Froogal: Project Design Document

By: Chris Adamson, Dylan Commean, & Johnathan Dickson

# 

[**Project Name**](#_j8xne9psqgck) **3**

[**Team Member Names**](#_xaicazl1ndqz) **3**

[**Updated Requirements List**](#_1iqbptc9pmnh) **3**

[(Onboarding) Sign-in Form:](#_m3t3gl2fh4g5) 3

[(Onboarding) Registration Form:](#_asnihytf98sa) 4

[(General) Dialog Box:](#_s5zome3xnjln) 6

[(General) Modal:](#_4w8c0aa4bnl) 6

[(Dashboard) Navbar:](#_9by67o6688ab) 6

[(Dashboard) Total Spending Tracker Component:](#_9immri3x3vy5) 8

[(Dashboard) Budget Watcher Component:](#_1w9kx23bauq7) 8

[(Dashboard) Over/Under Signaler Component:](#_3rlsx37w9x7t) 8

[(Dashboard) Category Breakdown Chart:](#_no03f46eokyl) 9

[(Dashboard) Expense Watcher List:](#_96xtp6an5qlz) 9

[(Dashboard) Receipt HUB:](#_f1s7d0kzvi2n) 10

[(Dashboard) Recent Receipts List:](#_t92dh2gi9kcq) 11

[(Dashboard) Budget Comparer:](#_2ciwl4bb249r) 11

[**Design Description**](#_pj94jzduqthu) **11**

[**Appendix**](#_tr92adrvqs2z) **16**

[Figure 1: Team Member Names](#_r8es51v8eu2n) 16

[Figure 2: Storyboard Sign In/Registration](#_dvv6ztpw9wrr) 17

[Figure 3: Dashboard](#_cgbz51qp9yef) 18

[Figure 4: Block Diagram](#_ci9qjtig8brj) 19

[Figure 5: Component Diagram (Sign In/ Registration Component)](#_2tr4gsgnkqss) 20

[Figure 6: Component Diagram (Dashboard to Navbar)](#_3h4t8ejsj4l) 20

[Figure 7: Component Diagram (Dashboard to Over/Under Signaler, Budget Watcher, and Total Spending Tracker)](#_f8odxsd3hpc6) 21

[Figure 8: Component Diagram (Dashboard to Budget Comparer and Category Breakdown)](#_9iph29g6vm9y) 21

[Figure 10: Component Diagram (Dashboard to Recent Receipts List, Expense Watcher, Receipt HUB)](#_y6hdmz3563oj) 22

[Figure 11: Component Diagram (Modals)](#_ambvz4aqxzsu) 22

[Figure 12: Entity Relationship Diagram (ERD)](#_1t2drzwq951d) 23

[Figure 13: Site Routing](#_llv0dq57co4b) 24

# Project Name

Being a finance-centric web application, primarily focused on managing personal finances, the word frugal came to mind. Playing off this keyword, we decided on the name of **Froogal** for our application.

# Team Member Names

See Figure 1 in the appendix for a detailed description.

# Updated Requirements List

## (Onboarding) Sign-in Form:

1.1: The sign-in form will be one component, with the following:

1.1.1: A label for an email address

1.1.2: An input box for the email label

1.1.2.1: The input box will have a placeholder of email@gmail.com

1.1.2.2: The input box will accept only valid emails

1.1.2.2.1: Emails must contain an @ symbol

1.1.2.2.2: Validation is to be done using HTML input attributes

1.1.2.2.3: On invalid email, the user is notified on the field.

1.1.3: A label for the password

1.1.4: An input box connected to the last password label

1.1.4.1: The input box will not have placeholder text

1.1.4.2: The input box will accept only valid emails

1.1.4.2.1: Passwords must be at least 6 characters long

1.1.4.2.2: Passwords must contain a special character (!, \_, or ?)

1.1.4.2.3: Passwords must contain one capital letter

1.1.5: Header text saying Sign in With Email & Password

1.1.6: Subheader text saying Sign in With Other Methods

1.1.7: A button to sign in using a Google account

1.1.7.1: On click, will trigger Google account authentication via firebase

1.1.8: A button labeled “Sign In”

1.1.8.1: On click, will query the database for matching user credentials

1.1.8.1.1: If a match is found, the user is taken to dashboard

1.1.8.1.2: If no match is found, a display box is shown to the user saying “Could not find the email or password, try again.”

1.2: The form should adhere to accessibility best practices

1.2.1: All labels should be keyboard navigable

1.2.2: All labels should target their connected input boxes when clicked

1.2.3: All labels should have aria text

1.2.4: All input boxes should be keyboard navigable

1.2.5: All input boxes should have placeholder text unless it is a password field

1.2.6: All input boxes should have aria text

## (Onboarding) Registration Form:

2.1: The sign-in form will be one component, with the following:

2.1.1: Header text of “Sign Up Now”

2.1.2: A label for First Name

2.1.3: An input box connected to the First Name label

2.1.3.1: The input box will have a placeholder value of “John”

2.1.3.2: The input box will only accept characters

2.1.3.2.1: On invalid input, user is notified that this field can only contain characters through the dialog box

2.1.3.3: The input box will be marked as required, denoted by \*

2.1.4: A label for Last Name

2.1.5: An input box connected to the Last Name label

2.1.5.1: The input box will have a placeholder value of “Doe”

2.1.5.2: The input box will only accept characters

2.1.5.2.1: On invalid input, user is notified that this field can only contain characters through the dialog box

2.1.5.3: The input box will be marked as required, denoted by \*

2.1.6: A label for email

2.1.7: An input box for the Email label

2.1.7.1: The input box will have a placeholder value of “123@gmail.com”

2.1.7.2: The input box will accept only valid emails

2.1.7.2.1: Emails must contain an @ symbol

2.1.7.2.2: Validation is to be done using HTML input attributes

2.1.7.2.3: On invalid email, user is notified on the field.

2.1.7.3: The input box will be marked as required, denoted by \*

2.1.7: A label for Password

2.1.8: An input box connected to the Password label

2.1.8.1: The input box will not have a placeholder value

2.1.8.2: The input box will accept only valid passwords

2.1.8.2.1: Passwords must be at least 6 characters long

2.1.8.2.2: Passwords must contain a special character (!, \_, or ?)

2.1.8.2.3: Passwords must contain one capital letter

2.1.8.3: The input box will be marked as a required field, denoted by \*

2.1.9: Normal text describing valid password structure

2.1.10: A label for Verify Password

2.1.11: An input box for the Verify Password

2.1.11.1: The input box will not have any placeholder text

2.1.11.2: The input box will hide the user’s input behind \* characters for each letter entered

2.1.11.3: The input box will accept only valid passwords

2.1.11.3.1: Passwords must be at least 6 characters long

2.1.11.3.2: Passwords must contain a special character (!, \_, or ?)

2.1.11.3.3: Passwords must contain one capital letter

2.1.11.4: The input box will be marked as a required field, denoted by \*

2.1.12: A button labeled “Create Profile”

2.1.12.1: On click, the email is checked to make sure an account with this email is not already created

2.1.12.1.1: If existing email, there will be a status message via a display box.

2.1.12.1.1.1: The message will say, “Email already taken”

2.1.12.1.2: If the email is open, no message is displayed

2.1.12.2: On click, the two password fields will be checked to see if they are equal.

2.1.12.3: There will be a password status message that displays the status of the password fields

2.1.12.3.1: The default status is blank.

2.1.12.3.2: If the two password fields do not match, the password status display will be set to “The passwords that were entered do not match”

2.1.12.3.3: If the two password fields match, the password status display will be set to “The passwords that were entered match.”

2.1.12.4: If any required fields are empty, a dialog box will open to the

Screen.

2.1.12.4.1: This dialog box will contain the message, “Please enter all required material”.

2.1.12.5: If there are no status messages and all the fields are filled out, the information is submitted to the back-end via JSON

2.1.12.5.1: The information will then be stored into Firebase

2.1.12.5.1.1: Data stored under User in Firebase

2.1.12.5.1.1.1: Email

2.1.12.5.1.1.2: Password

2.1.12.5.1.1.3: First Name

2.1.12.5.1.1.4: Last Name

2.1.12.6: The user is logged into their account

2.1.12.7: The application navigates to the dashboard.

## (General) Dialog Box:

3.1: The dialog box will have a button labeled “OK” which will close the dialog box.

3.2: Will display a message depending on which error is triggered

## (General) Modal:

4.1: All Modals will have an ‘X’ Button that will close the modal

## (Dashboard) Navbar:

5.1: There will be a place to include a user avatar/profile picture.

5.1.1: A user’s active avatar is displayed, set based on the user’s settings

5.1.2: A default avatar is displayed if the user has no active avatar in their settings.

5.2: There will be a link called Your Finances

5.2.1: On click, this will take the user to the “Your Finances” section of the dashboard.

5.2.2: This section will display the current user’s financial information, including 5.2.2.1: Current budget amounts for week and month periods

5.2.2.1.1: Budget amount is pulled from Firebase

5.2.2.2: Current net spending for week and month periods

5.2.2.2.1: Current net spending amount is pulled from Firebase

5.2.2.3: Current all recurring expenses (subscriptions, bills, rent, any Item marked as recurring)

5.2.2.3.1: Recurring expenses are pulled from Firebase

5.2.2.4: Show all created custom categories

5.2.2.4.1: Custom categories are pulled from Firebase

5.2.3: This section will allow the current user to update their financial information and subsequently store their financial information into firebase, including

5.2.3.1: Setting their annual income

5.2.3.2: Setting their annual budget amount

5.2.3.3: Setting their monthly income

5.2.3.4: Setting their monthly budget amount

5.2.3.5: Setting their weekly income

5.2.3.6: Setting their weekly budget amount

5.2.3.7: Ability to remove a custom category

5.2.3.8: Ability to add a custom category

5.2.3.9: Ability to remove a recurring expense

5.2.3.10: Ability to create a recurring expense

5.2.3.11: Ability to modify a recurring expense

5.3: There will be a link called Profile Settings

5.3.1: On click, this will render the Profile Settings section of the dashboard

5.3.2: This section will display the user’s current profile settings, including:

5.3.2.1: Their current first name

5.3.2.1.1: This field should not be editable

5.3.2.1.2: This field is pulled from Firebase

5.3.2.2: Their current last name

5.3.2.2.1: This field should not be editable

5.3.2.2.2: This field is pulled from Firebase

5.3.2.3: Their current email

5.3.2.3.1: This field is pulled from Firebase

5.3.2.4: Their current avatar resource

5.3.2.4.1: This field is pulled from Firebase

5.3.3: A button should be displayed with the label “Adjust Info”

5.3.3.1: On click, a modal will appear on screen.

5.3.3.1.1: The modal should have a close button denoted by X

5.3.3.1.2: The modal will have an Email tab

5.3.3.1.2.1: On click, displays an emailview, displaying

5.3.3.1.2.1.1: A label for Old Email

5.3.3.1.2.1.2: The current email

5.3.3.1.2.1.3: A label for New Email

5.3.3.1.2.1.4: An input box for the new email

5.3.3.1.2.1.5: The input box should validate emails

5.3.3.1.2.1.6: A button to set old email to new email

5.3.3.1.2.1.6.1: As long as all info is valid, the user’s email will be updated in Firebase

5.3.3.1.3: The modal will have an Avatar tab

5.3.3.1.3.1: On click, display an avatar view, displaying

5.3.3.1.3.1.1: A label for Old Avatar

5.3.3.1.3.1.2: The current source of avatar pic

5.3.3.1.3.1.3: A label for New Avatar

5.3.3.1.3.1.4: An image uploader to receive new src

5.3.3.1.3.1.4.1: If the image is valid it will then be updated in Firebase

## (Dashboard) Total Spending Tracker Component:

6.1: This component will be card-like and it will show the following information:

6.1.1: The user’s current total spending amount for the time period

6.1.1.1: Current total spending is pulled from Firebase

6.1.2: Text display of the current time period (e.g. month, week, year)

6.1.2.1: Current time period is pulled from Firebase

## (Dashboard) Budget Watcher Component:

7.1: This component will be card-like and it will show the following information:

7.1.1: Current set budget amount for the time period

7.1.1.1: Current budget amount is pulled from Firebase

7.1.2: Text display of the current time period (e.g. month, week, year)

7.1.2.1: Current time period is pulled from Firebase

7.2: There will be a button labeled “Adjust Budget”

7.2.1: On click, a modal will open & contain

7.2.1.1: A label for the New Budget Amount

7.2.1.2: An input box connected to the new budget amount label

7.2.1.3: A button labeled Submit

7.2.1.3.1: On click, the button will

7.2.1.3.1.1: Change the budget amount in the user object

7.2.1.3.1.1.1: The user object will be updated in Firebase as well

7.2.1.3.1.2: Display the updated budget amount in the component

## (Dashboard) Over/Under Signaler Component:

8.1: There will be a message that displays if the user is over or under budget.

8.1.1: If the amount is over, the message will display “Over”.

8.1.2: If the amount is under, the message will display “Under”.

8.1.3: Budget will be pulled from the user object from Firebase

8.2: There will be an inspirational message that displays from famous financial persons of interest.

8.2.1: The message will be random  
 8.2.2: The message will change when the user refreshes the page.

## (Dashboard) Category Breakdown Chart:

9.1: The chart will display predefined categories for a user’s spending

9.1.1: The predefined categories will include:

9.1.1.1: Food

9.1.1.2: Entertainment

9.1.1.3: Utility

9.1.1.4: Transportation

9.1.1.5: Rent

9.1.1.6: Miscellaneous (“misc”)

9.2: The component should have a button that labeled “Add Category”

9.2.1: On click, the modal should contain

9.2.1.1: An exit button denoted by X

9.2.1.2: A label for Category To Add

9.2.1.3: An input box that accepts characters

9.2.1.3.1: The input box should be validated and throw an error if non-characters are entered

9.2.1.4: A button to submit the custom category

9.3: The chart should reflect custom categories as well

9.4: Each category breakdown should include:

9.4.1: The total amount spent for that category during the period

9.4.2: A percentage of how much that category represents in net total spending for the period

## (Dashboard) Expense Watcher List:

10.1: This component will be a list of recurring expense blocks

10.1.1: Each expense block will contain

10.1.1.1: Expense Name

10.1.1.2: Expense Due Date

10.1.1.3: Expense Amount Due

10.1.1.4: Expense Category (bill or subscription)

10.2: The list should be sorted by the closest due date by default

10.2.2.1: Expenses can be sorted by the following values as well

10.2.2.1.1: Expense Due Date

10.2.2.1.2: Expense Name

10.2.2.1.3: Expense Amount Due

10.2.2.1.4: Expense Category

10.3: Expenses should be removed at midnight of the due date, this should be automatic

10.4: The user should be notified on the dashboard via a Toast Box that their expense is due soon. An expense is considered “Due Soon” two days before the expense’s Due Date

## (Dashboard) Receipt HUB:

11.1: The receipt will show a selected receipt.

11.1.1: The selected receipt is defaulted to the most recent user receipt

11.1.2: The selected receipt will contain the following:

11.1.2.1: Transaction date

11.1.2.2: Total price

11.1.2.3: Each item and its total price

11.2: The receipt hub will have a button labeled Delete Receipt

11.2.1: On click, a confirmation modal will show with options for delete or cancel

11.2.1.1: If delete is selected, current selected receipt is moved and the next current receipt is chosen

11.2.2:

11.3: The receipt hub will have a button labeled Create Receipt

11.3.1: On click, a ReceiptCreationModal is opened

11.3.1.1: The modal will have a label for company name

11.3.1.2: There will be an input box connected to company name

11.3.1.3: There will be a section to add Item Objects to the receipt

11.3.1.4: There will be a button labeled “Add Item”

11.3.1.4.1: On click, brings up ItemCreationModal

11.3.1.4.1.1: The modal will ask for item name

11.3.1.4.1.2: The modal will ask for item quantity

11.3.1.4.1.3: The modal will show a total price

11.3.1.4.2: After filling out the ItemCreationModal, the item is added to the receipt

11.3.1.5: There will be an X button that will remove items from the receipt list

11.3.1.6: The total of all items will be represented as the receipts Total Price

11.3.1.7: There will be a button labeled “Submit Receipt”

11.3.1.7.1: On click, the receipt object is added to the user’s receiptCollection prop

11.4: There receipt hub will have a button labeled Search Receipts

11.4.1: On click, this will bring up a SearchModal

11.4.1.1: The modal will have a search bar that can search receipts by

11.4.1.1.1: Receipt company name

11.4.1.1.2: Receipt transaction date

11.4.1.1.3: Number of items on the receipt

11.4.1.1.4: The total price of the receipt

11.4.1.1.5: A specific item on the receipt

11.4.1.2: Search results will be given as a list in the modal.

11.4.1.3. There will be a button called “Select Receipt”

11.4.2.1: On click, this will bring the highlighted receipt into the ReceiptView on the Receipt Hub

## (Dashboard) Recent Receipts List:

12.1: This component will be a list of cards, each displaying brief information about a receipt, including

12.1.1: The title of a receipt

12.1.2: The transaction date of a receipt

12.1.3: Number of items in a receipt

12.1.4: The transaction location

12.1.4.1: If no transaction location is set, no value is displayed

12.1.5: The total price of a receipt

12.2: Clicking a card will open a modal with the receipt’s information.

12.2.1.: Information the modal should show is:

12.2.1.1: Full item breakdown

12.2.1.1.1: Item quantity

12.2.1.1.2: Item name

12.2.1.1.3: item total price

12.2.1.2: The receipt’s id

12.2.1.3: The receipt’s location if any.

12.2.1.3.1: If no location is set, display no location on the receipt

12.2.1.4: The receipt’s transaction date

12.2.1.5: Total price of the receipt

## (Dashboard) Budget Comparer:

13.1: There will be a graph with three different data points displayed

13.1.1: The data points shown on the graph will be the following

13.1.1.1: Last period net spending

13.1.1.2: Current period spending

13.1.1.3: Current budget amount

# Design Description

The user will first be greeted by a sign in/registration page. See Figure 5 for a diagram of the Sign in/Registration page. Through this page, the user will have the option to sign in or register for a new account. If they do not already have an account with Froogal, they would begin to enter information within the registration component of the front page. On the registration component, the user will be able to enter information such as their first name, last name, email, and password. This information will be passed through a regular expression test to examine their initial validity. After passing the regular expression validity test and pressing the submit button, the user’s information is sent to the backend for email validation. Email validation requires that the email does not already exist within the database. As long as all the information is valid, the user’s information will then be entered into the database. The information stored in the database is the user’s first name, last name, email, and password. If any of the information is invalid after the regular expression validity, the user will be greeted by a dialog box that will notify them that one or more pieces of their information is not valid. If the email validation fails in the backend, the dialog box will contain, “Email is already taken. Please enter another email.” When the dialog box is escaped either by clicking the “OK” button on every dialog box or clicking outside the boundaries of the box, the user will need to try again with a different set of information. See Figure 5 for a diagram of the Dialog Box component.

The user also has the opportunity to sign in from this page as well. See Figure 5 for the diagram of the Sign-In component. Similar to the registration form, the sign-in component will require the user to enter their email and password into the form. This email and password are sent to the backend for authentication. Should the information be correct, it will lead them into the dashboard. If the information fails authentication, it will lead them to another dialog box informing them, “Email or password is incorrect. Please enter the correct information”. The dialog must be escaped to return to the login page. On successful registration to the site or upon successful sign-in, the user will be transported to the Dashboard and automatically signed into their new account. The user will then be transferred to the Froogal Dashboard and have a wide variety of options at their disposal.

The dashboard page is the main page in which the user can view, access, and adjust various pieces of information. See Figure 6 - 12 for a diagram of the Dashboard page.If the user is creating their account for the first time, they will be transferred to an onboarding page where they will be able to enter financial information which is necessary for every component of the application to function properly. Users are then able to access many features of the application. One of the main features that the users will have access to is the navigation bar which will hold links to profile settings and financial settings. Upon clicking the profile settings link, the user will now have access to adjust and alter different features that are related to their profile. See Figure 6 for a diagram of the Profile Settings component. These features include the ability to change email, password, and their avatar’s profile picture within the application. The profile settings will also let the user view their account’s first name and last name as well. Whenever a user wants to change their email or password, they must input the correct information in the labeled text boxes. If this information is validated through the backend, then the user’s email or password will be updated accordingly within the database. The user can now login with the newly changed email and/or password. The user also has the ability to click using the same navigation bar as before to access the financial settings. See Figure 6 for a diagram of the Financial Settings component. This page will have features related to altering information about the user’s financial settings. Features include the ability to view current budget amounts, net spending for the week and month, current expenses, and all custom categories. This page will allow the user to alter and adjust many financial settings as well such as setting their annual, monthly, and weekly budget and income amount. They will also see forms that will allow them to create, delete, and add custom categories. Similar to the user’s profile settings, if the information is validated then the user’s information will be updated in the database as well. After the user is done viewing or changing many of these settings, a user can simply close the modals and will be returned to the main dashboard where their updated information will be present.

Through the dashboard, the user will be able to see a multitude of different components and widgets where the user can view and interact with their financial data. One of the main components that the users can interact with is the receipt hub. See Figure 10 for a diagram of the Receipt Hub.Through the receipt hub, the user has a few options at their disposal. The user is able to access four buttons which gives them the ability to create, modify, view, and delete receipts. When clicking on the “Create Receipt” button, the user will be presented with the receipt creation modal that has input boxes where the user can insert information to create a receipt. The information that the user will be able to input is the company name and the items on the receipt. Whenever a user wants to enter an item, they will have to click on the add item button which will lead them to an item creation modal. See Figure 11 for a diagram of the item creation modal.In the item creation model, the user is able to add an item to their receipt. The information that the user can enter to create an item is the name, quantity, and total price. Once the user has entered the required information to create an item, they can click the “Create Item” button which will exit out of the modal and return them to their original receipt creation modal. The user can create as many items as needed when inputting their receipt. After the user is finished creating their receipt, they will need to click the “Create Receipt” button, which will then lead them back to the receipt hub. They can now view the receipt that they have just created. Another feature that a user can access through the receipt hub is the delete receipt button. This button when clicked on allows the user to delete a receipt. Upon clicking, a deletion modal appears and requires the user to confirm that they want to delete the currently selected receipt. If the user wishes to delete the receipt, then they are able to, by clicking the “Delete” button. The user can also change the selected receipt by clicking the “Search” button. Upon clicking the search button, the user can change the selected receipt by searching through all the receipts using different criteria. The users can search for a receipt by the receipt’s name, transaction date, number of items, total amount, and total price. After the user has a receipt, the user can access more information and views in the other components are modified given the new receipt information on the dashboard.

Another feature that users will have access to is the recent receipt list. See Figure 10 for a diagram of the recent receipt list.The user can view their most recent receipts in a card-like structure. The information that the users will be able to see in the receipt cards is the title, transaction date, number of items in the receipt, and transaction location. The user has the ability to click on the cards which pops up a detailed description modal. See Figure 11 for a diagram of the detailed description modal.Through this modal, the user sees more in-depth information about the receipt. Information includes that of the receipt’s full item breakdown which is the item’s name, price, and quantity while showing the receipt’s name, location, and total price.

Another component of the dashboard the user has access to is the expense watcher list. See Figure 10 for a diagram of the expense watcher list.The expense watcher list will contain a list of expenses. Users will have the capability to view their most recent expenses. The user can view the expense’s name, due date, amount due, and the category of the expense. The user will also be able to sort through the list of expenses by those categories as well. The users will also have the ability to “pay” the expense by clicking on the expense and clicking the “paid” button. The expenses will also showcase a toast message notification on the dashboard that will notify the user two days before the expense is due. This will let the user know that they have an upcoming expense that needs to be paid. The expense will be eliminated from the list the day after the expense was supposed to be due regardless of whether it was paid or unpaid.

Users will have access to a component on the dashboard called the budget comparer. See Figure 8 for a diagram of the budget comparer. The budget comparer is a data visualization of the user's last period spending amount, the current period’s spending amount, and the current budget for the user. In this component, the user will not be able to adjust or click on it as this component is strictly for viewing purposes. Another component the user will view is the category breakdown chart. See Figure 8 for a diagram of the category breakdown chart.

The category breakdown chart will allow the user to see the amount that they are spending per category. The different categories that the users will be able to view based on their spending breakdown is food, entertainment, utility, transportation, rent, and miscellaneous. The users will also see their custom categories on the chart as well. There will be a button that the users will be able to click which is called “Add a Custom Category”. This allows the user to add a category, which they would be able to use later whenever creating a receipt. Whenever the user clicks on that button, a modal will pop up allowing the user to create that custom category. This modal will simply be an input box where the user will enter the name of the category, and then be able to submit it.

Another component that the user will want to view on the dashboard is the total spending tracker. See Figure 7 for a diagram of the total spending tracker. The total spending tracker is a tracker that will showcase the user’s total spending for the period. The total spending tracker will also show the current period that is being tracked. This component is in a card-like structure and is not able to be clicked. This component is used strictly for viewing purposes only.

On the dashboard, the user will also be able to view components such as the over/under signaler component. See Figure 7 for the diagram of the over/under signaler component. Users can view a component called the over/under signaler component. This component will display “Over” if the user is over their budget amount for the period. The component can also display “Under” if the user is under their budget amount for the period. Another portion of this component is that the user will be able to see an inspirational quote that will help the user to keep doing well in their budgeting endeavors. This component is strictly a viewing component and will not have any other instances where the user can interact with it.

Users can however interact with the budget watcher component which they can access on the dashboard as well. See Figure 7 for the diagram of the budget watcher component. The budget watcher is a component that the user can use to see the current budget for the current period. Users can also use this component to set the budget for the given period. Users can view the component on the dashboard as it will display the user’s current period and budget. There will be a button called “Adjust Budget” that upon clicking, the user will be brought to a new modal that will allow them to enter a new numerical value to set as the new budget for the current period. This allows the user to be able to not only see their budget on the dashboard but be able to easily change it.

# 

# Appendix

## Figure 1: Team Member Names

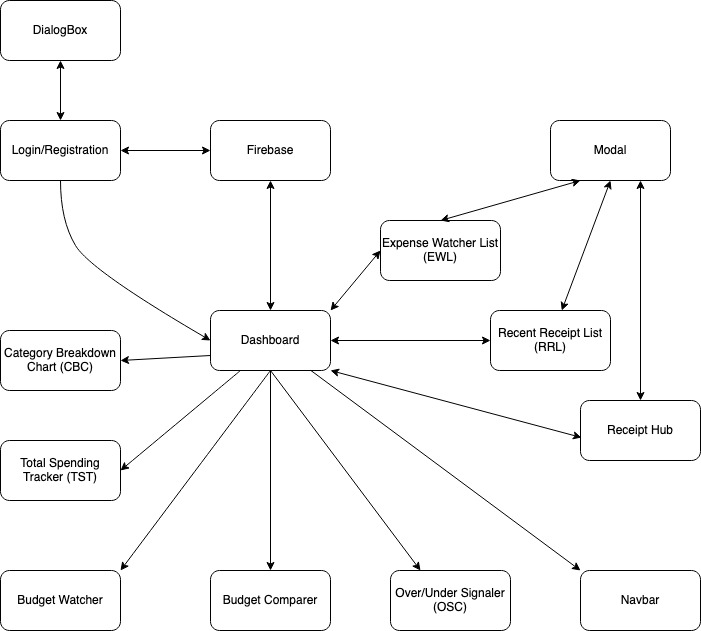
| Name | Role |
| --- | --- |
| Dylan Commean | Developer, Architect |
| Chris Adamson | Developer, Architect |
| Johnathan Dickson | Developer, Architect |

## Figure 2: Storyboard Sign In/Registration

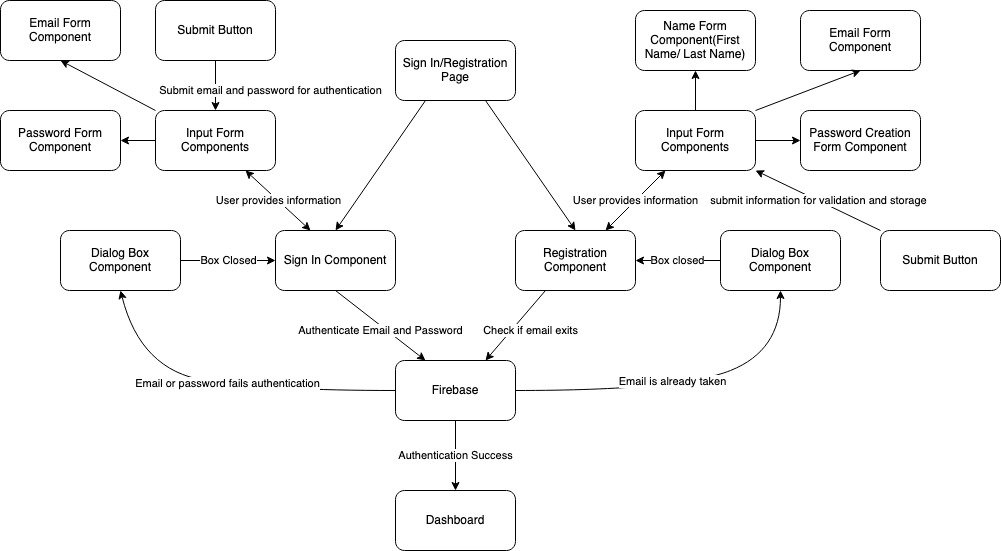
## Figure 3: Dashboard

## 

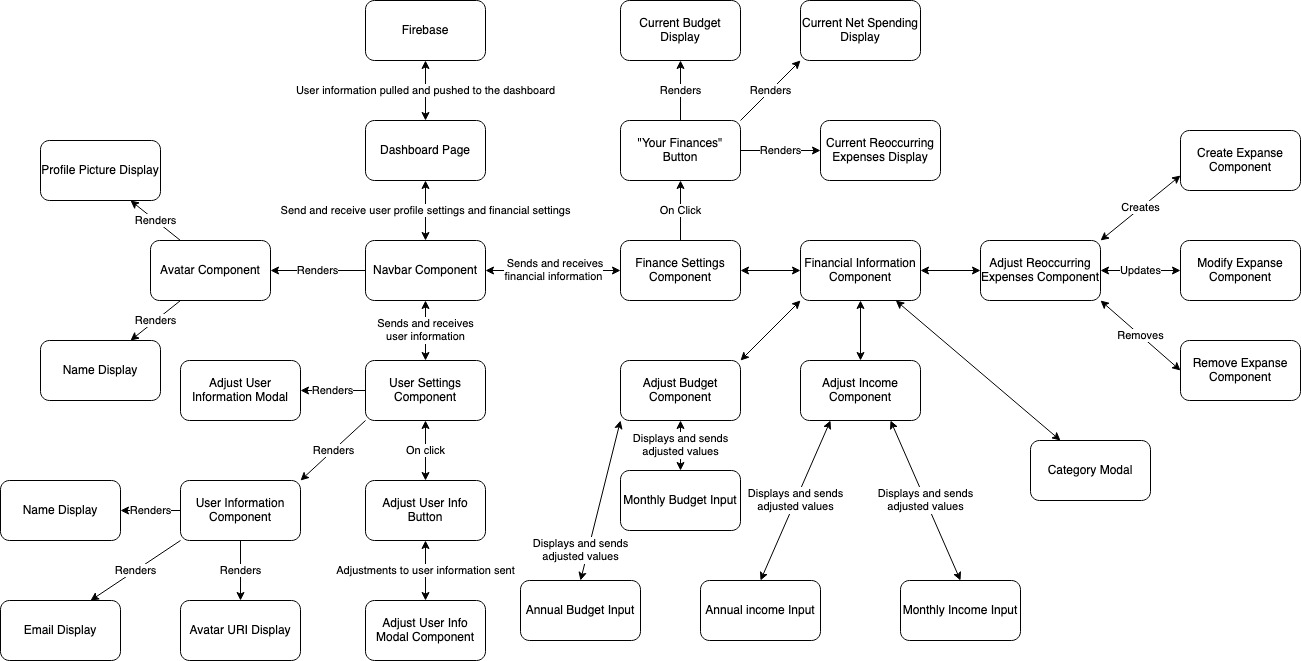
## Figure 4: Block Diagram



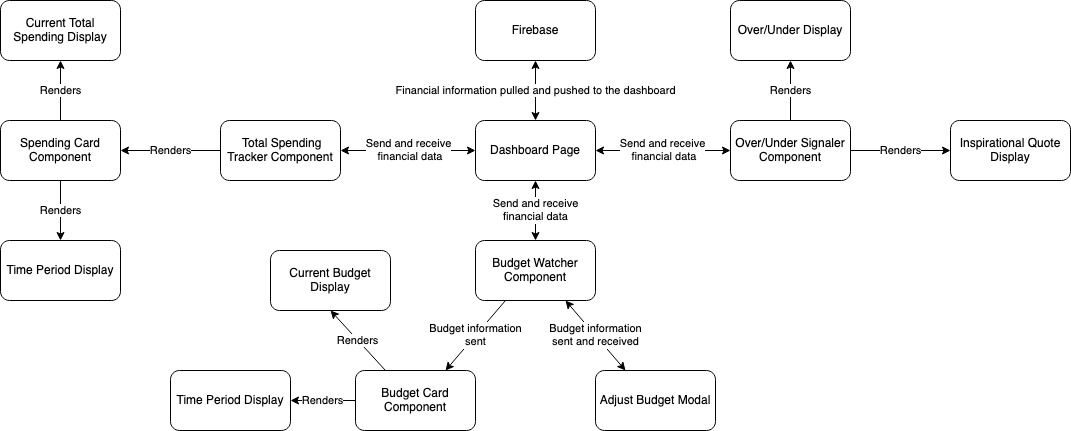
## Figure 5: Component Diagram (Sign In/ Registration Component)



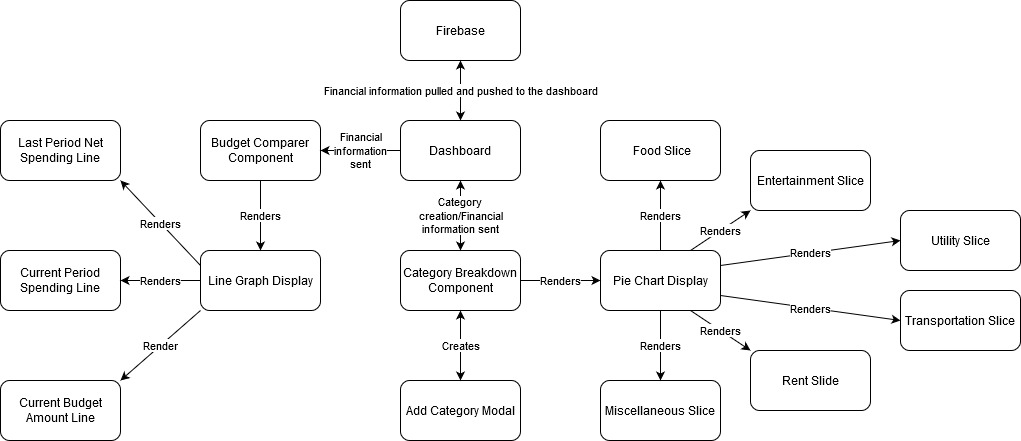
## Figure 6: Component Diagram (Dashboard to Navbar)



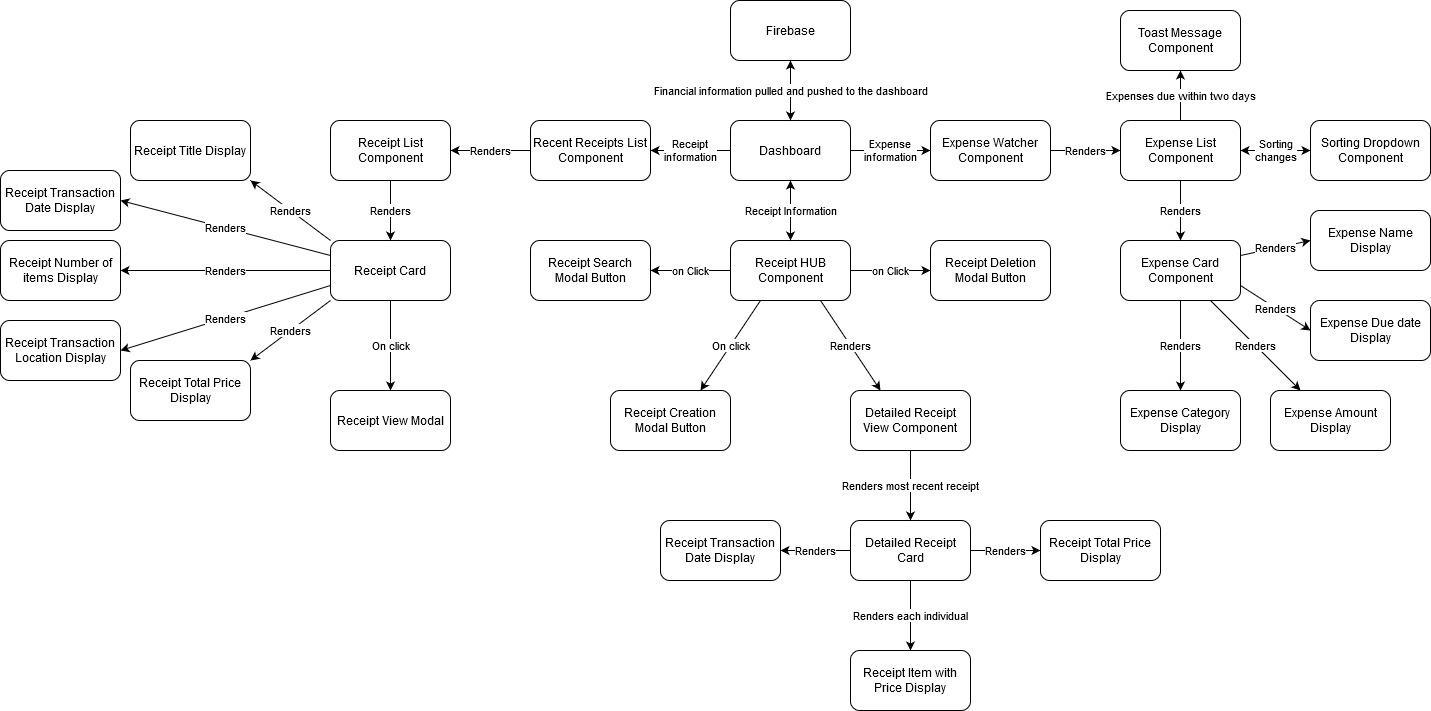
## Figure 7: Component Diagram (Dashboard to Over/Under Signaler, Budget Watcher, and Total Spending Tracker)



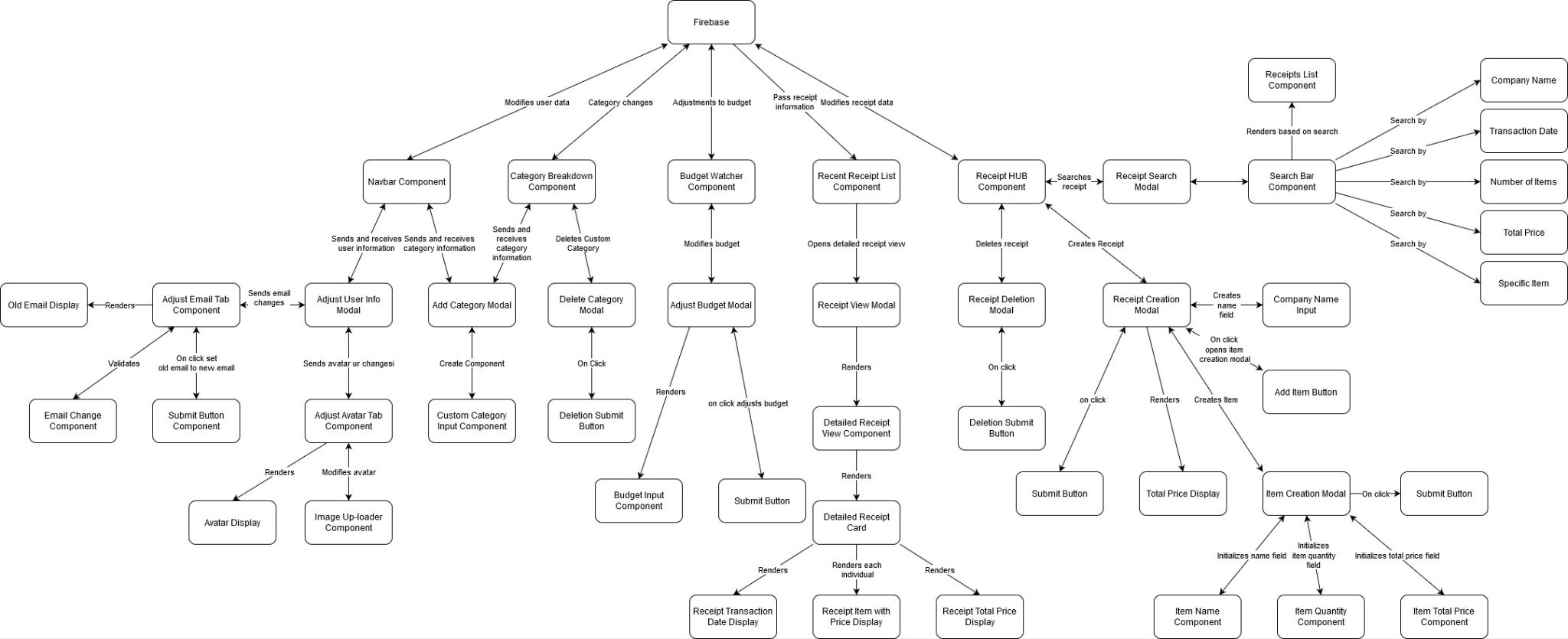
## Figure 8: Component Diagram (Dashboard to Budget Comparer and Category Breakdown)



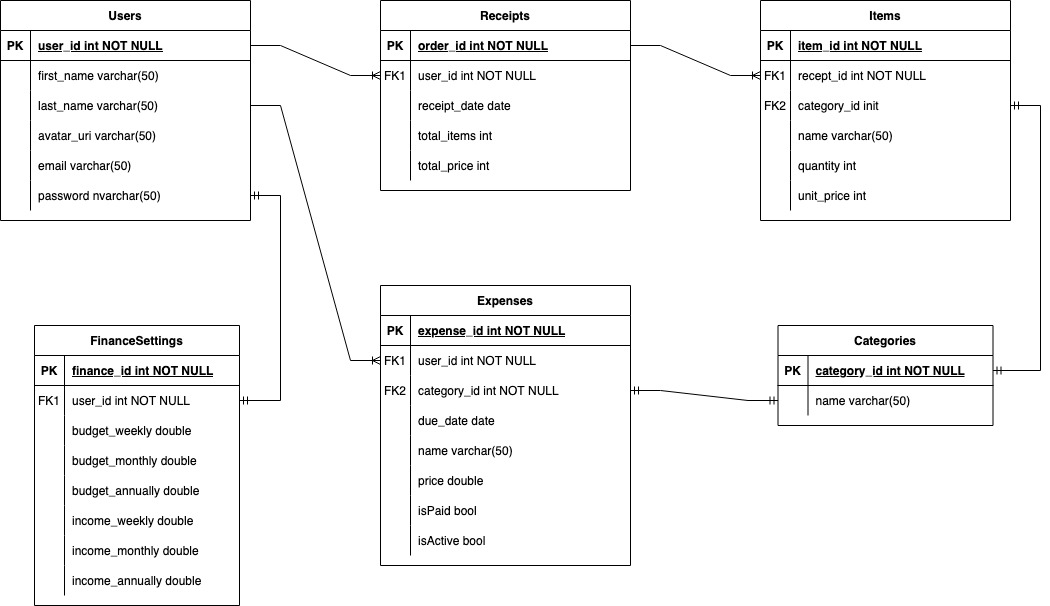
## Figure 10: Component Diagram (Dashboard to Recent Receipts List, Expense Watcher, Receipt HUB)



## Figure 11: Component Diagram (Modals)



## Figure 12: Entity Relationship Diagram (ERD)



## Figure 13: Site Routing

