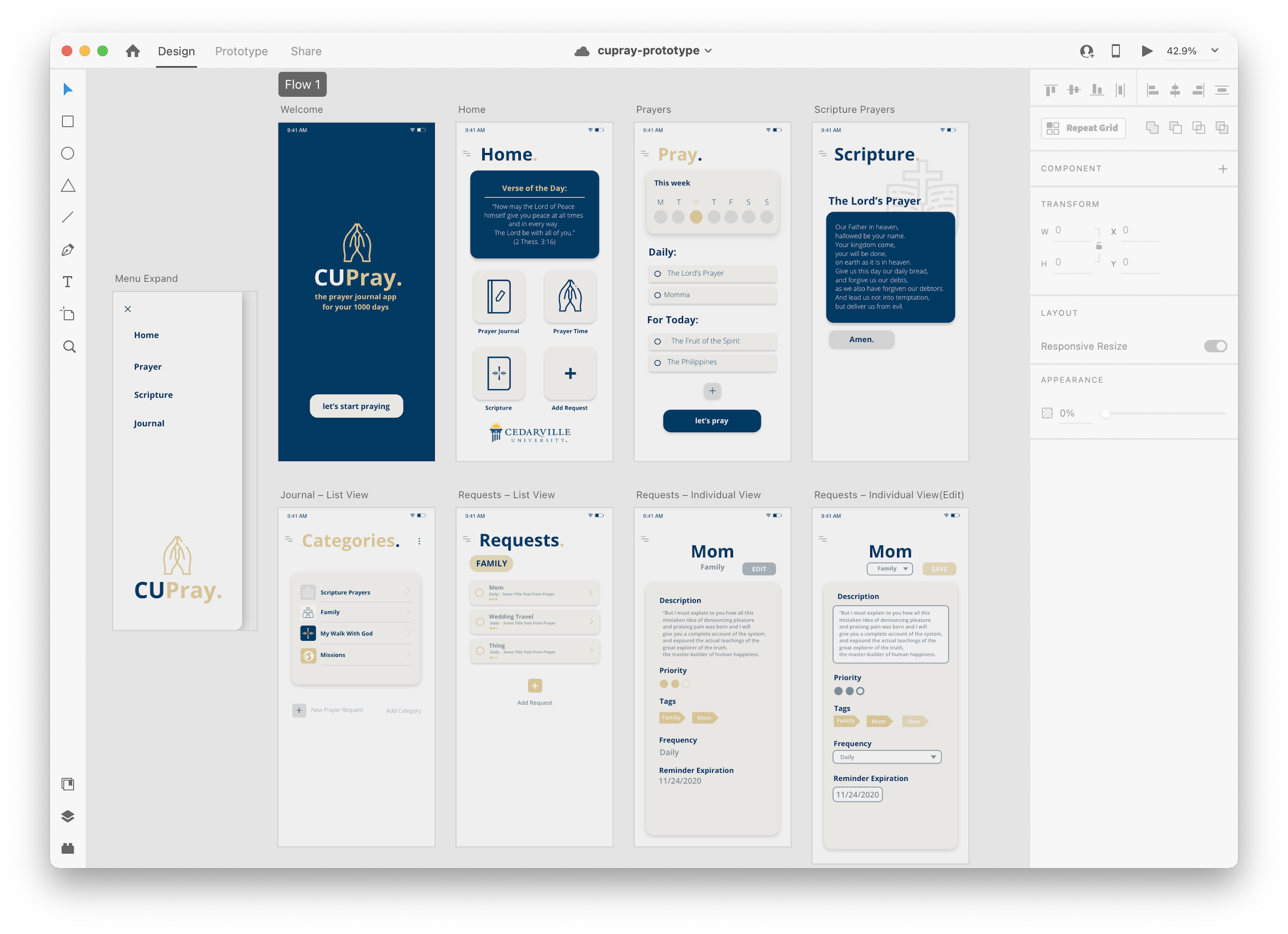


# Front End User Guide

**Overview**

Welcome to the brain dump for the visual side of the GOPray app! So as is mentioned in the other brain dump docs, we really learned RN through building this app so I believe there are still quite a few things that could be optimized, the most significant being the stylesheets of this app.

**Design Process**

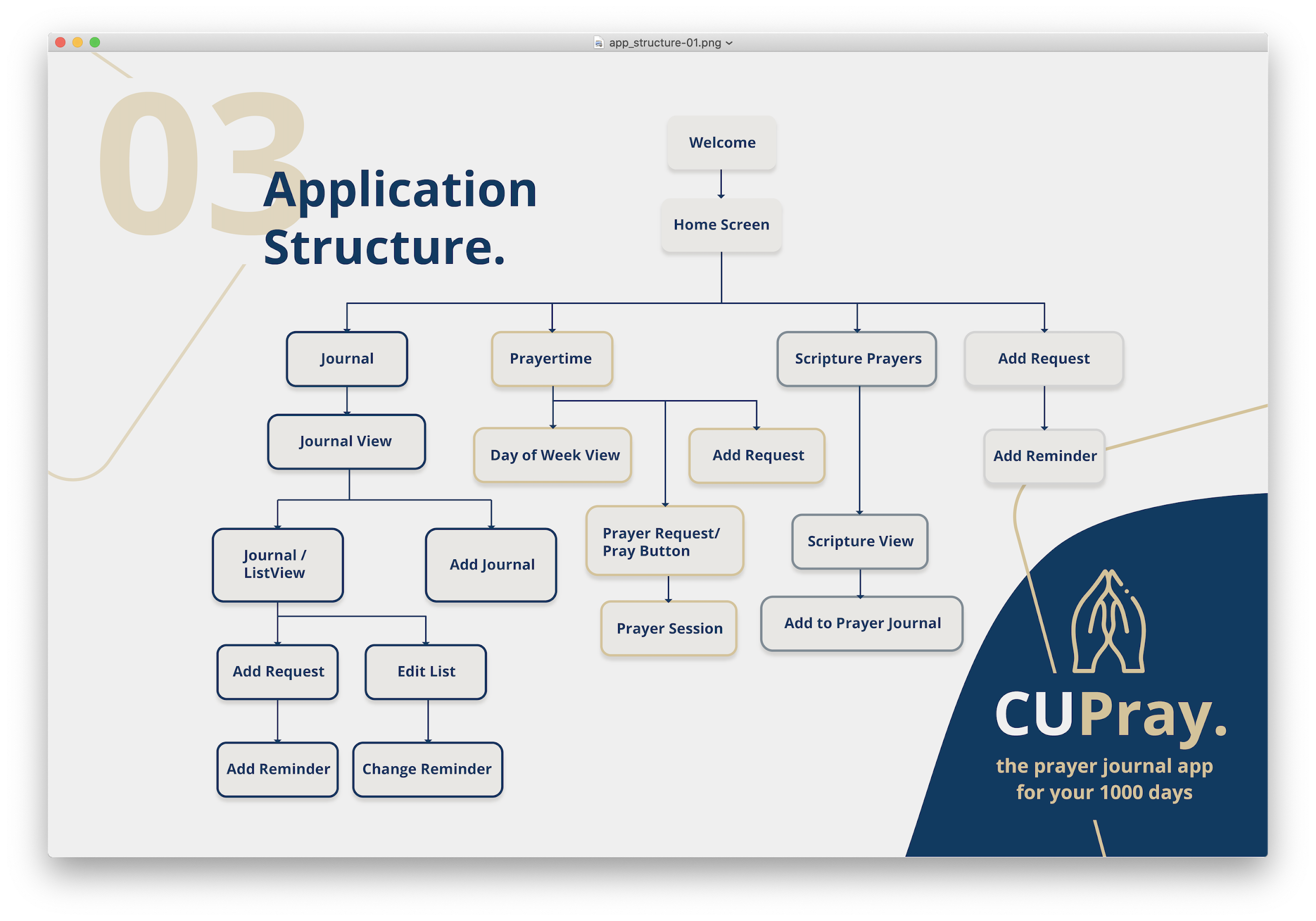
So the initial design for the app was created in Adobe XD. It’s a super helpful prototyping platform that allows you to design your app and literally visualize how it should react to button clicks and user interaction as well. It was the first thing we did after being assigned the project since we were working from the ground up and did not have a super clear vision of the project yet. Additionally, once you’re comfortable with the platform, it’s a great way to test out new features or views before you actually implement them -- HUGE timesaver. 

It is also super helpful as a guide to make sure your final project looks like you planned.

**App Structure**

The flow of the app was also decided in the design and requirement stage. This initial design has been revised with certain things removed/added as our requirements changed. One of the noticeable changes is the removal of the Scripture Prayers section to be replaced with the QR code generation. It was really confusing for us and our clients to figure out how we wanted the Scripture section to work so we scrapped it.

Additionally, from the prayer time screen, the flowchart is shorter and just ends at the prayer request button.



**Color Palette**

Since the app initially was going to be marketed and branded by Cedarville. we used Cedarville’s blue, white, and gold hues. When we got to the time of deployment and realized that we would not be publishing under Cedarville, for sake of time, we kept the same colors. I did do a sample color variation but with the limited amount of weeks left, other things were higher priority than changing the colors of the app. It’s definitely something to consider for the future though.



**Assets**

All of the illustrative assets were drawn by me (Yayira Dzamesi). All the necessary assets for the app’s function are in the assets folder and the additional design based assets will be uploaded to the repo as well for future tinkering. You will need an adobe account or free trial for the XD and PSD files.

**Views**

So the <div> of HTML becomes the <View> of React Native. There are multiple kinds of views such as the scrollview and safe area view, both of which we use in the app. There was a good bit of view manipulation that had to be done to make sure the screens scrolled correctly on different platforms. The best way we found to solve the scrollable issue with our flatlists (the lists of information such as categories, tags or prayer requests) was to utilize the ListHeader and ListFooter components for the Flatlist. This allowed us to have information that stayed at the head and foot of the page while still appropriated scrolling the requests as expected.

**Stylesheets**

This right here is the heart and soul of the visual aspect of the app. Stylesheets are what make it all come together. I apologize in advance, future reader for the fact that you will have to go through and try to make sense of my terribly inconsistent stylesheets I have tried to make your job a little bit easier by starting that transition process. Most if not all of the stylesheet items are names similarly across the pages but some have minor tweaks based on the content of the page and so that is where the bulk of the nitpicking will be.

Within the stylesheets, there are a key thing to keep in mind :

* You are developing for **multiple platforms** of **multiple screen sizes** so when assigning values like height and width, it is essential that you use the global height and width dimensions identified by this line:

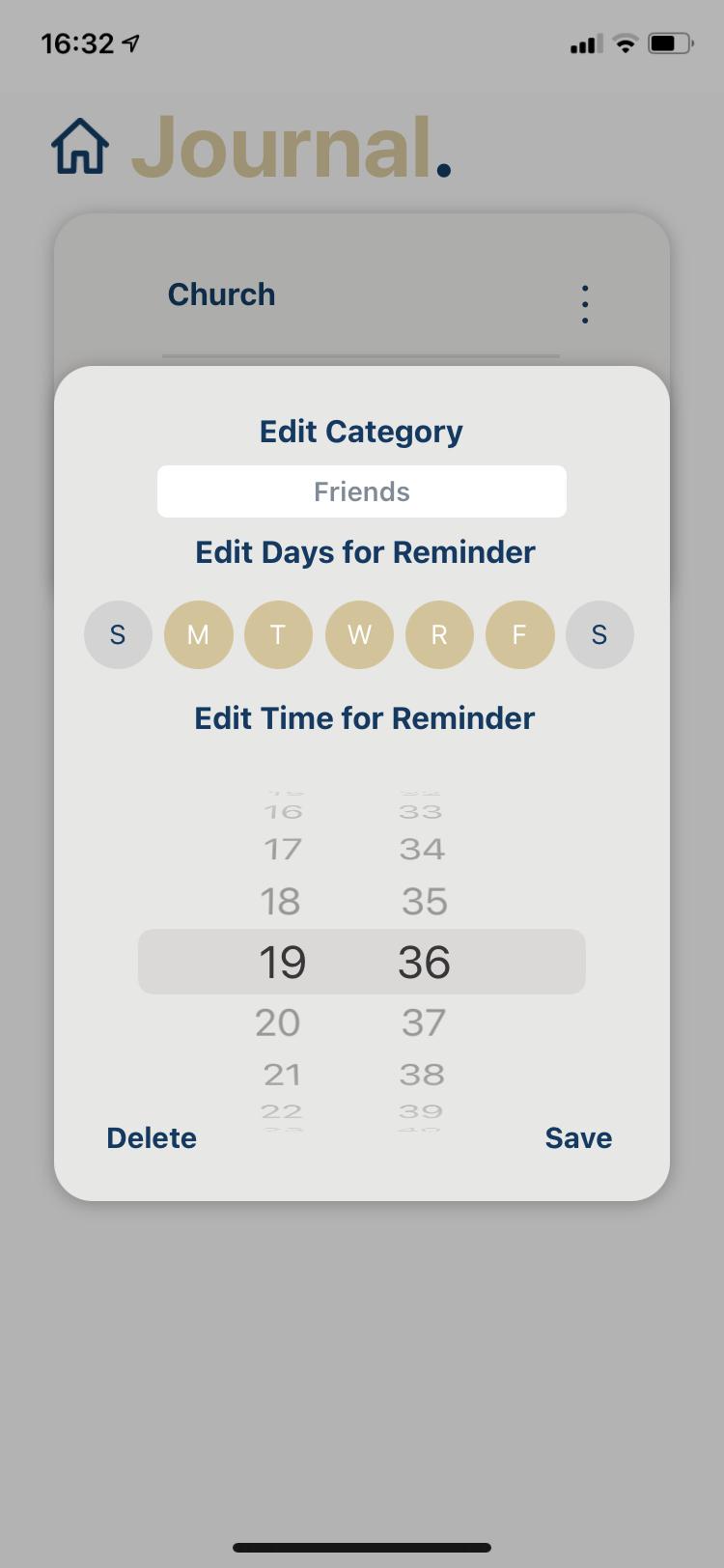
var { height, width } = Dimensions.get("window");

at the top of every page. This line takes the height and width of the screen the app is currently running on and saves them into the vars **height** and **width.** So we use fractions of these vars to place and size screen assets in a way that should react similarly on different phones of different screen sizes.

**Importing Assets**

The asset file within the code repository holds all the necessary items for the app to run. As mentioned earlier in this guide, I have also uploaded a zip file called assets that has all the working documents from this project. As for importing assets into the screen, it’s similar to other imports in just grabbing the path name and adding that to the required source like so:

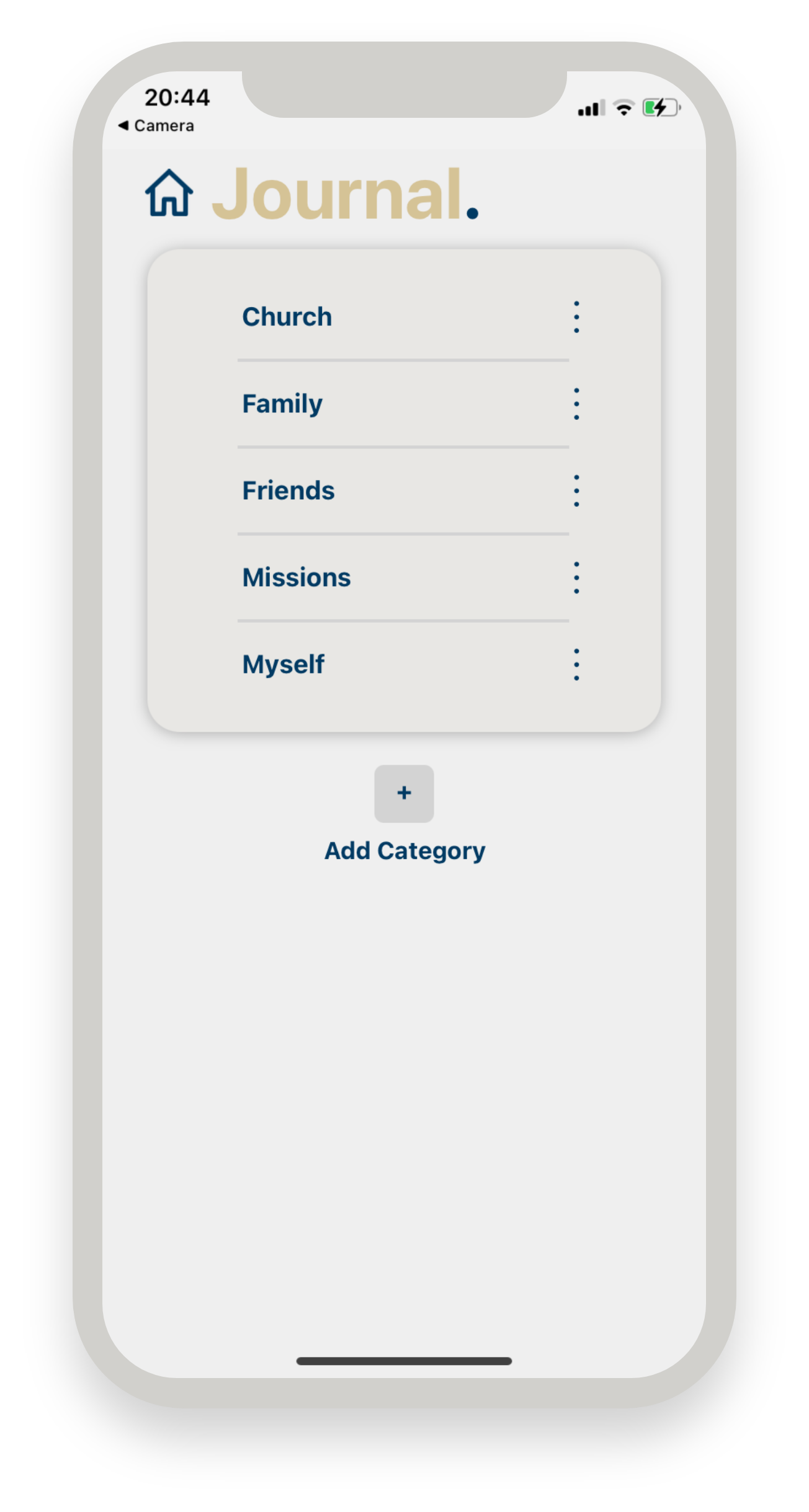
source={require("../assets/hamburger.png")}



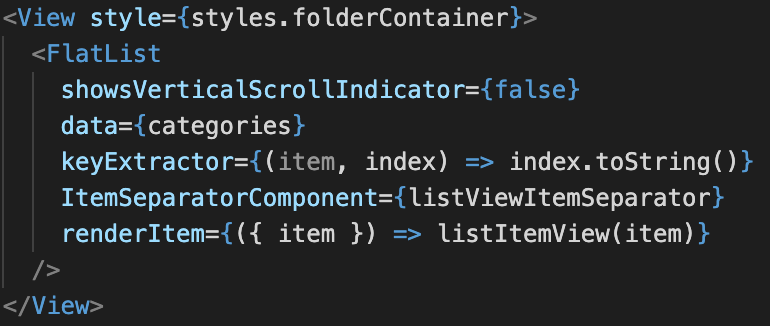
**Modals & Flatlists,**

M**odals** are all the little popups (as shown on the right) where we hide additional edit/manipulation information for the app. They are all triggered by long press and/or by vertical ellipses. For their attributes, they are all styled similarly, have a quick animation in and out time and an isVisible property for the tap on and off feature.

Note: for future fixes, we saw that the tap off of the modal to deactivate its visibility doesn’t always stay tapped off. So if you pull up the modal on a request or category, tap off, and then try to go back home, sometimes the modal turns on again.

**Flatlists** are the lists of requests, categories, tags etc. seen throughout the app. Essentially, wherever there is a list of data, that is a flatlist. Their purpose is to display the database information of the type for the page. They are rendered entirely via a query command and are styled individually for each item in the list as shown below.

For visuals sake, we also deactivated the flatlist scrollbar.

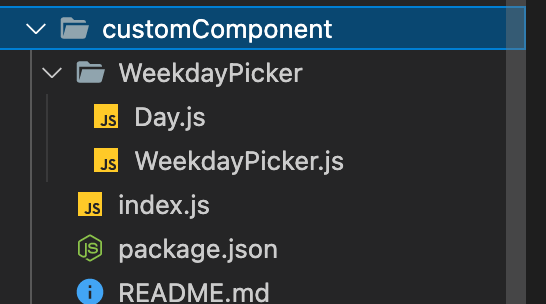


**Modes**

There are a few pages that utilize a mode-based design. The primary is the Individual Request page. For example, we have what we call the Edit mode and the Save mode. In the edit mode, the user is, as the name implies, editing the request that the page is up on. In this mode, the only button on the top of the screen will be save. Another difference is that all the tags in the app database that have been created for that request and others will be present there as a yellow outlined color. The currently selected or active tags will be the same shape but with the filled in color and the tag title in white. Below the tags is the date picker for the expiration date of the request. This picker will be the native date picker to the user’s phone so it is pretty intuitive how they should use it.

The save mode (which is really just the else statement of the clause), is the identical page except only the selected tags are shown and there is also the share button should the user desire to generate a QR code of the request.

**Checkboxes**

We discovered that RN’s checkboxes were extremely limiting and difficult to work with so we found a pretty efficient work around that allowed us the flexibility to design them like we wanted. This is used particularly for the priority bubbles, and the day bubble pickers in the category manipulation modal. Essentially, we overrode a react native component and adjusted it to our liking. This can be found in the customComponent folder of the project. 

This actually touches on a very powerful part of react native in which you are able to override almost any component in its libraries. We did not take full advantage of this due to time and learning constraints but in the future, that could be an optimization route that could be most profitable. For example, overwriting a view or button component so that every view and button in the app has the same aesthetic and visual aspect. Had we more time, I definitely would have enjoyed learning more about that.

**Summary**

There are a few other things that could have been mentioned but these are some significant highlights of the front end aspects of our GOPray app. I hope you have a jolly good time continuing the project, reader!