Ctrl-Alt-Elite
Stock Portfolio Tracker

CSPB 3308 - Team #3 Final Presentation

Introduction

We are Ctrl-Alt-Elite! (aka Team 3)

Brendan McDonald
Shammi Pereira
Scott Wilson
Jimmy Hundertmark
James Giannoni

Vision: create a lightweight and easy-to-use stock tracking tool

Project Goals

- Simple user interface
- Summarize key financials - avoid info overload
- Maintain user data with login functionality
- Stock “Watchlist” - focus on what matters to me
- Easy links to news and recent filings

Overall, we achieved our vision! Here’s how…

Agenda

Project Tools Used, Evaluation, and Challenges Encountered…

- Project Management: Git VC + GitHub Project Repo, Trello
- Data Management: Python and SQL, Supabase
- Web Design: HTML and CSS, Javascript, Flask, Render

Demo & Audience Questions

Project Tools - Code + Work Management

Git version control + Github project repo

- Industry standard good reason; it is fully featured, but easy to use
- Enforced code reviews on pull requests

Trello

- Managed the work by breaking it into small pieces
- Allowed us to track our progress
- The entire team created and moved cards

Project Tools - Data Management

Python

- get_stock_data.py:
  - Libraries included: Edgartools (API wrapper of EDGAR), yfinance, Pandas, Numpy.
  - Functions defined: get_stock_data(), get_watchlist_data(), get_stock_news(), get_stock_filings().
  - Built a main() function that supported barebones user interaction to print out values for debugging.
- test_get_stock_data.py:
  - ‘unittest’ module and its assertions were used.
  - 6 functions were created to test for valid input and return values for all functions defined in get_stock_data.py.

Project Tools - Data Management

Supabase + SQL

- 50MB database for free
- ***supabase-js*** client library
  - the JavaScript interface to the PostGreSQL database hosted by Supabase
  - https://supabase.com/docs/reference/javascript/introduction
- 3 Tables built: user_information, stock, watchlist

Project Tools - Web Design

HTML + CSS

- Structured web page layouts to define static and dynamic content
- Styled interface for consistent visual design across web pages and included page-specific styling

Javascript

- Used to handle DOM dynamically
- Connect to *Supabase* for database operations (via *supabase-js* client library)

Flask

- Route functions to render and navigate between HTML pages
- Intermediary logic to integrate front end Javascript to backend API functions

Project Challenges

1. Tried to build Agile, but kinda delivered Waterfall
    a. MVP came later than planned - *”if you’re not embarrassed by v1.0, you waited too long”*
    b. May have achieved more functionality if we broke up core pieces into smaller chunks
2. Typical integration issues
    a. Python package management and IDE issues occured; nearly all were mitigated
    b. Supabase connectivity issues slowed development of login functionality
3. Git branching conflicts
    a. Teammates were relying on other teammates’ changes to be merged to code their features
4. Volatile stock market made us doubt our data a few too many times…

Demo

Homepage

Login page

Watchlist functionality

View News

Account Settings (WIP)

Documentation

Questions?