Froogal: Update Document

By: Chris Adamson, Dylan Commean, Johnathan Dickson

**Table of Contents**

[**Changes Outlined**](#_fn12lqll6wbl)

[Initial Proposal 2](#_uwyduvcwq0dr)

[Requirements](#_tmh15gfreahh) 2

[Design](#_5ngupf86m3b4) 2

[Items that are complete](#_b3om5hfxcloz) 3

[Items that are partially complete](#_hrarb5e9x0c9) 3

[What still needs completed](#_tcr3abvddb46) 4

[Issues that are unresolved](#_73zw4txjusiy) 4

[**Will these issues be major or minor to the success of the project?**](#_8agbmmtit6do)4

## Initial Proposal

Building and scaffolding this website has been a complete learning experience for the whole team. With the initial proposal, we surmised Froogal to be a challenge that requires just enough hard work for the team; however, we wanted the heavy lifting of design to be removed through our technology. This is the reasoning behind our choice of either Tailwind Cascading Style Sheets (CSS) or Chakra User Interface (UI). The team was undecided about the database that would be used. Deciding on Firebase has been the best option even with its challenges, learning curves, and poor documentation. Other than the technologies we intended to use and a few features we have deemed as “Will build if time permits”, our proposal has stayed relatively the same. The team is aiming to give user’s the tools and visuals they need to assist in tracking spending habits. This has not changed since we started.

## Requirements

Requirements have been revised and shaped throughout this project’s journey. Originally, there was no information regarding where user data was being processed, stored, or pulled from. This information was crucial in determining what type of data would be collected from the user as well. Furthermore, establishing the data that would be collected from the user, was vital in the creation of the team’s Entity Relational Diagram (ERD) for our database architecture. The requirements have stayed consistent across the development cycle. The team tried to be very thorough with requirements to alleviate any ambiguity in the various components and their functions for the user.

## Design

The design of Froogal is constantly being updated as we develop. There was a consensus to utilize Modal components across a wide range of functions for the site. Modals assist in showing, creating, deleting, and searching user receipts, as well as persist the user to the site dashboard. The team strived for a variety of components that increase interactivity between the user and the dashboard. This resulted in the Drawer component. Initially, there were dedicated pages for user financial and profile settings; however, these pages have been functionally changed to utilize Drawers which activate and display respective data and inputs when their links are clicked in the Navbar. A few notable changes that aren’t reflected in the requirements as well. The database was intended to be only a user document that housed all the subcollections such as receipts, financial settings, expenses, and categories collections. After careful consideration, the decision was made to make receipts and expenses a top level document to assist with faster acquisition and deletion. A quotes document was added which will provide the Over/Under Signaler the quotes it needs.

## Items that are complete

Throughout the development of Froogal, we have been able to complete many of the user interface features. Some examples of this is that we have built the forms where the users can input their information to sign up and login into our web application. On the dashboard, we have completed the designs and layouts of the components for our application. The navbar for our application has implemented the buttons for the finances tab and the user settings tab. Whenever you click these buttons, it will produce a side modal where the user can input their settings or financial information. To reiterate, these components are completed in terms of their design and user interface. In terms of the actual dashboard, we were able to implement visual representations of our charts, and all of the other components that we planned on showcasing to the user. For the over under signaler component, we have completed the quote component, where the user will be able to see a quote that will help them become more inspired to make better financial decisions. This feature is properly connected to the database and is completed.

## Items that are partially complete

For most of the application many of the user interface portions are complete. However all of the data that is shown throughout the components are not complete and are placeholders representing the data that will be displayed once we have a more established backend. A feature that is still ongoing that we have not fully completed yet is that of the light/dark mode feature. For the most part, the functionality of it works, however there are still components such as some of the charts that do fully change all of the colors whenever the dark mode has been toggled. Another design feature that is partially complete is that of the fonts. We have some different fonts that we have implemented, however we have not settled on the exact font choices as of right now. A lot of the data for the application is currently being worked on and being passed to the database. For example, passing data from the register form to the database is currently being worked, and financial settings from the dashboard to the database.

## What still needs completed

Since the initial proposal, development of the Froogal application has made substantial progress. However, there are some key features that still must be implemented and components that must be adapted to fit requirements. Starting with the app’s UI design, the following should happen to meet requirements: create a modal component that displays a list of items which are either expense or receipt items, modify the receipt item tiles’ modal bodies to render correct receipt data, continue working on color scheme values for dark and light mode, and redesign the receipt item’s list to show an icon indicating that an item can be added. If time permits, the app’s responsiveness should be polished for mobile and tablet-sized devices. Regarding functionality, the following should be implemented to meet requirements: state should be correctly handled for finance and user settings drawers and all modals which change user data (e.g. BudgetWatcher’s modal, CategoryBreakdownChart’s modal, etc.), watchers should be updated to pull in user data, charts should be updated to reflect user data, and a creation modal should be added to receipt list which enables creating items for a receipt. If time permits, finance settings should have buttons that display modals of scrollable list of expenses or items respective of a button.

## Issues that are unresolved

The main issue that is unresolved is state management per component basis. This is critical in displaying and updating the data for a component. Moreover, proper state management is needed to correctly structure data to send to the Firestore DB. Having correctly formatted JSON is important so that it can easily be pulled from the DB when other components need to reflect data.

# Will these issues be major or minor to the success of the project?

The issues above will be highly important to the success of the application because without properly handling state, the data which is used for components could potentially be stored, retrieved, and/or formatted incorrectly. Thus, leading to incorrect data being shown on dashboard components which defeats the whole purpose of the application.