Stock Prediction App

Week 1 Progress

CS361 HW 6

Open	1,504.58	Div yield	-
High	1,510.94	Prev close	1,516.99
Low	1,478.49	52-wk high	1,530.74
Mkt cap	1.02T	52-wk low	1,027.03
P/E ratio	30.17		

Instructions and Program available at:
https://github.com/cs361-stock-prediction/stock-predictor

Team 16:
Felix Brucker
Robert Detjens
Remi Kendig
Dominykas Zobakas
Lyell Read

Table of Contents

Week 1 User Stories

Completion by 3/7/2020

Note: **[✓]**: Complete; **[●]**: Partially Complete; **[X]**: Not Complete

[✓] Basic Site Framework (not a user story)
Due Date: 3/4/2020
Prerequisites: None
Workers: (Robert, Lyell), (Robert, Remi)
Unit Tests: Code Linting (JS, Py, CSS, HTML)
Problems: None
Left to be Completed: None
Time to Completion: 3ε

[✓] Search by Ticker Symbol
Due Date: 3/4/2020
Prerequisites: Basic Site Framework
Workers: (Robert, Lyell), (Robert, Dominykas)
Unit Tests: Code Linting (JS, Py, CSS, HTML)
Problems: Linting CI did not function, but was updated.
Left to be Completed: None
Time to Completion: 3.5ε

[●] Account Creation
Due Date: 3/4/2020
Prerequisites: Basic Site Framework
Workers: (Robert, Felix)
Unit Tests: Code Linting (JS, Py, CSS, HTML)
Problems: None
Left to be Completed: Database Integration to Server
Time to Completion: 2ε

[●] Personalize Account
Due Date: 3/7/2020
Prerequisites: Basic Site Framework, Account Creation
Workers: (Felix, Dominykas)
Unit Tests: Code Linting (JS, Py, CSS, HTML)
Problems: None
Left to be Completed: Database Integration to Server
Time to Completion: 1ε

[●] Account Login

Due Date: 3/5/2020
Prerequisites: Basic Site Framework, Account Creation
Workers: (Robert, Felix)
Unit Tests: Code Linting (JS, Py, CSS, HTML)
Problems: None
Left to be Completed: Database Integration to Server
Time to Completion: 2ε

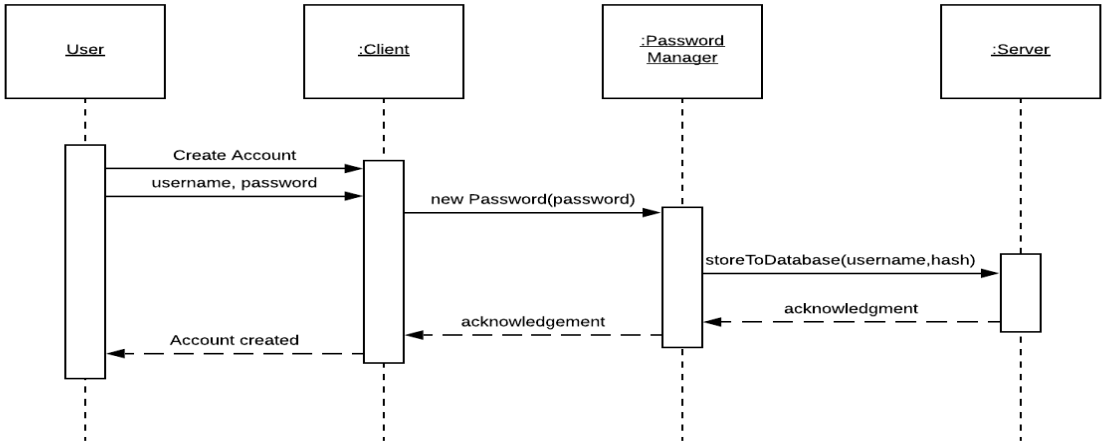
[X] Edit Account

Due Date: 3/6/2020
Prerequisites: Basic Site Framework, Account Creation, Account Login
Notes: This task was killed as it was redundant to an above task. For information about the completion of this task, see Personalize Account.

Diagram Reflections

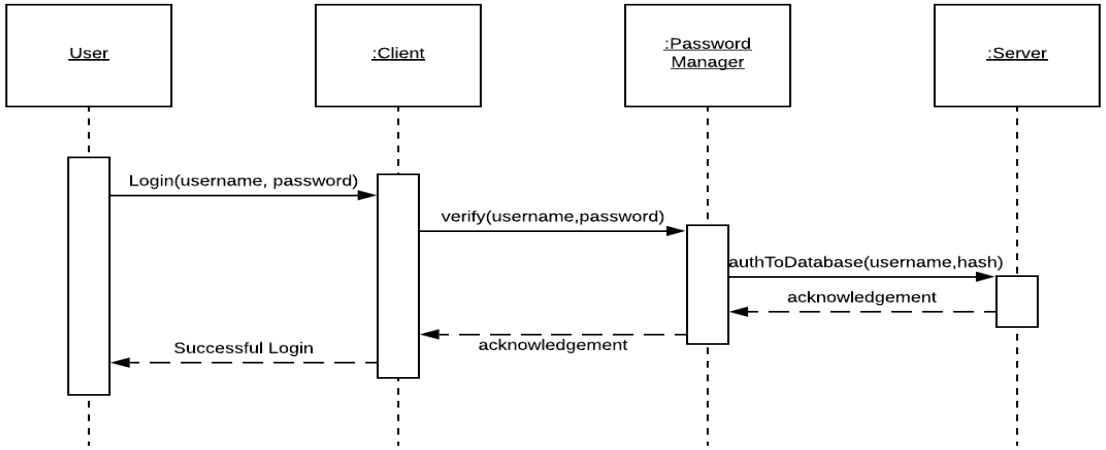
Account creation

This diagram was very helpful as a reference to how each part of the system was divided. The coding process and syntax also had a distinct separation of user, client, password manager, and server. Although the management of passwords on the site is not checked for the utmost of security, it is still one of the more complex operations and a visualization of how its data transferred from the manager to the server especially was helpful. It would be helpful to know how to make a more security-robust password, and have the diagram for it.



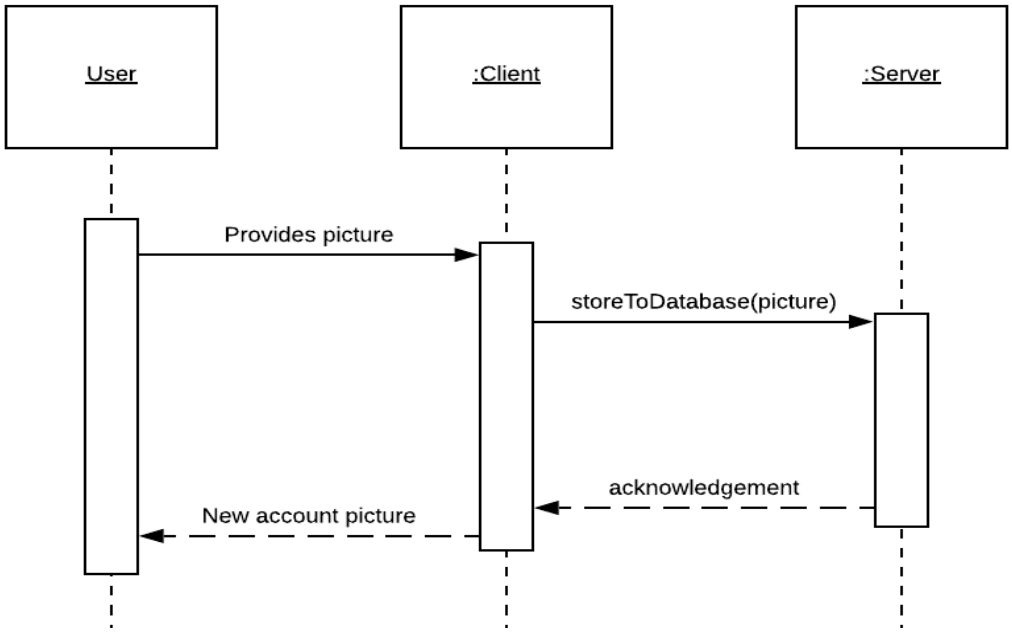
Account Login

This process was simpler, but the diagram still helped clarify the role of password manager, which isn't as intuitively defined as the user, client, and server. Just like password creation, it would be helpful to have the resources to make a truly security-robust authentication system.



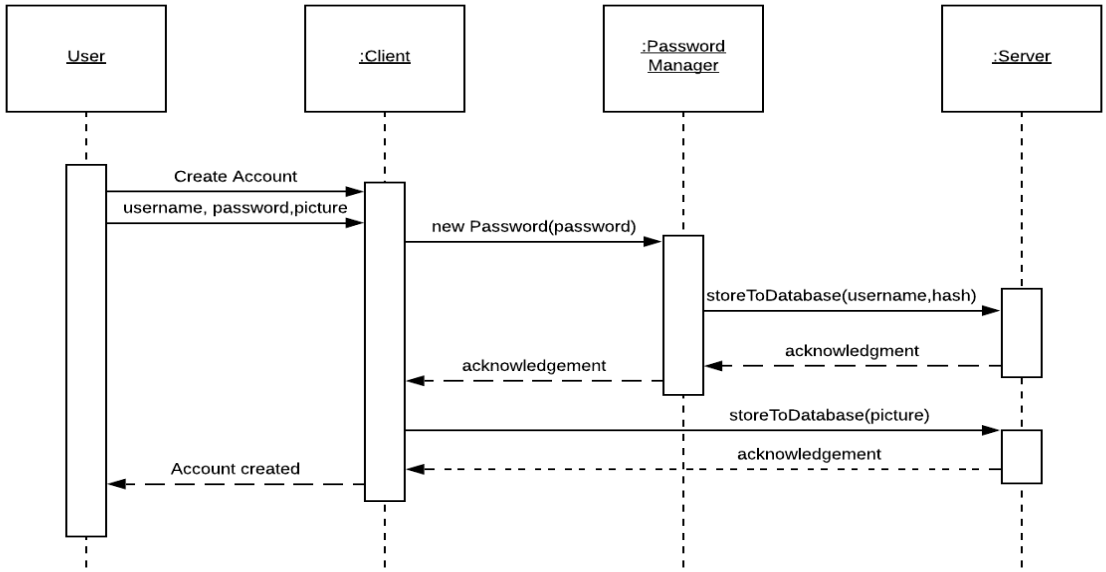
Edit Account

See Personalize Account - this user story was merged with it as discussed previously, as it is a part of the same process involving picture, password, and username.



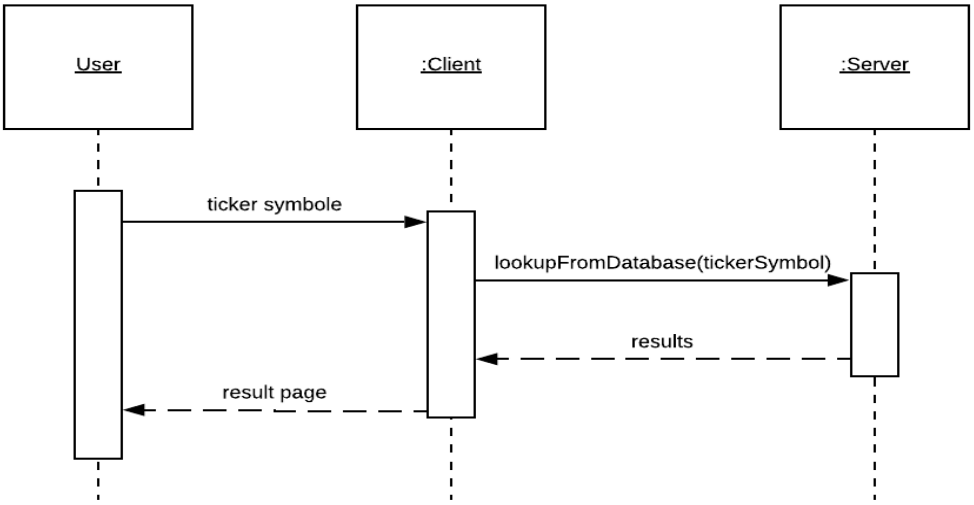
Personalize Account

Apart from the novelty of processing an image in addition to text, the personalizing of an account was expected to be a simpler affair. This was mostly due to it building upon existing database architecture and experience from account creation. The diagram reflects this - despite its density it conveys a simple back-and-forth - and doesn't say much more than is intuitively obvious. It might be helpful to see more in depth how our specific framework handles images, as proper approach could lessen loading time of a heftier than normal block of data such as an image.



Search by Ticker Symbol

While having the simple flow of data that account personalization also has, search relies more on novel architecture and was tricky. It's completion was not terribly helped by this fairly simple diagram, though the base concept helped. It might be useful to have a more in-depth diagram related to performance time, as search is a more distinctly live service that has to process lots of exterior data.



Burndown Diagram

Note: This burndown diagram does not line up perfectly, as there were a couple tasks that were not complete at the end of Week 1. These tasks total the difference between BurnDown (actual) and ToDo (forecasted). This was a result of many unforeseen circumstances including sickness and large final projects in other classes.

Refactoring

Overall

We refactored multiple sections of the website during design. The login page went through multiple iterations both in name and in design. It went from **/createaccount.html** (for both login and account creation) to **/accounts.html**, then **/accounts**, to finally separate account creation creation and login pages. Our install scripts were refactored to handle dependencies better and simplify repeated code into a helper function. The basic site, reused from a previous class's assignment, was split up into templates, one for the main site with the common header and footer, and separate templates for each page. The CSS for the site was also split up from one monolithic file into a main file for the header and footer, and then each site's rules were separated into their own file. We also changed our logo from a PNG file to a vector graphic image so that it looks sharp at all resolutions.

Server

Serverside, our server Python program was not refactored much at all during our design, as the design of the Flask web server makes it so that everything is simple and it needs to be done in the same manner, so it cannot really be changed without impacting the way it works.

CI Workflow

Our CI workflow was refactored several times as we settled on one linting service. There were multiple failed attempts at using Github Actions to lint our project, but after refactoring the workflow file, everything was working.

Some initial linting setups led to trivial amounts of refactoring minor details, in order to fix conflicts in the git flow caused by the timing of the automatic lint program.

Customer Questions

Our customer was very responsive as usual. At one point, we asked him if it was okay if we scheduled a user story to be “completed” in week 3 (i.e. if we did not do it for this class). This was a user story that would take a much longer time to complete, and it was at the bottom of the priority list. He promptly responded and told us that it was okay if we did not complete said user story. He also communicated with us to ask about his prioritization of the user stores. In addition, we kept him apprised of the development of the webpage in order to gauge his approval, and he was satisfied by what we showed him.

Integration Tests

As far as CI tests, we only used integration workflows consisting of Linters that check all the code files we push to make sure they follow certain style guidelines, i.e. enforcing indentation, CSS rule order, etc. We have set these linters to automatically fix simple errors, such as indentation and whitespace. These workflows are run every time a pull request (github's term for the merging of two branches) is made. All checks must pass for the branch to be allowed to merge.

Linters

Python

Our server is written in python. This server is checked using Integration workflows that run both Flake8 (https://flake8.pycqa.org/en/latest/) and Black (https://github.com/psf/black). These two linters check the code against a set of style rules every time we make a pull (merge two branches) request. We later abandoned black in favor of Flake8 as they provided redundant outputs, and for simplicity.

JavaScript

Robert has previously used ESLint (https://eslint.org/) in personal projects to lint his JS code, so he reused his existing config to lint our Javascript code. We currently do not have any JS code on our website, but this will enforce and fix errors in our programming.

CSS

We set up Stylelint (https://github.com/stylelint/stylelint) to lint our CSS files. We went with the standard configuration, but set the indentation level from 2 spaces to use tabs instead.

HTML

We attempted to get a HTML linter to work, but we could not find one that integrated into Github Actions. As such, we decided not to use an HTML linter.

Week 2 Schedule

Leftover from Last Week:

Account Creation

Due Date: 3/10/2020
Prerequisites: Basic Site Framework
Tasks:

- Client:
  - Password Hashing
- Server:
  - Account credential store
  - Creating user entry in account database

Projected time: 3ε

Account Login

Due Date: 3/10/2020
Prerequisites: Basic Site Framework, Account Creation
Tasks:

- Authentication Flow
  - Client: Login form, hashing function
  - Server: credential store, authentication matcher
- Session Management
  - Client: store session cookies to keep login across pages
  - Server: associate session activity to account

Projected time: 3ε

Edit Account

Due Date: 3/10/2020
Prerequisites: Basic Site Framework, Account Creation, Account Login
Tasks:

- Creating account settings page (Done)
- Updating user info in account db

Projected time: 2ε

Completion by 3/14/2020:

Save Favorites

Due Date: 3/10/2020
Prerequisites: Basic Site Framework, Search by Ticker Symbol, Account Login
Tasks:

- Client: Create button on stock visible when logged in
- Client: Send event to server when button is clicked
- Server: Adds/removes stock ticker to account db entry when event received
- Server: Checks if requested stock is saved to an account

Projected time: 3ε

Search History in Account Info

Due Date: 3/12/2020

Prerequisites: Basic Site Framework, Search by Ticker Symbol, Account Login
Tasks:
- Client: Sends event to server when stock is viewed
- Server: Adds/removes stock ticker to account db entry

Projected time: 1ε

Similar Stock Recommendations

Due Date: 3/14/2020
Prerequisites: Basic Site Framework, Search by Ticker Symbol, Account Login
Tasks:
- Client:
  - Button on stock visible when logged in
  - Click to toggle save or not save stock
  - Changes to show if saved or not
- Server:
  - Adds/removes stock ticker to account db entry
  - Checks if requested stock is saved to an account

Projected time: 3ε

Search by Stock Name

Due Date: 3/14/2020
Prerequisites: Basic Site Framework
Tasks:
- Search functionality:
  - Client: search entry field, results page and query to server.
  - Server: Performing lookup in name database, generating results page

Projected time: 3ε

Search by Category

Due Date: 3/14/2020
Prerequisites: Basic Site Framework
Tasks:
- Search functionality:
  - Client: search entry field, results page and query to server.
  - Server: Performing lookup in category database, generating results page

Projected time: 3ε

Change Site Theme

Due Date: 3/14/2020
Prerequisites: Basic Site Framework, Account Login
Tasks:
- Client:
  - Field on account page to change theme
- Server:
  - Store accounts' themes
  - Data containing site themes

Projected time: 3ε

Quick Account Creation

Due Date: 3/14/2020

Prerequisites: Basic Site Framework, Account Creation

Tasks:

- Client:
  - Button to create account via social media or Google account on "Create Account" page
- Server:
  - Google login service connection
  - Creating user entry in account database

Projected time: 3ε

Contributions

Customer

HW6

- Talked with our customer on Discord on 3/6/2020 to discuss the potential failure to complete one user story in the coming week.

Team Members

HW6 Document

- Felix Brucker: Refactoring, Diagram Reflection
- Robert Detjens: Refactoring, Integration Tests
- Remi Kendig: Customer Questions
- Dominykas Zobakas:
- Lyell Read: Customer Contact, Document Setup, Week 2 Schedule, Burndown Diagram, Week 1 Tasks, Integration Tests.

HW6 Coding

- Felix Brucker: Login and Account Creation pages, Settings Page
- Robert Detjens: Basic Site, Search by Ticker, Account login forms, CI Setup, API Documentation.
- Remi Kendig: Basic Site
- Dominykas Zobakas: Search by Ticker
- Lyell Read: Basic Site, Site Framework Server, Repository Setup, CI Linters for Python.

Appendix A: Historical Contribution Log

Customer

HW1

- We met with our customer Ghaith Shan after class on 1/23/2020.
- We have maintained communication with our customer on Discord since 1/16/2020.

HW2

- We met with our customer Ghaith Shan after class on 1/30/2020 to present the paper mockups.
- We discussed requirement revisions and went over the mockups with our customer over Discord on 1/31/2020.

HW3

- We did not meet with our customer as we had already verified that we share a common understanding of the goals and requirements for this app so far.

HW4

- We did not meet with our customer as we had already verified that we share a common understanding of the goals and requirements for this app so far.

HW5

- We met with our customer on Discord on 2/17/2020 to start the discussion of User Stories.
- We met with our customer in person on 2/18/2020 and discussed the User Stories further.
- We contacted our customer on Discord on 2/29/2020 to verify that we were interpreting the requirements of each user story correctly.
- In our meeting with our customer, he was very reasonable about what was needed.

Team Members

HW1

- Felix Brucker: Document Creation, Data Flow Diagram, Proofreading
- Robert Detjens: Functional Specifications, Non-functional Specifications, ERD Revision, Use Case 2, Use Case 2 MSD
- Remi Kendig: Use Case 1, Data Flow Diagram
- Dominykas Zobakas: ERD Revision, Use Case 1 MSD
- Lyell Read: ERD First Draft, Document Formatting, Customer Contributions, Functional Definitions, Non-Functional Definitions, Use Case 3, Use Case 3 MSD

HW2

- Felix Brucker: Mockup proofing, Use Case 1 Revision
- Robert Detjens: Paper mockups, Mockup meeting w/ customer, Definition revision
- Remi Kendig: Changes made since HW1
- Dominykas Zobakas: MSD Revisions, ERD revision
- Lyell Read: Mockup proofing, Digital Mockup Adaption, Appendix, Corrections to UC1 & UC3 MSDs.

HW3

- Felix Brucker: Implications
- Robert Detjens: Architecture 2 Data Flow Diagram, Fault Trees, Document Formatting
- Remi Kendig: Quality Attribute Assessment, Architecture Validation
- Dominykas Zobakas: Decomposition
- Lyell Read: Architecture 1 Data Flow Diagram, Team Member Contributions, Interfaces Revision

HW4

- Felix Brucker: Design Patterns, Interfaces
- Robert Detjens: UML Class Diagram, Interface Revision
- Remi Kendig: Incremental Development
- Dominykas Zobakas: Class Sequence Diagram
- Lyell Read: Document Formatting, Exception Handling, UML Class Diagram

HW5

- Felix Brucker: User Stories Implementation Plan
- Robert Detjens: User Story Descriptions, User Story Sequence
- Remi Kendig: UML Diagrams
- Dominykas Zobakas: User Story Diagrams
- Lyell Read: Customer Contact on Discord, Plan Organization by Group, User Story Sequence, User Stories Implementation Plan, Document Formatting.