Daily Diary

Mobile Devices Programming (2023-2024)

Final Project

Spencer Johnson
Daniel Strens
Matteo Firenze
Nikita Voronin

- Table of Contents

Introduction of your project **3**
SDK's **4**
API **4**
Design **5**
Problems you've found **6**
Conclusions **7**

Introduction of your project

As mental health has become very important these days, our group has taken an approach to try and assist our peers as much as possible. Our idea is a journaling app, that allows the user to receive a push notification at a certain time to remind them to do their journal entry. This entry will be associated with a date, title, description, photo and voice note. The information is saved depending on the user and added specifically to a list of that user's entries. This is all managed using Google Firestore for both managing the user information and the entry information. Our Firestore Database was set up by having a folder of userData that stores a unique identifier that is associated with a user login. This ID is then given a folder of entries in which all of their journals are saved. When the user logs in, a list of the entries are shown, and the user can click on the date to open a page to view the entry, or use the right side 'garbage' icon to delete it. Similarly, they can add another entry to their list by providing the attributes mentioned above that are associated. This way the user will be able to monitor and log their mental health throughout the month and take care of themselves.

SDK's

We utilized many different libraries throughout the development of the project as they are critical in the modern day to develop efficient software. Our journey through our sdk's begins with the firebase_auth which helps us to get the user into our app (login or signup) while also checking the information for correctness. This is very important because it allows us to save time on authentication and focus on things like the second sdk involving information with the database. This would include adding, deleting and retrieving the information. They seems to be pretty simple tasks but sometimes we run into fun problems that force us to think a little more. Leading us to a third and important sdk used for permission. This would prompt the user for the choice on whether to allow certain features of the app, such as camera or voice, to receive information. This would also branch into the actual sdk's involved with the recording or picture taking. Each task had its respective libraries that would have a controller (or equivalent) that required initialization or some other signal to function properly. After accommodating these factors and providing the correct information when the system queried, we were able to put the able together properly.

API

The API we chose to use for this project is the google calendar api : https://developers.google.com/calendar/api/guides/overview

The api functions by beginning by asking for permission for access to the user's calendar. Doing this allows us to create an event to the primary calendar, which will serve as a reminder for the user to record a journal entry. We chose this API because we had a tiny bit of experience using this API in python from past projects from other years and this allowed us some similarity from years previous. The outcome was found to be relatively easy and did not throw many errors when it came to implementing. Furthermore, the Google Calendar API serves as a robust and reliable tool for seamlessly integrating calendar functionalities into our project. By initiating the API with the necessary permissions to access a user's calendar, we can efficiently create events that act as reminders for the user to record a journal entry. The API's straightforward authentication process, coupled with its comprehensive documentation, has facilitated a smooth implementation, ensuring that our application can seamlessly interact with the user's calendar data.

Design

Our group proceeded forward with a generally simple design, nothing too fancy. But we did not want the app to be plain so we attempted to add hints of colors in certain places to allow for the user to enjoy their experience with our app. The reason for the simple design is that our group feels that things today are quite crowded and clustered. We wanted the user to feel like they had space because after all we should provide a safe space to track emotions. Building on this, color theory says the color blue has a calming effect on the mind, for this reason one can see the main color element throughout the project is blue. The widgets are constructed in a basic manner, by using SingleChildScrollView we are able build the list of entries to show the user when they sign in. Other widgets such as TextFormFields allow us to receive information from the user by means of keyboard. Similarly we take this information and display it back to the user using the class JournalEntry and building strings with its attributes. These strings are passed to Text with Padding and allow the user to visualize the information they have submitted to the database. This was all prepared through figma to allow the team to gradually move forward with an idea of how the final result should appear. Although there was not much needed to plan as most of the screens are generic app screens. What this means is that the user is expecting to sign into an app a certain way, as well as, there is a specific way to show a list of information to the screen and our group followed these formats.

Problems you've found

The first main problem we encountered was that the program would not compile on IOS. Attempting to compile the program on both the internal simulator from xcode and numerous external iPhones, both resulted in unsuccessful compilation. As normal we took to stackOverflow to find a solution for our problem of 'symbols not found'. However we were unable to locate a solution, so we moved to ask Alex, our teacher, who was unable to enlighten us as well. This proved strange that it would not compile on IOS as it compiles on many android devices, and as there was only one member of the group with an iPhone, we chose to continue forward as an android based app.

The second problem was finding old libraries. Not so much of a problem as a waste of time as occasionally implementing something would be in vain as the library has since been renamed. Also there are obviously many more resources on the older material. At first this seemed like trying to find a needle in a haystack. But at least this haystack was code and we could learn something along the way. It began to become easier to determine what we did not need to do and what was actually not allowing us to move forward. What this meant is that when we found the random obscure line of code to fix everything, we knew it was exactly what we were missing.

Another problem with libraries we have to mention is their compatibility. A lot of dependencies depend on other dependencies (so called transitive dependencies), and those can come in different versions. If Dependency A and Dependency B both depend on Dependency C, but Dependency A depends on version 1.1.1 of Dependency C, and Dependency B depends on version 2.2.2 of Dependency C, we'll experience a conflict between them. These conflicts are very common. In fact the programming community has invented a word to describe the situation when there's a lot of transitive dependency conflicts: it's called dependency hell. The solution, in our case, turned out to be pretty simple: we just set the transitive dependencies versions to "any", and that made the code function properly.

Microphone usage caused us a new issue. The recording would start and stop properly, and it was also stored in the system correctly and could be played. The only problem was that there was no sound. Apparently the problem was not in the code, but rather in the emulator settings - it didn't have a microphone input set up. What we did was going to the emulator's extended controls, opening the Microphone tab, and turning on the "Virtual microphone uses host audio input" setting.

Conclusions

Overall the group is extremely satisfied with how the project turned out. We were able to demonstrate not only the knowledge we have already acquired but also showcase newly learned material. This can be seen in something simple as displaying a string and progressing up to working with live databases that can receive many different types of media input. As we began the year working only with IOS and moved into Flutter it was interesting to watch the development process change with the languages. As in IOS we could have had problems with a constraint and something of this nature does not exist in Flutter. As the dart language can be seen as similar to the nature of web development, the group was already a little more comfortable when constructing the scaffolding of the particular screen. This allowed us to focus more on the logic and creativity of the app itself rather than constantly worrying about utilizing a new programming language. The dart language itself is very intuitive and has many libraries that assisted our group along the way. From something like the camera, to the permission handler, to Google Firebase, the dart language was constantly providing different options to implement what was required of the group for the project. Even though the group experienced the occasional speed bump that required debugging, or using outdated libraries, these were easily overcome as different members of the group were more experienced and were able to alleviate these problems as they arose. By utilizing good communication and good programming techniques the group was able to build a fun project that we are proud of.