# Froogal: Requirements Specification

By: Dylan Commean, Chris Adamson, & Johnathan Dickson

# Table of Contents

## 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 detailed description.

## Abstract

Financial literacy is the cornerstone of wealth building. Budgeting is a powerful tool in one's financial journey; however, it alone is not sufficient. An analysis on one's spending habits is a key metric that reflects who the consumer is at their core. The process of analyzing spending can be a daunting task; where does one even start? Froogal aims to streamline this process by providing rich financial reports about one's spending, over various periods (day, weeks, months, and years), so our users can have a deeper understanding of their spending habits. This gives our users the upperhand in controlling their personal finances, thus combatting overspending.

## Tools & Technologies

See Figure 2 in the appendix for a detailed description.

## Requirements List

### 1. (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, 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, 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

## 2. (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 not 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: Message will say, "Email already taken"
  - 2.1.12.1.2: If 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

2.1.12.6: The user is logged into their account

2.1.12.7: The application navigates to the dashboard.

## 3. (General) Dialog Box:

3.1: 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

## 4. (General) Modal:

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

## 5. (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 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

5.2.2.2: Current net spending for week and month periods

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

5.2.2.4: Show all created custom categories

- 5.2.3: This section will allow the current user to update their financial information, 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
  - 5.3.2.2: Their current last name
    - 5.3.2.2.1: This field should not be editable
  - 5.3.2.3: Their current email
  - 5.3.2.4: Their current avatar resource
- 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
    - 5.3.3.1.3: 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

## 6. (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
- 6.1.2: Text display of the current time period (e.g. month, week, year)

## 7. (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
- 7.1.2: Text display of the current time period (e.g. month, week, year)

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
      - 7.2.1.3.1.2: Display the updated budget amount in the component

## 8. (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”.

8.2: 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.

## 9. (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 a 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

## 10. (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 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

## 11. (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

11.3: 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 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

## 12. (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 set, display no location on receipt
12.2.1.4: The receipt's transaction date
12.2.1.5: Total price of the receipt

### 13. (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

## Updated Timeline

See Figure 3 in the appendix for a detailed description.

## Appendix

**Figure 1**: Team Member Names

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

**Figure 2**: Froogal Tech-Stack

| Place In System | Dependency Name | Dependency Type | Why |
|---|---|---|---|
| Frontend | React | JS Library | Implement component based, reactive UI's to configure pages based on user's state. |
| Frontend | Chakra UI | CSS Framework | Styling React components with a consistent predefined or custom theme |
| Frontend | Framer Motion | Animation Library | Incorporating animations, is a dependency of Chakra UI |
| Frontend | Redux | React Package | Alternative option of reducing the complexity of handling app state for users. If React's useContext hook becomes overwhelming, we will utilize this package. |
| Frontend | Recharts | React Package | Handle data visualization such as creating charts, line graphs, and various other models from user data. |

|  |  |  |  |
|---|---|---|---|
| Backend | Firebase Platform | Cloud Platform | Utilizing the Firebase platform to handle the harder parts of backend development (e.g. user auth, security practices, etc.) |
| Data Source | Cloud Firestore | Database | NoSQL real-time cloud-based database to handle user's information using JavaScript Object Notation (JSON). |

**Figure 3**: Tentative Schedule*

|  |  |
|---|---|
| Week 1: August 21st - 27th | **All Members:**<br>Application proposal, product discovery, determining softwares and languages, assessing team members strengths and abilities, information gathering. |
| Week 2: August 28th - September 3rd | **All Members:**<br>Team setup with the tools for development. GitHub/Trello setup for sprints, research and resolve any issues with hardware or tools. Information gathering. |
| Week 3: September 4th - 10th | ***No Class - Labor Day (5th)***<br><br>**Johnny:**<br>Setup application base in GitHub. Creation of master branch and development branches.<br><br>**Dylan:**<br>Discovering/thinking about the Front-end components. Designing database architecture and back-end structure.<br><br>**Chris:**<br>Information gathering and getting a feel for the languages and stack we are using. |
| Week 4: September 11th - 17th | ***Production officially starts.***<br><br>**Dylan:**<br>Establishing connections from the back-end to database.<br><br>**Johnny & Chris:**<br>Getting a feel for the languages we are using, asking |

| | |
|---|---|
| | questions, and solving any issues that arise. Information gathering. |
| Week 5: September 18th - 24th | **Dylan:**<br>Dialog Box,<br>Server-side components and connections<br>Database materials<br><br>**Johnny:**<br>Sign-in Component<br><br>**Chris:**<br>Registration Component |
| Week 6: September 25th - October 1st | **Dylan:**<br>Navbar,<br>Server-side components and connections<br>Database materials.<br>**Johnny:**<br>Total Spending Tracker<br><br>**Chris:**<br>Over/Under Tracker<br><br>**All Members:**<br>Testing and debugging of initial work alongside programming. |
| Week 7: October 2nd - 8th | **Dylan:**<br>Navbar,<br>Server-side components and connections<br>Database materials.<br><br>**Johnny:**<br>Total Spending Tracker,<br>Budget Watcher<br><br>**Chris:**<br>Recent Receipts List<br><br>**All Members:**<br>Testing and debugging continue. |
| Week 8: October 9th - 15th | ***No Class - Fall Break (10th - 11th)***<br><br>**Dylan:** |

| | |
|---|---|
| | Navbar,<br>Server-side components and connections,<br>Database materials<br><br>**Johnny:**<br>Category Breakdown Component,<br>Receipt Hub<br><br>**Chris:**<br>Budget Comparer<br><br>**All Members:**<br>Assessments are done to determine how the team is doing. Teammates assist in other areas that are falling behind if necessary. |
| Week 9: October 16th - 22nd | **Dylan:**<br>Navbar,<br>CRUD Hookups<br><br>**Johnny:**<br>Category Breakdown,<br>Receipt Hub<br><br>**Chris:**<br>Budget Comparer<br><br>**All Members:**<br>Production, testing, and debugging continues. |
| Week 10: October 23rd - 29th | **Dylan:**<br>CRUD Hookups for various components<br><br>**Johnny:**<br>Receipt Hub<br><br>**Chris:**<br>Budget Comparer<br><br>**All Members:**<br>Production of the base site is nearing completion. Areas needing help  are accessed and completed. |
| Week 11: October 30th - November 5th | **Dylan:**<br>CRUD Hookups for various components |

| | |
|---|---|
| | **All Members:**<br>Base site and base components are ready for testing. Solving any problems that arise with the initial designs and remedying those issues. |
| Week 12: November 6th - 12th | ***No Class - Veterans Day (11th)***<br><br>**All Members:**<br>Optional components are looked over for their viability and ordered by deliverability. As a team, decide which components should and could be implemented in a timely manner. |
| Week 13: November 13th - 19th | **All Members:**<br>Email and Push Notifications<br>Bug fixes and polish work. |
| Week 14: November 20th - 26th | ***No Class - Thanksgiving Holiday (23rd - 27th)***<br><br>**All Members:**<br>Cut off week for the production of optional components. Testing, bug fixes, and polish work. |
| Week 15: November 27th - December 3rd | **All Members:**<br>Finishing touches and polish work. Gear up for presentations the following week. |

*Schedule subject to change