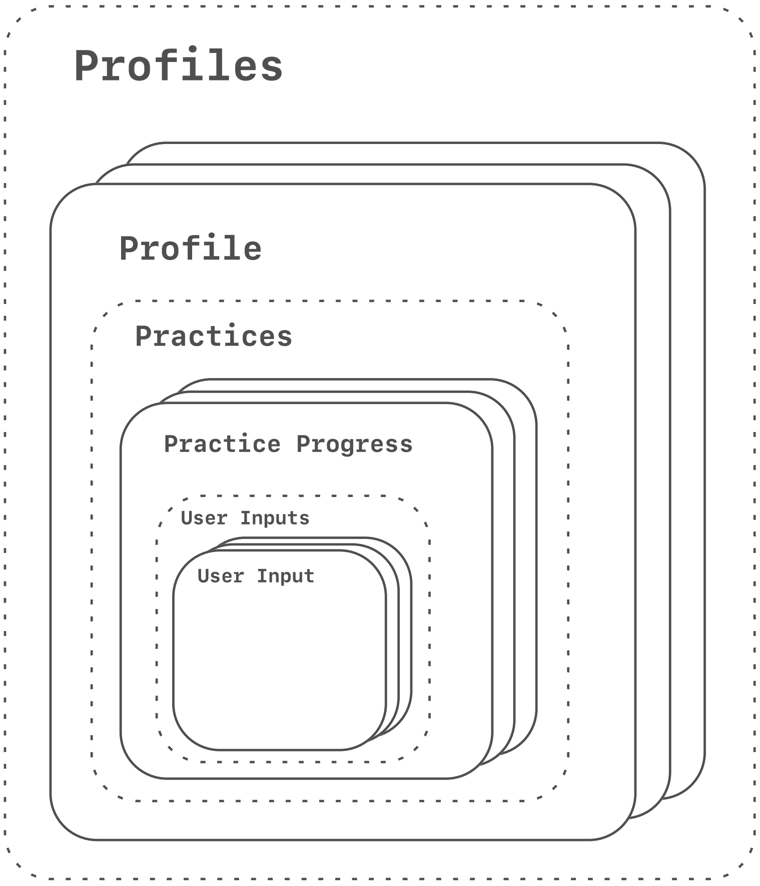

FIGURE 3.6: Firestore Collection Example of Profiles

The structure is designed this way to match with the practices table. Here go two paths: first to the actual exercise and the second to the exercise input.

```
"journeys/<journey-id>/practices/<practice-id>/exercises/<exercise-id>"
```

```
"profiles/<profile-id>/practices/<practice-id>/inputs/<exercise-id>"
```

To know the progress, the system checks what practices exist in the user's practices table.

Chapter 4

App Structure

4.1 General Structure

As the project is primarily content-driven, the Firebase is a "brain" of it, Mobile App, referred to as Client, is an interface that the end-user will interact with. Furthermore, we need a way to moderate the content. Basically, CRUD operations, that mean: creating new content, having access to all of it, updating it, and deleting. Under the content, I mean new practices, journeys, images, and overall structure. Admin Tool deals with all of that stuff.

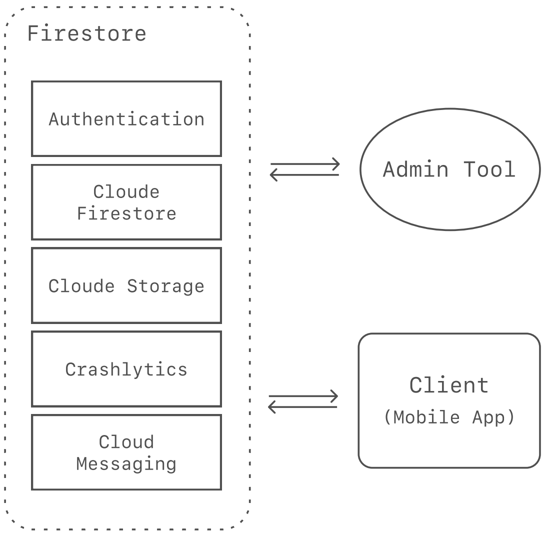

FIGURE 4.1: Project Infrastructure with Firebase services

The illustration above is general and straightforward, but it shows the way of communication between main instances. Admin Tool may have different interfaces, like web service, macOS, iOS, or iPadOS application. Using Apple's platforms for this purpose has a substantial advantage, as they can be implemented as a submodel for the Client app and share the same entities. However, it requires having a device running that operating system. At the same time, web service can be opened on any device, which makes it easier to delegate this duty to content managers that may not have Apple's device.

4.2 User Flows

User Flows are all different pathways a user can take when interacting with a product. The diagram below illustrates all of them for the Client app, where most of blocks units represent a single screen in the application. It is important to remember that some blocks can be omitted as they depend on the user's current state. It will be described more deeply in the following sections.

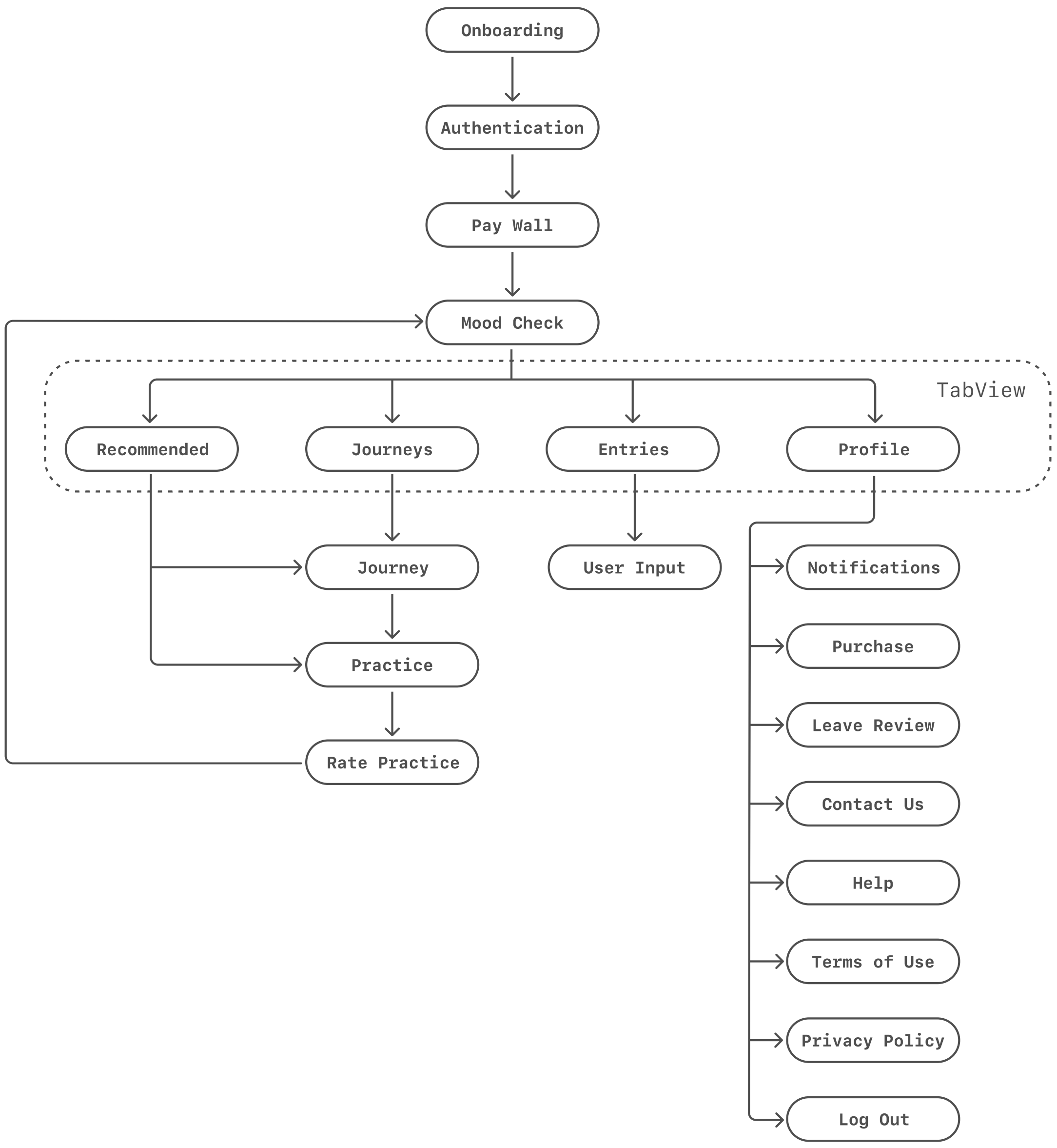

FIGURE 4.2: User Flows Structure

Making such a diagram is a part of the UX process. Many developers would like to skip this step and start developing right away. However, it can help prevent different issues like recurrent flow cycles by creating a new instance for the screen that has already been created and persists in the memory, which will cause memory leaks and may mislead users when they want to return back by screens stack.

TabView is a container from iOS SDK that holds all the screens in memory and allows easy switching.

4.3 App Screens

4.3.1 Onboarding

Onboarding, in general, serves as a way to demonstrate the application's main functionality and features. But you can also include some kind of questioner if the app can customize its content for user's preferences. As the app has a screen with recommended practices, those questions adjust the recommendation algorithm. Those screens also are displayed only once for the first time user opens the app. Then it stores the user's responses and defines the onboarding process as completed, so it'll not be shown anymore.

4.3.2 Authentication

Authentication is required for allowing access to purchased content, storing user-specific data such as inputs, progress, and preferences. Apple recommends delaying sign-in as long as possible. It's a good practice to show your app before the users give their data. For this reason, Firebase Authentication has an Anonymous sign-in feature that generates a unique id for a user and allows them to access the data in the database. So you can delay the actual sign-in till when it's indispensable. Firebase also has the functionality to check the user's state. The system will rely on that flag to know if it should display the authentication screen or let them in.

4.3.3 Pay Wall

Pay Wall is a screen, which blocks users from paid content if they didn't purchase. And presents a way to pay for it. It displays several plan options to choose from and uses Apple's In App Purchase mechanism to process the payment.

4.3.4 Mood Check

Mood Check is a way to know what the user feels at the moment. We want to check if he or she makes some progress, so we're designing the screen to collect those data on regular practice.

4.3.5 Recommended

For the beginning Recommended screen remains the same for all the users. Later we'll process users' responses from the onboarding stage to feed them into a regression model to display more relevant journeys and practices to their preferences. For now, it's just a separate table into the database that stores the references to the specified practices.

4.3.6 Journeys

Journeys is an entry to the main flow of the app. This is a list of all available journeys.

4.3.7 Journey

Journey represents a collection of practices with correlated subjects as well as an introduction to the topic.

4.3.8 Practice

Practice is the core of the app. It's where a user has a presentation of the topic, makes self-reflection, and enriches his self-awareness.

4.3.9 Rate Practice

Rate Practice is where we collect some feedback about recently finished practice and add documents to the table with progress in the database.

4.3.10 Entries

Entries is a place where a user can come back anytime to reflex on his previous thoughts. It's basically a list of all answers from the practices.

4.3.11 Profile

Profile screen stands for settings and some additional information about regulations we need to follow.

4.4 App Architecture

SwiftUI + Combine is a perfect combo for Model-View-ViewModel architecture

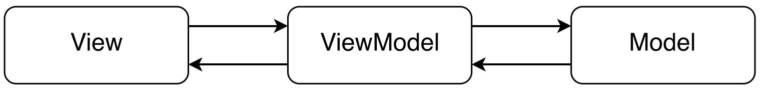

FIGURE 4.3: MVVM

SwiftUI has a built-in MVVM nature, as it was designed to have a single source of truth and react to its changes. With its Bindings system, unlike UIKit's UIView-Controller, you don't need a Controller anymore, you just bind a view state directly to the model. Any changes will immediately trigger the rendering mechanism to make relevant changes to the view.

Chapter 5

Future Improvements

First of all, we need to release the first version as soon as possible to see users' feedback. Then the stage of improving user experience. And after that, we're going to add an audio experience, where you can listen to subject-related stories and record your answers. There is also an idea to train a neural network model on philosophers' works to make a chat with them.

Chapter 6

Conclusions

In this work, we discussed the technical solution for making people healthier and happier. What parts the solution consists of, how it works in-depth.How to work with Firebase. Discussed the differences between databases. And the overall experience I gained doing this project.

Bibliography

Kerpelman, Todd. *The Firebase Blog: Cloud Firestore vs the Realtime Database: Which one do I use?* URL: `https://firebase.googleblog.com/2017/10/cloud-firestore-for-rtdb-developers.html`. (Online, accessed: 16.05.2021).