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**
          <u>Correct Output:</u> 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**
          <u>Correct Output:</u> All files in our dataset are renamed correctly
        - **Return labels in chucks to client backend**
          <u>Correct Output:</u> 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**
          <u>Correct Output:</u> Multiple files can be read from at any given time

Sprint-3:

Features:

Feature 1:

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

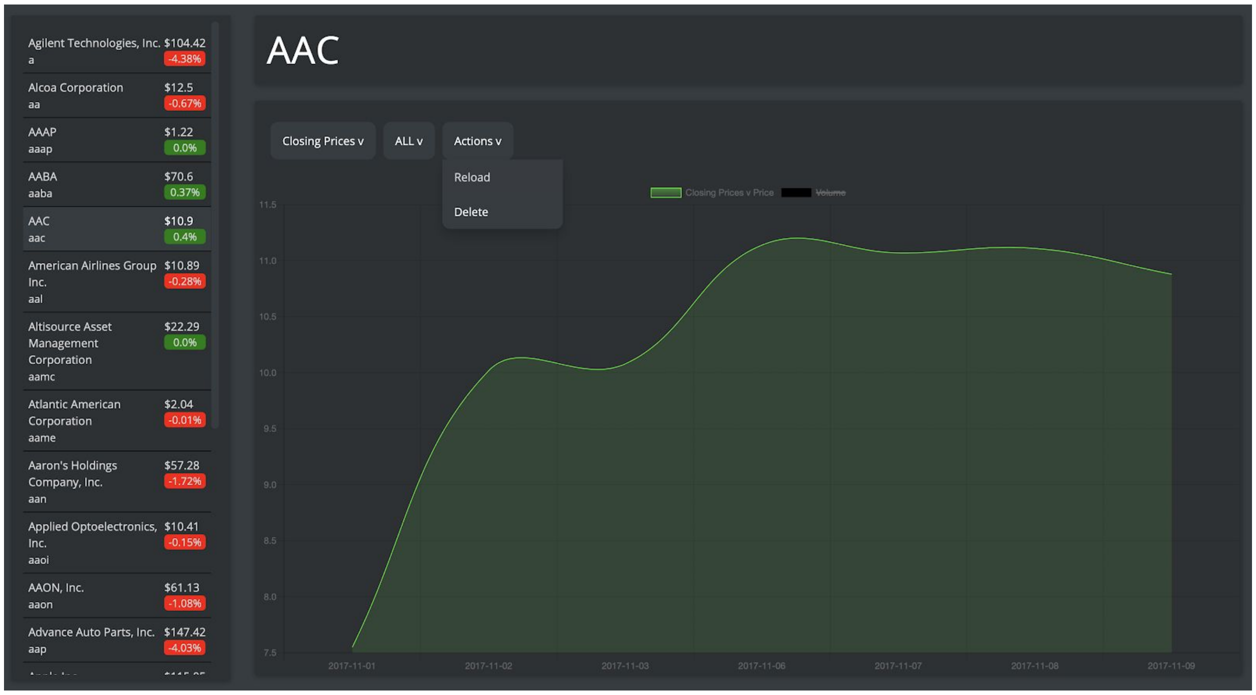
Tasks:

Step 1: Get access to new stock data from stock data source
[Completed by]
Step 2: Write functions that inserts lines into CSV and memory
[Completed by]
Step 3: GUI is updated to reflect the newly inserted data
[Completed by]
Step 4: Write function that deletes a selected stock from memory (front and back)
[Completed by]
Step 5: Back up deleted stock by keeping reference in CSV
[Completed by]
Step 6: GUI is updated to reflect the newly deleted data from portfolio
[Completed by]
Step 7: Write function that reloads a selected stock from server
[Completed by]
Step 8: Reload chart with newly imported data
[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 Test: (INSERT) Write functions that inserts records to CSV file
  Correct Output: The inserted records are successfully written to the CSV file
  [Completed by]
- Step 3 Test: (INSERT) Stock chart UI is updated with new data
  Correct Output: The chart now shows the latest data after the data was pulled from server
  [Completed by]
- Step 4 Test: (DELETE) Write functions that delete records (server and local)
  Correct Output: Once the delete function is ran, the server's dictionary should have one less record
  [Completed by]
- Step 5 Test: (DELETE) Stock List UI will remove deleted stock from list
  Correct Output: The chart now shows one less stock in the stock list and the chart data is cleared
  [Completed by]
- Step 6 Test: (RELOAD) Write function that reloads a selected stock from server
  Correct Output: Local datastore now hold a newly pulled set of data from the server
  [Completed by]
- Step 7 Test: (RELOAD) Reload chart with newly imported data
  Correct Output: Once the data is reloaded from the server the chart is reloaded for the user to see.
  [Completed by]

UI Example



Agilent Technologies, Inc. a	$104.42	-4.38%
Alcoa Corporation aa	$12.5	-0.67%
AAAP aaap	$1.22	0.0%
AABA aaba	$70.6	0.37%
AAC aac	$10.9	0.4%
American Airlines Group Inc. aal	$10.89	-0.28%
Altisource Asset Management Corporation aamc	$22.29	0.0%
Atlantic American Corporation aame	$2.04	-0.01%
Aaron's Holdings Company, Inc. aan	$57.28	-1.72%
Applied Optoelectronics, Inc. aaoi	$10.41	-0.15%
AAON, Inc. aaon	$61.13	-1.08%
Advance Auto Parts, Inc. aap	$147.42	-4.03%

AAC