ITSC 4155 - 001
Group 10
03/26/24
Trevor Richardson, Shail Patel, Justin Tubay, Devon Lemasters, Minh Anh Nguyen

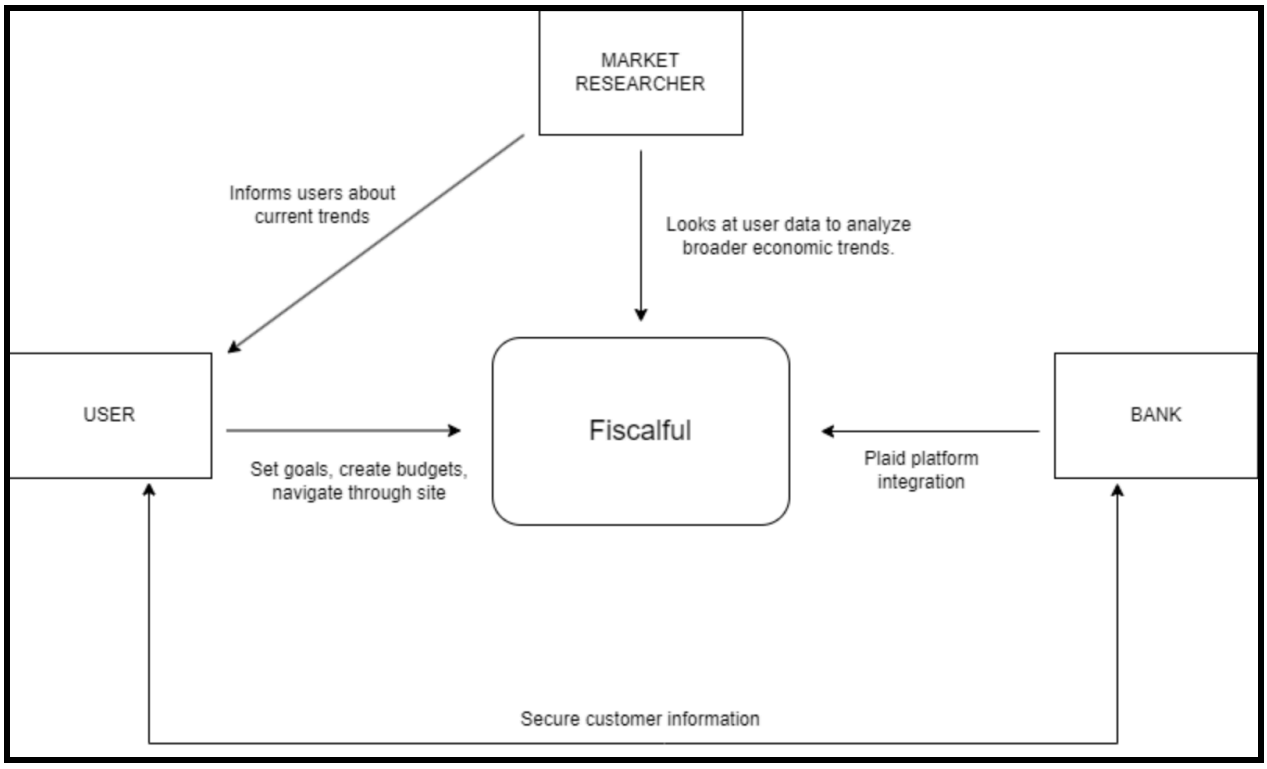
ITCS 4155
SD^2: Software Design Document

1. Project Overview

Hundreds of millions across the globe are currently faced with the perils of economic tribulations stemming from various external and internal factors. It is imperative that individuals are given access to technologies that will aid and support their financial decisions in order to promote financial security and stability. The software will offer ways to provide information on economic trends, create budgeting plans, and to connect to the user's bank account via Plaid. Users can gain personalized financial insights and advice through an intuitive AI chatbot. Users can create customized budget plans based on their individual income, expenses, and financial goals.

- **Individual Users:** The primary users of the software are individuals seeking to manage their personal finances effectively.
- **Financial Institutions:** Banks and other financial institutions would have an interest in the Plaid platform integration, allowing users to connect their accounts securely. These institutions would benefit from an increase in user engagement.
- **Financial Advisors:** Financial Advisors may find value in the software as it can help follow and analyze the current economic trends.
- **Market Researchers:** Professionals in market research could use user data to analyze broader economic trends and consumer behavior.

CONTEXT DIAGRAM:



USER STORIES:

Story #	Card Front	Card Back
ST-0	As a <user who wants this functionality> I need a <what the user wants> so that <why the user wants it>.	Acceptance Criteria <what the user should be able to do with the functionality>
ST - 1	As a user, I want to be able to create and login into my account so that I can manage my finances.	A user should be able to create and login to an account via email and password, be able to reset password.
ST - 2	As a user, I want to be able to see my bank account information in my dashboard so I can manage my finances from a centralized point.	The user should be able to connect their bank account to the user dashboard via PLAID API.
ST - 3	As a user, looking for financial awareness, I want to track my spending habits and identify areas where I can make adjustments to improve my financial health.	The user should be able to view their spending and recent transactions.
ST - 4	As a user, I want to easily access information on economic trends, so I can stay informed on the market and make smarter financial decisions.	A user should be able to view the economic trends so they can make decisions that fit their needs.
ST - 5	As a user, I want to ask financial questions in a secure AI chat so that I may gain information that is specific to my situation and goals.	A user should have access to an AI chatbot that is trained in economic trends and market information to ask questions that might be unique to them.
ST - 6	As a user, I want to be able to set customizable goals for spending, saving, and investment so that I can ensure I'm staying on track.	A user should have access to an AI chatbot that is trained in economic trends and market information to ask questions that might be unique to them.
ST - 7	As a user, I want to be able to create budgets so that I can better understand my weekly or monthly expenses.	A user should be able to create bill reminders and budget plans that visually display and communicate their spending/expenses.
ST - 8	As a user, I want to be able to communicate with site administration so that I can provide user feedback and report issues.	The user should be able to submit a contact form to communicate with site administration via email upon response.
ST - 9	As a user, I want to be able to easily navigate through the application and access the features the application provides without any trouble.	A user should be able to access the information they need so they effectively manage their finances.
ST - 10	As a site administrator, I want to be able to recieve user contact requests so that I can effectively resolve consumer issues.	A site administrator should be able to view user emails from a centralized point/automated system.
	As a site administrator, I want to be	A site administrator should have

PRODUCT BACKLOG:

Group Number: 10
Project Name: Fiscalful
Description: Personal Finance Website

PRODUCT BACKLOG (TO-DO)

Team Members: Trevor Richardson, Minh Anh Nguyen, Justin Tubay, Devon Lemasters, Shail Patel

Story #	Card Front	Card Back	Sprint Number	Priority	Assigned To	Comments	Estimated Points	Estimated Hours	Estimated Velocity (Estimated Points / Estimated Hours)	Actual Velocity (Estimated Points / Actual Hours)
ST-0	As a <user who wants this functionality> I need a <what the user wants> so that <why the user wants it>.	Acceptance Criteria <what the user should be able to do with the functionality>								
ST - 1	As a user, I want to be able to create and login into my account so that I can manage my finances.	A user should be able to create and login to an account via email and password, be able to reset password.	Sprint 2	1st		React built user interface with backend firebase authentication/account creation.	3	2	1.50	#REF!
ST - 2	As a user, I want to be able to see my bank account information in my dashboard so I can manage my finances from a centralized point.	The user should be able to connect their bank account to the user dashboard via PLAID API.	Sprint 1	2nd		Implement plaid API bank connection and update values on user dashboard.	5	4	1.25	#REF!
ST - 3	As a user, looking for financial awareness, I want to track my spending habits and identify areas where I can make adjustments to improve my financial health.	The user should be able to view their spending and recent transactions.	Sprint 2	3rd			3	2	1.50	#REF!
ST - 4	As a user, I want to easily access information on economic trends, so I can stay informed on the market and make smarter financial decisions.	A user should be able to view the economic trends so they can make decisions that fit their needs.	Sprint 2	4th			13	10	1.30	#REF!
ST - 5	As a user, I want to ask financial questions in a secure AI chat so that I may gain information that is specific to my situation and goals.	A user should have access to an AI chatbot that is trained in economic trends and market information to ask questions that might be unique to them.	Sprint 3	5th			21	16	1.31	#REF!
ST - 6	As a user, I want to be able to set customizable goals for spending, saving, and investment so that I can ensure I'm staying on track.	A user should have access to an AI chatbot that is trained in economic trends and market information to ask questions that might be unique to them.	Sprint 3	6th			5	4	1.25	#REF!
ST - 7	As a user, I want to be able to create budgets so that I can better understand my weekly or monthly expenses.	A user should be able to create bill reminders and budget plans that visually display and communicate their spending/expenses.	Sprint 4	7th			3	2	1.50	#REF!
ST - 8	As a user, I want to be able to communicate with site administration so that I can provide user feedback and report issues.	The user should be able to submit a contact form to communicate with site administration via email upon response.	Sprint 4	8th			8	6	1.33	#REF!

2. Architectural Overview

In designing the Fiscalful website, there were several architectural design options that were considered in order to promote scalability, security, usability, and efficiency.

Alternative Designs:

1. Monolithic Architecture (Single Unit): This architectural design was considered due to its ease of development and simplicity however, due to scalability concerns as the website grew, this architecture was ultimately ruled out.
2. Serverless Architecture (AWS Lambda): This architecture design was considered due to its scalability and cost effectiveness by only accounting for services and website usage however, due to the concerns in regard to latency issues and site uses this architecture was ruled out.

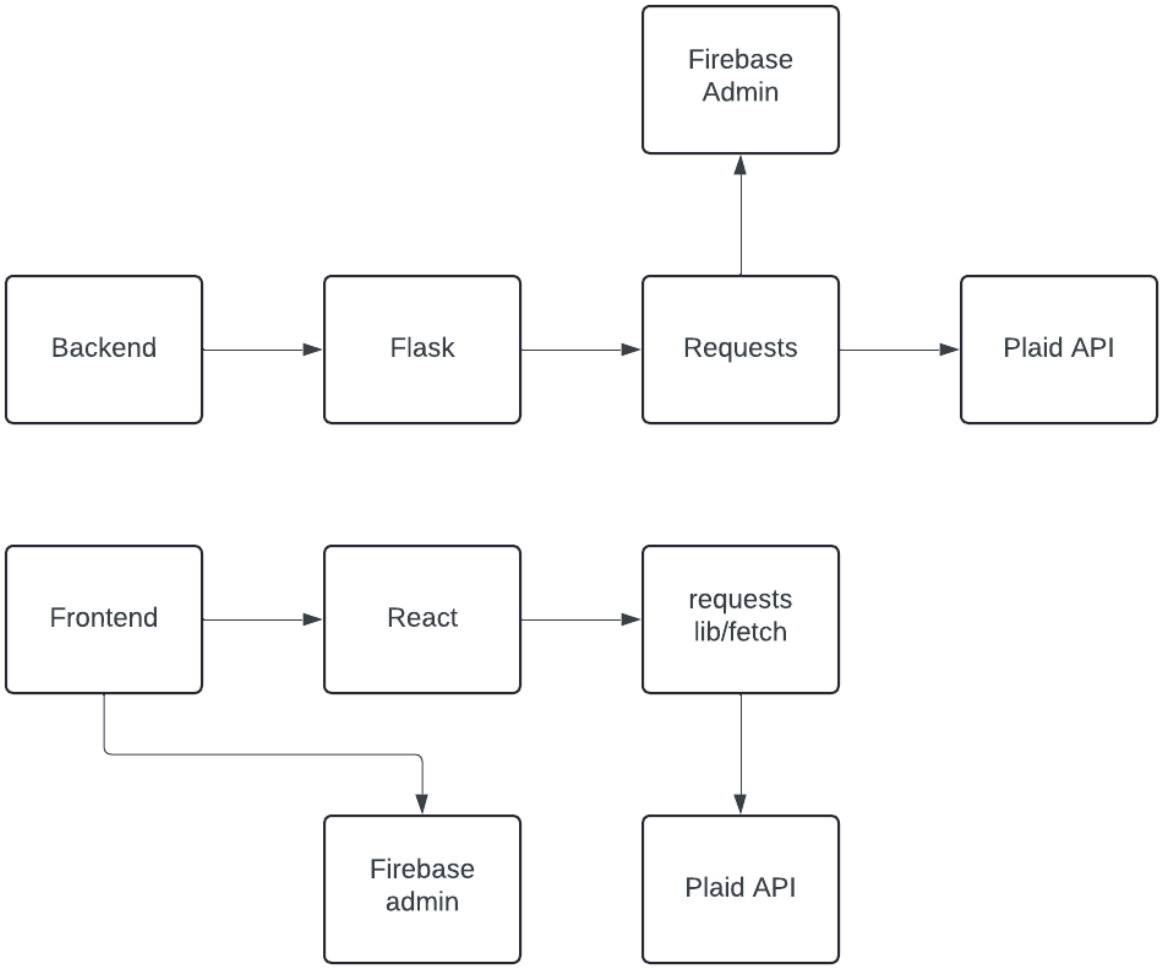
Chosen Design:

1. Microservice Architecture: This architecture design was chosen for a variety of reasons including but limited to scalability, technology diversity, data management, and etc. This architecture design allowed the team to isolate certain website services with certain technologies which in turn if there was ever a issue in regard to one of the site services (contact, login/signup, bank account connection, chatbot usage, budget planning) then the team would be able to identify the technology that was used for that specific service and quickly resolve the issue.

By utilizing the microservice architectural design, we have been able to implement various technologies into our project such as Firebase, Email.js, PLAID, React, and etc.

Utilizing these technologies will not only allow us to better test the system and identify weak points, but also provide an efficient, scalable, and usable website for users.

2.1 Subsystem Architecture



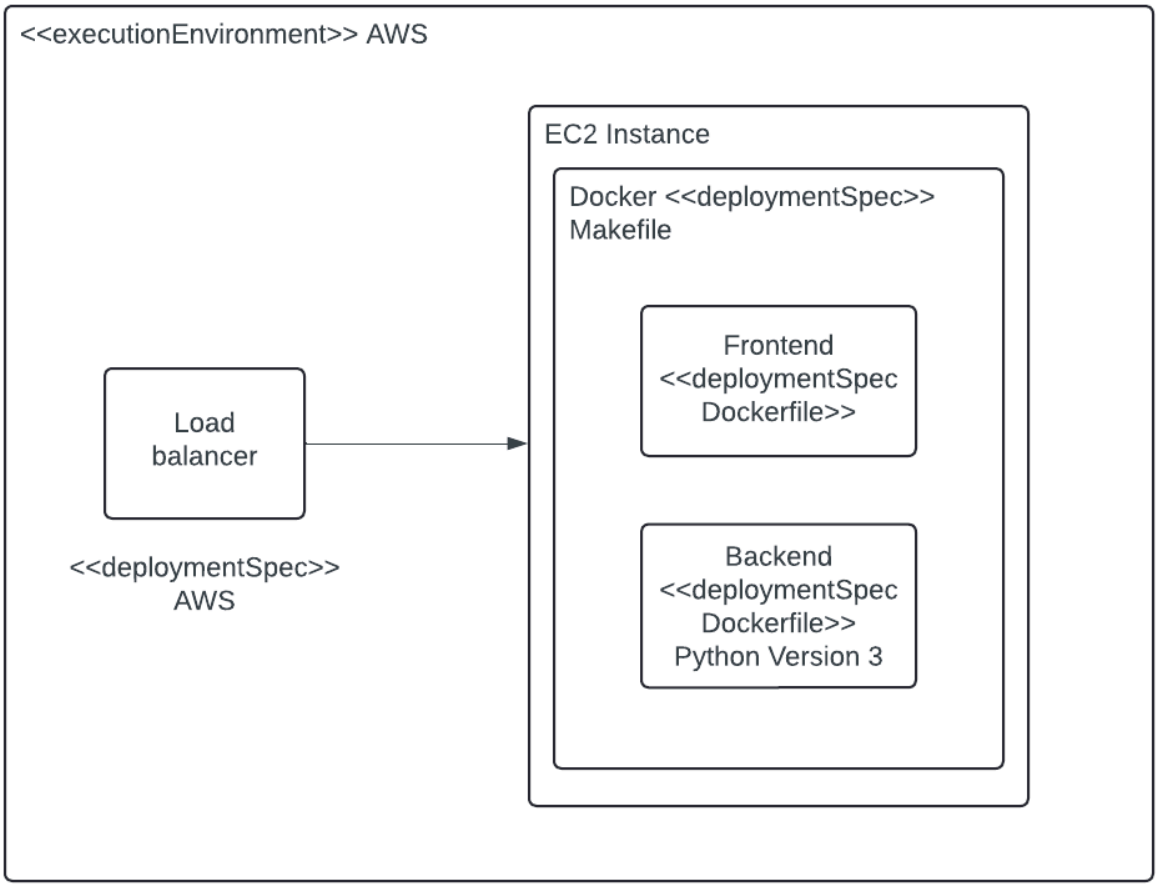
For the backend, Flask is used to make an http application. The requests library is used to make requests to the firebase admin module as well as the plaid API module.

On the frontend side react is used and the firebase admin package. React has a fetch library to handle requests which calls the plaid api.

2.2 Deployment Architecture

Everything will run on an AWS environment. AWS will handle stuff like load balancing and spinning up EC2 instances based on load. An EC2 instance will contain a docker

image of the frontend and backend code.



2.3 Data Model

Our approach to storing data will be through Firebase, a cloud, noSQL database, and the Plaid API via JSON. The Plaid API shall store information about the user's banking, such as transactions and purchase history. Firebase will store login information, which will be held via access tokens. It will also store the user's budgeting plans, and will consistently update as the user enters changes into the application.

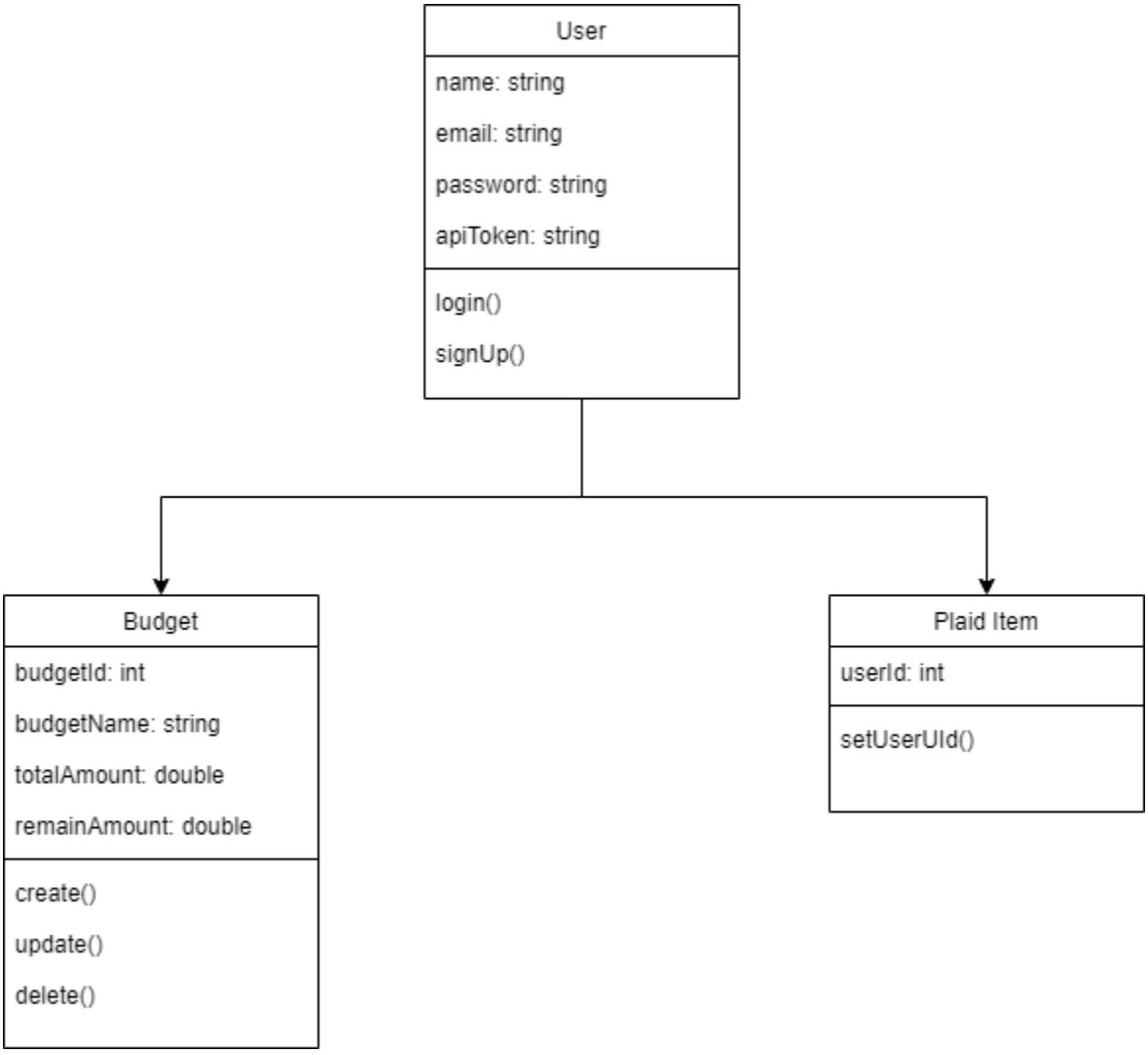
2.4 Global Control Flow

The application is event-driven, where events will be held in their own specific loops and functions. For example, when a user logs in, they have access to many features, and if they want to check their balance, that event will be executed when the user does so. There are not any timer-controlled actions in our system, and it is more of an event response type. Our system doesn't use multiple threads, since every feature would be its own separate event. For example, your budgeting plans won't really effect what your balance is.

3 Detailed System Design

Since we are using the microservice architecture approach, data is retrieved and stored through a variety of services/technologies. We mainly use React.js on the frontend while using Flask on the backend. Everything from the pages, images, header, footer, and navigation that you see on the application are displayed using React. Both services have libraries to request data from Plaid API and Firebase for functions such as login and sign up. Currently, the three classes for our application are user, budget, and plaid item.

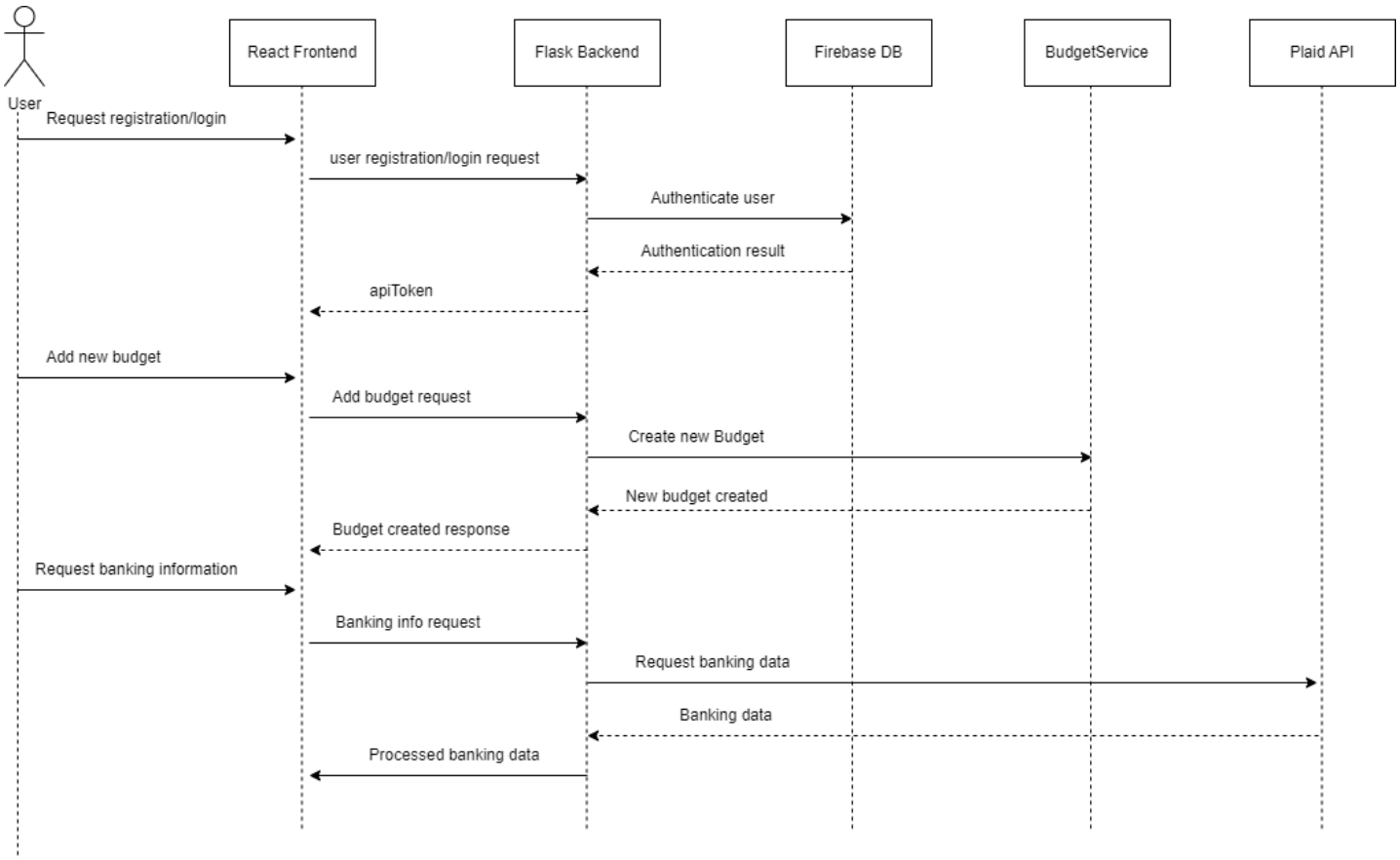
3.1 Static view



For the user class, we have name, email, password, and apiToken to store login and sign up credentials from Firebase. When a user adds a new budget, a budget class will be created with attributes: budgetId, budgetName, totalAmount, remainAmount. TotalAmount will be used to represent the starting budget, while remainAmount will

represent how much budget is left. For the plaid item class, we simply have userId that Plaid will use to retrieve the rest of the user's banking information.

3.2 Dynamic view



Key components include the React.js frontend, Flask backend, Firebase for authentication and Plaid API for banking data. Users interact with the frontend to register or login. The frontend sends a request to the Flask backend, which handles authentication via Firebase. Upon successful authentication, an API token is generated for subsequent requests. Users can also create budgets using the React frontend. The frontend sends a request to Flask, and the budget service creates a new budget instance. This is then sent back to the frontend for user view. Users can also view their banking information in the app using Plaid API. When the information is needed, the frontend sends a request to the backend. The backend then requests the users banking information from the Plaid API using their userId. Plaid fetches the banking data, which Flask processes and sends back to the frontend for user view.

Github Repository
https://github.com/ITSC4155MDSp24Group10/Fiscalful.git