CS 407 - Software Engineering Senior Project

Design Document

Team 5:

John Robinson

Wesley Turnbull

Henry Wellman

Patrick Spitzer

Index

Purpose

Introduction

Financial planning is an important part of everyday life. However, many college students do not actively keep a budget. This largely stems from the budget software they would potentially use either being a paid service, or the free versions lack many key features and have a poor UI experience. That is when our project, FinanceFriend, comes in. We aim to deliver a completely free financial well-being project that can be used as the last financial software a user will ever need. This includes adding features that normally are not seen in free budgeting applications like 401k / Stock Market price tracking, so users can manage their investments and retirement goals with more insight, or an interactive calendar to keep track of things like bill deadlines or important meetings. We will also have an optional feature that will enable users to see extra info about their spending, like where they were and what time it was when they made the transaction.

Project Objectives:

- Develop a web app that allows users to set a budget with weighting based on several different spending categories
- Make the project cross-platform and usable on both iOS and Android devices along with the web app
- Include the ability to track your expenses and categorize them
- Create the ability to add and see bill deadlines and other relevant dates
- Include the ability to track the current performance of their investments
- Include the ability to visualize the user's spending via tools like graphs
- Design and use a database so that a user's information is saved for the next time they log on

Functional Requirements:

L: Login
UI: UI/UI Elements
B: Backend (Database, website functionality)
F: Part of a feature
If time allows

ID	Function Requirement	Category	Hours
1	As a user, I would like to be able to create a *FinanceFriend* account	L	2

2	As a user, I would like to login to my Finance Friend account.	L	2
3	As a user, I want to be able to reset my password.	L	1
4	As a user, I want to be able to delete my *FinanceFriend* account.	L	1
5	As a user, I want to be able to update my login information.	L	2
6	As a user, I would like to be greeted with an overview of my finances.	UI	4
7	As a user, I would like for my data to be saved for the next time I login to *FinanceFriend*	B	5
8	As a user, I would like there to be a dedicated page for investment information.	UI	2
9	As a user, I would like there to be a dedicated page for tracking bill deadlines and other important dates.	UI	2
10	As a user, I would like a calendar interface on the tracking page so I can easily see how many days I really have left.	UI	2
11	As a user, I would like to be able to add new bills and their deadlines to the project and see them on the tracking page.	F	3
12	As a user, I would like to generate graphs to see my percentages of spending within my budget in a new way.	UI	3
13	As a user, I would like to be able to create a new budget with a custom name and categories.	F	3
14	As a user, I would like to see the percentage and exact amounts I have spent in each category.	F	2
15	As a user, I would like to create a target spending amount in each category and see how close I am to the target as new spending information is added.	F	4
16	As a user, I would like to see each transaction in the budget section in an itemized list/table format.	UI	3
17	As a user, I would like to track the prices of my investments in the investments section.	F	6
18	As a user, I would like to generate graphs to see the percentages of my total investment portfolio.	UI	3

19	As a user, I would like to see how my investment prices have changed over time and see the percentage growth.	F	6
20	As a user, I would like to be able to create spending goals and have a system notify me if I achieved the goals or not.	F	5
21	As a user, I would like to be able to see a history of the places I visited.	UI	3
22	As a user, I would like the option to get notified at a time of my choosing when to enter my expenses at stored locations.	F	3
23	As a user, I would like to enter how much I spent at the places I visited in my location history.	F	3
24	As a user, I would like the option to get notified to enter my expenses whenever I leave a location.	F	3
25	As a user, I would like to be able to use *FinanceFriend* on multiple devices.	B	4
26	As a user, I would like to get specific feedback on my spending and areas where I can save	F	4
27	As a user, I would like to be able to add/omit any spending data from the budget tracking	F	2
28	As a user, I would like to customize the dashboard to prioritize the information that matters most to me.	UI	3
29	As a user, I would like to see how my spending would look on multiple different credit cards	F	4
30	As a user, I would like to be able to set specific saving goals for big purchases	F	2
31	As a user, I would like to receive warning alerts if my spending has reached a certain threshold for the different categories	F	3
32	As a user, I would like the option to enable or disable push notifications for the app.	F	2
33	As a user, I would like to be able to personalize push notifications based on category and how much my budget has been exceeded.	F	2
34	As a user, I would like the option to see my expected future costs for certain categories based on data from previous	B	5

	periods.		
35	As a user, I would like tips on how to use a credit card to keep my credit score from getting low	F	2
36	As a user, I would like to highlight certain transactions in the itemized list that were not planned.	F	2
37	As a user, I would like there to be different colors for spending categories shown in the generated graphs	UI	1
38	As a user, I would like to see different colors for investments based on overall ROI results.	UI	1
39	As a user, I would like to be able to request my friends to see their calendars.	F	4
40	As a user, I would like to generate graphs based on the expected future spending reports.	F	2
41	As a user, I would like to see tips on what stocks to invest in based on my level of risk.	F	1
42	As a user, I would like to be able to request my friends to see their budgets.	F	4
43	*As a user, I would like my budget to be exportable to a CSV-type format*	B	4
44	*As a user, I would like an encouraging summary at the end of the month telling me about my spending habits*	F	6
45	*As a user, I would like there to be a social hub where I can share my spending goals with my friends on the app*	UI	2
46	*As a user, I would like to add fellow users as friends*	F	3
47	*As a user, I would like to be able to comment on my friend's goals and their progress*	F	3
48	*As a user, I would like to be able to make posts on the social hub that my friends can see*	F	3
49	*As a user, I would like there to be a dedicated page for creating budgets.*	UI	2
50	*As a user, I would like the ability to direct message users that are my friends.*	F	3

51	*As a user, I would like the ability to block other users.*	F	1
52	*As a user, I would like to be able to see what money-saving challenges my friends are participating in.*	F	3
53	*As a user, I would like to be able to export an overview of my spending suggestions and analytics to a PDF format*	F	3
54	*As a user, I would like to see a color indicator for how effectively I am managing my budget.*	UI	1
55	*As a user, I would like to see cohesive branding across the app in the form of images, stylized text, and quality logos.*	UI	5
56	*As a user, I would like to see animations when switching pages.*	UI	3
57	*As a user, I would like to see a loading animation when logging in.*	UI	4
58	*As a user, I would like to see a list of challenges for me to join based on what I need to save money on*	F	3
59	*As a user, I would like to join a challenge to save money with other users*	F	3
60	*As a user, I would like to add a short description to my account describing what my financial goals are*	F	1
61	*As a user, I would like to add a profile picture to my account*	F	2

Non-Functional Requirements

Usability

The user interface of FinanceFriend should show the user's budget, expenses, and any investments graphically when they open the application. It should then be capable of editing expenses and budgets, along with more detailed information on their investments. The application will also make manual entry of expenses as easy for the user as possible, like using location to ask the user how much they spent at particular places. FinanceFriend will also be available on a variety of devices, so it should be able to work properly on all of them.

Performance

Managing personal finances often feels like a mundane task, which is why offering a finance application geared toward productivity perfectly aligns with user expectations. Navigating the application and editing fields should be a speedy, enjoyable experience. Hence, adopting a

high-speed framework like Flutter and placing emphasis on optimizing loading times emerge as pivotal factors in ensuring a good user experience.

Scalability

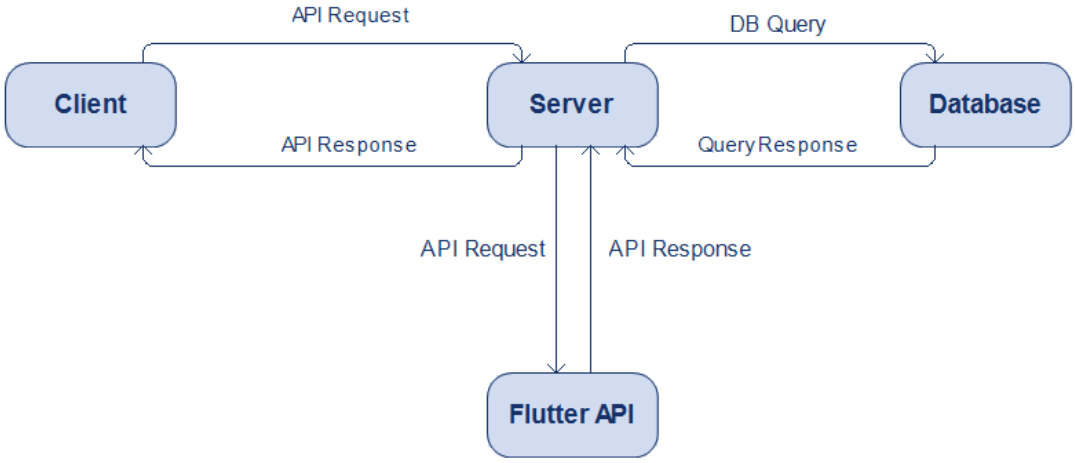
To ensure the long-term success of Finance Friend, it must be designed and developed with the capability to seamlessly handle increased user loads and data volumes. This includes the ability to efficiently scale both horizontally and vertically, ensuring that as the user base grows, the app remains responsive and performs optimally without significant degradation in speed or functionality. Scalability considerations should also extend to the underlying infrastructure, ensuring that servers, databases, and other resources can be easily expanded or upgraded to accommodate future growth, providing users with a reliable and enjoyable budgeting experience regardless of the app's popularity and usage levels.

Security

Ensuring the confidentiality, integrity, and availability of user financial data is of utmost importance. The app should employ robust encryption mechanisms to protect sensitive information during storage and transmission. Additionally, stringent access controls and authentication measures should be implemented to prevent unauthorized access to user accounts. Prioritizing security will not only safeguard user trust but also ensure compliance with relevant data protection regulations, making Finance Friend a safe and reliable platform for personal financial management.

Design Outline

Client-Server Model And Component Interactions



Our application will use the client-server architecture.The client makes requests to the server which, depending on the request, may query data from the database before returning the information to the user. We will also use Flutter's built in API's and libraries to help setup a local development server for ease of use and testing purposes.

User Client:

a.) The client will primarily be run for desktop, in the user's browser, and shall serve as the primary interface for our project.
b.) The client will request information from the database, utilize APIs from Flutter, and have information returned and distributed to them by the server.
c.) Depending on response validation, the views displayed to the client will change accordingly.

Server:

a.) Used to connect and distribute user account info to other users.
b.) Primarily local development server based on Flutter's "run" build command to deploy dart code in a chrome browser environment.
c.) If time allows, we may transition over to an external host like AWS.

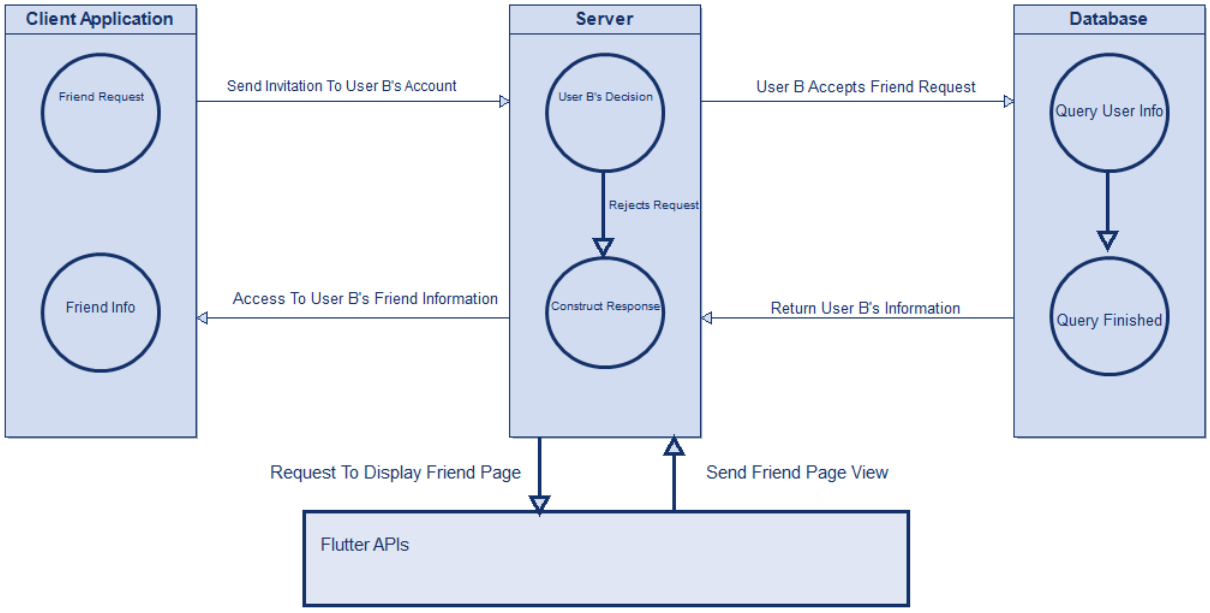
Database:

a.) The SQLite database will hold all relevant user account information like budgets, bill deadlines and the user's friends.
b.) The database will receive queries, and store information inputted by the users.

Third-Party APIs:

a.) Flutter APIs: These APIs and libraries will enable us during the development periods to leverage unique functions and modules regarding database usage with dart and SQLite.
b.) Google APIs: These will be utilized to help enable location tracking at time of purchases to help provide insights for the user's financial situations.

Event Sequence



The diagram above shows the high level structure of our application through the example of a friend request system. When User A sends a friend request to another user (User B), this request is handled by the server so different clients can interact with each other. Through the server, User B can choose to either accept or reject User A's request. If they reject, then the server's request to gain access to the friend page view tab will be denied by the Flutter API and the notice that the friend request was rejected by User B is sent to User A from the server. However, if User B accepts the request, the server goes to the database and queries for User B's relevant friend information, and returns the information to the server. The flutter API will also grant access to the new "friends page" tab since they have a friend, and User A is notified by the server that User B is now a friend and that they can access relevant friend information from User B.

Design Issues

Functional Issues:

1. What features should we focus on in our implementation?
    a. **Budgeting**
    b. **Investing**
    c. **Bill Tracking**
    d. Social Network

We decided to implement all of the features in a, b, and c and also to implement a social aspect to our app if we have enough available time. We of course need to implement budgeting features in our app as it is a budgeting app, but we also decided to add bill tracking and investing pages because these are also important aspects of personal finance. The social network aspect of out app will be added as a sort of bonus feature if we have enough time. We figured users may appreciate the added motivation to achieve their finance goals if they are able to see others doing the same.

2. What style of framework should we design our app around?
    a. **Client <-> Server <-> Database**
    b. User <-> Local Database

We decided to use a client, server, and database structure for our app as opposed to a purely local structure. This is primarily because our app will need a server and nonlocal database for the social network component and we believe it will be easier to simply implement this structure now instead of starting with a purely local structure and potentially having to add a server and online database component when implementing our planned social network features.

3. How should our app collect user budgeting and investing data?
    a. Connect to user's financial accounts
    b. **User input**

We decided to only allow users to input their financial data personally. We considered the option of allowing users to connect their banking and investing accounts to the app but realized that there would be the potential for issues with privacy law if we got the data directly from a user's accounts.

Non-functional issues:

1. What development kit should we use to develop our app?
    a. React
    b. **Flutter**

c. VueJS

We decided to develop our app using Flutter. Both Flutter and React are excellent platforms and although several of our team had prior experience with React, we decided that the new experience gained by working with Flutter would be valuable for us. We also considered the available resources for both Flutter and React and found them to be comparable, and so made our choice based on which would give us the most future benefit.

2. What platform should our app be usable on?
    a. Mobile
    b. Web
    c. Both

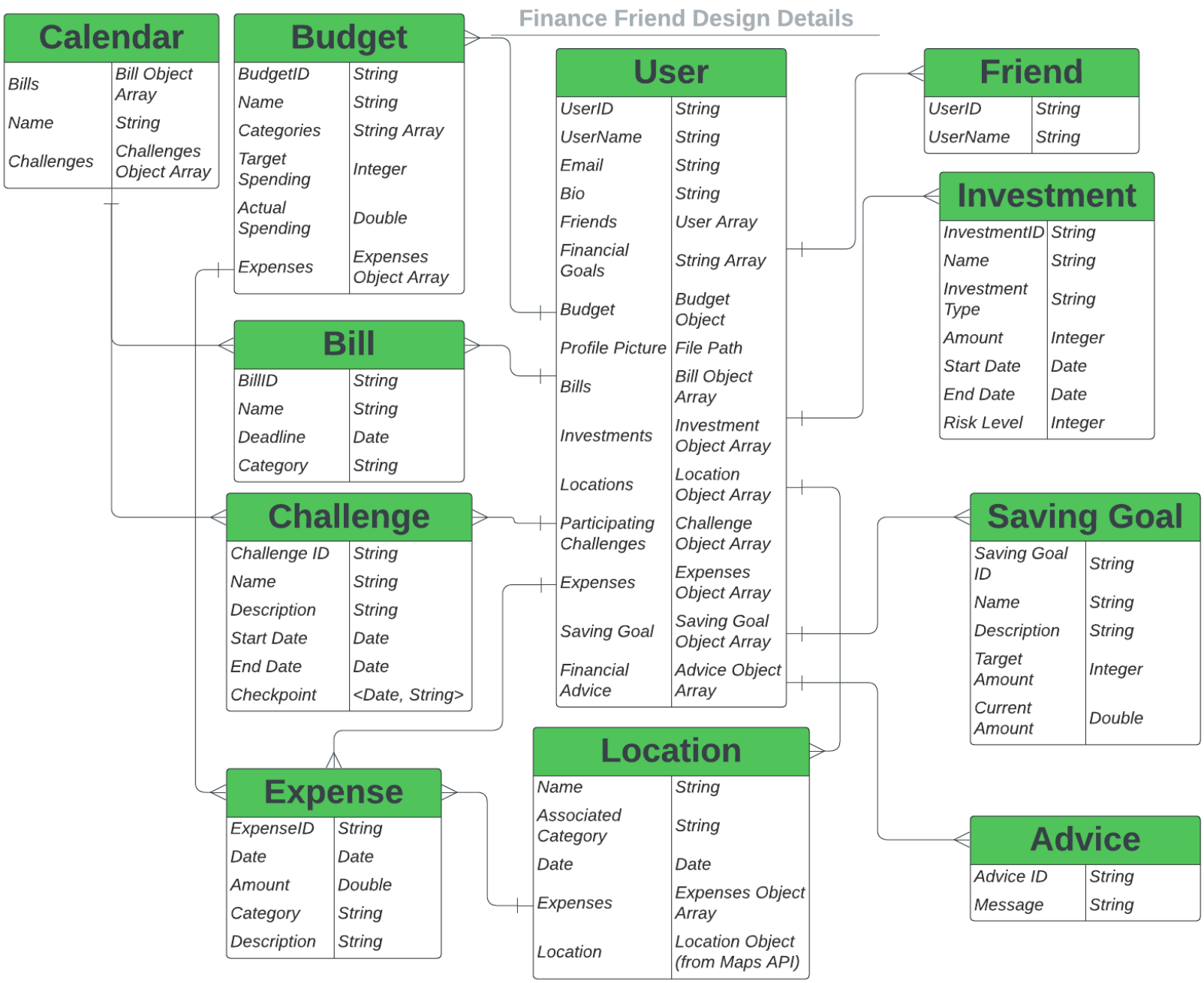
We decided to develop our app primarily for web platforms. We figured that if we prioritize the quality of our app over platform accessibility then we would be more happy with the resulting product. We chose web specifically for this because we believe that it will be much easier to develop than mobile. We may also port the app to a mobile version later if we have enough time.

3. What type of database will we implement?
    a. MongoDB
    b. SQLite
    c. Firebase

We decided to implement a SQLite database for our app. We settled on SQLite because we found documentation for integrating SQLite with Flutter. We also had a team member with experience using SQLite who found it to be simple to work with.

Design Details

Data Class Level Design



Description of Data Models/Classes and Interactions

User Class:

- Represents application users.
- Interactions:
    - Manages bills, investments, budgets, transactions, locations, challenges, friends, notifications, saving goals, and financial advice.
    - Participates in challenges, receives notifications, and sets preferences.

Bill Class:

- Represents bills to be paid.
- Interactions:
    - Belongs to a user.
    - Can be added, deleted, and tracked in budgets.
    - Can be seen in a calendar

Investment Class:

- Represents financial investments.
- Interactions:
    - Belongs to a user.
    - Calculates Risk of Investment and tracks investment performance.

Budget Class:

- Represents user-created budgets.
- Interactions:
    - Belongs to a user.
    - Contains categories, expenses, and generates graphs.

Expense Class:

- Represents financial transactions.
- Interactions:
    - Belongs to a user, budgets, and locations.
    - Helps categorize user spending.

Location Class:

- Represents visited locations.
- Interactions:
    - Belongs to a user.
    - May trigger expense notifications.
    - Contains expenses and the associated budget categories based on the type of location

Friend Class:

- Represents user's friends.
- Interactions:
    - Forms friendships with other users.
    - User ID can be used to find similar challenges, saving goals, etc. to be visible to user

Saving Goal Class:

- Represents saving goals for big purchases.
- Interactions:
    - Belongs to a user.
    - Tracks progress toward financial goals.

Challenge Class:

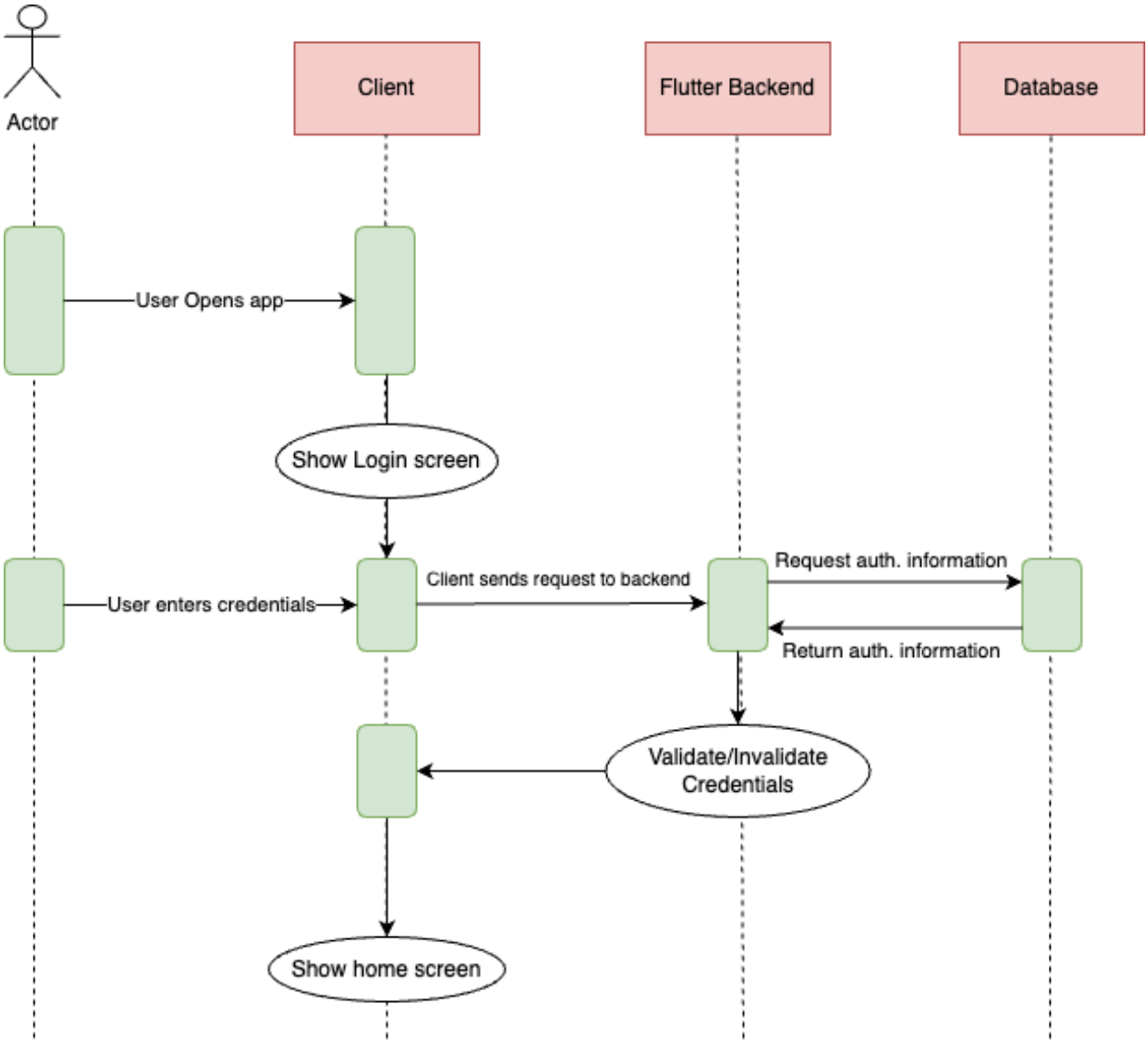
- Represents a money-saving challenge for a user to participate in.
- Interactions:
    - Belongs to a user
    - Tracks progress and status.
    - Tracks start/end and checkpoints to add to user’s calendar
    - Involves challenge participants.
    - Shows up on the user’s bill calendar if preferred

Advice Class:

- Provides financial advice to users.
- Interactions:
    - Advice for a user is available to them.
    - Offers tips and recommendations on how to budget and build good financial habits.

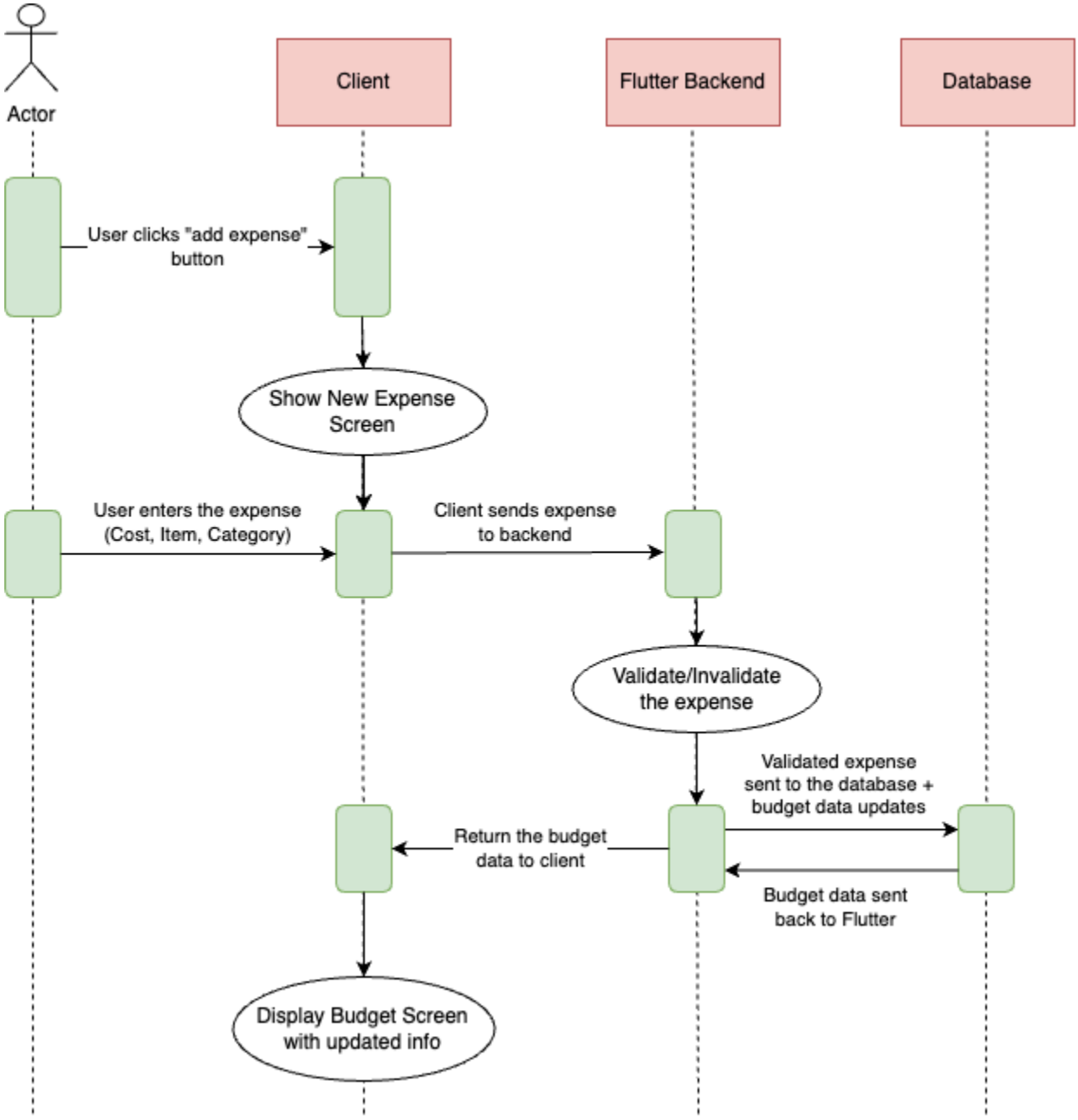
Sequence Diagrams

User Login



This diagram depicts a user logging into FinanceFriend. Once a user opens the app, they are greeted with the sign in screen, where they enter their credentials. The client then sends the credentials to the Flutter backend, which then requests the authentication information for the account to the Database. Once it receives the authentication data back, it validates the credentials, then opens up the home screen on the client side.

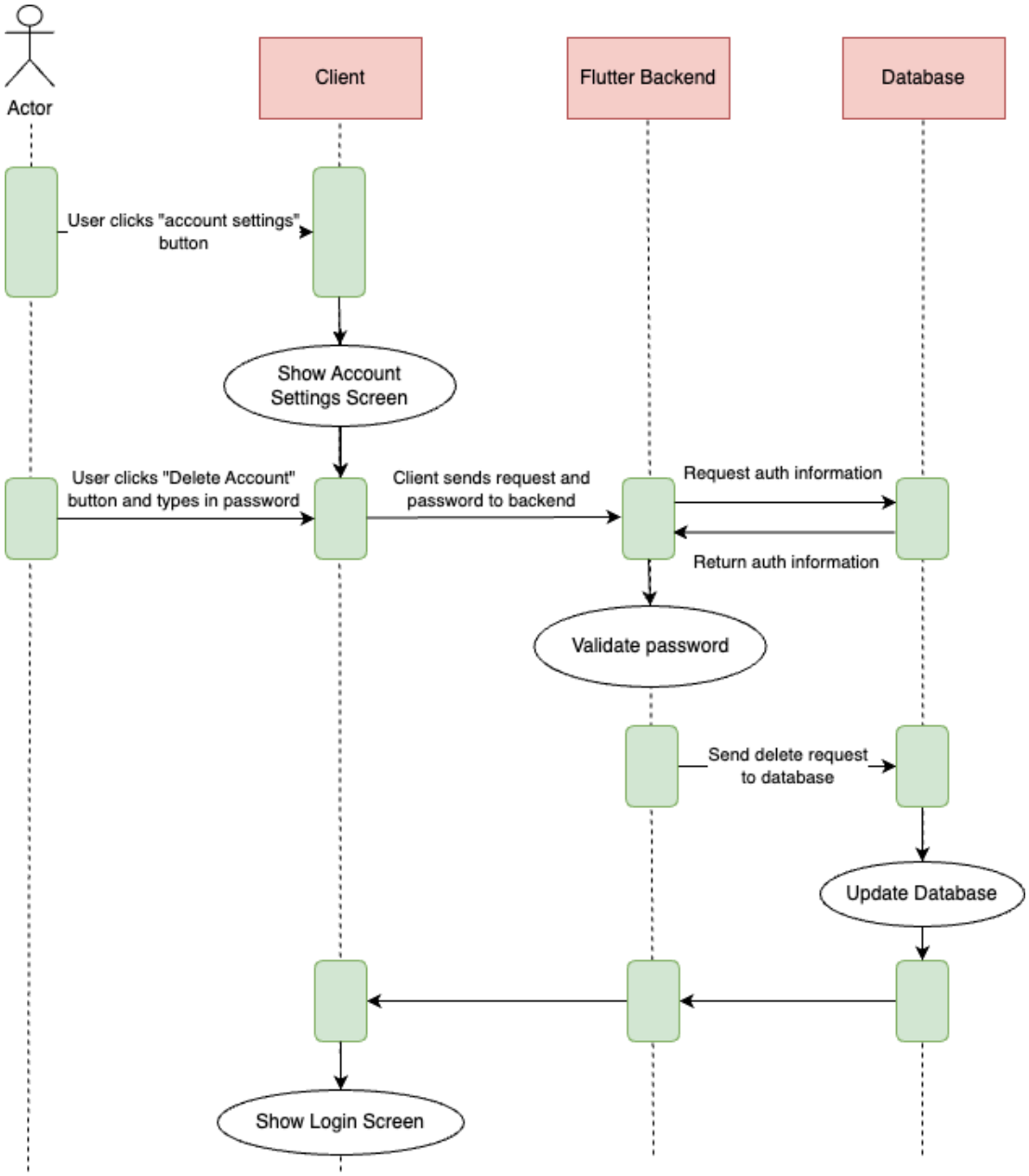
Adding an expense



This depicts a user adding an expense to their budget tracking service. Once a user clicks the “add expense” button, they are prompted by the client to input the expense data (cost, item, category, etc.). Once the data is inputted, it is validated on the backend before being sent to the database. After it is stored in the database, the budget data is updated and sent back to the backend, which goes back to the client. Once the data is client side, the budget screen

displays with the latest information, showing how the new expense impacted the existing budget.

Delete Account

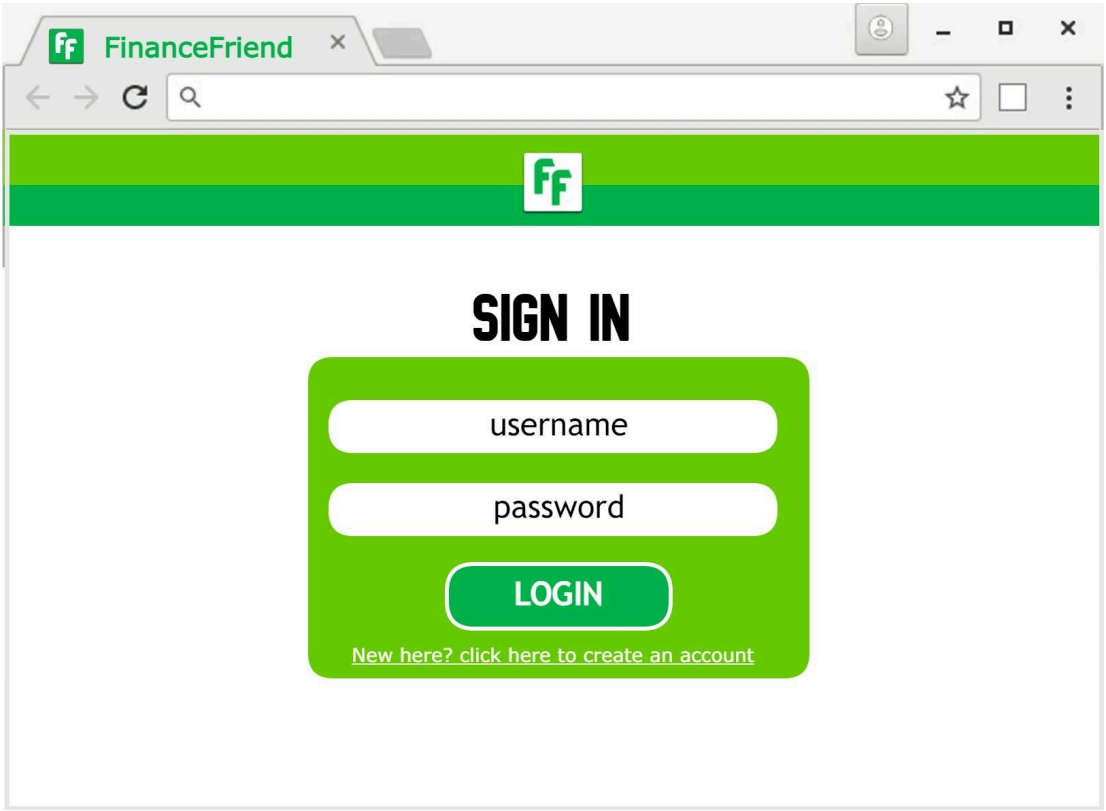


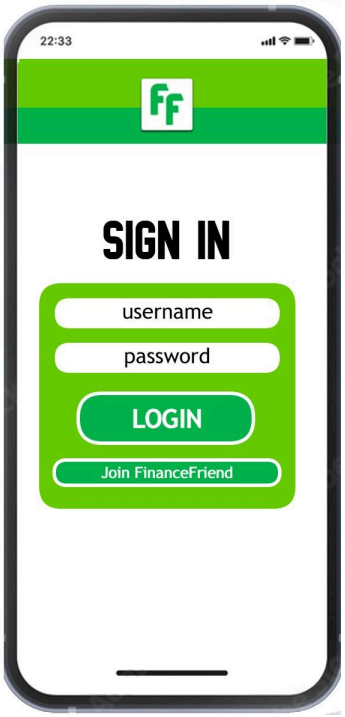
This diagram depicts a user deleting their account and removing their information from the database. It starts with the user clicking the “account settings” button, which shows a screen with several account settings options. The user then clicks the “Delete Account” button and enters their password, which is authenticated by the backend and the database. Once it is

validated, the backend sends a deletion request to the database, which removes the account from the database. After the database is up to date and the account is removed, the Login Screen is displayed.

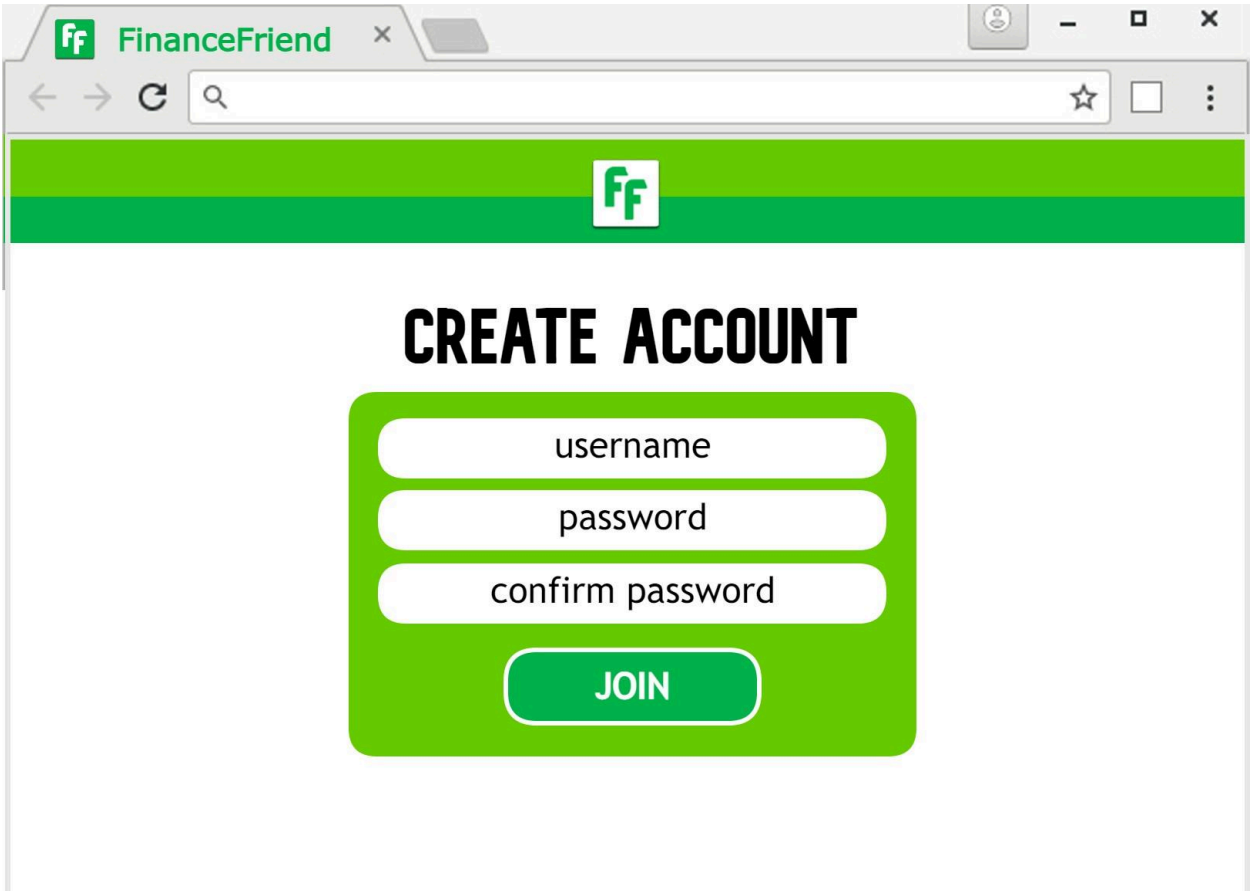
UI Mockups

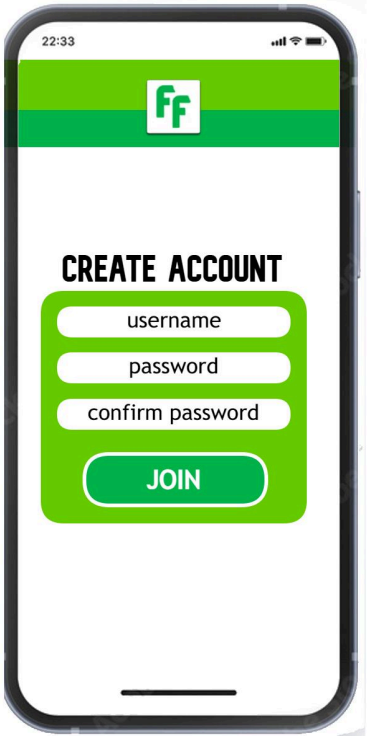
Login Screen (web and mobile)





Create Account (web and mobile)





Budget Dashboard (web and mobile)