



Team 5 Design Document

Team Members

Sarah Pushparaj, Rosie Ajish, Aditi Barla, Angela Joseph

Table of Contents

Purpose	3
Functional Requirements	4
Non-Functional Requirements	6
 Design Outline	 8
High-Level Overview	8
 Design Issues	 9
Functional Issues	9
Non-Functional Issues	10
 Design Details	 12
Class-Level Diagram	12
Class Descriptions and Interactions	13
Sequence Diagrams	15
Navigation Flow Map	21
UI Mockups	22

Purpose

Recent studies have shown that 90% of college students struggle with effective money management. This highlights the prevalent issue of inadequate financial knowledge and budgeting skills in our generation. To make budgeting less of a daunting task, we introduce Budget Busters, a user-friendly app used to create a personal budget and to foster a sense of financial independence.

In more detail, Budget Busters goes beyond traditional budgeting applications by actively tracking user spending habits and providing data visualizations to monitor their progress towards monthly budget goals. Users can easily set and customize these goals, plan for major spending categories, and input daily purchases with ease, ensuring a high level of accountability in managing their finances. Existing budgeting applications like Mint and RocketMoney offer budgeting for specific scenarios, and it can be overwhelming to navigate for young adults who are just embarking on their personal finance journey. Budget Busters, on the other hand, focuses on providing a daily reinforcement of tracking expenses to ensure that financial awareness sits at the forefront of one's mind. Ultimately, our aim is to empower young adults to make responsible financial decisions in their day-to-day lives, setting them on a path to long-term financial success.

Functional Requirements

- **User account**

1. As a user, I would like to be able to create an account (including name, email address, password, phone number) for Budget Busters using the official website.
2. As a user, I would like to be able to create an account for Budget Busters using Gmail.
3. As a user, I would like to be able to delete my Budget Busters account.
4. As a user, I would like to be able to login to my Budget Busters account.
5. As a user, I would like to be able to reset my password if my account is linked to the official website.

- **User Profile Settings**

6. As a user, I would like to be able to create my profile page with a profile picture, age, and notes section, and phone number (inputted from User Account page).
7. As a user, I would like to be able to edit my profile page with a profile picture, age, notes section, and phone number.
8. As a user, I would like to be able to upload my profile picture from my computer.

- **Home Page**

9. As a user, I would like to view my personalized home page with my monthly goal and spending pie chart.
10. As a user, I would like to access the monthly goal setting page through a button on my home page.
11. As a user, I would like to access the input daily spending page through a button on my home page.
12. As a user, I would like to access the category breakdown page through a button on my home page.
13. As a user, I would like to access the settings through an icon on my home page.

- **Monthly Goal Setting Page**

14. As a user, I would like to be able to create my budget goal at the start of each month.
15. As a user, I would like to be able to add my spending categories at the start of each month by selecting from the provided options.
16. As a user, I would like to be able to add my spending categories by manually typing them out.
17. As a user, I would like to set my monthly goal spending only once for the month.
18. As a user, I would like to add in my categories only once for the month.

- **Inputting Daily Spending Page**

19. As a user, if I have not inputted any of my purchases for the day yet, I will see a default message on the Daily Spending Page stating "You did not spend anything today."
20. As a user, I would like to click on the plus button to manually enter one item, amount, and category for the day.

21. As a user, I would like to select the category from a dropdown menu that is associated with the entered purchase.
22. As a user, I would like to be able to edit my spending amount for a purchase for that day.
23. As a user, I would like to be able to edit the category for each spending amount for that day.
24. As a user, I would like to view my daily purchases.
25. As a user, I would like to only be able to input numerical inputs for the spending amount field.
26. As a user, I would like to be notified with an error message if I try to add a duplicate purchase, so I can verify if I accidentally logged the same purchase twice.

- **Category Breakdown Page**

27. As a user, I would like to view each of my spending categories with their own separate progress bars indicating how it fits into the monthly goal.
28. As a user, I would like to color code each spending category.
29. As a user, I would like to be able to modify the names of my spending categories.

- **Notification Settings**

30. As a user, I would like to be able to select the method of notifications (email and/or text message).
31. As a user, I would like to receive mobile text notifications if I choose the text message option.
32. As a user, I would like to receive email notifications if I choose the email option.
33. As a user, I would like to edit my method of notifications.
34. As a user, I would like to be able to set my daily input spending notification time.
35. As a user, I would like to be able to set a budget limit warning notification via a percentage threshold.
36. As a user, I will receive a notification if I reach the budget limit.
37. As a user, I would like to receive a notification on the 1st day of every month to fill in my monthly goals and spending categories.
38. As a user, if my account is through gmail, I would like to receive a Google Calendar reminder on the 1st day of every month to fill in my monthly goals and spending categories (if time allows).

- **Data visualization**

39. As a user, I would like to view my monthly spending in a pie chart, where it is divided based on spending category.
40. As a user, I would like to view the percentage amount of each category in relation to my budget goal when I hover over each section on the pie chart.
41. As a user, I would like to view my cumulative daily spending trend compared with the budget goal in a line graph.
42. As a user, I would like to view a predictive line graph based on current monthly spending (if time allows).

Non-Functional Requirements

a. Response Time

Budget Busters is a web application with multiple user accounts, so there must be a minimal server lag for enhanced user experience. Some mechanisms to prevent this is to utilize Java's concurrency features, like thread pools or asynchronous processing, to ensure that multiple server requests can be handled concurrently and efficiently. Additionally, we can utilize load balancing to distribute the user traffic among multiple server instances. This would prevent any overloading for a single server and eventually improve response times. We will also be using the Firebase Realtime Database so that the application can handle a large amount of concurrent connections. It also offers real-time data synchronization, implying that any changes to the user data would be immediately shown to the connected clients. This ultimately enhances the response time for users since it reduces the latency for real time updates.

b. Security

The main security concern with Budget Busters is the storing of user details, specifically their login credentials. The login credentials will give them access to their account with all their budgeting goals, which places a high emphasis on security. We will be using Firebase's Authentication System to manage the user credentials and information. Firebase offers robust security features to protect users data by using industry-standard security practices, like using encryption. Additionally, Firebase is a cloud-based platform which implies that it can easily scale to accommodate a growing user base, without changing much of the applications infrastructure. For storing specific information related to the users budgeting goals, we will use Firebase's Database. This database will store the users' specific categories, budgeting goals per category, daily input spendings, and etc.

c. Usability

Budget busters will take the form of a web application. For easy navigation, the home page will display noticeable buttons denoting each of the available pages: profile, monthly goal setting, input daily spending, and category breakdown. Each of these page interfaces will be simple, employing input fields and dropdowns where necessary. Additionally, the home page will display visual cues such as a welcome message with the user's name, the numerated monthly goal, and the spending breakdown pie chart to make the budgeting data easily understandable. Users can also color-code their spending categories for further personalization. Moreover, users can choose how and when they receive notifications by clicking on the settings icon from the home page.

d. Hosting/Development

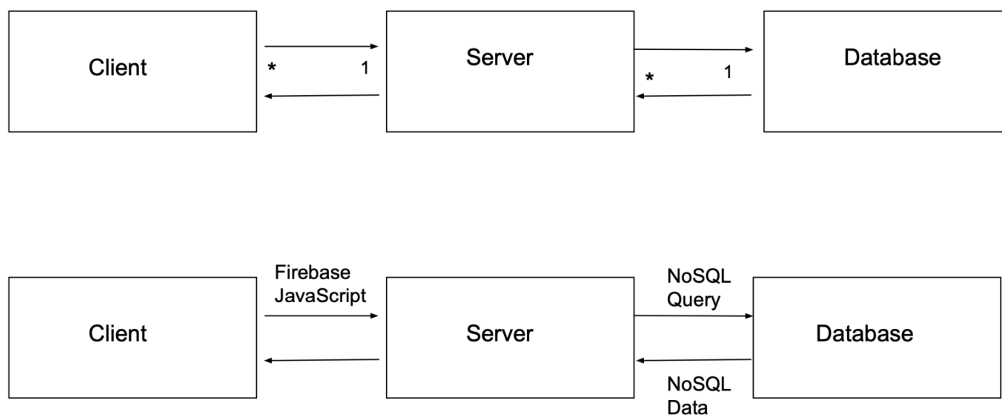
Budget Busters will be hosted on Firebase, as it offers a NoSQL cloud database. Specifically, this database utilizes data synchronization, allowing data from multiple devices to be stored and updated in real time. We will implement this hosting by configuring each user's JSON data to set and subscribe to data changes, and it will then listen to incoming requests to make necessary

changes onto Firebase. We will receive up to 10 GB of hosting storage, which will be enough to complete the Budget Busters application given the scope of this project.

Design Outline

High-Level Overview Diagram

This project is a web application that allows users to create an account, login, and interact with an interface to set monthly budget goals and actively monitor their daily spending habits. In order to implement this web application, the client-server-database model can help in handling a large number of users and send requests to store and gain access to data. The client has interfaces to help interact with the user, and the server will accept client requests, access and/or store data in the database, and provide feedback to the client, if necessary.



1. Client:

- a. Client provides the user an interface to interact with the budgeting app by creating budgeting functionalities.
- b. Client sends requests to validate authentication information to Firebase.
- c. Client sends requests to retrieve user profile information and budgeting statistics.

2. Server:

- a. Server requests database to store and/or access user inputted data (login, profile, budgeting goals) in the database.
- b. Server provides feedback to the client when requested or needed.

3. Database:

- a. A Firebase database (NoSQL) will store all user information such as login information, user profile data, and budget goals.
- b. Firebase will validate user login information through an authentication API.

Design Issues

Functional Issues

1. What information is required for creating an account?
 1. Option 1: username, password
 2. Option 2: email, password
 3. Option 3: email, password, phone number, firstName, lastName, age
 - Choice: Option 3
 - Justification: We chose to go with Option 3 to provide a comprehensive user profile. This allows for more personalization with the full name and age. We are utilizing the full name to prevent any potential duplication of user accounts. Additionally, we are requiring age, so that we can include a feature where the user can compare their budgeting goals with other users around/with the same age. The email and phone number are essential for the user to get notifications from our website. We opted out of creating a username and to simply use their email as their username to make it easier for the user.
2. When can the users change their budget goal?
 1. Option 1: monthly
 2. Option 2: weekly
 3. Option 3: can customize as user feels needed
 - Choice: Option 1
 - Justification: We felt users are allowed to change their budget goal monthly more specifically on the first day of the month. For every other day in the month this particular action would be disabled. This allows for the web application to collect enough data for more valuable charts and graphs as opposed to customizing to whenever. We also felt weekly goals would be too short of a time frame while a month would be a decent amount of time.
3. What kind of information must be inputted to add a purchase?
 1. Option 1: only cost of item
 2. Option 2: cost of item and name of item
 3. Option 3: cost of item, name of item, and category of item
 - Choice: Option 3
 - Justification: Having the cost of item, name of item, and category of item were all necessary details to track daily purchases. The combination of the cost of item and name of item allows us to verify if there are duplicate purchases when not intended. We also wanted the user to keep track of the category of items to create visuals for the user to monitor the categories that their purchases come from.
4. How should users know they are over their budget goal?
 1. Option 1: disable inputting in daily purchases
 2. Option 2: send exceeds budget goal notification
 3. Option 3: do nothing differently

- Choice: Option 2
- Justification: The goal of this web application is to build good budgeting habits and become more financially aware. By sending a budget goal notification the user is aware of their goal not being met. However this should not stop their purchases entirely but instead keep tracking their daily purchases for identifying patterns and better improve in the following month. Also doing nothing differently would not help the user become aware of going over budget.

Non-Functional Issues

1. Which languages are appropriate for implementing our back-end services?
 1. Option 1: Javascript
 2. Option 2: Python
 3. Option 3: Java
 - Choice: Option 3
 - Justification: We chose Java as it offers robust and high-performance backend services. It is designed to handle heavy workloads and efficiently manage concurrent requests, which is suitable for our web application as we hope to have multiple users and a lot of user data.
2. What type of database is most appropriate for our data?
 1. Option 1: MySQL
 2. Option 2: NoSQL (Firebase)
 - Choice: Option 2
 - Justification: We chose Firebase as a NoSQL database since it is known for its scalability and can handle large volumes of data if need be. NoSQL databases like Firebase have flexible schema which means we can easily adapt the data structures as our application evolves.
3. How to store passwords securely?
 1. Option 1: store it directly in Firebase
 2. Option 2: use Firebase Authentication
 - Choice: Option 2
 - Justification: We chose to use the Firebase Authentication system to store the passwords. This increases security for the user's account as Firebase Authentication provides features like account verification and multi-factor authentication. Storing directly into the database as a plain text can be damaging if there are any data breaches.

4. What framework should be used?

1. Option 1: React JS (web)

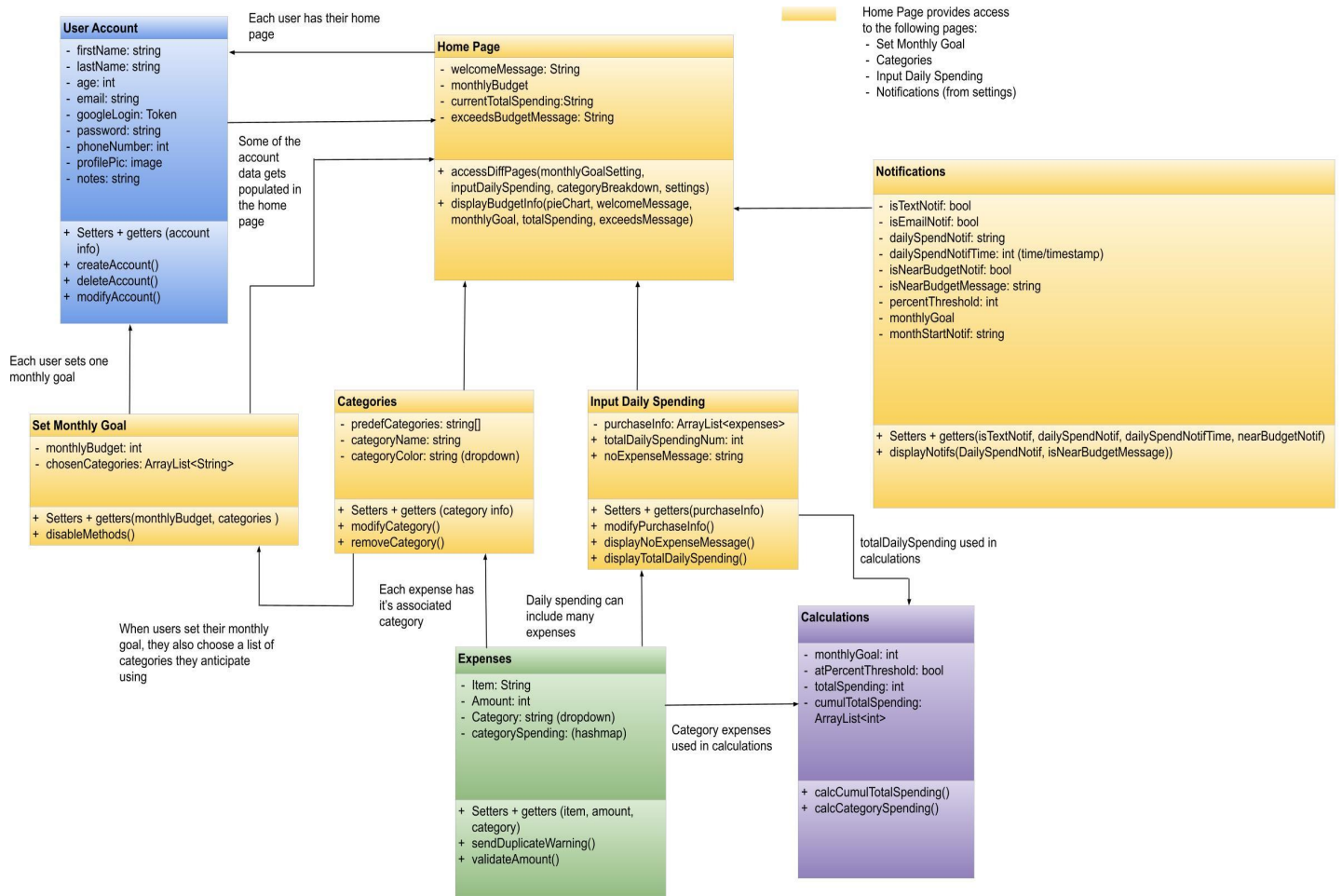
2. Option 2: React Native (mobile)

➤ Choice: Option 1

➤ Justification: We chose React JS as it is specifically designed for web applications. React Native, on the other hand, is intended for mobile app development. Since our application is intended to be a web application React JS would be the most appropriate.

Design Details

Class-Level Diagram



Class Descriptions and Interactions

1. User Account

- a. The user account class is used when a user creates an account and provides their information.
- b. The user will provide the following input:
 - i. First and Last Name
 - ii. Age
 - iii. Standard email or Gmail
 - iv. Password
 - v. Phone number
 - vi. Profile picture
 - vii. Any notes
- c. The home page will use the name input to populate its welcome message.
- d. The user can modify their user account via settings.
- e. Once the user creates their account/signs in, they are directed to the home page, where they can engage in the other features of the app.

2. Home Page

- a. Once users are directed to the home page, they'll view a personalized welcome message that lists their name, their monthly budget goal with a visual pie chart, their current total spending, and a warning message if their current spending exceeds their monthly goal.
- b. The home page will display buttons that'll redirect users to the essential features of the app. They can choose to set their monthly goal and categories, view a categorical breakdown of their spending, input daily spending, and/or access settings to modify their account or customize their notifications.

3. Set Monthly Goal

- a. Users will input an integer as their monthly goal and choose the categories they anticipate their spending will be divided into.
- b. This monthly goal will be used in the home page as part of the welcome message.
- c. When users set their monthly goal, they will be choosing/creating categories using the categories class.

4. Categories

- a. When the user sets their monthly goal or inputs an expense, they will use the category class to choose/create/assign a category. We will provide a predefined set of general categories that users can choose from, but they also have the option to create their own. For added customizability and differentiation, users can color-code their categories.
- b. The categories users choose and their respective colors will be used in the category breakdown page.

5. Input Daily Spending

- a. Each day, users will receive a notification to input their daily spending. They can choose to input as they spend, or input them all at once. On the input daily spending page, users can click an add button to enter an expense.
- b. The expenses for each day is also used in the Calculations class.

6. Notifications

- a. Our app utilizes notifications to remind users to input their daily spending, but also to receive warnings if their spending is approaching their budget and/or reminders at the first of the month.
- b. Users can customize their notification settings by going to the notifications section of the settings page. Home page > settings > notifications.
- c. Users can also choose whether they receive their daily input spending notification as an email or SMS message.

7. Expenses

- a. When a user presses the add button in the input daily spending page, an expense object is created.
- b. Each expense has the following fields the user has to input:
 - i. Item
 - ii. Price
 - iii. Category
- c. Once an expense is made, the info is used in the calculation class, which will later be utilized for the graphs.

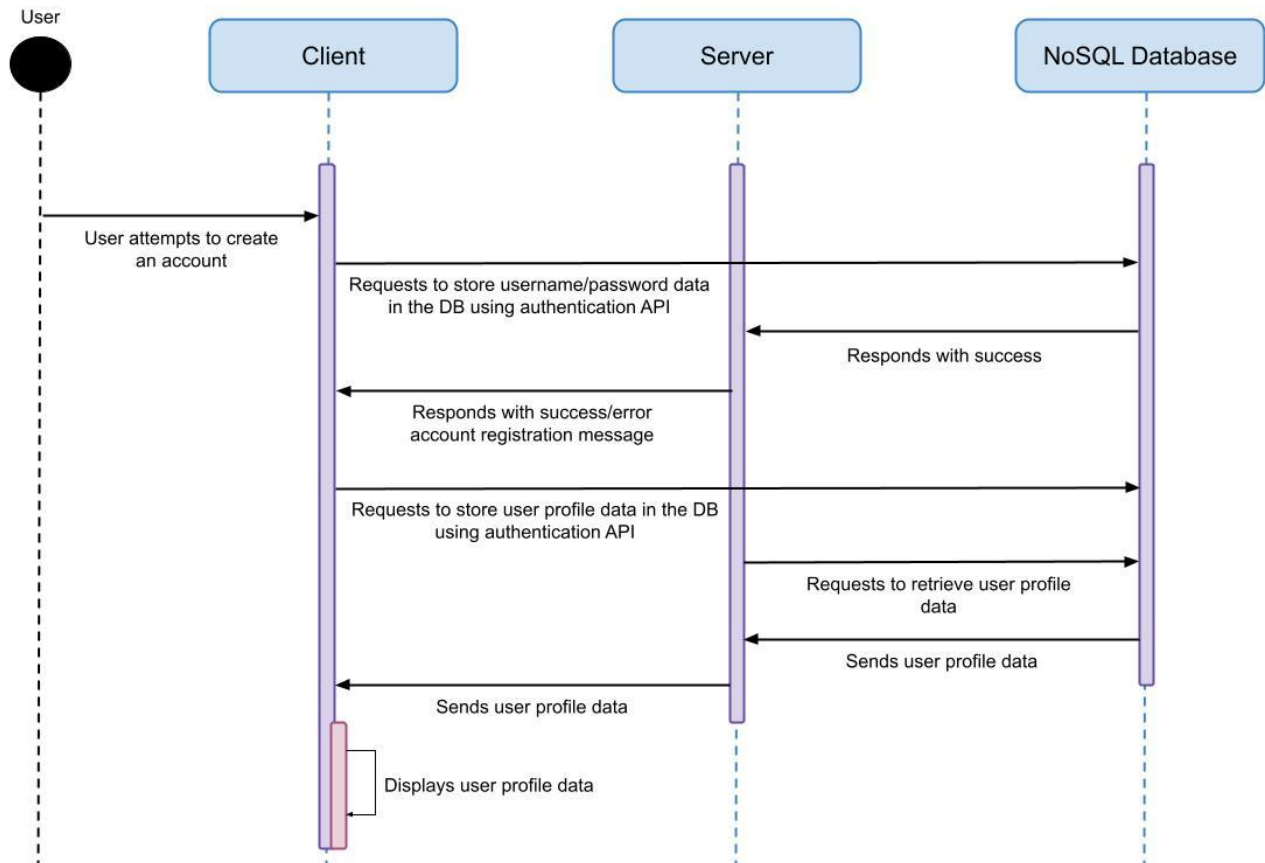
8. Calculations

- a. The calculations class is used primarily for data visualization.
- b. Each expense and daily set of expenses is used in the Calculations class.
- c. If users opt-in for a warning notification if their spending is reaching their budget goal, they can choose to provide a specific percent threshold.
- d. As each expense is logged, the total daily spending amount will be updated accordingly.
- e. The percent threshold and the monthly goal is used in this class to determine at what spending point the user will receive the warning.

Sequence Diagrams

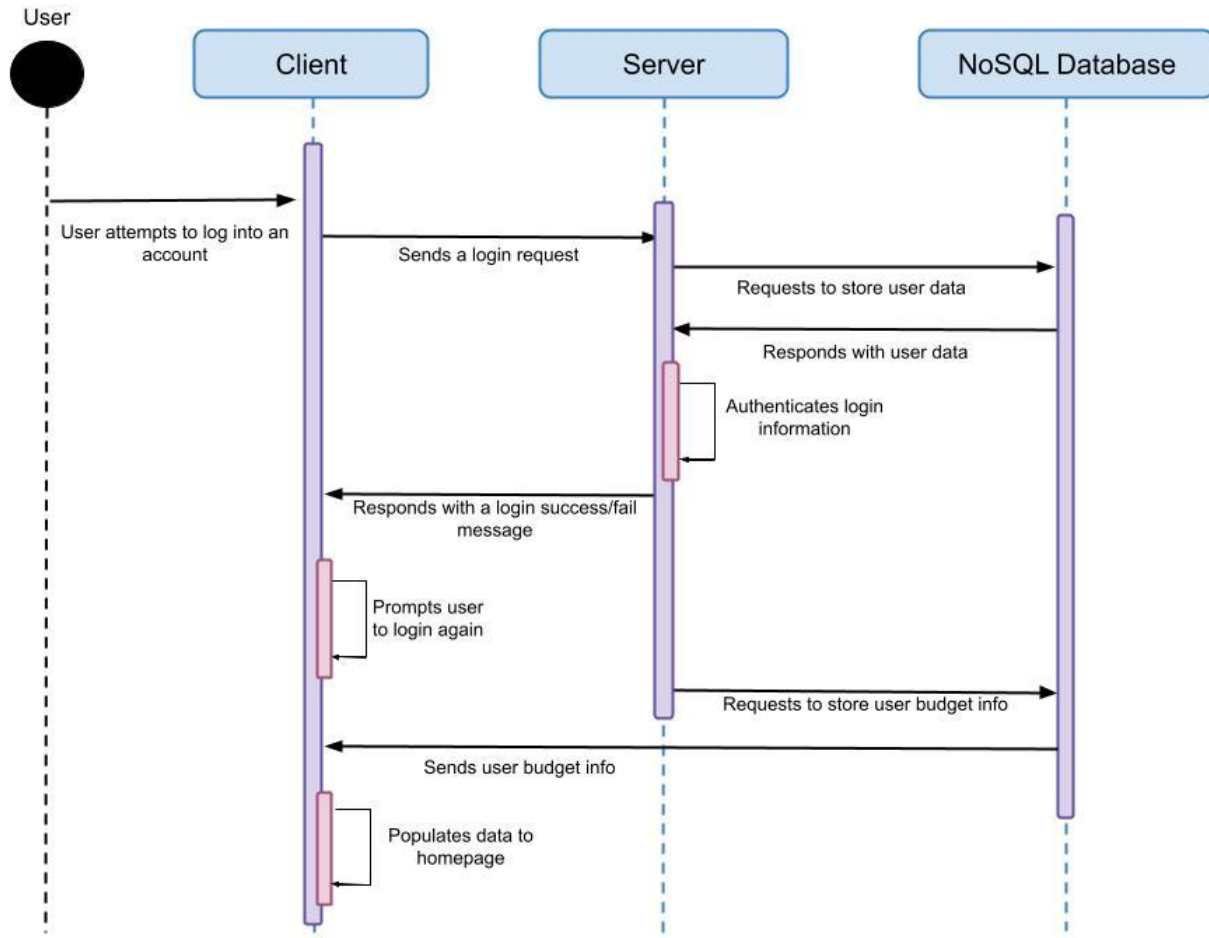
The following diagrams depict the sequence of the major events in this application including registering on the app, signing in the app, updating user profile, setting monthly budget goal, and inputting a purchase. We've shown the interactions between the user, client, server, and database in these sequence diagrams.

Registering on the app



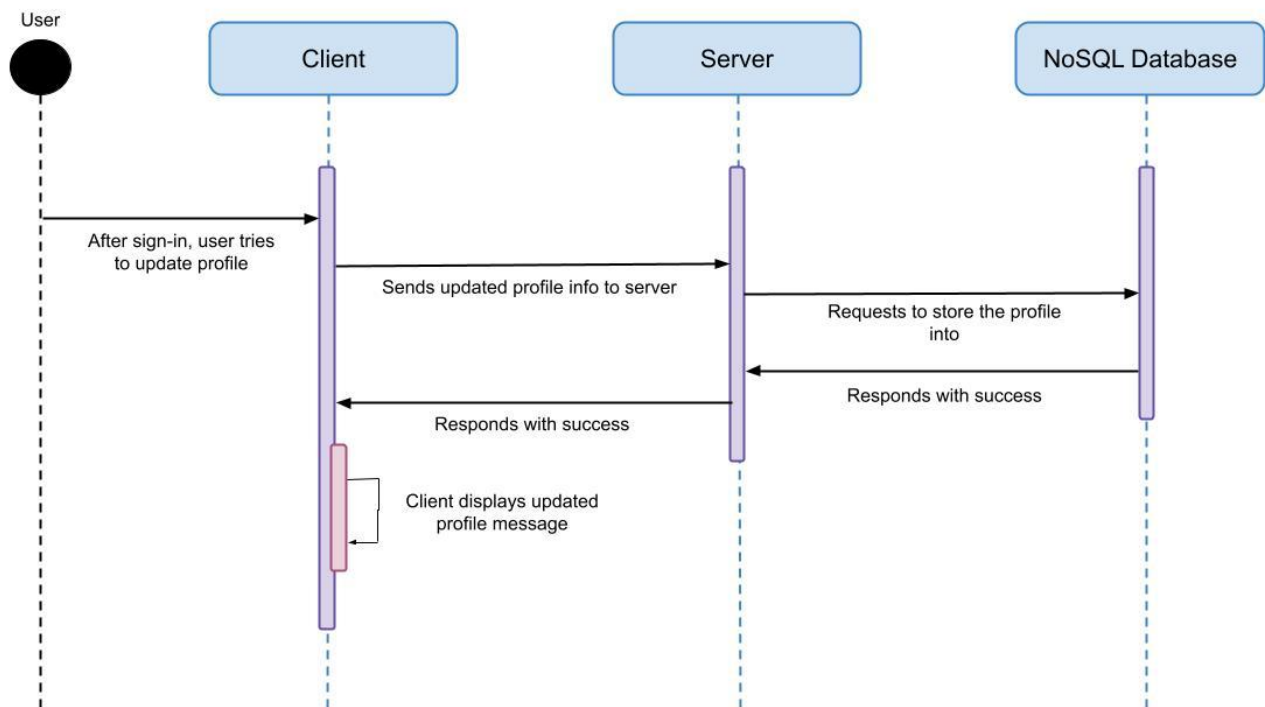
When the user first enters into our website they can create an account. This will send a request from the client to the databases to store the username/password. If this request is successful, the client will be notified from the database to proceed. Then, the client will collect the other registration information and send it to the database. Lastly, the server requests to retrieve the user profile data from the database. This user profile data is then sent and displayed on the client side.

Signing in the app



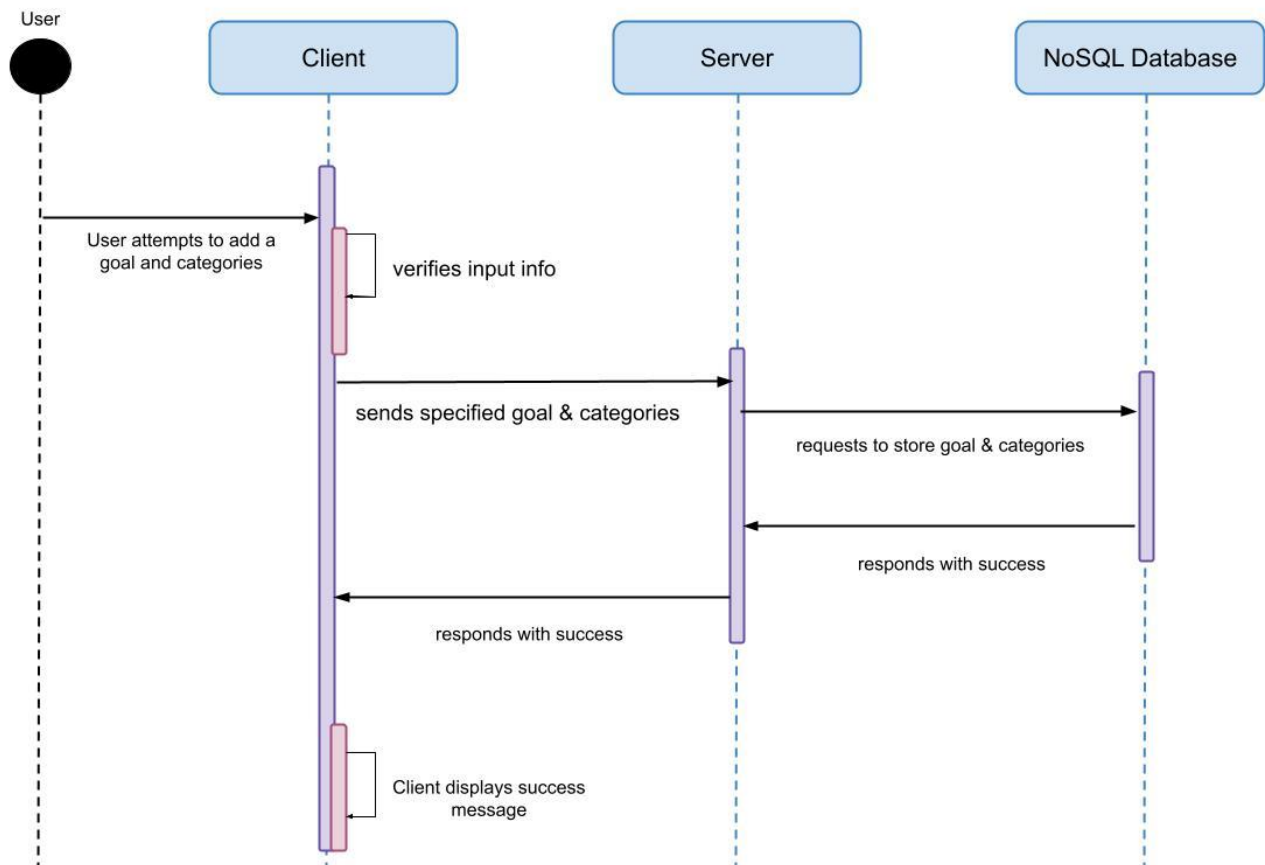
For a user to sign in to their account on Budget Busters, the client will first send a login request to the server, in which the server requests Firebase to store the sign in information. If that's successful, Firebase responds with the sign in information, to which the server would authenticate the user's inputted login information. If the validation is successful, the server will communicate that with the client. If not, the user is prompted to login again. During this time, the database sends over the users budget information and statistics. Then, the data from Firebase would be populated on the homepage.

Updating user profile information



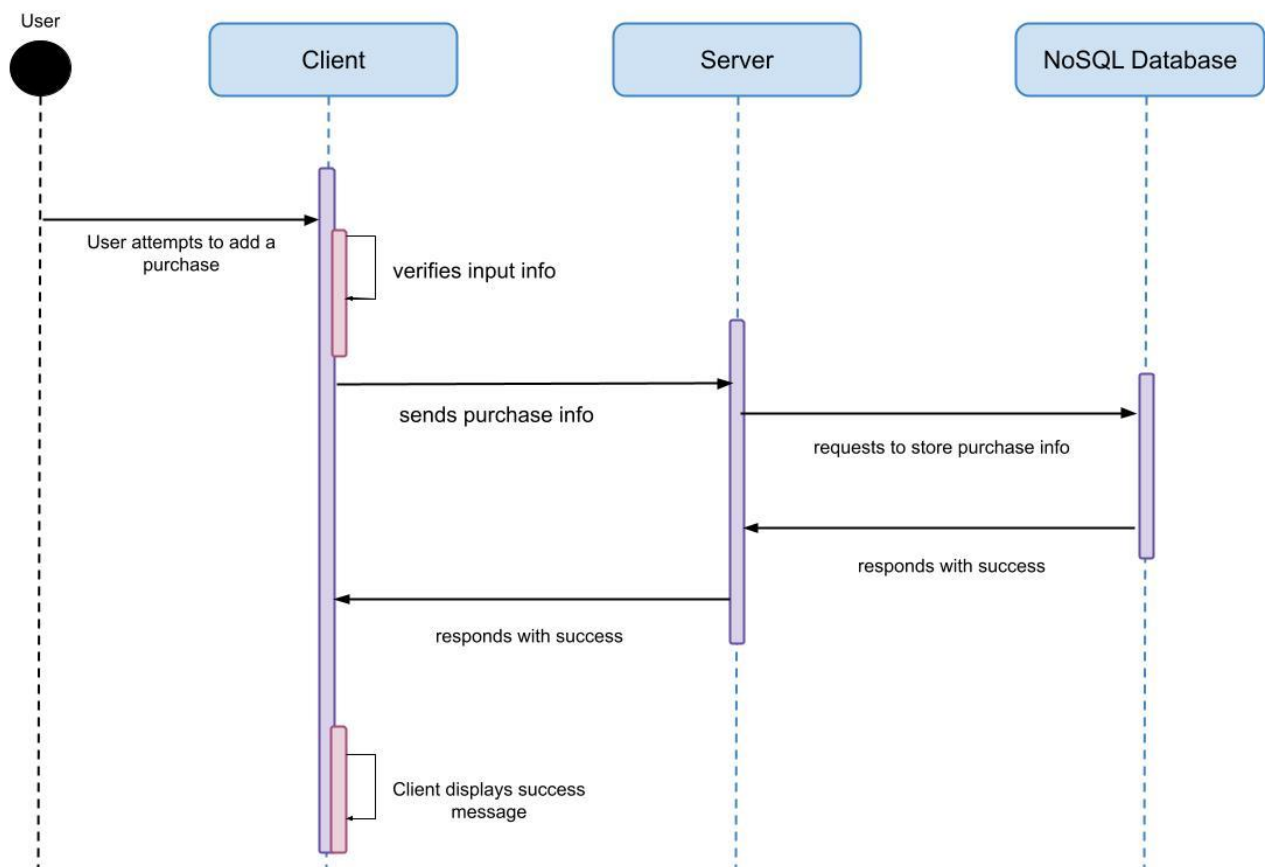
Once the user signs in, they can modify their user profile information. If they decide to do so, the client will send the updated profile information to the server, and the server will request the NoSQL database to store the info. Upon completion, the database sends a success message to the server, which will in turn send a success message to the client. Once the success message is passed from the database to the client, the client will display the updated profile message.

Setting monthly budget goal



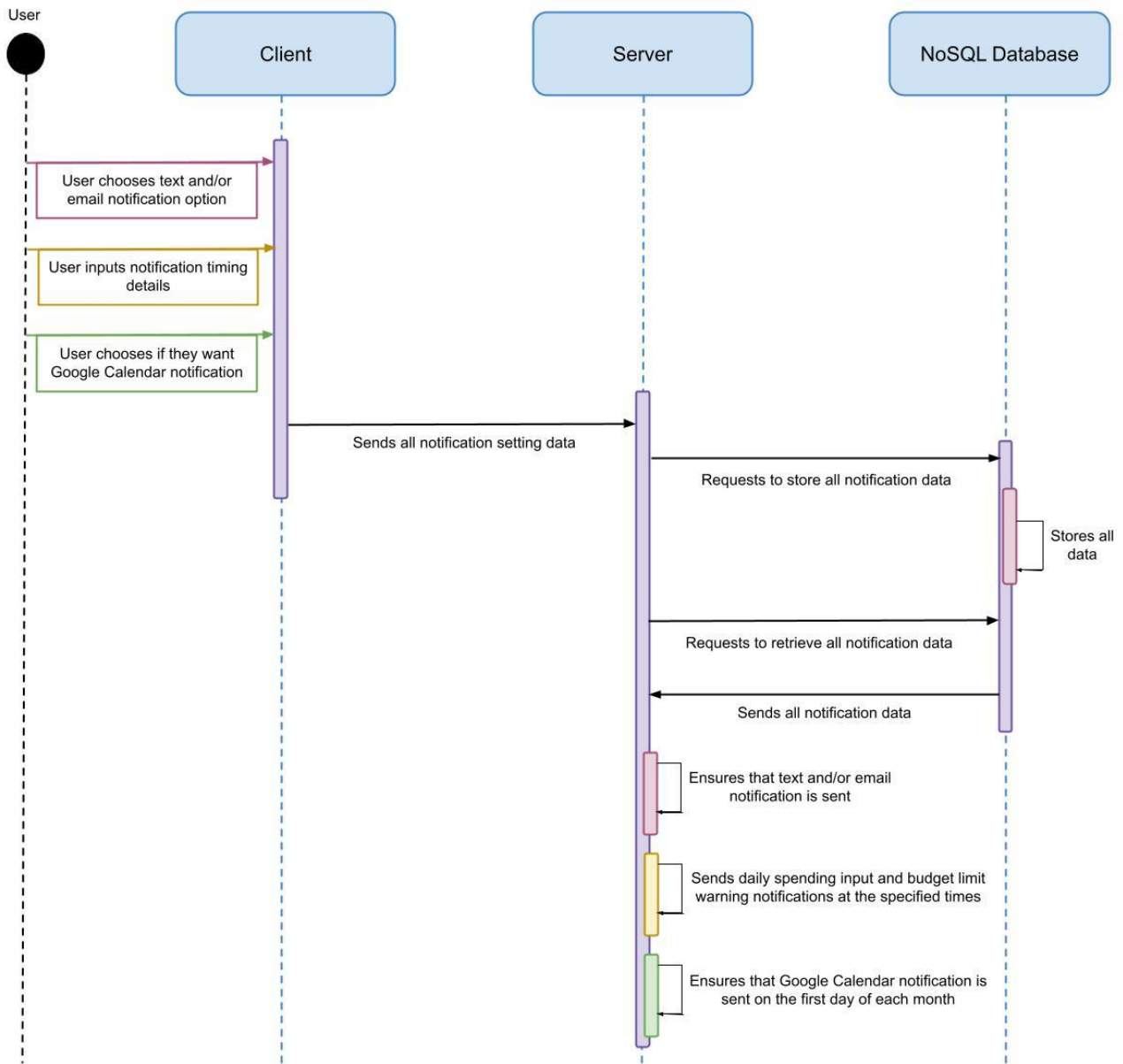
When the user decides to add their monthly goal along with some categories, the client verifies whatever the user inputs. Once the verification is complete, the client sends that information to the server, which then requests the NoSQL database to store. Upon completion, the database sends a success message to the server, which in turn sends a success message to the client. Once the success message travels from the database to the client, the client displays the success message.

Inputting purchase information



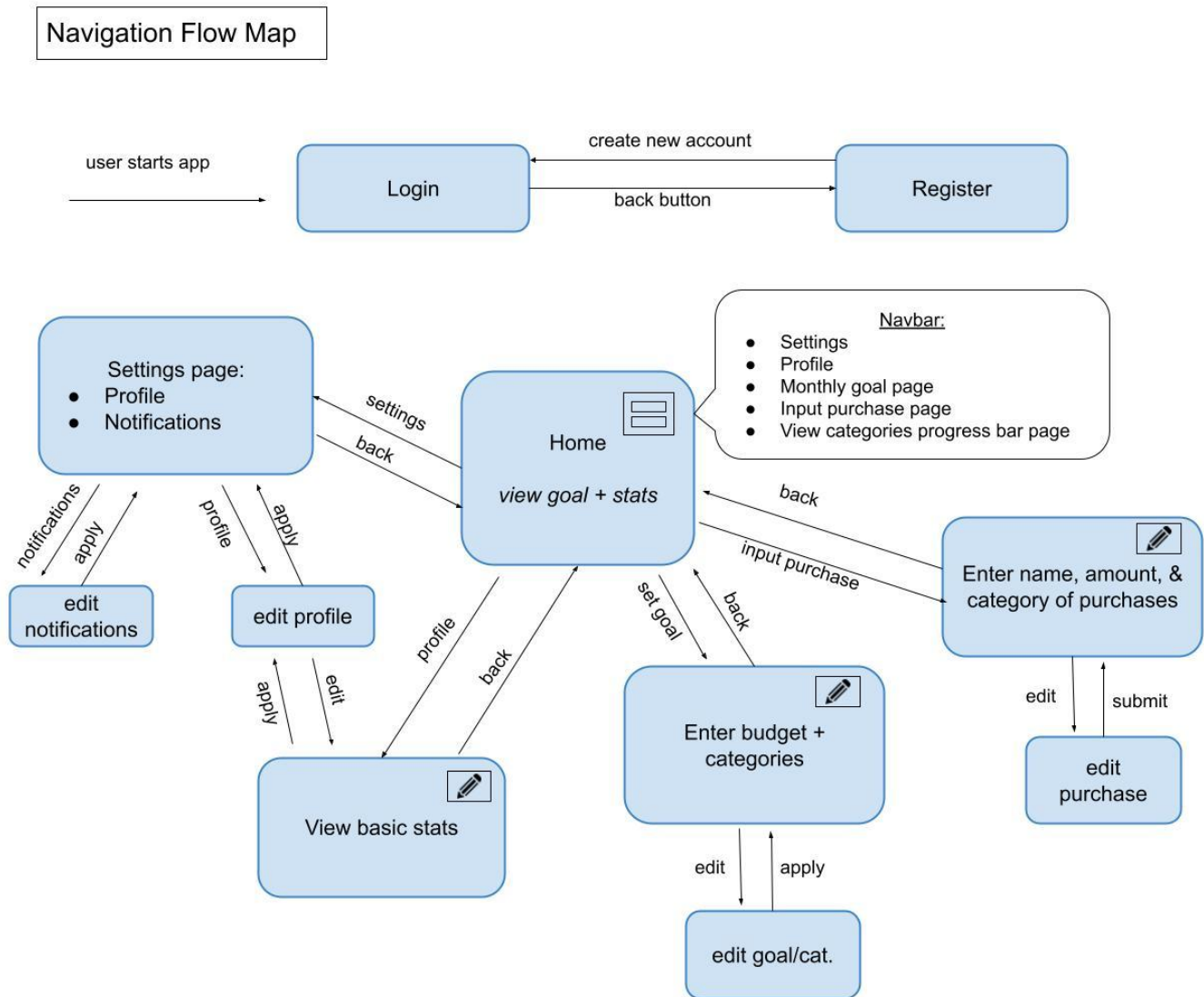
After the user purchases an item, they record the information on the inputting page. The user attempts to add the information on the client, which verifies the input by checking valid input types. Then, the information is sent to the database to store the purchase information. If this request succeeds, the client displays that the purchase information was recorded.

Customizing notification settings



A user can make the following customizations on how they receive their notifications: they can choose to get them through text and/or email, they can specify the time that they get their daily input spending and budget limit warning notifications, and if they want a Google Calendar alert at the beginning of each month to enter their monthly budget goal. All of this information goes to the client first, and then it gets passed on to the server and database, respectively. Then, whenever it is time to send a notification, the server requests to retrieve the user's inputted customizations from the database. The server then uses the notification data to send out the notifications in the specified manner.

Navigation Flow Map

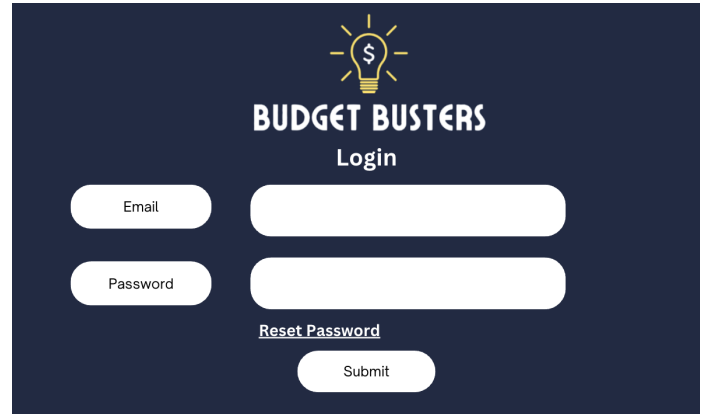


When the user opens the app, they will see a screen showing options to login or create an account. Upon registering or logging in, they will see a home page, which details their monthly budget goal and the categorical percentages in relation to that goal. They will also have access to the navigation bar, where they can access settings (which contains notification and profile information), input budget and categories page, category breakdown page, and their input daily spending page. Each screen stemming from the home page will contain a back button that will lead the user back to the home page.

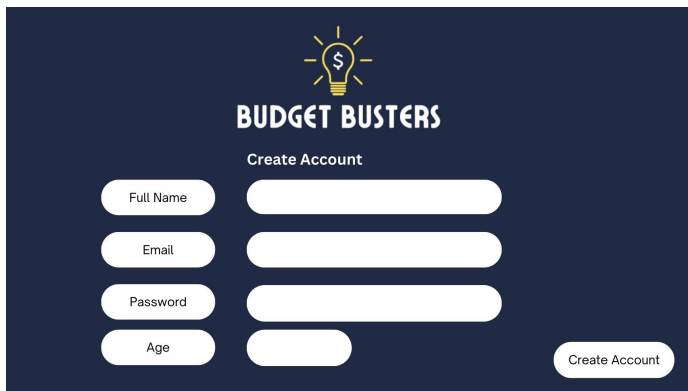
UI Mockups



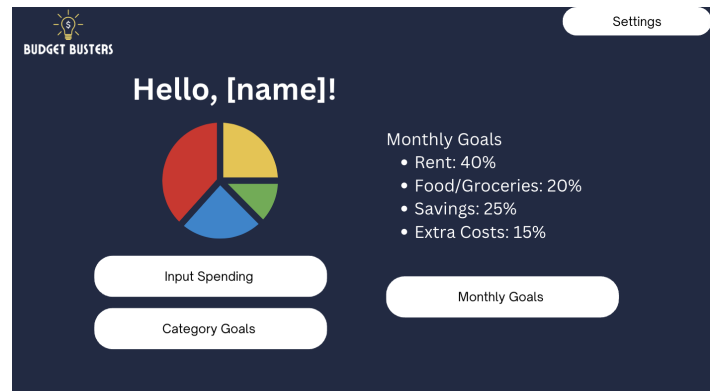
Opening Page



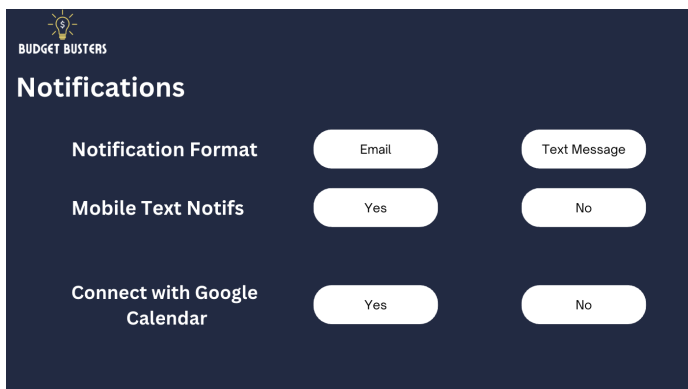
Login Page



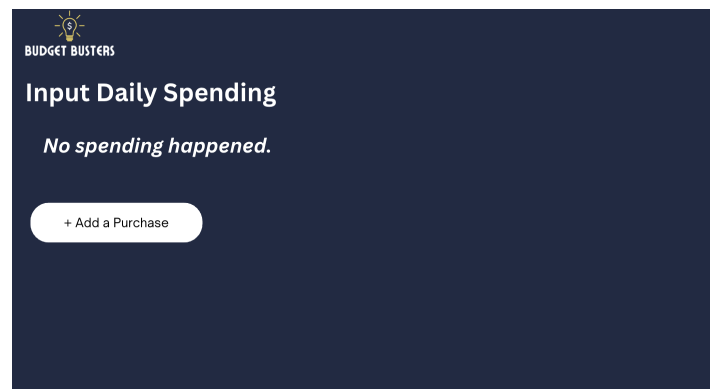
Create Account Page



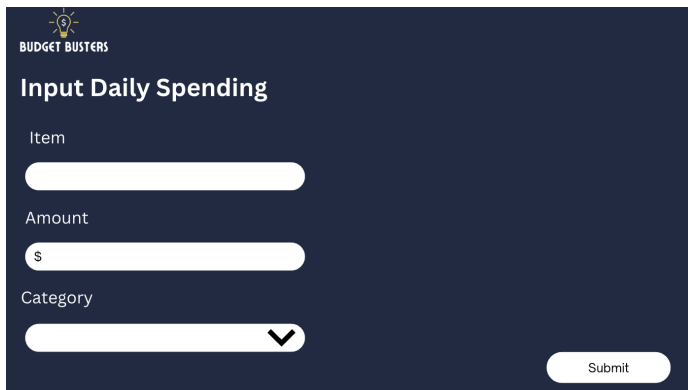
Home Page



Notification Settings Page



Input Daily Spending Page
(with no previous purchases)



BUDGET BUSTERS

Input Daily Spending

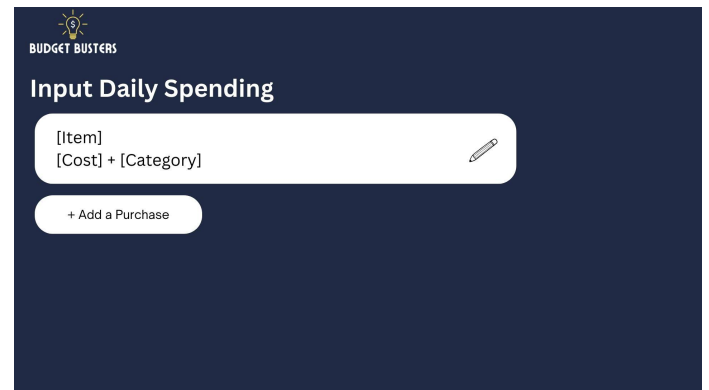
Item

Amount

Category


Submit

Input Single Purchase Page



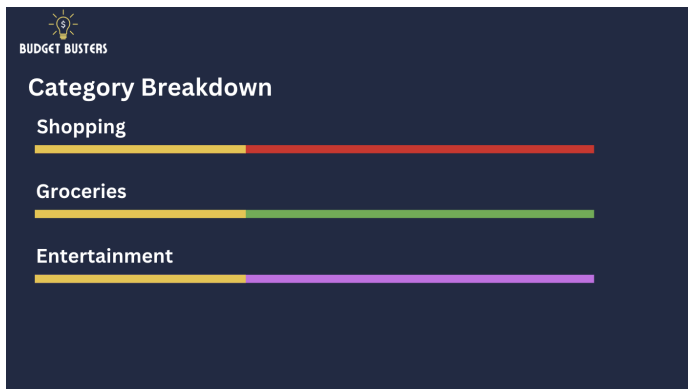
BUDGET BUSTERS

Input Daily Spending

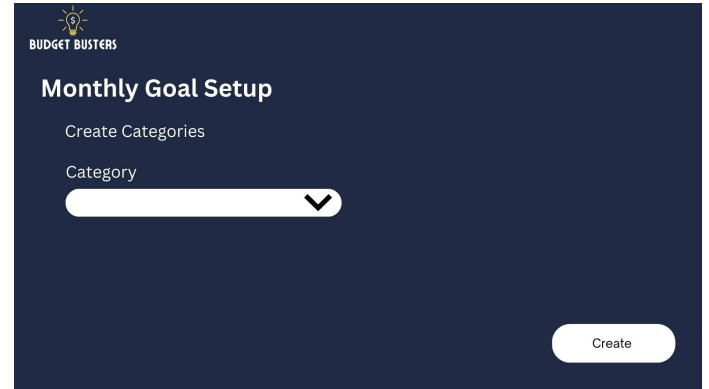
[[Item]
 [Cost] + [Category] 

+ Add a Purchase

**Input Daily Spending Page
(with previous purchases)**



Category Breakdown Page



BUDGET BUSTERS

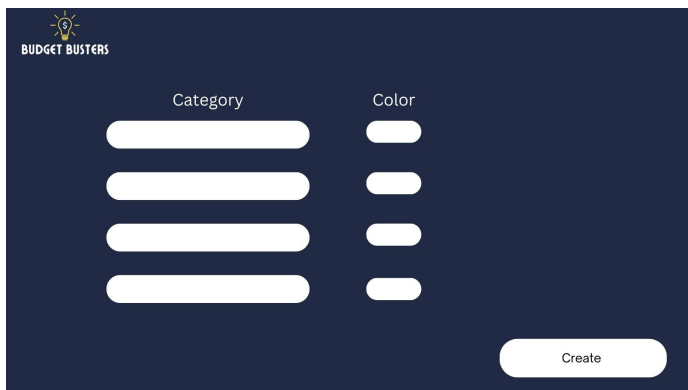
Monthly Goal Setup

Create Categories

Category

Create

Monthly Goal Setting Page



BUDGET BUSTERS

Category	Color
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>

Create

Category Customization Page