F.T.A.H.

Hayden Pour, Julian Beaulieu, Mohamed Rayyan, Padraic Reilly, Nicholas Cardinal

Artifact

[Historical daily prices and volumes of all U.S. stocks]

Data we have:

1. Trading Date
2. Opening Price
3. Daily Price (High)
4. Daily Price (Low)
5. Closing Price
6. Volume Sold

Platform:

Web Application / Desktop Application

Programming Languages:

- JavaScript
- HTML
- CSS
- Python

Feature List (Question of Interest):

Important features:

1. ***How does a stock's price change over a given period?***
2. ***How does the volume of stock sold change over a given period?***

3. ***What is the moving average for a stock over a given period?***
4. ***What is the highest/lowest closing price over a given period?***
5. ***Which stock has the largest margin over a given period?***
6. ***What days had the largest increases or decreases in price? (Useful for correlating to real world events)***
7. ***How did a specific stock's daily change compare to the market average change? (High or low performing stocks)***

Implement if we have time, because we need to use data outside of the database:

1. ***How do real-world events affect stock prices?***
2. ***What is the predicted opening and closing price of a stock?***
3. ***What stocks are the best to trade for the day?***

Sprint-2:

Action Items:

- **Client/UI**
  - **JS Promises**
  - **Create import routine for data retrieved from server**
  - **Sort retrieved labels alphabetically**
  - **Sort retrieved data by date**
  - **Display graph from label click**
  - **Display other various information on label click**
- **Server**
  - **Update csv file name to stock ticker name - stock name**
  - **Return labels in chucks to client backend**
  - **Parallel processing when importing from csv**

Tests:

- **Client/UI**
  - **JS Promises**

    Correct Output: The user's data will load completely from server before displaying information to the UI

  - **Create import routine for data retrieved from server**

    Correct Output: The website displays stock information when user clicks the stock they want to view

  - **Sort retrieved labels alphabetically**

    Correct Output: The website will display the stock list in alphabetical order
  - **Sort retrieved data by date**

    Correct Output: The data received by the server is properly sorted by date

- **Display graph from label click**
  Correct Output: When a label is clicked, the website will display all pertinent information on the right side of the UI

- **Server**
  - **Update csv file name to stock ticker name - stock name**
    Correct Output: All files in our dataset are renamed correctly

  - **Return labels in chucks to client backend**
    Correct Output: When called, the website requests data from the server and it is returned in a format that can be understood by the website

  - **Parallel processing when importing from csv**
    Correct Output: Multiple files can be read from at any given time

Sprint-3:

Features:

Feature 1:

User Story: As a user, I would like to search a list of stocks in order to find the one that I want.

Tasks:

Step 1: Create UI box on form
[Completed by]
Step 2: Create function to read UI box value
[Completed by]
Step 3: Create function to load stock list by UI box value
[Completed by]

Tests:

- Step 1 Test: Create UI box on form
  Correct Output: Box is visible on form and can type in it.
  [Completed by]
- Step 2 Test: Create function to read UI box value
  Correct Output: The function uses DOM to select the senders value to be passed to load function
  [Completed by]
- Step 3 Test: Create function to load stock list by UI box value
  Correct Output: Function returns data matching what user enters in box
  [Completed by]

Feature 2:
User Story: As a user, I would like to be able to switch between stocks and ETFs

Tasks:

Step 1: Create UI object to switch between ETF and Stock
[Completed by]
Step 2: Create a function that loads ETF or Stock depending on UI
[Completed by]
Step 3: Create function to reload stock list with newly filtered data
[Completed by]

Tests:

- Step 1 Test: Create UI object to switch between ETF and Stock
Correct Output: The website would display radio buttons that say “ETF” and “Stocks”
[Completed by]
- Step 2 Test: Create a function that loads ETF or Stock depending on UI
Correct Output: The server will return the selected type of data to the client
[Completed by]
- Step 3 Test: Create function to reload stock list with newly filtered data
Correct Output: The website will refresh the stock list when the user changes the selected type
[Completed by]

Feature 3:
User Story: As a user, I would like to view my stock data in a candle graph

Tasks:

Step 1: Display Candle Chart on web page with dummy data
[Completed by]
Step 2: Create a function to gather correct data to display on chart
[Completed by]
Step 3: Create function to reload chart with newly filtered data
[Completed by]

Tests:

- Step 1 Test: Display Candle Chart on web page with dummy data
  Correct Output: The website will display a candle chart
  [Completed by]
- Step 2 Test: Create a function to gather correct data to display on chart
  Correct Output: The website will display a candle chart with data selected the user
  [Completed by]
- Step 3 Test: Create function to reload chart with newly filtered data
  Correct Output: The websites UI will refresh when the user requests a change in stock
  [Completed by]

Feature 4:

User Story: As an administrative user, I would like to delete, and insert new stock data into the CSV files and know when it is done.

Tasks:

Step 1: Get access to new stock data from stock data source
  [Completed by]

Step 2: Write functions that insert, and delete from CSV files
  [Completed by]

Step 3: A pop up displays when the data import is complete
  [Completed by]

Tests:

- Step 1 Test: Get access to new stock data from stock data source
  Correct Output: Backend server can receive data from some stock data source
  [Completed by]
- Step 2.1 Test: Write functions that inserts records to CSV file
  Correct Output: The inserted records are successfully written to the CSV file
  [Completed by]
- Step 2.2 Test: Write functions that deletes records to CSV file
  Correct Output: Once the delete function is ran, the CSV file should contain less lines than it did before the delete
  [Completed by]
- Step 3 Test: A pop up displays when the data import is complete
  Correct Output: Once the function has completed the website will display a form that lets the user know that it is done.

[Completed by]

UI Example