

GOPray.

Team Oreo

Noah Duncan

Joshua Tarwater

Yayira Dzamesi

Patrick Dudenhofer

CS4820

3/25/2021

**Table of Contents**

1. Abstract 2

2. Project Definition Statement 2

3. Introduction and Background 2

4. Requirements 4

5. Constraints and Assumptions 4

6. Design 5

7. Individual Achievement

7.1 Noah 6

7.2 Joshua 8

7.3 Yayira 10

8. Societal Impact 14

9. Conclusion 15

10. Bibliography 15

11. Appendices 16

**1. Abstract**

Our project’s purpose is to develop a cross platform prayer app that will allow users to create and manage prayer requests and then the app will send notifications throughout the day to remind users to pray for those requests. Over the course of two semesters, we succeeded in creating such an app. Our app allows users to create notifications and organize them into categories or “Journals”. Reminder days and times can be scheduled for these categories and the app will send the user reminders to pray. Users can also navigate to a page that will randomly select up to five prayer requests for the user to pray for during his or her daily quiet time. Users can also share requests with each other locally through scannable QR codes. The app is available on both the iOS and Google Play Store for download.

**2. Project Definition Statement**

Christians are called to pray without ceasing, but it is easy to get distracted and forget. The goal of this project is to create an application that will enrich the user’s prayer life. The app should have features that improve both morning prayer time and “without ceasing” aspects of prayer time. The user should be able to add prayer requests that the app will store. Then, during morning prayer time, the user can navigate to a screen that will display a few of the prayer requests for that day. Throughout the day, the user will continue to receive reminders of various requests as a reminder to “pray without ceasing”.

**3. Introduction and Background**

As aforementioned, the goal for this project is to provide a minimal and focused resource for individuals who are seeking to spend a more intentional and consistent time in prayer. Though the app is focused on the individual, it is developed in a way that encourages uplifting and supporting one another in prayer. The primary customers for this app were Professor Dudenhofer and Keith Holcomb with the Global Outreach Office. However, over the course of the project, we gained a secondary customer, Dr. Kevin Jones, the dean of the Department of Education. Patrick Dudenhofer is a Cedarville University Computer Science professor and one of his desires for the past few years was to proctor a Senior Design team to create an intentional prayer app. When the opportunity arose, the Global Outreach office also expressed interest in the project. Global Outreach exists to prepare and organize the Cedarville University community to know, live, and share the Gospel. One of the ways they do this is through the annual Missions Conference at the beginning of the Spring semester. The original goal was for the app to be released in tandem with the conference and students would be able use the app to store prayer requests from the booths of the visiting mission agencies. Due to COVID-19, releasing the app in tandem with the conference was not feasible. Instead, news of the prayer app is being spread through word of mouth and it is being used by those who are serious about praying more consistently throughout the day.

This project started from the ground up with no former affiliations and so the research gathering and requirement building phase of the project lasted significantly longer than anticipated. That said, it was easier to get up and running once the above was decided because once development began, the only bugs and hiccups we had were those we wrote instead of having to try and understand the thought process of another developer from multiple years ago.

**4. Requirements**

1. Create a cross platform app.

2. The app should allow users to create and manage prayer requests

3. User can set the expiration date for prayer requests, after which requests are archived, and removed from the main page

4. Prayer requests can be sorted into various categories of “folders” or “journals” (e.g., family, local church, etc.)

5. Prayer requests periodically send the user reminders. Frequency can be determined by the user in one of two ways

1. The user puts the request in a category. Based on the frequency set for the category, the user will get periodic reminders about a random request in that category
2. The user sets the frequency of reminder for the request independently from the category

6. Prayer requests can be quickly loaded into the device without network capabilities

7. A prayer request can be marked as “answered”, after which it will be archived

8. Expired and answered prayer requests can be viewed in a separate folder

9. The app should be primarily local and should not be dependent on information from the web

10. The app should have a page that displays requests in a way that augments daily prayer time

11. The app should have a page that display various scriptures that the user can pray through

12. Folders/Journals and prayer requests should have various display formats (list, image etc.)

13. This app will not have social networking features

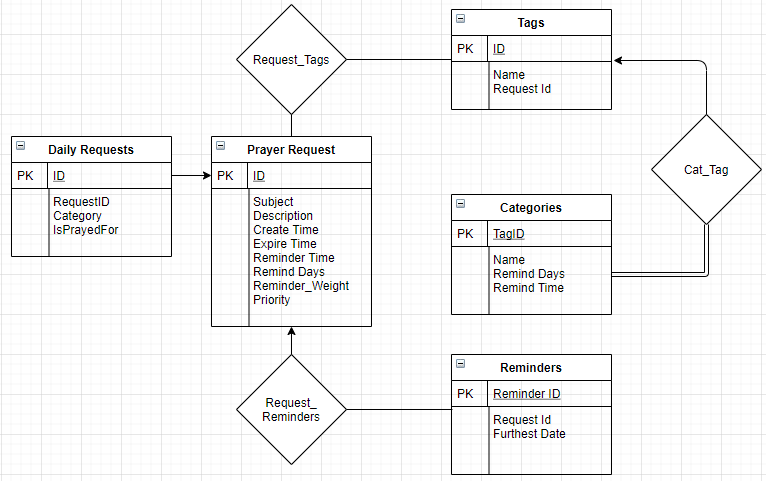
**5. Constraints and Assumptions**

We were not to build a network heavy or complicated app. Apps such as YouVersion start out as a bible app but then over time, they gain features like streaks, sharing, daily passages, and a number of other extra, unnecessary features. Our app was not to be like that. It was understood that we would build a minimal app with no web connectivity.

**6. Design**

The biggest change that we ended up having to implement was changing the app’s name from “CUPray” to “GOPray”. This change took place because the app was denied permission to be created with any affiliation with the University due to some ongoing marketing issues outside of the project. It was an unfortunate hiccup and a process that took weeks to come to a conclusion but in the end, the best solution to get the app released on time was to change the name and personally pay the developer account fees. Aside from that, and perhaps being a little over ambitious about what we could accomplish, we stayed pretty close to the design requirements and our initial design.

Our actual implementation consists of a series of screens. Each screen has a method called a useEffect (a react native hook) which runs when a screen is first pulled up. In the use effect, we do things like get data out of the database to show the user, or cleanse the data in a certain way before displaying it to the user. Some of the screens pass data between each other. For example, the screen that displays the information about a given request only knows which request to pull out of the database because the screen that navigated to this screen told him that he should pull up the request with the id of such and such a number. Each of the screens are connected to each other through react native navigators. We have 2 kinds. We have a drawer navigator. And we have a nested stack navigator. The screens in the drawer navigator can be navigated to arbitrarily, but the screens in the stack navigator have a bit more order. You can’t just jump to the requests screen, because that screen displays the requests in a certain category, which you have to select on the category screen, which you have to navigate to from the dashboard, etc… All of these screens are able to access a local database to get the various requests for each screen. The database schema is shown below.



The last design element to discuss is the scheduler. It works by using a lottery system. Each request is given a certain number of tickets depending on its priority and how long it has gone without being selected. Then requests are selected randomly and scheduled for the days and times that were predetermined by the user. One way we could improve this design is to move some of the scheduling load off of the user. Instead of asking in a sense, what days and times do you want to be reminded about each category, we could ask, how many requests do you want to show up across this 8 hour period? Or across this 12 hour period? Then we just handle internally how to spread that many requests throughout the day.

**7.1. Individual Achievement: Noah**

I began the semester by putting a lot of work into following tutorials. Through this, I was able to gain a basic understanding of the technologies we used as a team. These tutorials focussed on react and react-native and, because of this preparation, I was able to communicate with the team better as well as understand the parts of the project that they were working on.

I also wrote the entire reminder scheduling system for the app. This included developing the algorithm as well as implementing it in actual code. Before I could work on the scheduler however, I researched how to send notifications in react-native. Rather than work with network-centered push notifications with a service like firebase, I decided that we should use ahead of time scheduling. The expo API’s were very helpful here. The algorithm design itself required a lot of thought because it required solving a largely synchronous problem in an asynchronous environment. Once the algorithm was developed, implementation proved to be difficult due to a lack of familiarity with javascript. However, because I worked on this issue over break, I was able to push past challenges and create a working piece. The scheduling algorithm I created manages both what requests are sent throughout the day as well as what requests show up in the “Prayer Time” section of the app. One extra feature that I added to the scheduler was the option for the user to specify a priority for each prayer request. The priority influences how often the request is chosen by the scheduler. This required more thought and effort, but I was very happy with the result because I believe that that functionality is very helpful from the user’s perspective.

Another important component of the app that I worked on was the use of QR technology. Specifically, I researched and experimented to find the right libraries and then showed Joshua what code he should include when he added the functionality to the app. Also, I wrote a program that automatically converts text prayer requests into shareable qr codes. I used this program to parse through a list of prayer requests from our client and convert them into qr codes in groups of three. That way, Dr. Jones can distribute the prayer requests in an easy-to-use way. He was very excited to see this in action when we demonstrated our app to him.

Another task I accomplished was publishing the GOPray app to the Google Play Store. This is something that I had never done before, but it was a good learning experience. It involved checking a lot of boxes and answering a lot of questions. Because we published the app, it can now be downloaded by virtually anyone with a smartphone. The published app includes a link to a Google form that I created so that our team can gather feedback and continue to improve the app.

Finally, I also did a lot of app testing to ensure that some features worked. Through this process, I found a few bugs and was able to alert the team. Because we caught these bugs, the users did not have to deal with them themselves.

**7.2. Individual Achievement: Joshua**

I worked on the majority of the functionality of the app. I am responsible for almost every button that does a visual thing, whether that thing be jumping to a new screen that pulls data out of the database to display, creating new categories, requests or tags, filtering requests, deleting requests, etc...To be fair, Yayira did create some buttons and screens, but she was basing them off of code that I wrote and she copied (not to take away from her achievements. My buttons would be grayscale and the wrong size if it weren’t for her steady work in the background to make things look good.) I created the database schema that you can find above in section 6 and wrote the functions to actually create the database within the app. For detailed explanations of what I developed and how, you can refer to the “Back End Brain Dump.pdf” that should be in the git repository. For here, I’ll give a top level overview of some of the things I’ve done.

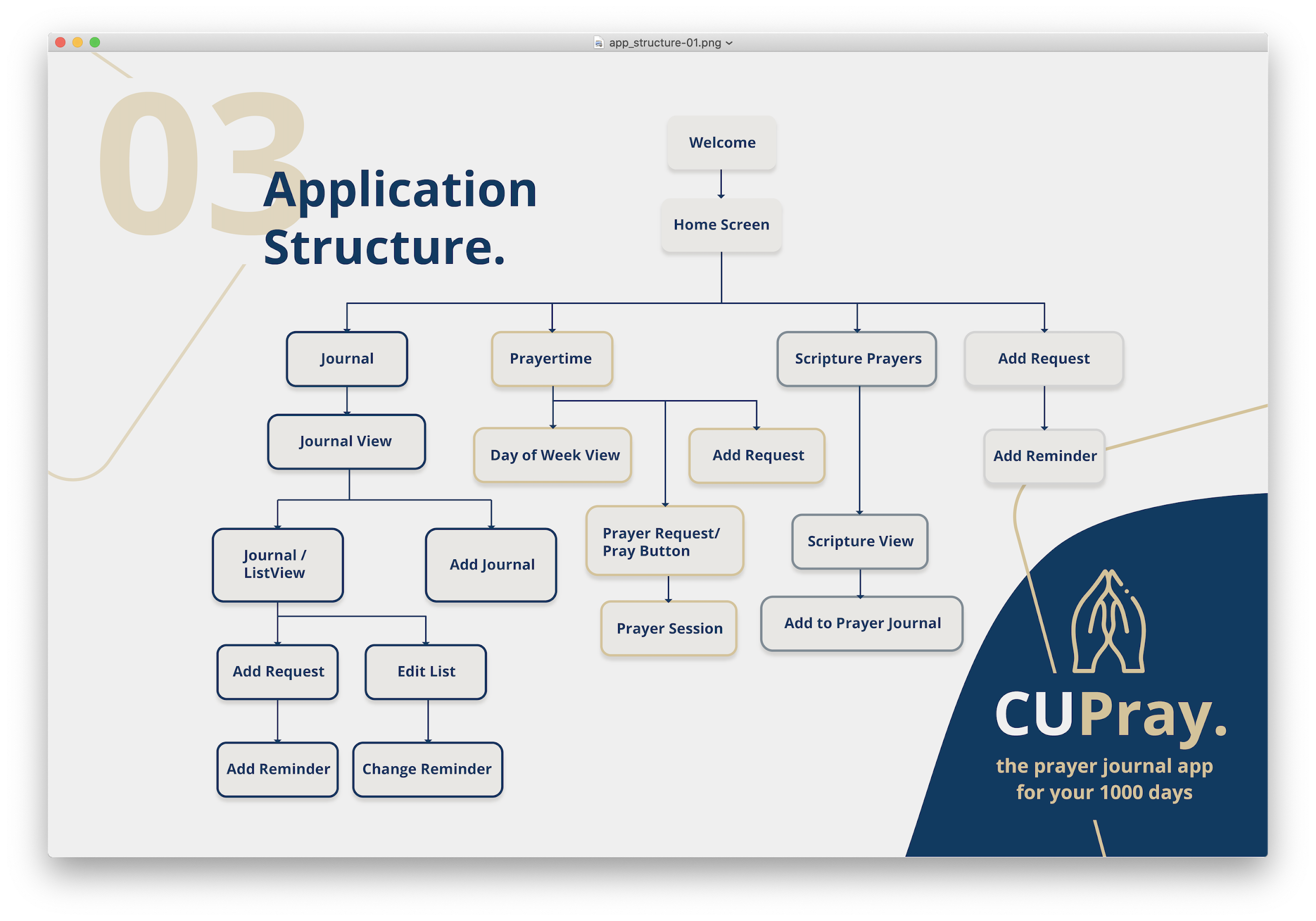
I figured out how to use React Native. I made sure our team used functional components and interacted with the database by using callback functions and useState hooks. These hooks trigger a re-rendering of the screen so that whenever the values change, they still update correctly on the screen. Some other hooks I used were useEffect and useFocusEffect. These hooks are called whenever a screen is loaded for the first time the app has been opened or whenever a screen is focused on. In these methods, we get whatever data we need to get out of the database and filter it when necessary so that it displays correctly for the user.

I figured out how to make toggle buttons. Originally, we were trying to use someone else’s library but we were having trouble getting it to work so I built my own React components which we used in a few different places. For example, when choosing reminder times, each day of the week is a toggleable button. Or when selecting tags for a request, each tag is a toggleable button. The styles were done by Yayira (selected styles vs unselected styles) but hooking them up to executable code so that we could dynamically add the buttons (which was especially important since we don’t know how many tags will be present) and using the information to interact with the database was pretty much all me. I was not the one who figured out how to do the dropdown menu or the drawer navigation. However, I did figure out how to connect the dropdown menu to the database so we could save and load the initial information. I also figured out why we couldn’t use the two navigators (drawer and stack navigators) in tandem, they had to be nested navigators. I came up with a solution which Yayira helped me implement to make navigation simple and intuitive for the user. While on the topic of navigators, I am also responsible for passing the data from screen to screen. For example, category passes its name and id to the requests page, the requests page passes a number of different fields to the individual request page so it can populate some fields that would be harder to get otherwise. Similarly, when scanning a QR code, more information than normal has to be passed to the individual request screen since we can’t pull the data out of the database, we’re pulling it out of a QR code.

I created the majority of the database calls. The exceptions are some database calls which Noah created for his navigator. He based his database functions off of mine and did some extra fancy stuff with callback functions to make it work with parameters and callbacks since he was having trouble. I did create at least a couple calls at his request since I was more familiar with the database at first. After Noah figured out how QR codes work, he and I hooked his abstract implementation to our actual project and made sure that we could load in a request (and later on multiple requests, which I did on my own) from a QR code. After Yayira added the search bar and the toggle button to the all requests page, I added the toggle functionality, as well as connecting the search bar and toggle button to 2 different queries to filter requests by tag and either hide or show expired requests depending on the toggle button state. I can’t explain every single thing I worked on here. I wrote a 15 page (single spaced) document where I really go into detail about what I did and how it works, and there’s no hope for it to fit here in 5 pages double spaced. But hopefully, this gives you an idea of some of the things I accomplished and if the reader wants to see more, they can see the document that I mentioned above.

**7.3. Individual Achievement: Yayira**

For the duration of the project, the entirety of the visual aspects and aesthetics of the app has been my responsibility. I relished the challenge to design and develop the GOPray app and I am proud of the fact that the product at the end of this year looks quite similar (with the exception of logo changes) to the initial prototyped design for it as you will see from the images at the end of this section. Every screen on the app was dynamically structured, designed, and illustrated through my efforts. Almost, if not all, of our marketing assets were organized and design by myself as well. Additionally, I was also a main point of contact for a significant amount of the correspondence between our clients and our team.

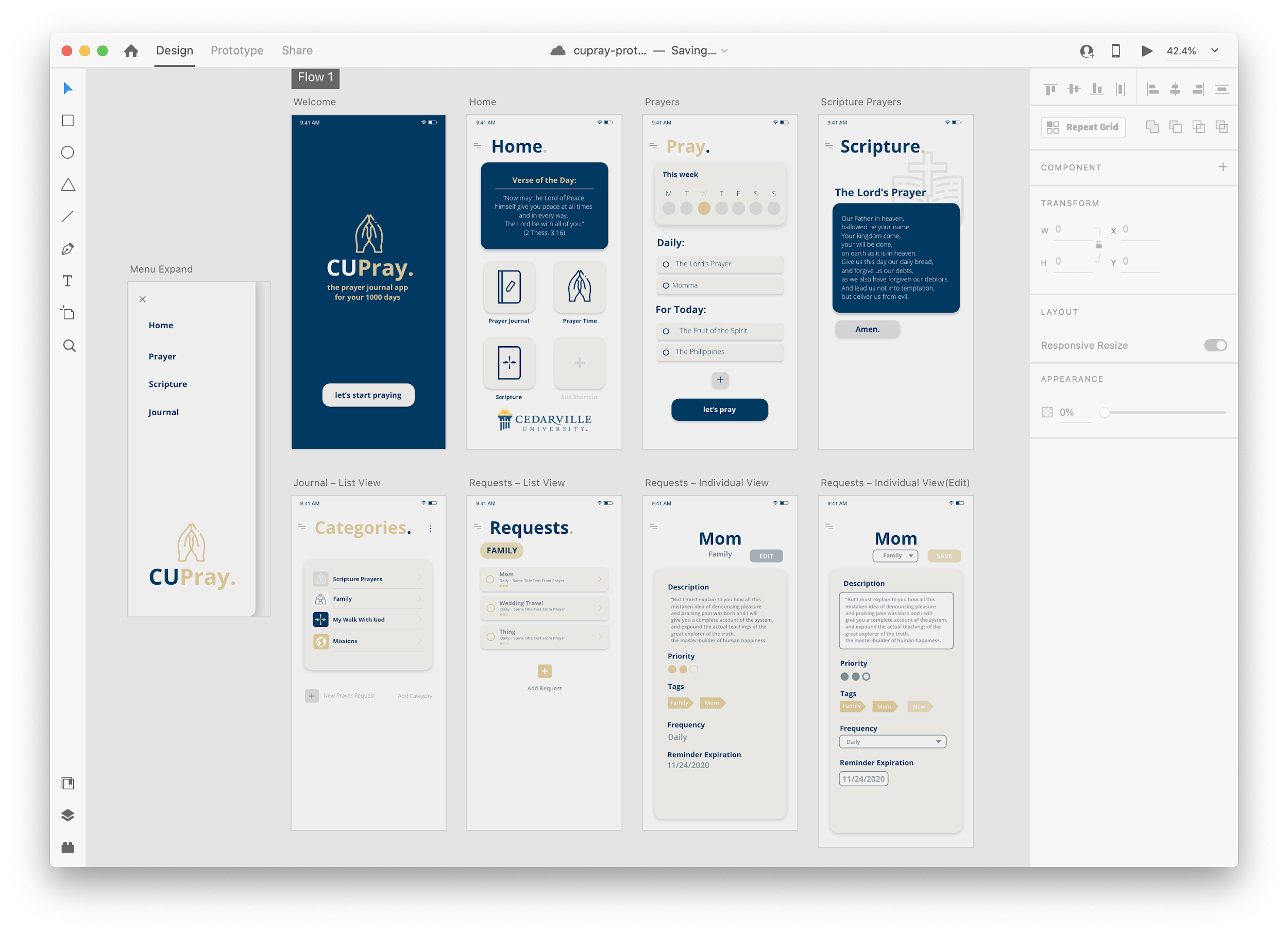
In the initial developmental stages of the app early in the project, I created a full prototype of the app through Adobe XD to exhibit a potential design to our client and advisor so we had a clearer goal to aim for. I also wrote a 10-page document titled App Specifications to explain in detail every interaction the user should be able to have with the application to flesh out the UI and UX to the team to make sure we had a grasp on what we wanted to accomplish. After that, I built an application flowchart (pictured below) that described the design and interaction the user should expect to have. 

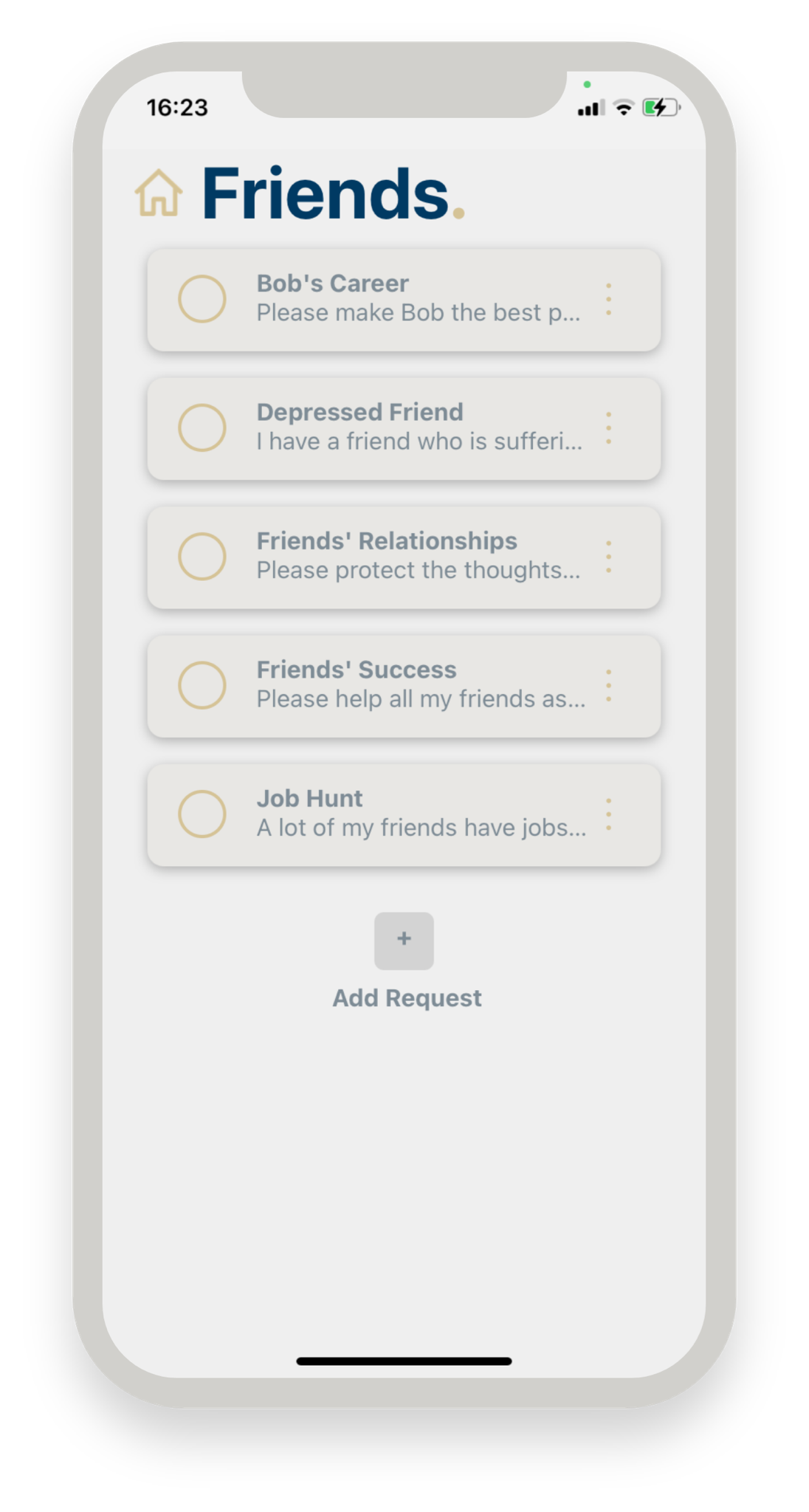
As the UI/UX and Front End Lead, I researched tirelessly many aspects and features of React Native to understand how to best put forward a product that users would enjoy, that would appeal to them, and most of all, satisfy our advisor and clients. I spent hours pouring over a variety of phones and simulators to ensure that the app behaved virtually identical on a spread of platforms, nitpicking on every little inconsistency and sometimes even changing the whole design to better accommodate the diversity of screens. Specifically, I was also able to sort out incorporating modals (popups) into our app and also understanding the incorporation of native date and calendar components for the Edit Category fields. Furthermore, all iconography and assets were illustrated and designed by yours truly. It was important to me that this application is designed in an appealing manner. Since the original idea was to publish with and for Cedarville University, the app is designed with the color palette of Cedarville and the logo was also illustrated to capture a core tenet of the university ( being founded on the Word of God). Though the app is no longer directly tied to the university through publishing, I still believe this design addition speaks to the importance of Scripture and prayer together as can be seen in the passage from 1 Thessalonians that is displayed on the home screen.

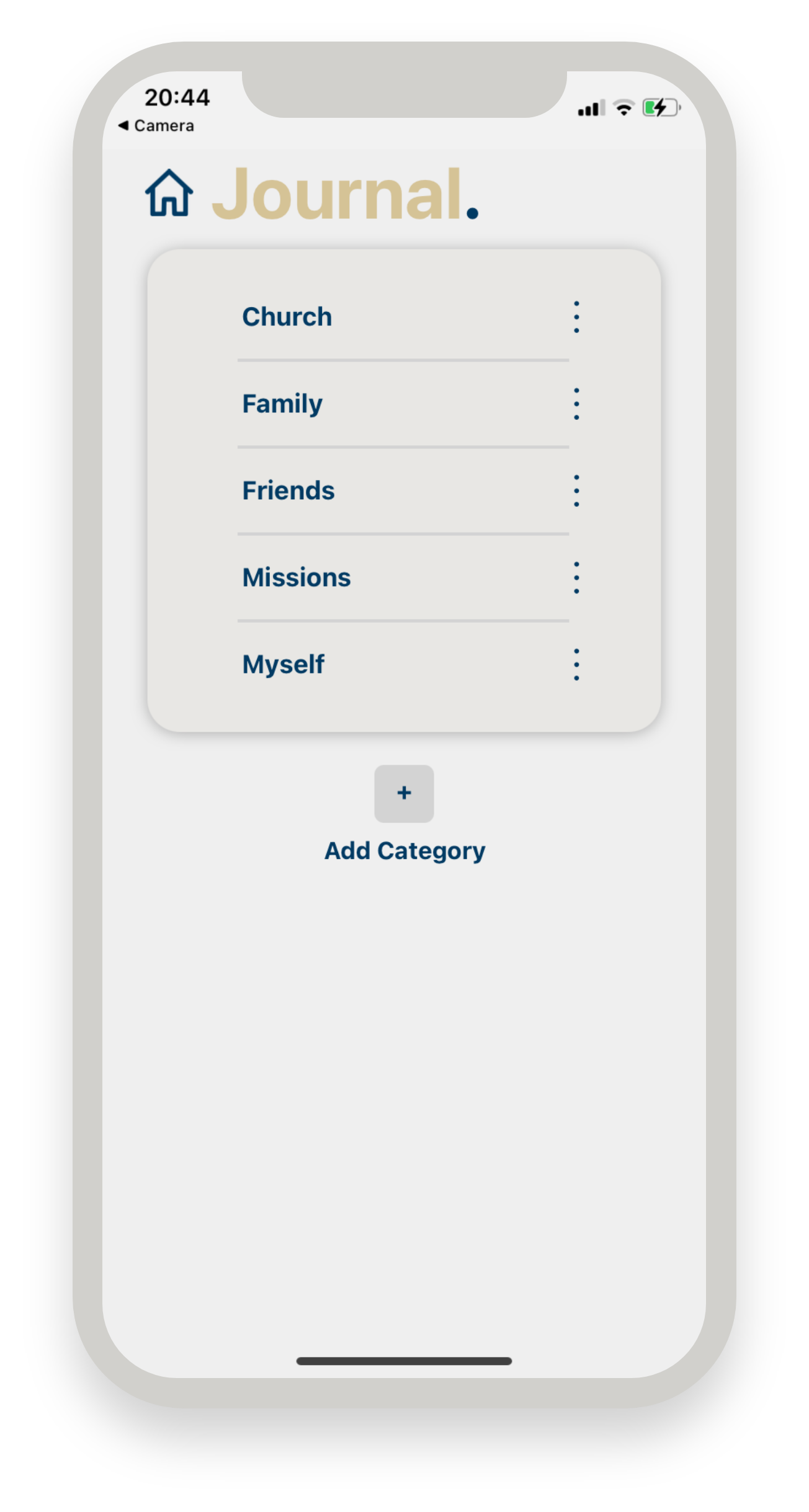
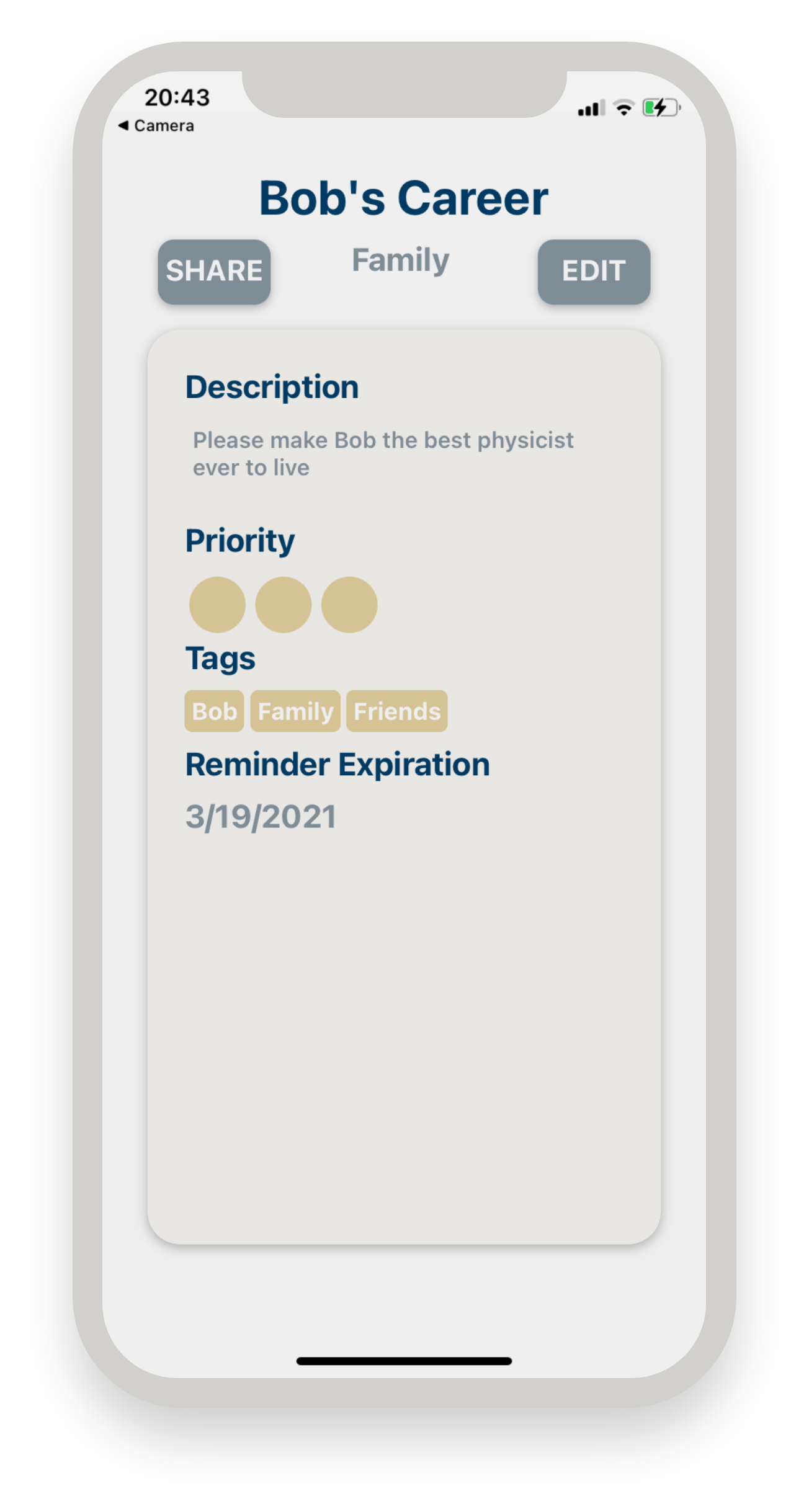
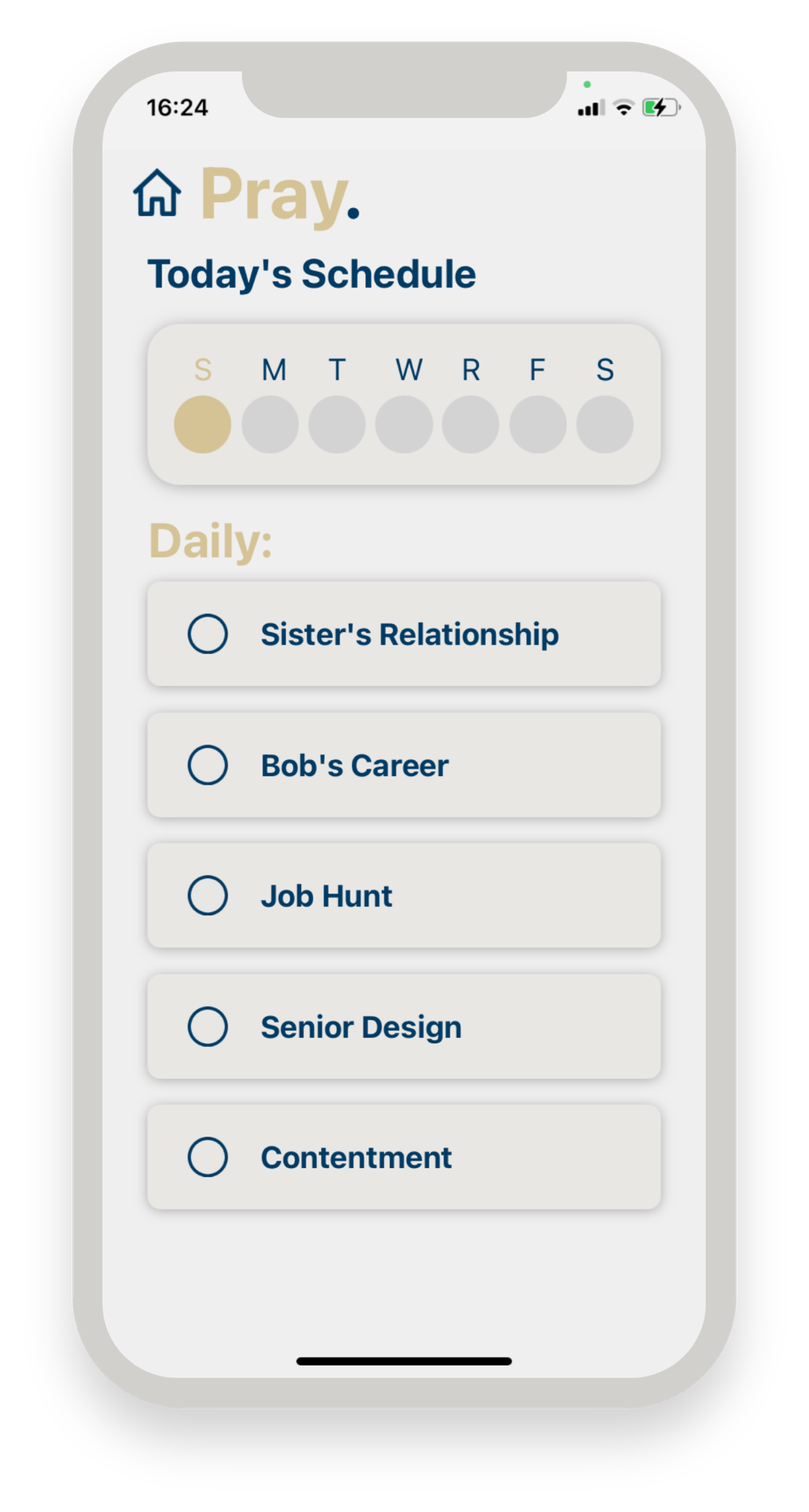
Working together with Joshua, our team lead, I was able to exponentially speed up the rate of creating new pages and putting together the visual assets for Josh to feed the ‘guts’ to through the database. We worked together very often throughout the project to knock out problems together, both database and design related. This teamwork gave each of us a better understanding of what the other was doing and how certains bugs that may have arisen may be connected to something either of us had done. I worked to be flexible and efficient and effectively to make anything my team desired possible. Any design asset, feature, or visual aspect or change that was mentioned and agreed upon was completed as quickly and efficiently as possible.

Lastly, regarding our marketing assets, presentations, and demonstrations, you will notice that they are all designed cohesively with an identical color palette and aesthetic. This was also my responsibility and throughout the project, maintaining our slide presentations, designing and compiling our demo videos, creating and printing our posters, and uploading the app to the iOS store have all been my responsibility to effectively complete. More details of the inner working of the design assets can be found in my multi-page Front End User Guide for the future benefit of any who desire to improve the project beyond us.

Initial Prototype:



Published App:



**8. Societal Impact**

Depending on how much publicity the app gets, and how well it has actually been designed, the societal impact could be somewhat significant. If every Christian took to heart the command to pray without ceasing, and chose to use this app to achieve that end, then society would notice. Christians all over the world could be using this app as a reminder to pray without ceasing, which can only be a good thing since prayer is so powerful.

**9. Conclusion**

In summary, our app met all the requirements. It allows users to create and manage prayer requests. It is a cross platform application. It has been published to both app stores. And it has been published for over a month, so we have had time to get feedback from a lot of people including both alpha and beta testers, as well as a software security team who downloaded, tested, and tried to break our app. We were able to implement a lot of the changes that these people suggested such as fixing a bug that the security team found or adding triple dots to requests as a way to pull up the request options, instead of just the long press method, which isn’t intuitive for the older generation of users.

**10. Bibliography**

Requirements\_Document\_v0.pdf: We wrote this document as a way to communicate our understanding of the expectations to our client last semester. We referenced it heavily in this report.

We Also might have referenced in passing the following documents:

Back End Explanation.docx

Front End Explanation.docx

Scheduler and QR Explanation.docx

**11. Appendices**