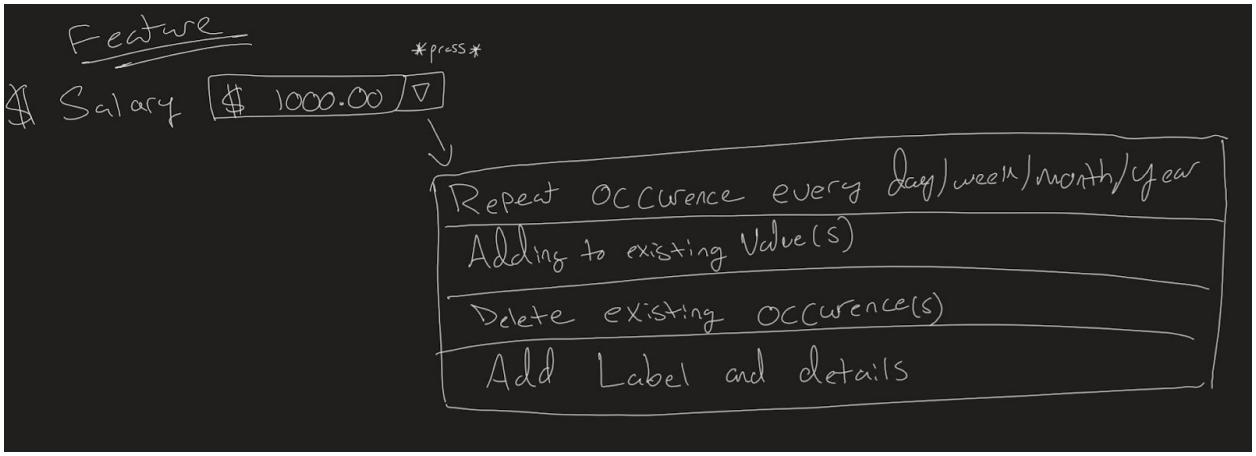
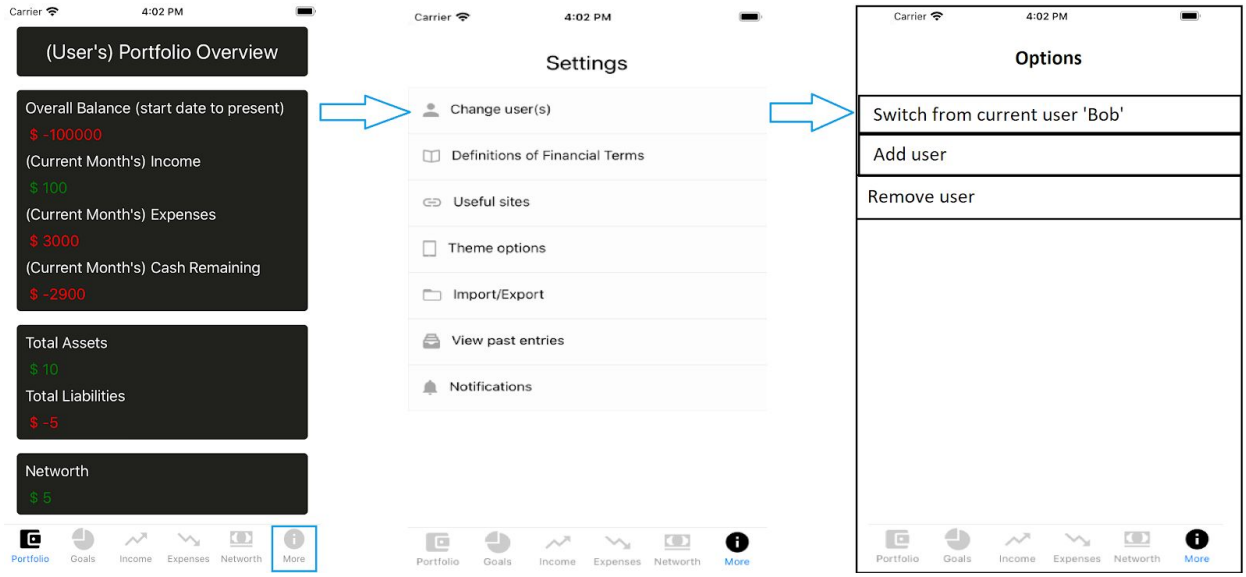
Low-Fidelity Prototyping and Heuristic Evaluation

Brainstorming session results:

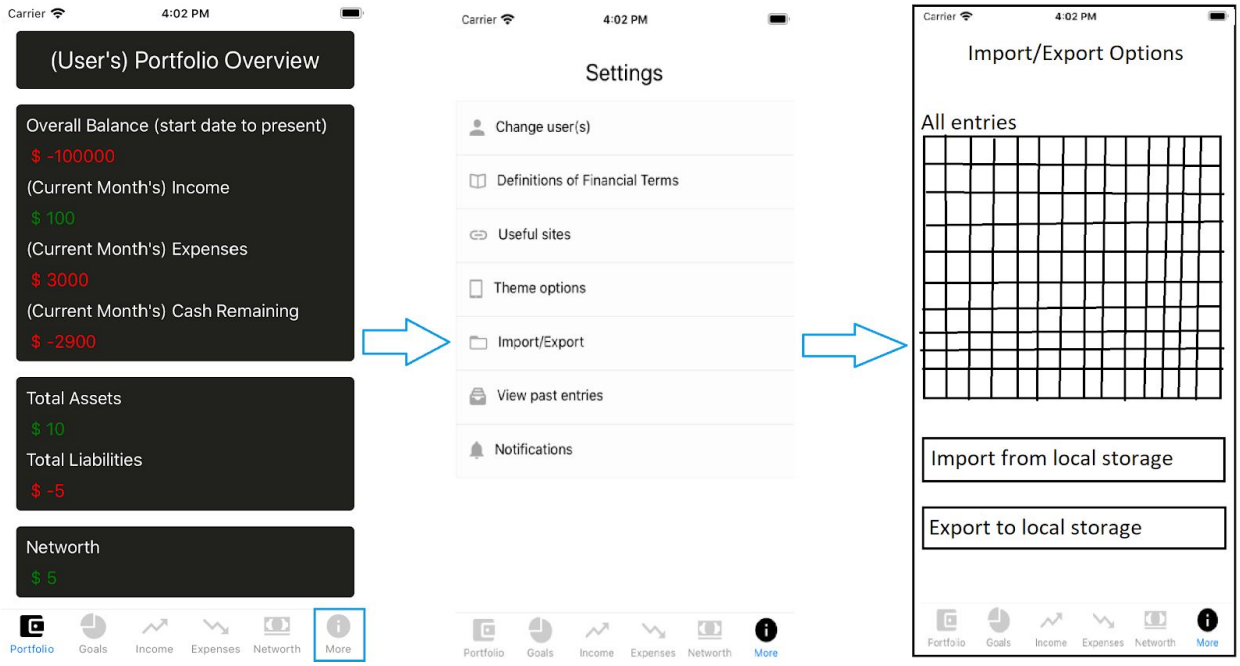
Task: Able to modify routine expenditures and income
Omar:



Task: Add and Manage Multiple Accounts
Omar:

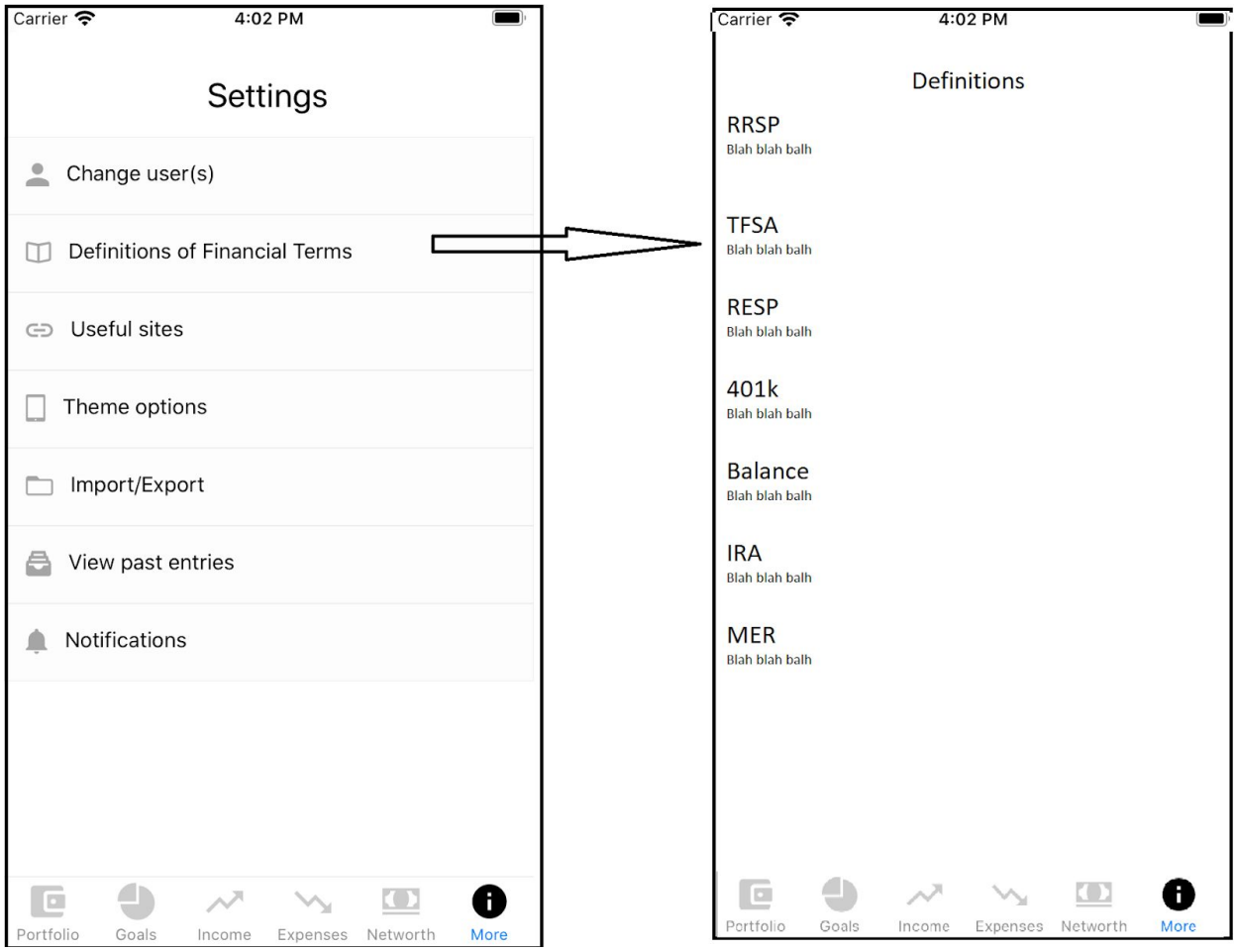


Task: Export monthly expenditures and income
Omar:

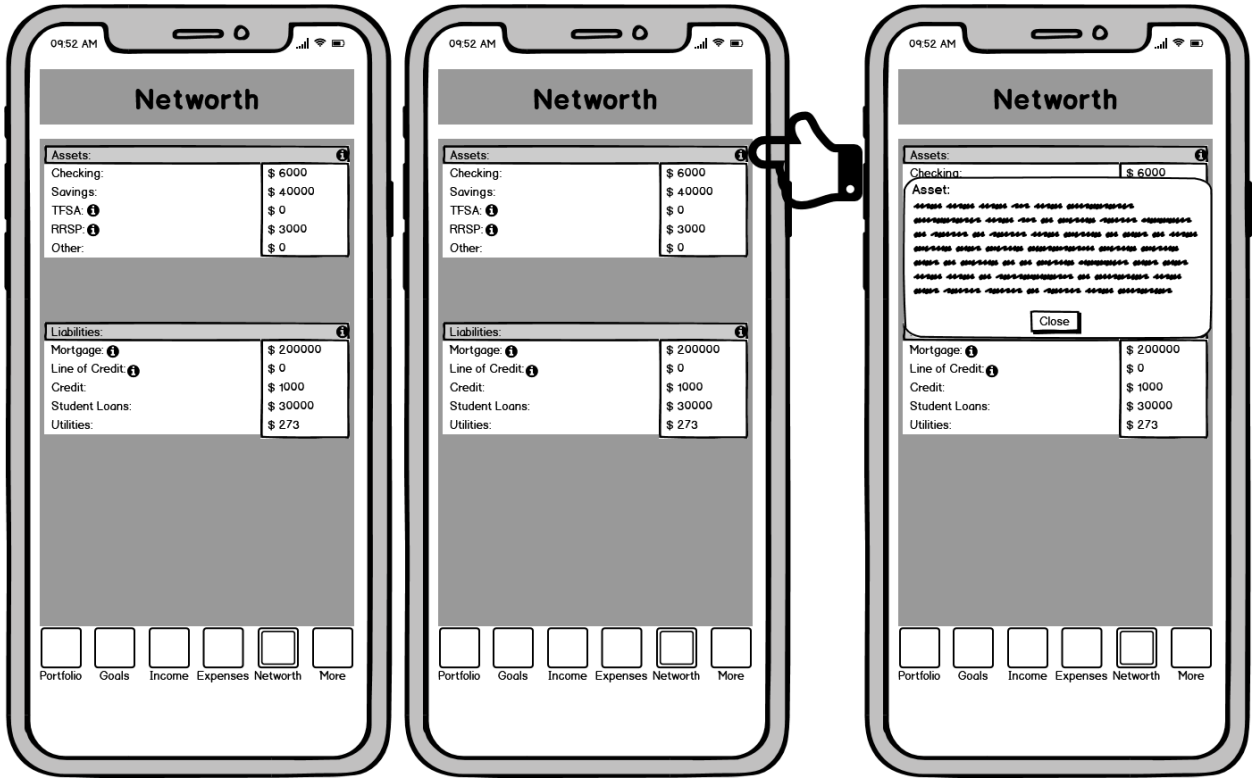


Task: Checking definitions of financial terms

Omar:



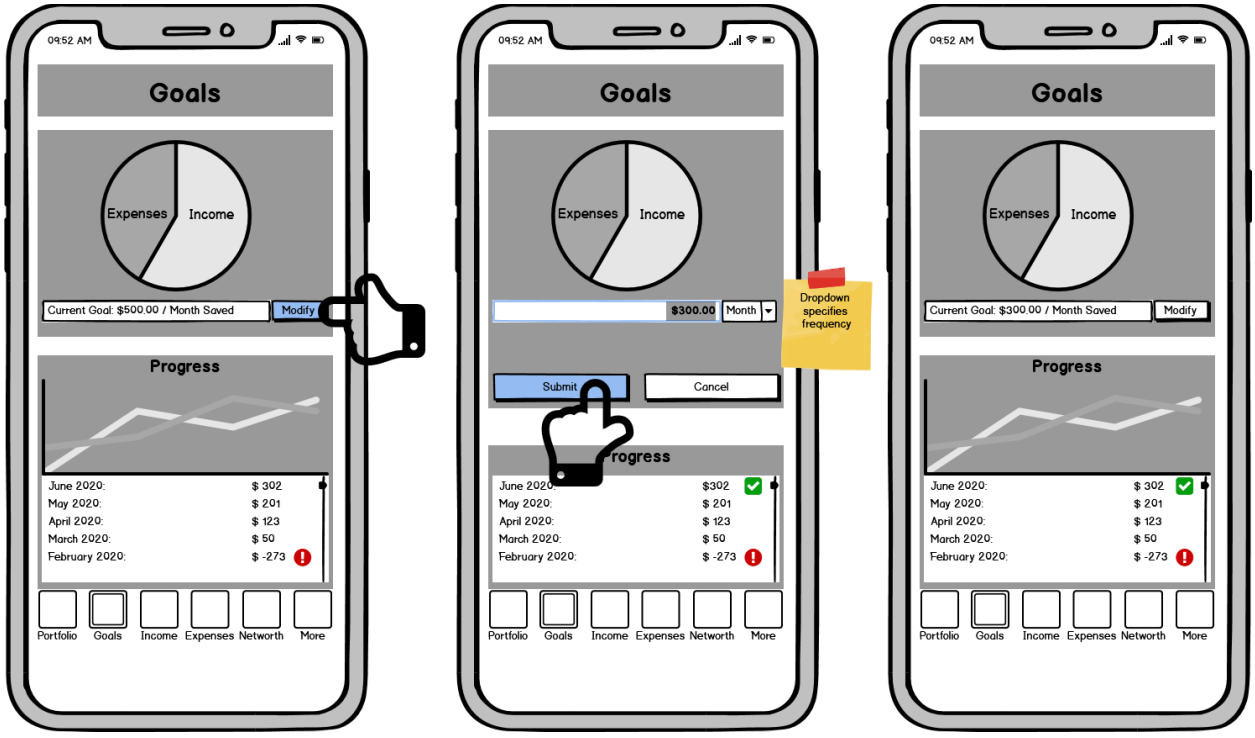
Cameron:



Other Tasks:

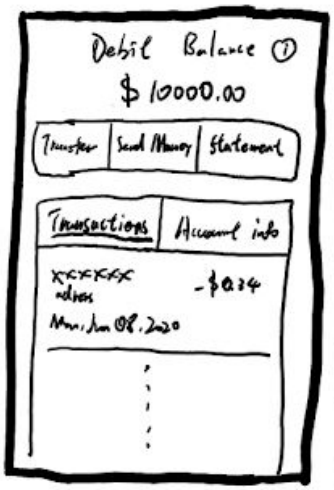
Specify Savings Goals:

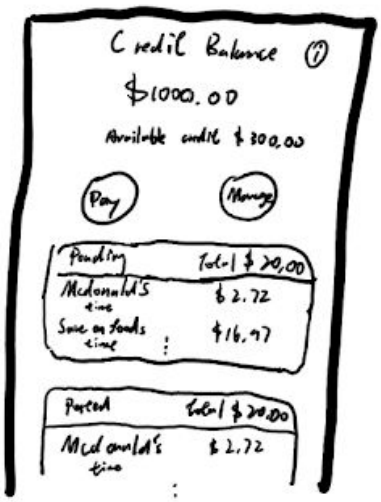
<u>Cameron:</u>

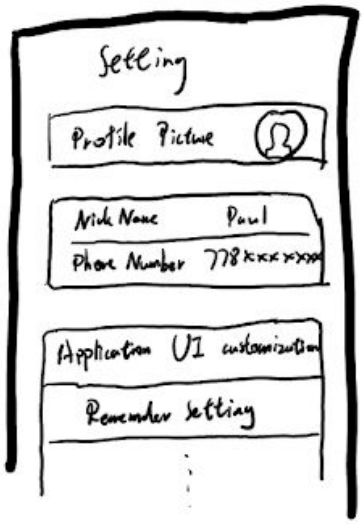


Add Recurring Income/Expense

Set login and multi-user function, and draft the main page
Add optional investment page
Add the detail of expenses, and the detail of credit pages
Set UI customise, set reminder, and customise portfolio functions on the setting page
Paul Zhang:

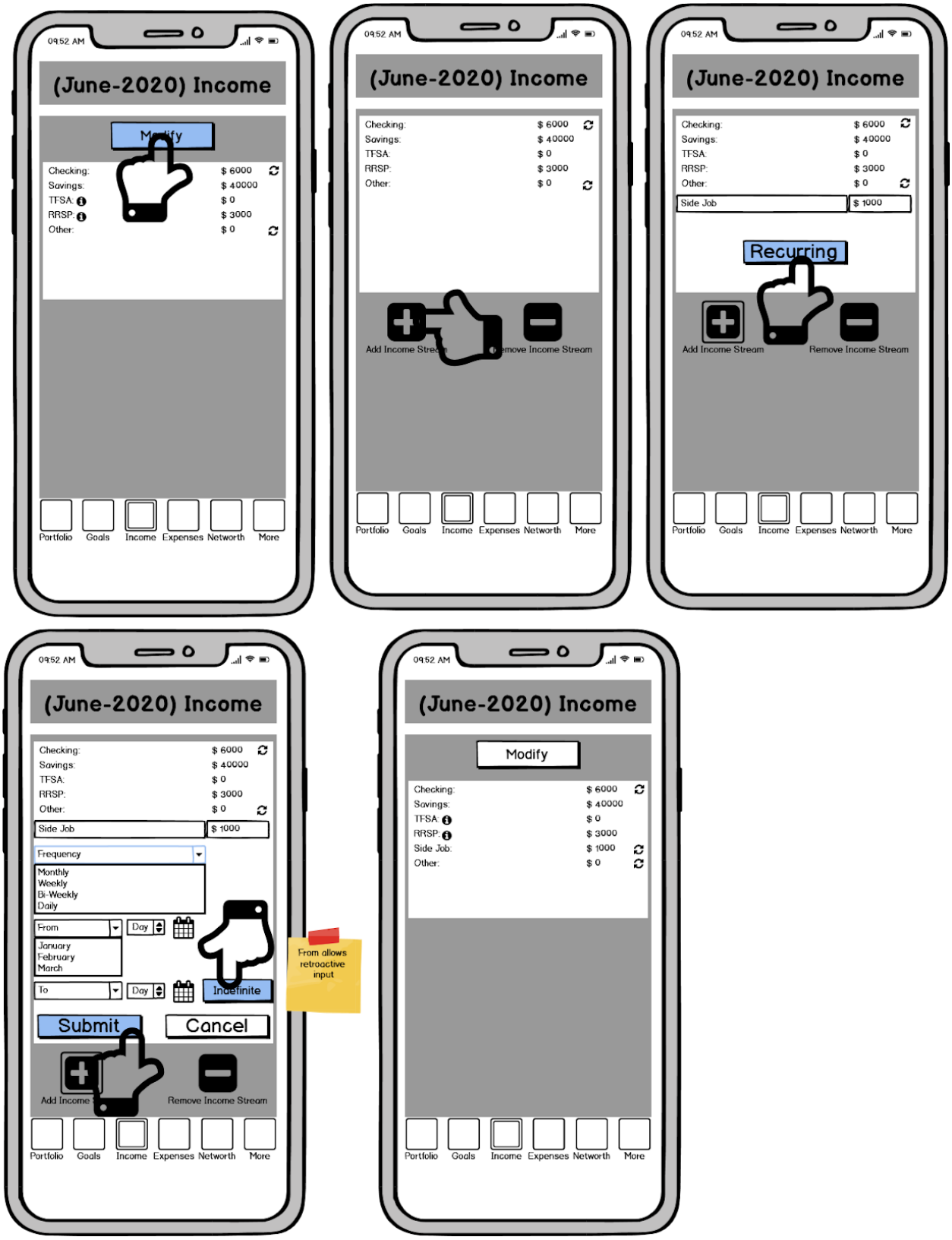




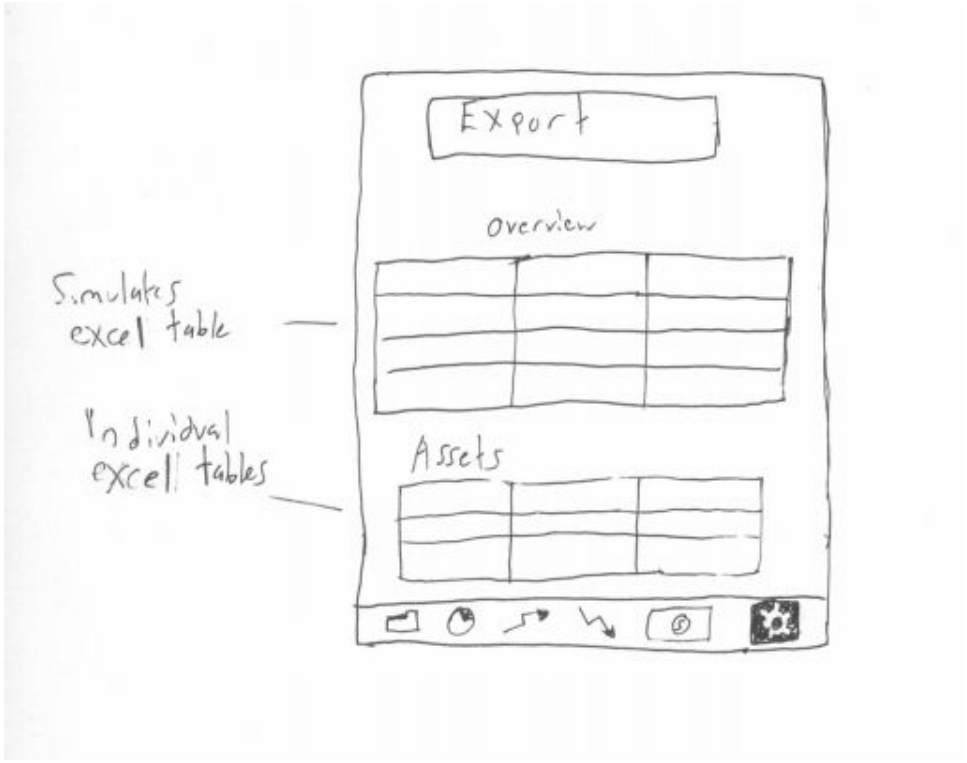


:

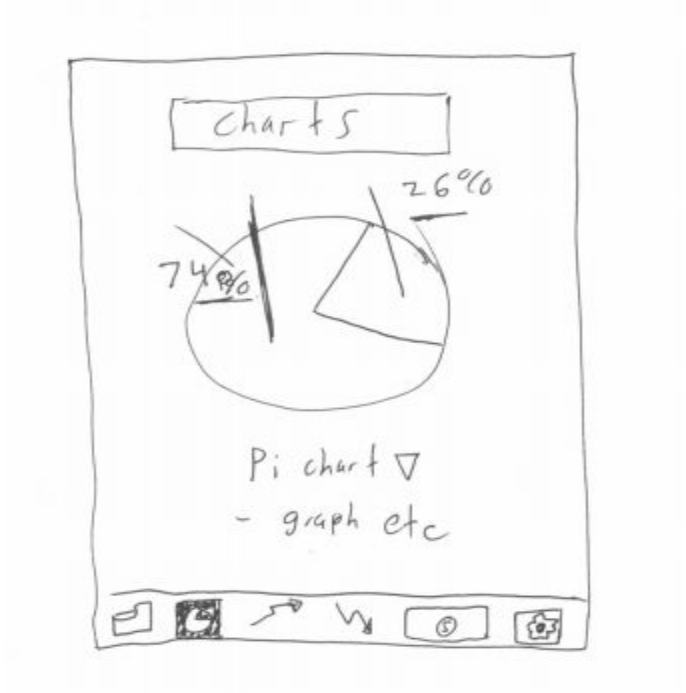
Cameron:


09:52 AM
(June-2020) Income
Modify
Checking: $ 6000
Savings: $ 40000
TFSA: $ 0
RRSP: $ 3000
Other: $ 0
Portfolio Goals Income Expenses Networth More
Add Income Stream
Remove Income Stream
Side Job $ 1000
Recurring
Frequency
Monthly
Weekly
Bi-Weekly
Daily
From
Day
January
February
March
To
Indefinite
Submit
Cancel
From allows retroactive input

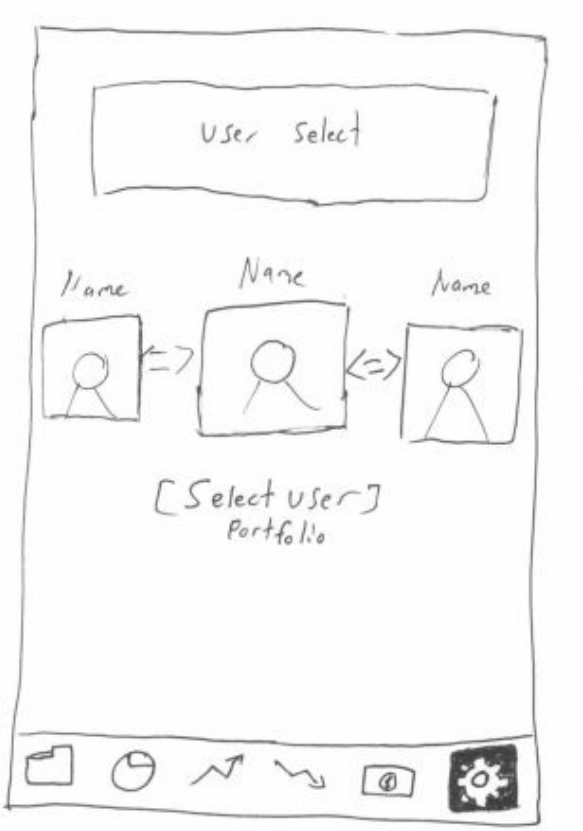
Kelvin:
Export Monthly Expenditures and Income



Kelvin:
Charts graphs



Kelvin:
Add and Manage Multiple Accounts



Kelvin:
Able to modify routine expenditures and income

After our brainstorming process, we reflected on which task was the most promising. Through a democratic approach, we agreed that the ability to modify recurring/non-recurring income/expenditures is of high importance for all users and requires the most attention to detail on our part. In relation to our user research, this feature was in our “absolutely must include” for design requirements. Unfortunately, users would have to modify their income/expenditures manually since our application won’t link to any outside institutions due to privacy concerns and financial constraints. On the other hand, users will be able to leverage security of their data, ease-of-use, and customizability. Customizability includes the following features:

- Add income/expenditures
- Delete income/expenditures
- Make income/expenditures recurring
    - Set entry to repeat every day/week/year until day/month/year
- Label custom entries
- Add entry descriptions

The features we plan to implement are quite feasible. The frontend needs to design an appealing modification screen for given income/expenses. The backend would need to develop a way to link features from the frontend so that all entries are securely stored into our exportable spreadsheet. The ability to modify spreadsheet entries is nothing novel, but our implementation will have its own unique flair when compared to other budgeting applications on the market.

The brainstorming process involved an evolutionary approach. We first looked at our initial mockup of the app and iteratively changed it to incorporate each peer's interpretation of improved design. The incorporations were based on the most intuitive ideas, but only for our storyboard was the most promising chosen. Intuitive ideas had great appearance, functionality, interactivity, and spatial structure. By modularity, we built a system (the app) and had separate components (tasks) connected. These components were sketched separately and the best ones will be combined into the final system. For one example of existing works we compared with, *Fudget* on the Apple iOS store was our inspiration on what not to do with modifying income/expenditures. We found their design unintuitive and unappealing, so we desired to do the opposite of that. Our design will have more features, its own modification screen, and cleaner design.

Reflecting on our brainstorming process, we found that spitballing a bunch of tasks and deciding what to sketch worked well. Difficulty arose when we had to decide how much time we wanted to put into sketching so many ideas and which ideas to sketch. Time constraints leave us with limited options. For next time, we would like to reduce the amount of ambiguous ideas, encourage criticism, and focus on breadth.

Storyboard and video prototype:

Financial Management App Storyboard

Cameron Lindsay, Omar Kawach, Cameron Drummond, Paul (Shuhao) Zhang, Kelvin Leung

University of Victoria

Persona: Bob the Builder, 40, works in construction as a general handyman.

Scenario: Bob has taken on a side-job that pays him every 2 weeks until December 2020. He wants to add this revenue stream to his previously tallied incomes.

Portfolio Overview:

Following login, forwarded to the Portfolio Overview screen.
Select 'Income' screen

June 2020 Income:

Select the 'Modify' option

Recurring Income:

Select the option to 'Add Income Stream'

Field Options:

Fill in specified information into the desired fields

Confirm Info:

Review information and select 'Submit'

Return to Income:

New income information has been updated

Audio for video:
https://www.youtube.com/watch?v=XCcN-IoYIJA&list=PL44UysF4ZQ23B_ITIqM8Fqt1UXgsA9yD6

OVERVIEW:

1. Login Screen, select account
2. See portfolio overview
3. Select Income screen
4. Select "Modify"
5. Select "Add Income Stream"
6. Fill in desired information
7. Select "Recurring"
8. Fill in desired recurrence information (fixed time)
9. Select "Submit"
10. Select Expense screen
11. Select "Modify"
12. Select "Add Expense Stream"
13. Fill in desired information

14. Select “Recurring”
15. Fill in desired recurrence information (Indefinite)
16. Select “Submit”

- Add concise descriptions of what the prototype is illustrating

Heuristic evaluation results:

Individual evaluations:

Omar

List of violations
- 3. User controls and freedom
- 10. Help and Documentation

Why list
- Usability heuristic #3 for UI’s is violated by the fact that the users are not asked to confirm their long operation of entering in recurring income/expenditures. Users may accidentally add an income or expenditure. This violation may result in lack of error prevention (6) as well.
- Usability heuristic #10 for UI’s is violated by the fact that there are no tips or tutorials provided to first time users who are entering in their first recurring income/expenditure

Severity rating
- Usability heuristic #3 for UI severity: 3, should be addressed
- Usability heuristic #10 for UI severity: 2, most users will likely figure it out

Cam

List of violations
- 8. Aesthetics and Minimalist Design
- 7. Flexibility and Efficiency of Use

Why list
- Usability heuristic #8 is violated when the remove button appears before an item is selected. This has no real impact on the usability of the app, but the button is essentially dead until the user selects an income to edit, thus it is an unnecessary button on the screen at that time which impacts the minimalism of the design.
- Usability heuristic #7 is violated when a user is adding a new income stream. The user enters the information and amount of the income, but then there is no other option but to make the income recurring. The user cant simply add the new income as a one time input. This violates usability heuristic #7 as it retracts from the flexibility of the application.

Severity rating
- Usability heuristic #8 severity rating: 1, it's a cosmetic problem, not serious

- Usability heuristic #7 severity rating: 3, this should be addressed

Kelvin:

List of violations:

- 3. User Control and Freedom
- 7. Flexibility and Efficiency of use

Why list

- There is no option to create a custom frequency duration which can make it inconvenient for some people with odd recurring scheduling. This violates the law of user control and freedom as it constricts the user's options.
- Checking previous monthly income from the current monthly income screen is not an option which may be convenient in some instances. This violates the flexibility and Efficiency of use law.

Severity Rating:

- Usability heuristic #3 severity rating 2, not serious as it appeals to a very niche market
- Usability heuristic #7 severity rating 2, it would make the app more usable but it only increases efficiency by a small margin.

Cameron

List of violations

- 8. Aesthetics and Minimalist Design
- 8. Aesthetics and Minimalist Design (2)
- 10. Help and Documentation

Why list

- Usability heuristic #8 is violated when the remove button is present prior to the selection of an item. While a minor issue, the remove button's presence detracts from the overall minimalistic design scheme.
- Usability heuristic #8 is violated when the add new remove button is present when an item is selected. Like the issue above, it is a minor issue and is only guilty of detracting from the overall minimalistic design scheme.
- Usability heuristic #10 the UI as a whole does not provide a tutorial or description of what each action does prior to selection.

Severity rating

- Usability heuristic #8 severity rating: 1, no impact on function, just cosmetic
- Usability heuristic #8 (2) severity rating: 1, no impact on function, just cosmetic
- Usability heuristic #10 severity rating: 2, user can likely intuit functions of each action. A tutorial could be implemented down the line.

Paul

List of violations

- 5. Help users recognize, diagnose, and recover from errors
- 7. Recognition rather than recall

Why list

- Usability heuristic #5 for UI's is violated by the fact that the name of the incomes/expenses does not contain enough detail for users to recognize the source of revenue and expenditure. Users might forget the details of how they spend money or get money from when they go back to check the statement after a long time. It will make users harder to recover from the errors when they manage their financial account.
- Usability heuristic #7 for UI's is violated by the fact that the function of the recycle icon in the income/expenses page is not clearly illustrated by the application. Some of the income/expenses has the recycle button beside it, but the others don't. It will make the use of the application hardly retrievable.

Severity rating

- Usability heuristic #5 for UI severity: 3, should be addressed
- Usability heuristic #7 for UI severity: 2, most user will figure it out, if the system could be improve

Group evaluation:

Key problems agreed upon as a group:

7) Flexibility and Efficiency of use

- Usability heuristic #7 is violated when a user is adding a new income stream. The user enters the information and amount of the income, but then there is no other option but to make the income recurring. The user cant simply add the new income as a one time input. This violates usability heuristic #7 as it retracts from the flexibility of the application.
- Severity: 3, major usability problem; important to fix.
- Difficulty of the fix: Easy, simply need to add a "Submit" button beside the recurring button when adding new incomes so users can make the new income/expense a one time event.

7) Flexibility and Efficiency of use (2)

- Checking the income of a previous month is currently not an option. This feature would make viewing past financial information convenient for users and may be desirable in some instances. This violates heuristic #7, the flexibility and efficiency of use.
- Severity: 2, minor usability problem.
- Difficulty of the fix: Moderate, a drop down list of past months or arrows to switch between months would need to be added, and the information stored from those months would need to be accessed and displayed.

8) Aesthetics and Minimalist Design

- Usability heuristic #8 is violated when the remove button appears before an item is selected. This has no real impact on the usability of the app, but the button is essentially dead until the user selects an income to edit, thus it is an unnecessary button on the screen at that time which impacts the minimalism of the design.
- Severity: 1, cosmetic problem.
- Difficulty of the fix: Easy, only need to change the timing of when the add/remove buttons are on screen. The “Remove” button should only be visible when an income/expense is selected for editing.

8) Aesthetics and Minimalist Design (2)

- Usability heuristic #8 is violated when the add new remove button is present when an item is selected. Like the issue above, it is a minor issue and is only guilty of detracting from the overall minimalistic design scheme.
- Severity: 1, cosmetic problem.
- Difficulty of the fix: Easy, only need to change the timing of when the add/remove buttons are on screen. The “Add” button should only be visible when no income/expense is selected for editing.

Closure

Each group member began by running through the interactive mockup in Balsamiq to understand the flow of the program. Afterwards, group members would perform individual heuristic evaluations to catch violations. When individual members had finished gathering their results, we compared our findings as a group to see which key problems should be addressed in our medium fidelity prototype. We found that working independently on the evaluations and then as a group was quite useful and brought forth novel observations that may have not been brought forth in a simultaneous viewing.

This strategy unfortunately leads to shortcomings in the users final interpretations. Observing the results of a low fidelity prototype describing a single task can result in communication errors not thought of by the prototype’s designer. It is difficult to convey all ideas of the designer in a limited view of the application. A low-fidelity prototype requires users to intuit features and make assumptions based on the application’s UI. This can result in users reporting that features do not exist, rather than have just not yet been drafted in the prototype. This is fortunately an error that can easily be resolved through the use of proper documentation throughout the prototype and direct communication with the designers and reviewers throughout the evaluation process.