ICS4U JAVA PROJECT

STOCKPREME

Kimberly Dao, Yousef Yassin, Shrish Mohapatra

May 2019

Stockpreme is a stock simulator program which allow users to view, purchase, and sell stock shares. This stock data is dynamically updated based on real-life stock changes through the use of Web Scraping. Stockpreme also allows users the ability to manage their financial account, allowing for withdrawals and deposits.

What are stocks?

When companies become a corporation, they release portions of their company using stocks. Public corporations allow the general public to purchase these portions (or shares), which means they are investing in the company. The value of these shares increase and decrease based on the market as investors buy and sell their shares. Also, company actions can positively or negatively affect their stock value (eg. Apple releasing new iPhone).

For our program, we are interested in the stock value over periods of time. Visually displaying this data is important for investors as they can observe stock trends and make smarter decisions. Using the python module **Pandas**, we can grab the following data of a stock for each day:

Open: This is the starting value of a stock (at the beginning of the day)

High: The highest value that the stock was purchased for

Low: The lowest value that the stock was purchased for

Close: The final value of the stock at the end of the day

Adjusted Close: Will talk about later

Volume: # of trades that occured

For our program, the main information we need is adjusted close. When corporations feel that the value of their shares are too high, they can split shares and reduce the value of each stock by adding more stocks (think of adding water to a concentrated solution). To account for this possibility, adjusted close takes into account any changes in stock volume, and gives the true final stock value.

Client

Who: Kim's father

Interest: Stock program

- Seeing how much money investment of stocks can earn
- Gives an idea/simulation on money management
- See stocks in real time

Requirements for the program:

- Buy and sell stocks
- Link information to user's bank account
- Keep transaction history (purchasing and selling stocks, deposit, withdrawal)
- Simple and organized UI

Suggestions of features:

- See percentage of company's stocks that you own
- Stock Ticker Graph (Stock value progress graph)

Feedback

[May 3] - Met with client to discuss potential ideas for program. Brainstormed list of requirements for Stocks Program.

[May 11] - Updated and showed client progress, with more detailed explanation of features to be included in final product (ex. Stock transactions linked to user's bank account).

[May 15] - Collection of feedback from client - Good progress. Suggestion of more potential features - final product could show how much percentage of the company's stocks the user owns.

[May 27] - Nice design or UI Dashboard. Simple and organized. Great how once logged in, past information is saved to the user's account.

[June 10] - Satisfied with the design of the program and its functionality.

Algorithms

LOGIN SYSTEM

1. Load Data
    a. Load array of users from text file and save to *masterList*
2. Create User Interface
    a. Title: ***Stockpreme***
    b. Username and Password fields
    c. Username and Password labels (to identify fields)
    d. An error message box (to show user if username/password does not meet criteria)
    e. “Sign Up” and “Log In” buttons
3. On click “Sign Up”
    a. Check that username is at least 5 characters
    b. Check password is at least 5 characters, has capitals, has lowercase, has numbers, has symbols
    c. If all conditions are met, create a new ***Account*****,** proceed to next page
    d. If all conditions are not met, inform user via error message box
4. On click “Log In”
    a. Look for account with same username and password in *masterList*
    b. If account is found, proceed to next page
    c. If account is not found, inform user via error message box

BANKING SYSTEM

1. Create a project called “BankingSystem”
2. Create a class called “AccountBalance” that is used to keep track of account balance.
    a. In the “AccountBalance” class, declare a variable for the balance, called “balance”.
3. Create a class called “Transaction”, that is used to keep track of transaction history.
    a. Store user’s name, date of transaction and amount deposited/withdrew for every transaction in a text file.
4. Create a new class called “Menu”, which will display the menu of the banking system.
5. In the “Menu” class, add a button called “Instructions” that will display the instructions.
6. In the “Menu” class, add a button called “Deposit” that will allow the user to enter funds into their account.
    a. Under the “Deposit” button, create a method that will allow users to input how much money they would like to deposit.
    b. Add the inputted amount to the variable “balance”.
    c. Once the transaction is done, add a new entry into the transaction text file.
7. In the “Menu” class, add another button called “Withdraw” for users to take out funds from their account.
    a. Under the “Withdraw” button, create a method that will users to input how much money they would like to withdraw.
    b. Subtract the inputted amount from the variable “balance”.
    c. Once the transaction is done, add a new entry into the transaction text file.
8. In the “Menu” class, add another button called “View Balance” that will display to users their account’s balance.
    a. Under the “View Balance” button, create a method that will display the account balance.
    b. Display value of variable “balance”.

STOCKS SYSTEM

User Interface

1. Set page colour to according value, display title name “Stocks page” , user name and ticker search field. Upon selection of stock, display ticker name, current live price and percent growth since day open.

Acquire Ticker Information

2. Import buffered stream reader, url, etc. (To read HTML code).
3. Ask user for ticker, transform string to all capitals.
4. Copy HTML code from yahoo finance website for given ticker under the format "https://finance.yahoo.com/quote/" + SYM.
5. Code comes in hash mapped dictionary format, find the required information by iterating through the code (with if statement for required line).
6. For the current live price, search for “post market value”. Return this string.

Buy/Sell Shares

1. Open smaller window
2. Display the user’s checkings balance or stock allocated balance at the bottom of the screen (total).
3. Display user’s current investments, with allocated money to each respective stock in addition to percent gain since last login (this will be loaded within save info file).
4. In case of increase, display as green else red.
5. Each stock tab represents an object, can sell the tab in which case initial invested amount in addition to global balance. Tab disappears.
6. Option to purchase current viewed stock. Slider will determine integer value of shares to be purchased. Upon confirmation, value times current price is deducted from account and new tab object for stock is created given it is not already present. Present stocks will be updated.

Graphing**

1. Acquire ticker history through Yahoo Finance API (figure out dependencies first).
2. Use history table for adjusted market close against date to plot stock price graph. Look into a time delta module that will allow plotting for a certain time period.
3. Alternatively, open the yahoo finance page for the stock which hosts the graph (plan B).

SAVING SYSTEM

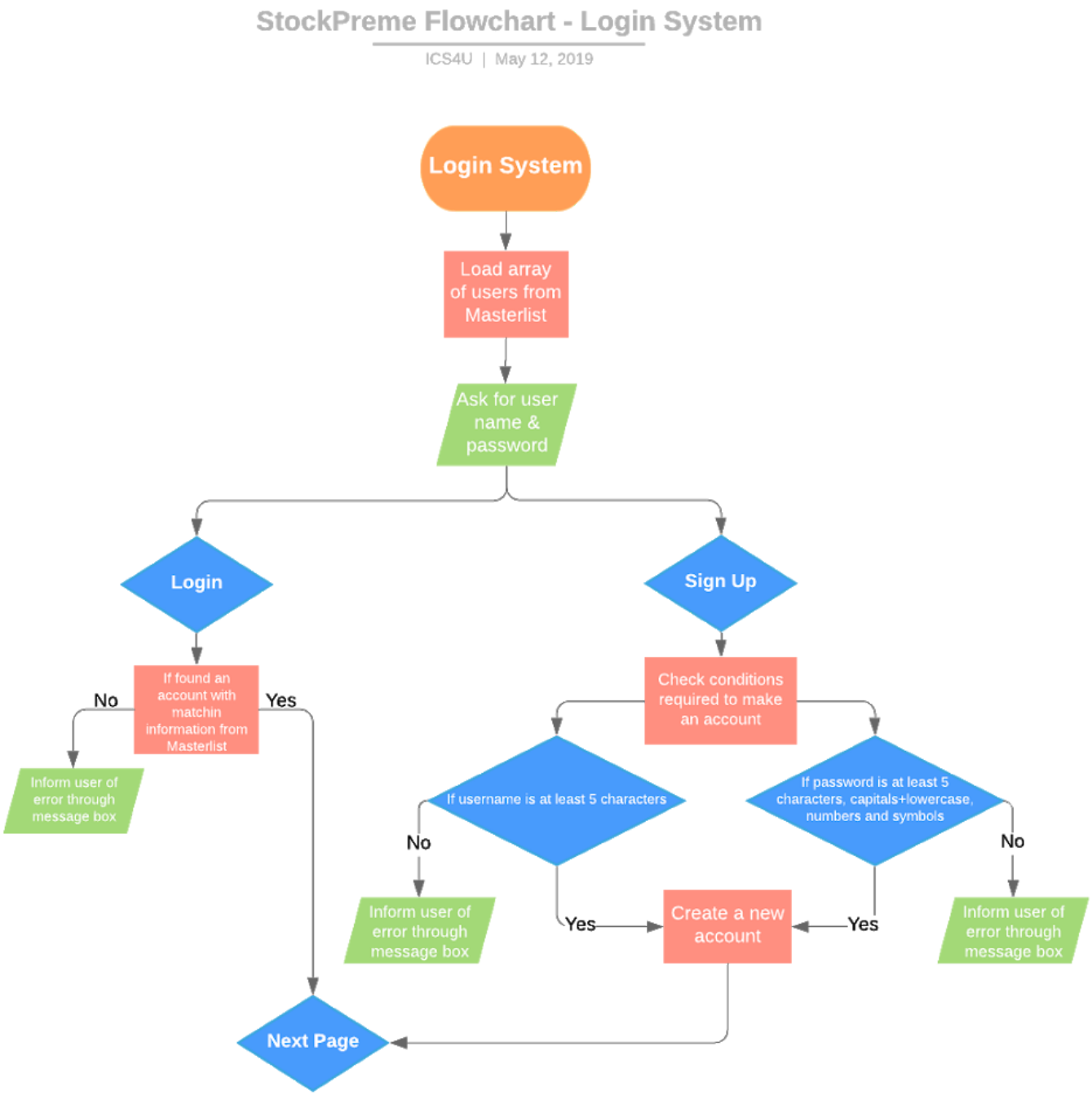
1. Open text file “userData.txt”
    a. If file does not exist, create new file
2. Loop through each account in *masterList*
    a. Print username and password on **Line n**
    b. Loop through each transaction in account *ledger*
        i. Print transaction date, type, amount on **Line n+1**
        ii. (each transaction split by “,”)
    c. Loop through each stock in account *portfolio*
        i. Print stock ticker, # of shares on **Line n+2**
        ii. (each stock split by “,”)
3. Close file

SAMPLE “userData.txt”

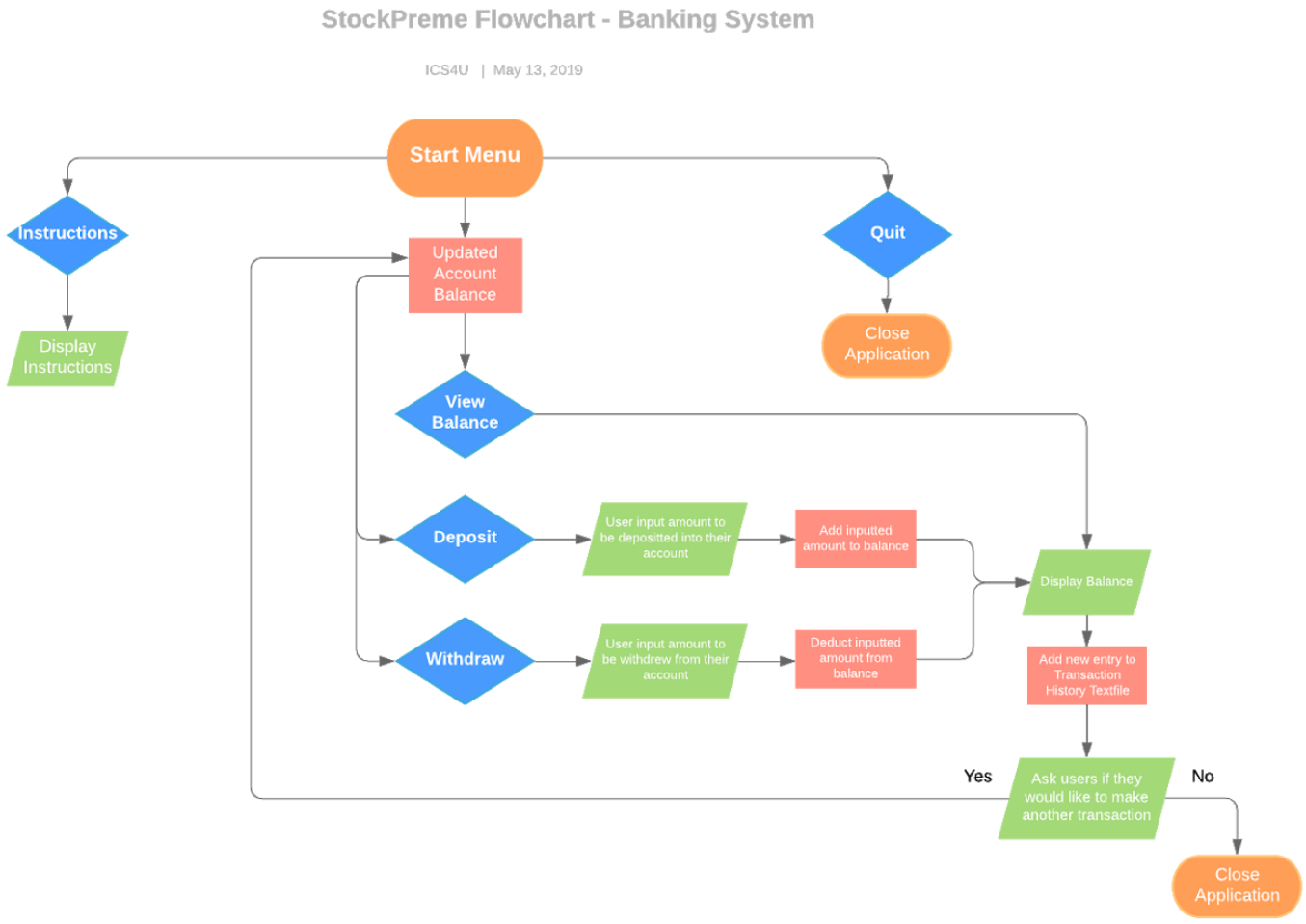
```
shrish.m ICSisFUN
2019/05/02 Deposit 500.00, 2019/05/05 Withdraw 450.00
AAPL 3 189.29, GOOG 10 5344.3, KL 4 3534.3

kim.d ICSisFUN2
2019/05/02 Deposit 4500.00, 2019/05/05 Withdraw 350.00
AAPL 3 189.29, GOOG 10 5344.3, KL 4 3534.3

yousef.y ICSisFUN3
2019/05/02 Deposit 9800.00, 2019/05/05 Withdraw 51.00
AAPL 3 189.29, GOOG 10 5344.3, KL 4 3534.3
```

Flowcharts

LOGIN SYSTEM

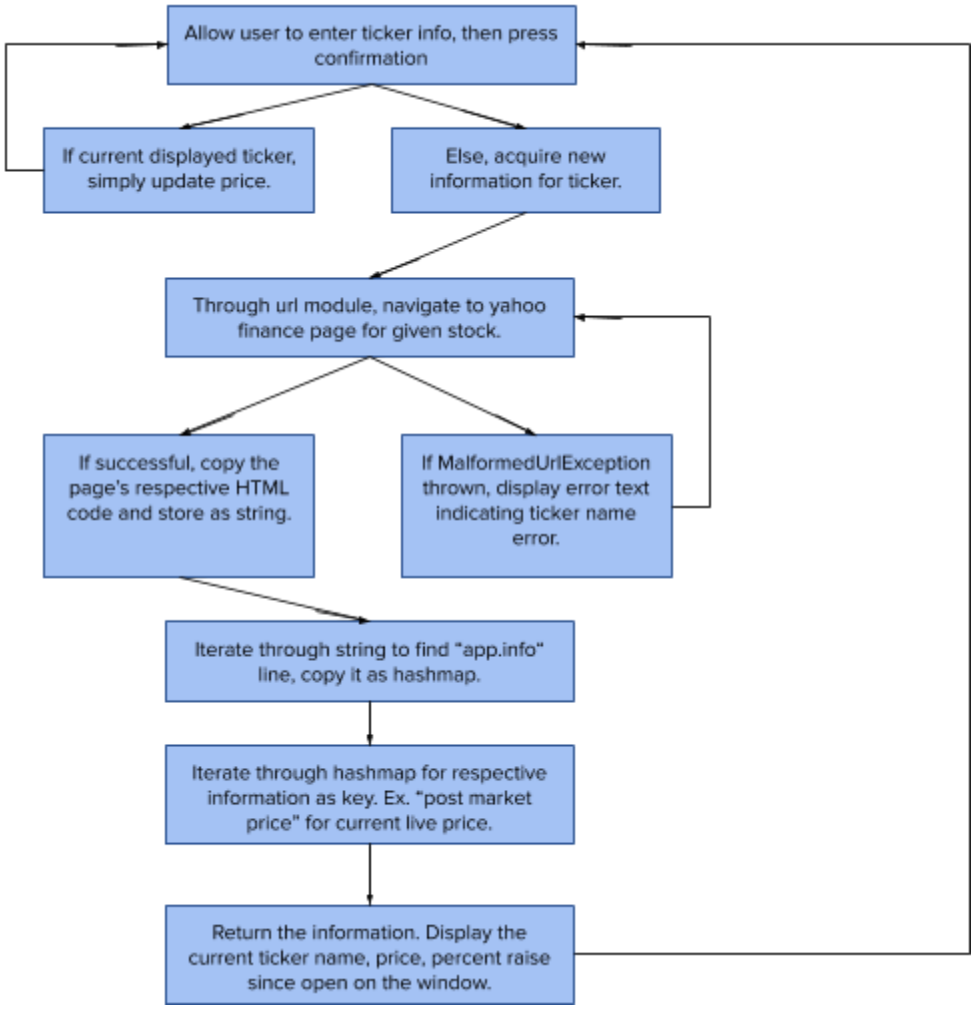


BANKING SYSTEM

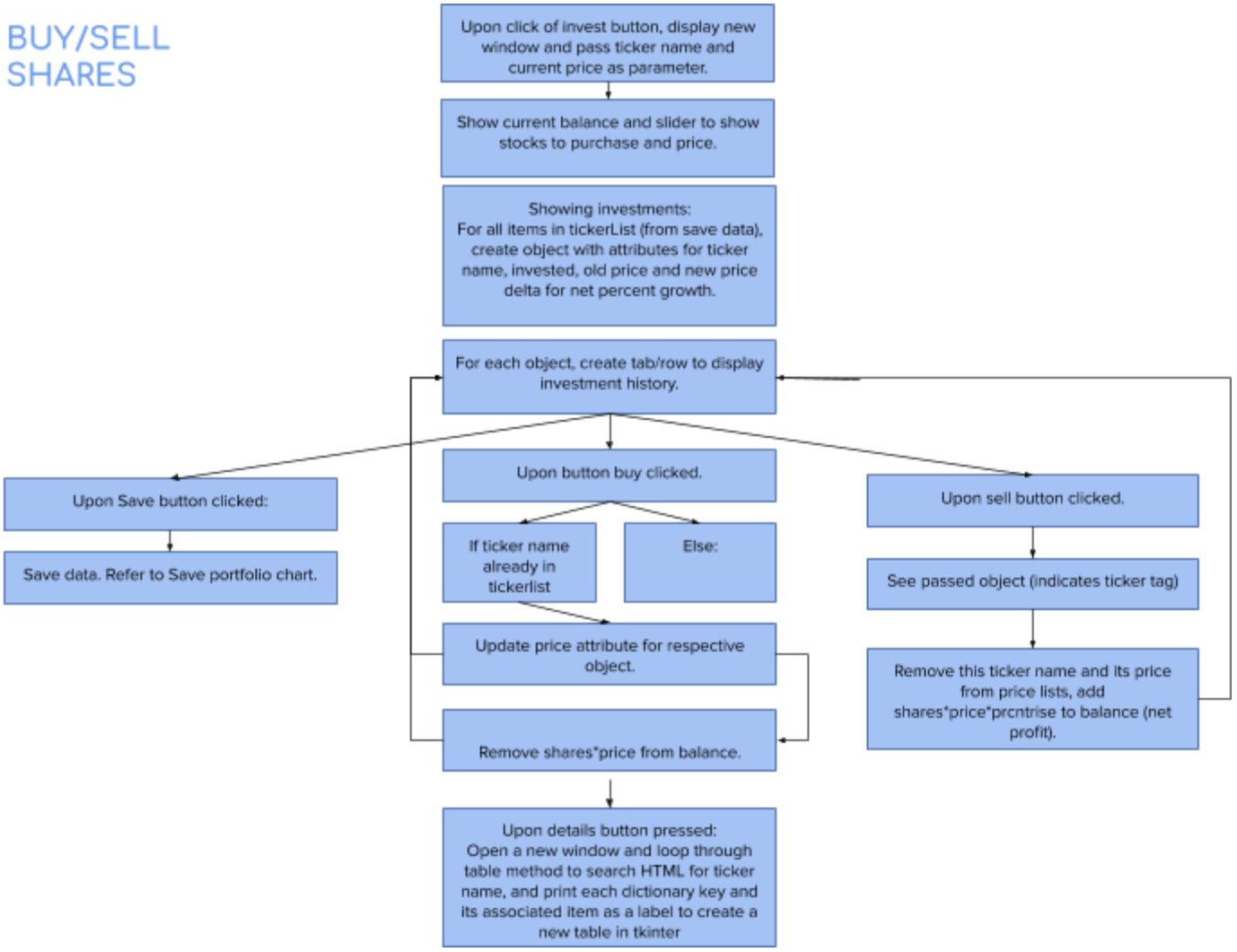


STOCKS SYSTEM

TICKER INFORMATION



BUY/SELL SHARES



Schedule

Date	Tasks
Friday May 3rd	- Meet with client, discuss ideas

May 3-May 13	- Have a solid Plan
	  - Client's requirements
	  - Prototype
	  - Algorithms
	  - Flowcharts

May 13-May 23	- Version 1 Ready to show client for feedback
	  - Code Login System
	  - Code Banking System
	  - Code Stocks System

May 24	- Meet with client, discuss, and take feedback
May 23-June 4	- Finish testing - Meet with client for more feedback
June 4-June 7	- Apply any additions or fixes, have final product ready
June 10/11	- Close project - Prepare Presentation

Teamwork times & Meetings

Date	Summary of meeting

May 10	- Clarified requirements and visions for the project
	  - Trial of different ways to incorporate web browser into project
	  - Trial of different ways to graph a stock
	  - Trial of different ways to import live stock prices
	- Discussed roles and assigned individual tasks

May 13	- Updated Schedule and Progress Update - (All assigned tasks are done) - Review Stocks System Code - New feature to work on: currency exchange rate (CAD→ USD)
June 10	- Discussed Closing of project and tasks to complete - Preparation for Presentation

Log

[Group] May 9 2019 - Found potential client, recorded requirements for the problem and began brainstorming potential ideas for solution.

[Group] May 10 2019 - Conducted research regarding methods to gain live stock and currency information. Found multiple methods, settled upon “web scraping” - gaining information through indexing an HTML.

[Group] May 11 2019 - Created Asana project page, finalized initial brainstorming stage and entered prototype/planning stage - assigning each group member their own tasks/classes to develop.

[Yousef] May 12 2019 - Developed code that would be able to access and copy the live HTML code of a web page.

[Yousef & Shrish] May 13 2019 - Expanded upon above code to be able to index and return a specific portion of the HTML code according to input parameter.

[Yousef] May 14 2019 - Through more research, enhanced code to be able to return valid information for several aspects regarding a ticker such as the name, ask price/size, bid price/size, market high, low, dividend, etc.

[Kim & Shrish] May 14 2019 - Developed code for currency exchange (from CAD to USD, as stock prices are in USD). Imported live HTML code of a webpage to return specific portion for USD currency rate.

[Yousef] May 15 2019 - Added the ability to be able to calculate and return the delta price change for the day, in addition to the net percent growth for the stock. Next step is to create a similar method to calculate the same for the live price and the previously purchased price for investments.

[Kim] May 15 2019 - Met with client to discuss prototype and project advancements. Collected and documented feedback.

[Yousef] May 16 2019 - Further enhanced code, adding comments and improving organization (more modular rather than sequential).

[Yousef & Shrish] May 17 2019 - Practically finalized Stock Data class, implementing a constructor that would create a ticker object, storing pertinent information such as price, name, share amount, etc.

[Shrish] May 18 2019 - Created two packages; stockLogic and stockUI. Created “Process” class in the “stockLogic” package which manages saving and loading of users, finding users from masterlist, and checking user input. Created classes for “UserAccount”, “BankProfile” and “StockProfile”.

[Shrish] May 19 2019 - Created “LoginMenu” class with ability for users to create new accounts or login with previous account information.

[Shrish & Yousef] May 20 2019 - Created saving and loading methods in the “Process” class to allow the ability to save and load user data to a text file. Imported Yousef’s “StockData” into the stockLogic package.

[Shrish] May 25-26 2019 - Created “Dashboard” UI.

- Added Portfolio - view stocks in user’s account
- Added Stock Viewer - search stock ticker and view ticker information
- Added buying & selling buttons
- Added withdraw & deposit buttons
- Added transaction viewer (view date, type, and amount)

[Kim] May 27 2019 - Met with client and showed advancements of Stockpreme. Collected feedback and presented client feedback to team.

[Shrish] June 2 2019 - Organized UI components into various methods. Improved “StockData” code to allow for efficient error checks.

[Shrish] June 8 2019 - Added currency converter to the “Dashboard”, allowing users to convert balances to CAD or USD. Fixed bugs regarding saving & loading.

[Kim] June 10 2019 - Showed client final version of Stockpreme and documented feedback.

[Group] - June 11 2019 - Closing of project.

Help

What are Stocks:

Plain and simple, a stock is a type of security that allows stockholders a share in the ownership of a company. They represent a claim on the company's assets and earnings. The more stocks invested, the larger the ownership claim on the organisation

Why Invest:

There exist various reasons to invest in shares, namely to support a company's products or mission. Often times, organisation may offer dividend percentages on the amount shareholders have invested as a token of gratitude. Other times, the success of an organisation may provide significant benefits.

How to Invest:

To begin your investor journey, simply head over to the "Stocks" tab where you will view your current balance and portfolio. To display a ticker's information, type the name under the "Market" title and press search. A list of information will be displayed to help inform your decision, such as live ask and bid prices. To invest in a stock, input the amount of shares you would like to purchase and select the "Buy" Option. To sell a stock, select the "Sell" option. We wish you the best of luck!

Popular Stocks:

New users are often interested in similar stocks. Here are a few to help you get started!

Apple Inc (AAPL), Shopify (SHOP), Nike (NKE), Sony Entertainment (SNE) and Google (GOOG).

What we learned

Web Scraping

When collecting information regarding stock data, we had to use something called web scraping. Web scraping involves the extraction of information from a website, which is often written in HTML. To grab stock data and information, we searched for key terms in the HTML page of Yahoo Finance. This allowed for the ability of getting live stock prices.

Object Oriented Programing

During the development of Stockpreme, the application of object-oriented principles was vital for the projects success. We learned the importance of using various objects and classes to create a modular ecosystem. User Accounts, for example, stored user information such as username, password, stocks, etc. In future projects, this same class could be used to model accounts.

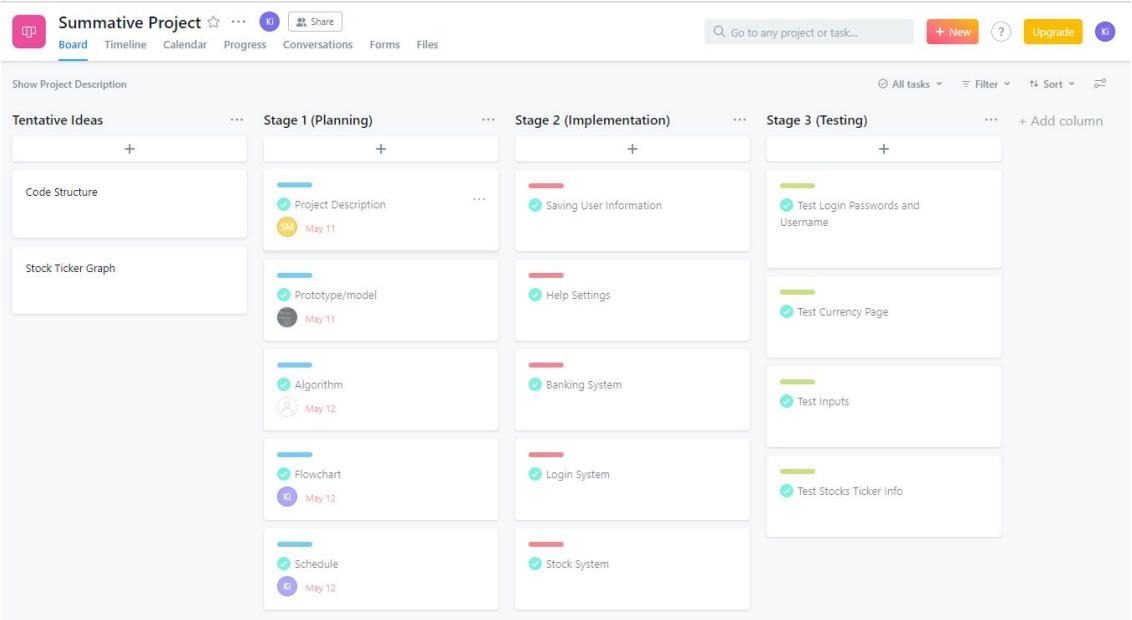
GUI

We also learned about how to use Netbeans' WindowBuilder to create user interface for the application. Moreover, we researched various other stock portfolio applications to understand basic UI principles to encourage a pleasant experience for the user.

Planning

Throughout this project, we truly learnt the importance of Planning. We found and learned it was important to break down our overall goal into smaller goals in order for all requirements to be met and not forgotten. From diligently planning out each task to be completed, the steps involved (as also seen in the Flowchart and Algorithm), and the time frame together did they allowed for the bigger picture, the creation of Stockpreme to be created successfully.

The PM Tool that we used to aid our project process was Asana. We chose Asana as it was a software that fit our team the best, being free, having a simplistic yet organized layout and being available on both the web and mobile device. With Asana, we were able to divide our project into layers (Planning, Implementation, Testing and Tentative Ideas), enhancing organization by making it easier to keep track of which tasks are done or need to be completed. In addition, Asana allowed for us to create tasks with a due date and assign them to a user. This feature made it easier to keep track of what needs to be done by date and which member/members are responsible for it. The organizational features used in Asana greatly contributed to the making of Stockpreme.



Teamwork/Organization

Learned the importance of cohesive teamwork to create a successful product. We also realized the significance of code organization, and how it allows for better understanding amongst team members. We learned how to implement each others' code and use them in conjunction. We also divided the code into two sections; stockLogic and stockUI. This allowed for distinction between user interface classes and background processing classes.