Department of Computer Science & Engineering
The University of Texas at Arlington

System Requirements Specification
CSE 4317: Senior Design II
Spring 2020

The Budgeteers
WhatchamaBudget

Anil Karki
Bobby Kemp
Emily Knowles
Hedges Omordia
Roshan Shrestha

Revision History

Revision	Date	Author(s)	Description
0.1	10.20.2019	AK, BK, EK, HO, RS	document creation
0.2	12.10.2019	AK, BK, EK, HO, RS	document revised for version 2
0.3	04.28.2020	AK, BK, EK, HO, RS	document revised for version 3

Contents

List of Figures

1 Product Concept

This section describes the purpose, use, and intended user audience for the budgeting app. The app is a web-based application that helps its users to manage their money using a variety of money management techniques. The app also performs various forms analysis on the user's spending patterns in order to better help them save money. The major goal of the application is to have the user better manage their budget/goal and introduce different types of financial management techniques.

1.1 Purpose and Use

Users of the app will be able to see and track their spending/savings by linking various bank accounts to the app. Once linked, the app will receive information about the user's balance on their accounts in real-time or near real-time. Using this data, the user will be able to set up budgeting plans in the app and also inspect past trends of spending using the built-in visualization tools. Figure 1 below depicts the general flow of user data from the Plaid banking API, to the back-end application server, to the end user. The user will be able to create a goal and respective budget for that goal and at the same time track progress in the overview section. The overview section contains graphical representation of the overall transaction/budget/goal for better financial understanding.

1.2 Intended Audience

This product, at the time of writing, is expected to be made available commercially as an all-in-one budgeting system. Any person literate in basic computer usage with an interest in budgeting will be able to use the app with little money management training required. Also, audience with curiosity to learn about future goal planning and keeping track of it would have the perfect application to start with.

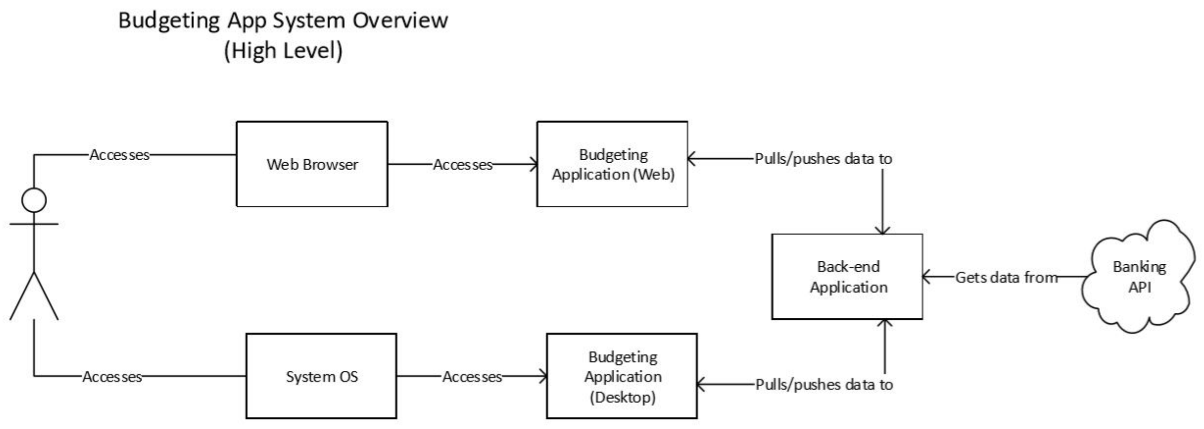


Figure 1: Budgeting app system overview conceptual drawing

2 Product Description

This section provides the reader with an overview of the budgeting app. The primary operational aspects of the product, from the perspective of end users, maintainers, and administrators are defined here. The key features and functions found in the product, as well as critical user interactions and user interfaces are described in detail.

2.1 Features & Functions

From a high-level perspective, the budgeting application is composed of two main parts: a front-end client-side application, and a back-end application server. The back-end application is a Django web application written in the Python programming language, and can be administered by an authenticated user in various ways, including changing database information (such as users). The back-end communicates with the Plaid banking API and the front-end application through the HTTP protocol, encoding transmitted data in the JSON format. The back-end is used for user and administrator authentication as well as persistent storage of user data. External elements to this project include the Plaid banking API and it's associated components (e.g. the drop-in JavaScript Link module).

The front-end of the application consists of all components of the product with which the user interacts directly. The front-end firstly consists of a user login screen for a user to authenticate with the application. The layout of the login screen has been changed since the one created in Senior Design II. As shown in Figure 2 below, the login screen will consist of branding for the application, such as the application name and logo, if applicable. There will be a text input field for a username and a password, as well as a button to submit this information to the back-end. There will also be a button to allow the user to initiate the process of a password reset (a "Forgot Password" button). The application also consists of the following components: a home page, an about page, a goals page, a budget page and an overview page where user can input transaction items either individually or in bulk.

As shown in Figure 3, the home page consists of various visualizations for past, current, and future budgeting information. It also displays recent user transactions and other information related to the user's set budgeting goals.

Figure 4 illustrates a basic layout for the settings page, which consists of various settings for the application, including various choices of budgeting styles with accompanying explanations. There is also an area to specify Transaction Rules. These will be used to classify/tag financial transactions into certain groups for easier user digestion. There is also an area in which the user can specify any Static Fees; fees that occur every so often that should be included in the budget by a certain point in time.

Figure 5 shows an example of the manual data entry page, which allows the user to enter transaction information that has not come from the back-end application. This will push it up to the back-end application so that it may be stored there and subsequently displayed back at the front-end for the user. The user may also import many transaction statements in bulk from a file on their device.

2.2 External Inputs & Outputs

The budgeting app expects to receive credentials from the user for initial account creation/login. After proper authentication, the app expects various input settings, such as type of budget to be used, types of projections to display, among other similar inputs. Moreover, the back-end application server expects to receive user credentials and send them to the Plaid API. It also expects to receive and relay to the front end information retrieved from the Plaid API. Besides that, the application expects user to input goals and budget information to set up their financial information.

Outputs for the system include charts and visualisations generated based on the user's budget types and account balance. The account balance itself is another type of output that is generated from the back-end via the Plaid API and displayed in the front-end.

Name	**Description**	**Use**
User Credentials (Input)	Username/password combination	Authenticate with the system
Application Settings (Input)	Settings used to determine application behavior	User sets settings to control how the application behaves
User Bank Account Information (Input)	A user's bank account information coming from the Plaid Banking API	Used to display user purchases/projections of user budget
Visualization Tools and Budgeting Reports (Output)	Various information concerning a user's budgeting information	Main product of the application for user consumption
User Bank Account Information (Output)	Output received from Plaid at the back-end app server and relayed to the front-end	Displayed to the user by the client-side code
Manual transaction (Intput)	Incase the user decides to add transaction that are not listed in the bank account	Comply with the overall transaction

2.3 Product Interfaces

User credentials will be entered at the login screen, depicted in Figure 2. The user will enter a combination of their username and password in order to authenticate themselves with the system. Alternatively, the user will be able to reset their password from the login page, or create a new account (not depicted in Figure 2).

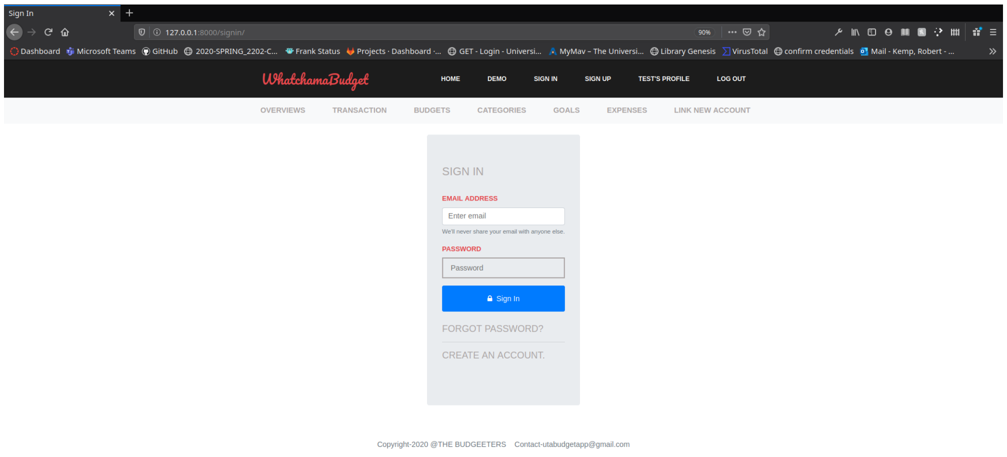


Figure 2: Application login screen

The home screen/overview screen, Figure 3, will be the source of virtually all application output to the user; it will contain various visualization tools and budgeting reports, and also display recent user financial transactions.

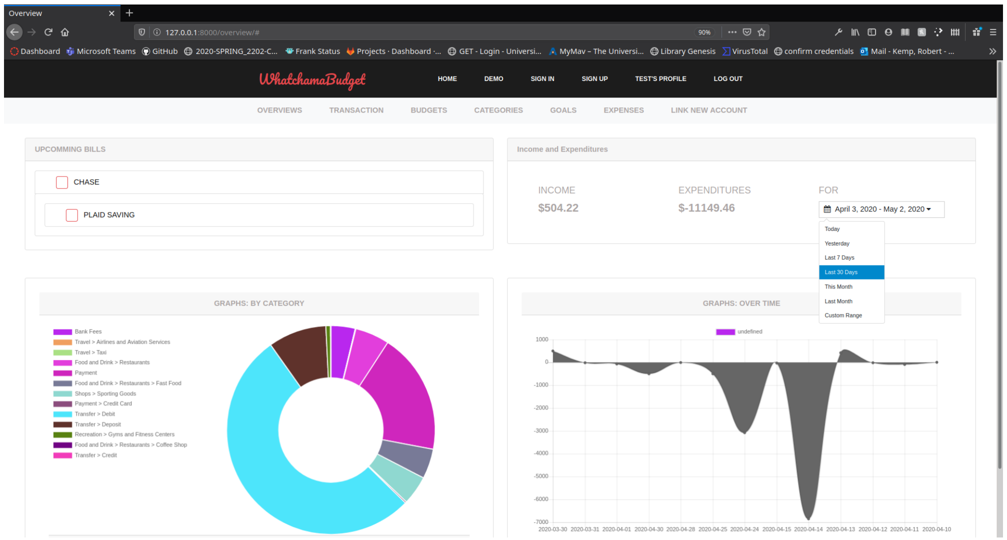
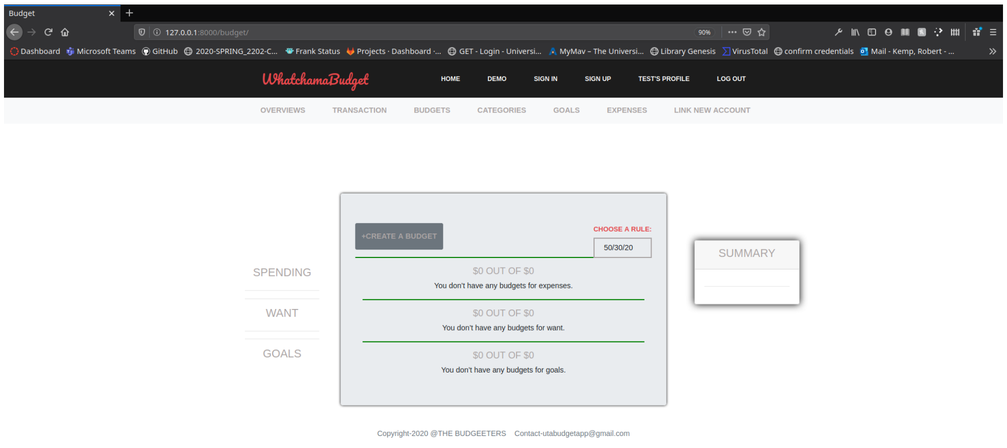


Figure 3: Application home screen

The budget screen, Figure 4, will control various budget settings and ability to create budget for a certain goal. The user will be able to choose the amount for goal, expenses and want separately.



$0 OUT OF $0

You don't have any budgets for expenses.

$0 OUT OF $0

You don't have any budgets for want.

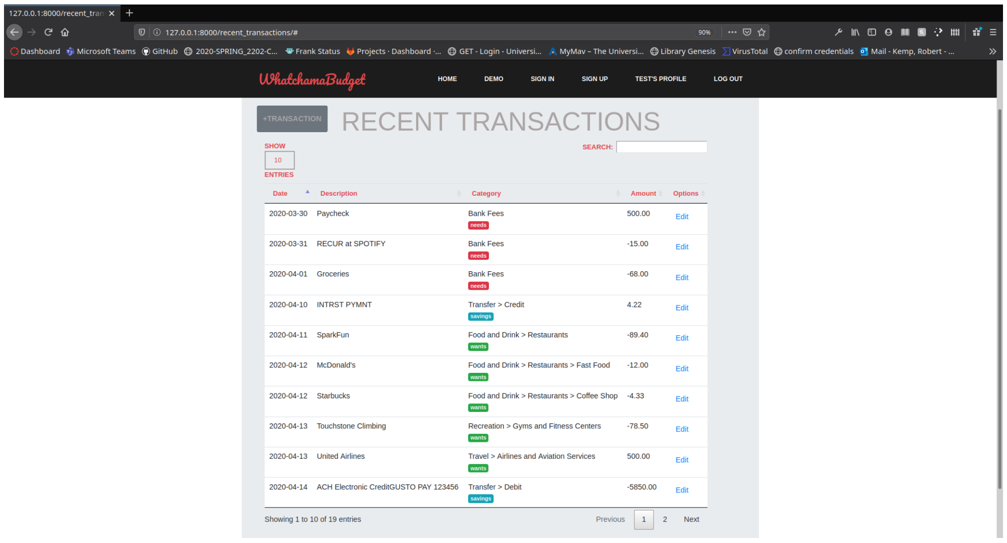
$0 OUT OF $0

You don't have any budgets for goals.

Copyright-2020 @THE BUDGEETERS Contact-utabudgetapp@gmail.com

Figure 4: Application settings screen

The manual data entry screen, depicted in Figure 5, will allow a user to manually enter transaction data and submit it to the back-end. This will allow for the monitoring of transactions not directly associated with a user's bank account (e.g. cash transactions). This feature is included in the transaction page where the list of all the transactions are shown.



RECENT TRANSACTIONS

Date	Description	Category	Amount	Options
2020-03-30	Paycheck	Bank Fees needs	500.00	Edit
2020-03-31	RECUR at SPOTIFY	Bank Fees needs	-15.00	Edit
2020-04-01	Groceries	Bank Fees needs	-68.00	Edit
2020-04-10	INTRST PYMNT	Transfer > Credit savings	4.22	Edit
2020-04-11	SparkFun	Food and Drink > Restaurants wants	-89.40	Edit
2020-04-12	McDonald's	Food and Drink > Restaurants > Fast Food wants	-12.00	Edit
2020-04-12	Starbucks	Food and Drink > Restaurants > Coffee Shop wants	-4.33	Edit
2020-04-13	Touchstone Climbing	Recreation > Gyms and Fitness Centers wants	-78.50	Edit
2020-04-13	United Airlines	Travel > Airlines and Aviation Services wants	500.00	Edit
2020-04-14	ACH Electronic CreditGUSTO PAY 123456	Transfer > Debit savings	-5850.00	Edit

Showing 1 to 10 of 19 entries

Previous 1 2 Next

Figure 5: Manual data entry screen

3 Customer Requirements

When we began work on this project, we had a couple of goals in mind: Proficient use of banking API and web/desktop development tools to provide better user interface and data security, grow from web platforms in order to be versatile and attainable to all potential users, and provide/suggest users to budgeting platform or options along with creating a specific budget plan which has a long term or a short term scenario. Our customer has graciously provided us with their list of requirements. Additionally, we trust that our customer's experience in the finance industry will be extremely useful in determining requirements for this project. There were update made on the requirements that the application should support web version instead of the desktop version. During the Senior Design II, the web version of the application was progressed into being final application.

3.1 Interface

3.1.1 Description

The program requires a web interface at a minimum.

3.1.2 Source

Customer.

3.1.3 Constraints

N/A

3.1.4 Standards

N/A

3.1.5 Priority

Critical.

3.2 Input of Line Items

3.2.1 Description

The user must be able to enter items manually (spending/income, type of purchase (if applicable), place of purchase/deposit) or pick among the given options in case of starting date and ending date for goals and budget. The user must be able to allow batch input of items directly from their bank account.

3.2.2 Source

Customer.

3.2.3 Constraints

The user will be able to batch insert items manually, the batch insert will be a feature of account connection and transaction update.

3.2.4 Standards

N/A

3.2.5 Priority

Critical.

3.3 Linking of Accounts

3.3.1 Description

The user must be able to link the app to their banking accounts. This feature is required to allow the app to run smoothly and automatically for our users. The account linking should feed directly into requirement 3.2 Input of Line Items. Through that connection the user will be able to add single transactions into the application dashboard.

3.3.2 Source

Customer.

3.3.3 Constraints

N/A

3.3.4 Standards

N/A

3.3.5 Priority

Critical.

3.4 Account Security

3.4.1 Description

The app must be password protected. This is to protect the user's confidential banking data. The user will be able to access their personal account by logging in with that password.

3.4.2 Source

Customer.

3.4.3 Constraints

We cannot reasonable handle actual banking security, this requirement may be handled as a document written of our security recommendations.

3.4.4 Standards

ISO 17799 and ISO 27001, these set the best practice for information security.

3.4.5 Priority

Critical.

3.5 Expense Deductions

3.5.1 Description

Putting money into funds for expenses that are paid not every month (such as UTA parking, property tax, car/house insurance, etc). This money needs to be deducted at the start of the month, not if thereâs money left over at the end. Every budget created will be related with specific goals set up by the user

3.5.2 Source

Customer.

3.5.3 Constraints

Funds can only go into expenses that have been defined by the user. The user cannot over fund the expense.

3.5.4 Standards

N/A

3.5.5 Priority

Critical.

3.6 Fund Allocation

3.6.1 Description

The user must be able to set a maximum amount for funds, if the fund cap is met, no additional money needs to go into that fund unless the fund is used. The user must be able to allocate extraneous income toward funds.

3.6.2 Source

Customer.

3.6.3 Constraints

The user must define the funds.

3.6.4 Standards

N/A

3.6.5 Priority

Critical.

3.7 Fund Rollover

3.7.1 Description

The user must be able to decide what to do with their extra money at the end of the month (add to funds, roll into next month, spend as "fun money", etc.). This functionality is completely user dependent.

3.7.2 Source

Customer.

3.7.3 Constraints

N/A

3.7.4 Standards

N/A

3.7.5 Priority

Critical.

4 Packaging Requirements

The minimum requirement of the application is that it should have web interface with available user functionalities. Once the development is completed, user can access the website that is published for public domain(digital delivery). But in order to use the services of the application, user must sign up first. The user will be able to view how the application and its features work on the demo tab without signing up.

4.1 Web Interface

4.1.1 DESCRIPTION

This is the basic requirement of the application. Anybody who has access to the application link and has internet access must be able to use it. The desktop version of the application will not be published since only the web based application fulfilled the updated requirements.

4.1.2 Source

At this point, the development team does not have direct contact with the sponsor. But with the help of Dr. Gieser, the team obtained the basic requirements of the application. Dr. Gieser provided us with information through out the sprint and Senior Design II.

4.1.3 Constraints

Functioning of the web application may depend upon a client's web browser and its configuration.

4.1.4 Standards

HTTP/HTTPS

4.1.5 Priority

High

5 Performance Requirements

Their are several competitive applications out there in the market which could have similar functionality as ours. So, one of the basic and important requirement for better performance is to have clear user interface with quick response time. Performances requirements also include the number of bank account that the user can add, no of users the application can support as well as no. of days that the application will track for budget management. Besides that, the application should have at least those features that are comparable to the other budgeting application such as Mint.

5.1 Clear User Interface with better response time

5.1.1 Description

The application's user interface should convey clear and transparent messages to the user. Many studies have shown that human's brain processes pictures quicker than text. So, including contextual images would be better solution. The user interface should also be simple and easy to understand along with its conveyance.

5.1.2 Source

Team decision.

5.1.3 Constraints

N/A

5.1.4 Standards

N/A

5.1.5 Priority

High

5.2 Number of Bank account that can be added

5.2.1 Description

If user has multiple accounts, he/she should be able to add them.

5.2.2 Source

User requirements.

5.2.3 Constraints

User should have valid bank account.

5.2.4 Standards

N/A

5.2.5 Priority

Moderate

5.3 Setting budgeting time

5.3.1 Description

Some budgeting application allow user to select the time frame (like weekly, monthly etc) but for now, the team will work for monthly planning.

5.3.2 Source

Team decision.

5.3.3 Constraints

The application should have access to user bank account.

5.3.4 Standards

N/A

5.3.5 Priority

Moderate

6 Safety Requirements

The Budget App, being a complete software based project, there are no other hardware components required that are assembled or necessary.But there are some safety factors to keep in mind listed below.

6.1 Laboratory equipment lockout/tagout (LOTO) procedures

6.1.1 Description

Any fabrication equipment provided used in the development of the project shall be used in accordance with OSHA standard LOTO procedures. Locks and tags are installed on all equipment items that present use hazards, and ONLY the course instructor or designated teaching assistants may remove a lock. All locks will be immediately replaced once the equipment is no longer in use.

6.1.2 Source

CSE Senior Design laboratory policy

6.1.3 Constraints

Equipment usage, due to lock removal policies, will be limited to availability of the course instructor and designed teaching assistants.

6.1.4 Standards

Occupational Safety and Health Standards 1910.147 - The control of hazardous energy (lockout/tagout).

6.1.5 Priority

Critical

6.2 National Electric Code (NEC) wiring compliance

6.2.1 Description

Any electrical wiring must be completed in compliance with all requirements specified in the National Electric Code. This includes wire runs, insulation, grounding, enclosures, over-current protection, and all other specifications.

6.2.2 Source

CSE Senior Design laboratory policy

6.2.3 Constraints

High voltage power sources, as defined in NFPA 70, will be avoided as much as possible in order to minimize potential hazards.

6.2.4 Standards

NFPA 70

6.2.5 Priority

Critical

6.3 RIA robotic manipulator safety standards

6.3.1 Description

Robotic manipulators, if used, will either housed in a compliant lockout cell with all required safety interlocks, or certified as a "collaborative" unit from the manufacturer.

6.3.2 Source

CSE Senior Design laboratory policy

6.3.3 Constraints

Collaborative robotic manipulators will be preferred over non-collaborative units in order to minimize potential hazards. Sourcing and use of any required safety interlock mechanisms will be the responsibility of the engineering team.

6.3.4 Standards

ANSI/RIA R15.06-2012 American National Standard for Industrial Robots and Robot Systems, RIA TR15.606-2016 Collaborative Robots

6.3.5 Priority

Critical

7 Maintenance & Support Requirements

The maintenance of the database holding the personal information, its accessibility and privacy is critical.These factors of the Budget App will be monitored and updated to make it bug free and efficient.The application will be required to constantly check for any memory leaks throughout the whole program.

7.1 Update notifications

7.1.1 Description

Any update for the new version of the application will be notified to the user and will be asked to the user being updating it.

7.1.2 Source

Back end Developers and Database administrators. The user will informed through a message on their account or an email that will be sent to them.

7.1.3 Constraints

Any credentials to log in will be required after a version update.

7.1.4 Standards

N/A

7.1.5 Priority

High Priority

8 Other Requirements

In essence the Other Requirements for the system requirement specifications of our budget software are specifications we have considered adding because they are valuable features to our customer although they still propose some limitations.

8.1 Allow users to implement custom rules

8.1.1 Description

Users should be given the ability to customize rules for each of their accounts that are linked to the app as well as their preference on the budget creation along with goal creation. For example, they cannot withdraw more than a certain amount or spend more than a certain amount of money a week.

8.1.2 Source

Customer

8.1.3 Constraints

It may be difficult to give a user that type of administrative control, determining exactly how much leeway that they will be provided with. Also, it will be difficult to decide exactly what would happen when a user tries to break a rule they implemented.

8.1.4 Standards

N/A

8.1.5 Priority

Moderate

8.2 Predefined budgets and custom budgets

8.2.1 Description

The application will come with in-built predefined budget schemes that the user can use (e.g. 50/30/20 rule, and the 60/20/20 rule). User can also implement whatever multiple types of budget and ratio assortments they desire. This will be available in a Budget features once the user has logged into his/her account.

8.2.2 Source

Customer

8.2.3 Constraints

User may not follow defined budget or may not have sufficient funds to facilitate selected budget. For the custom budgets the user may construct impassable budgets.

8.2.4 Standards

N/A

8.2.5 Priority

Moderate

9 Future Items

9.1 Expanding desktop version features

9.1.1 Description

We have considered developing and expanding features such as budget creation, goals creation and overview into more user-friendly environment. This way the application will be able to showcase versatile financial information.

9.1.2 Source

Developers

9.1.3 Constraints

Time constraint Intellectual constraint, some developers are unfamiliar with Java, css and back end functionalities.

9.1.4 Standards

N/A

9.1.5 Priority

Future

9.2 Stock functionality

9.2.1 Description

The stock functionality is another requirement that was considered and decided to be pushed to future requirements. The ability to give the user the option to invest, save, and earn through the trading of stocks.

9.2.2 Source

Developers

9.2.3 Constraints

- Time constraint
- Stock API

9.2.4 Standards

N/A

9.2.5 Priority

Future

REFERENCES