Project Report [012-2]

Title
Stock Sandbox: A Paper-Trading Stock Simulator

Who
- Grant DeBernardi (Grant-DeB)
- Danny Toy (dannytoy)
- Guilherme Avila Loges (guilhermeavlog)
- Josh Lettau (jlettau12)
- Luke Robinson (luker-robinson2)
- Peter Benda (peterbenda)

Project Description
Stock Sandbox is an online paper trading platform for anyone who wants to get better at stock trading without risking any real money. It uses live price data from the real U.S. stock market, so users can learn strategies on actual market conditions.

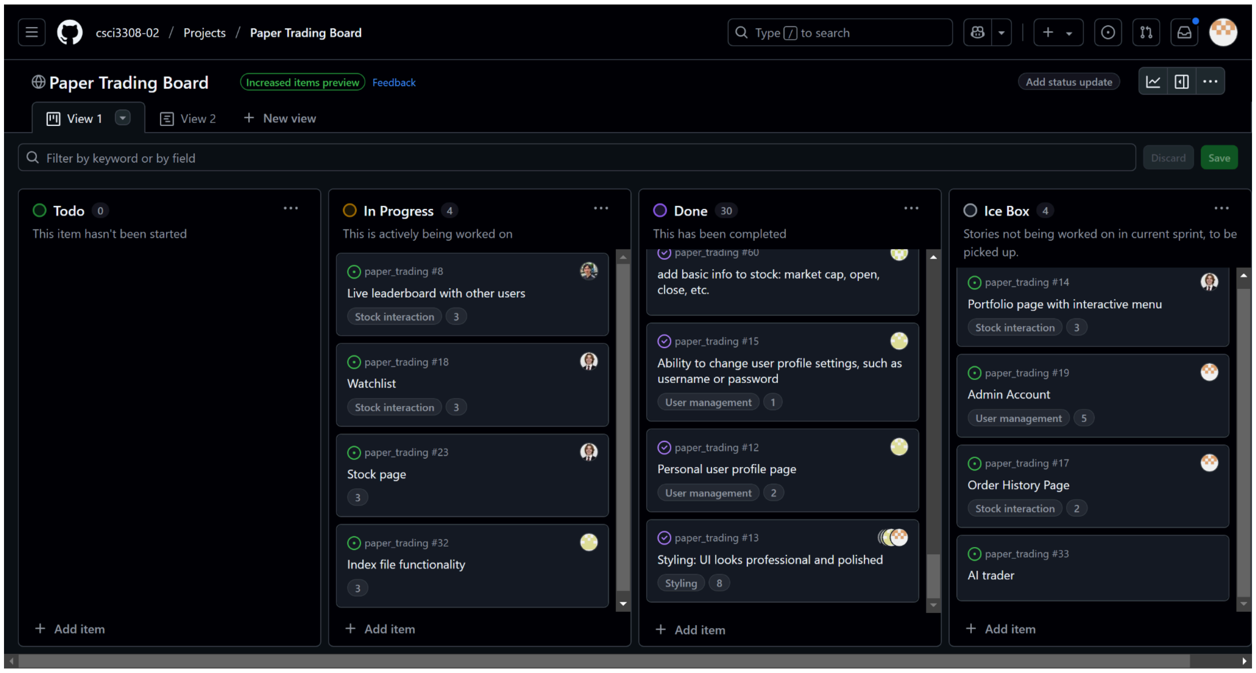
One of the main features is the interactive charts page, where you can search any U.S. stock ticker and view multiple charts at once. The charts have essential details like current price, open and close, daily highs and lows, 52-week range, volume, and market cap. It's an easy way to compare stocks side by side.

The trade page lets users buy or sell fractional shares, with real-time pricing updates based on the latest market data. Your holdings are displayed on the same page, so it's easy to track your stocks and sell with a single click.

There's also a leaderboard that ranks top virtual portfolios, a news section for recent headlines, and a detailed portfolio page showing account stats, performance, and trade history.

Whether you're new to trading or trying out a new strategy, Stock Sandbox can offer a realistic way to practice your trading skills.

Project Tracker – GitHub Project Board
Link: https://github.com/orgs/csci3308-02/projects/1
Screenshot:



Todo 0
This item hasn't been started
+ Add item

In Progress 4
This is actively being worked on

paper_trading #8
Live leaderboard with other users
Stock interaction 3

paper_trading #18
Watchlist
Stock interaction 3

paper_trading #23
Stock page
3

paper_trading #32
Index file functionality
3

+ Add item

Done 30
This has been completed

paper_trading #60
add basic info to stock: market cap, open, close, etc.

paper_trading #15
Ability to change user profile settings, such as username or password
User management 1

paper_trading #12
Personal user profile page
User management 2

paper_trading #13
Styling: UI looks professional and polished
Styling 8

+ Add item

Ice Box 4
Stories not being worked on in current sprint, to be picked up.

paper_trading #14
Portfolio page with interactive menu
Stock interaction 3

paper_trading #19
Admin Account
User management 5

paper_trading #17
Order History Page
Stock interaction 2

paper_trading #33
AI trader

+ Add item

Video Demonstration
Link: https://youtu.be/460i-M83oxQ

Version Control System
Repository: https://github.com/csci3308-02/paper_trading

Contributions

- **Grant DeBernardi** – Created Login, logout, and registration pages, as well as initially templating the index file and the docker compose file. Created user struct and database, allowing for users to have personal accounts and info. Also helped others with integrating team member changes, stylizing the UI, and fixing bugs.

- **Danny Toy** – Created the news page and portfolio page using FinnHub API and a controller for statistical calculations. Worked on general styling, UI and QOL features such as multiple charts, user settings, price display for trading, persistent pages, and overall website appearances. Deployed the site on Render.com.

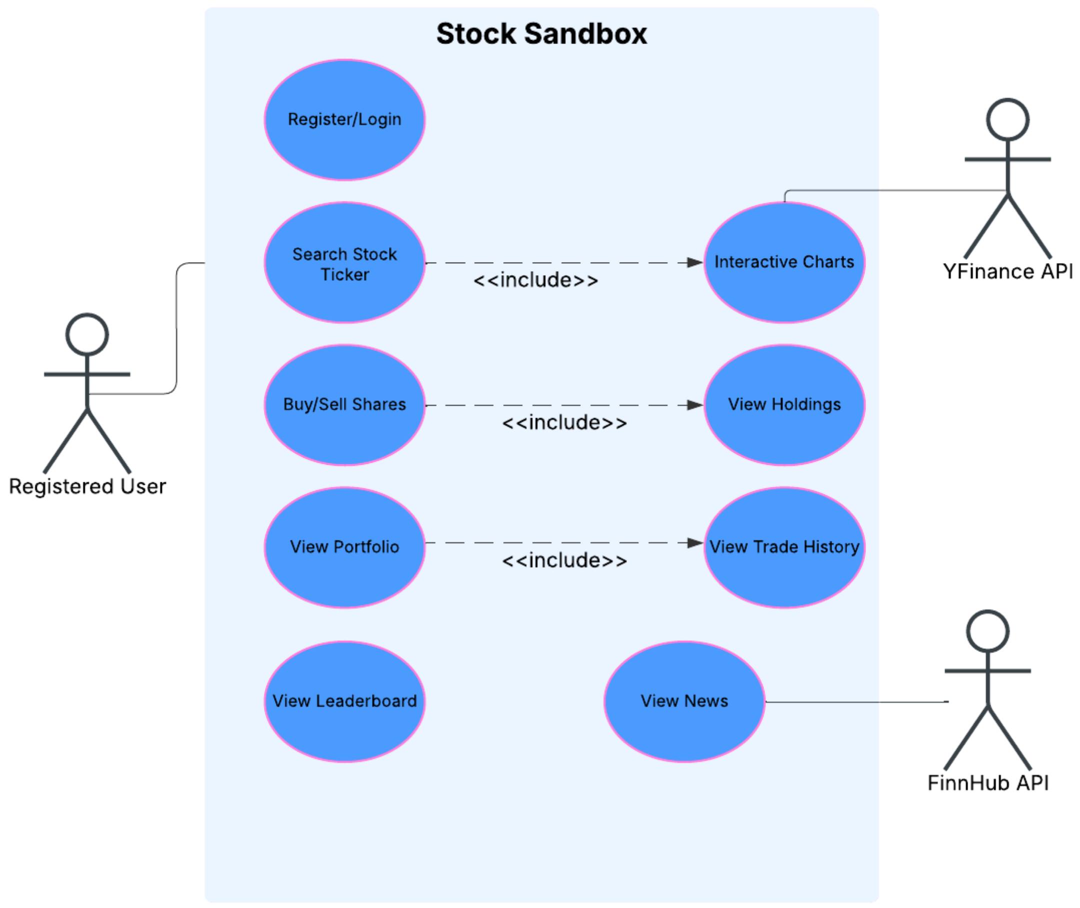
- **Guilherme Avila Loges** – Designed and styled the Leaderboard to showcase top-performing stocks from the database. Implemented routing for all pages and made minor enhancements to the backend.

- **Josh Lettau** – Created the charts and the backend for them using the free python library for Yahoo Finance, using it to display both real time and historical data, as well as important information about a given stock. Lead the overall UI design of the website. Integrated different branches into main and fixed many bugs found throughout the project as a whole.

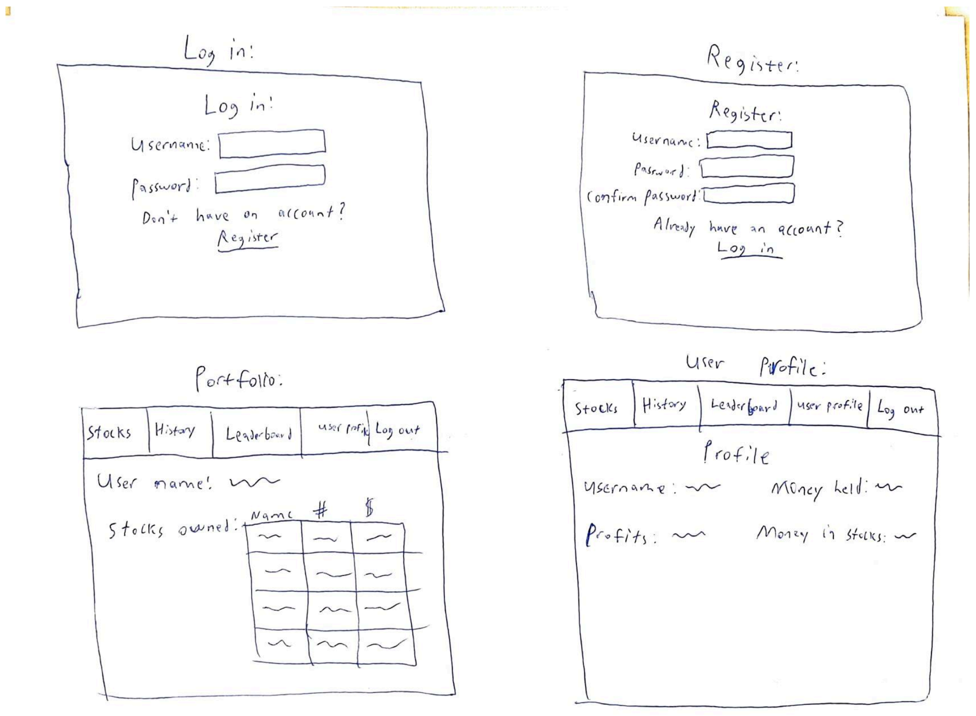
- **Luke Robinson** – Created the database schema to keep track of user information, user holdings, and stock prices. Created it in postgres docker image with sql scripts. I also wrote a lot of the python backend, which interfaced with the database, the node frontend, and the yfinance api. The python backend was deployed in a docker container. I also worked on the docker-compose environment.

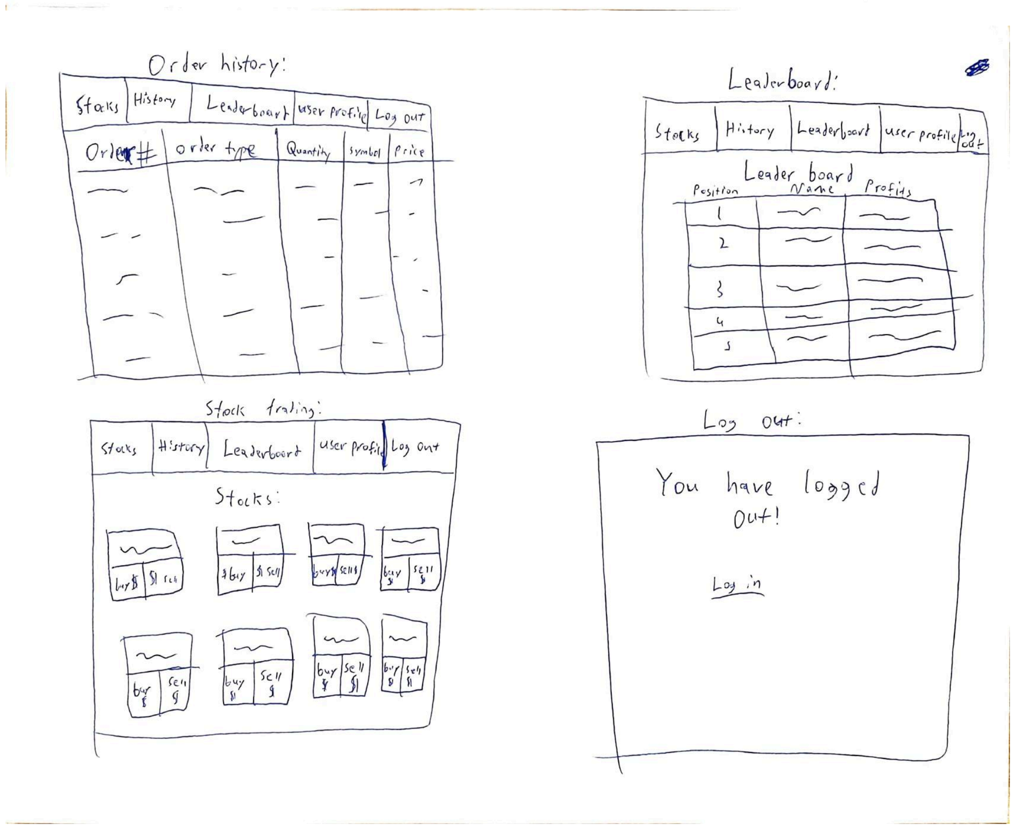
- **Peter Benda** – Created the partials and layout for the site as well as other handlebar documents like the header, nav, main, and footer. I implemented the search bar, enabling users to search for stocks by name or ticker symbol, frontend/back-end routing, and search functionality utilizing the Python API for Yahoo Finance.

Use Case Diagram


Stock Sandbox
Register/Login
Search Stock Ticker
Interactive Charts
<<include>>
Buy/Sell Shares
View Holdings
<<include>>
View Portfolio
View Trade History
<<include>>
View Leaderboard
View News
Registered User
YFinance API
FinnHub API

Wireframes





Test Results

1. When a user enters incorrect login information, they are given a message telling them that authentication failed and suggesting to navigate to the registration page if they don't have an account already. This will be tested by passing a username and password to the login page which does not correspond to any user in the database. We will also try an account that has already been registered and working to ensure that it is not denied by the system. This test can be conducted on a simple docker webpage, easily confirmed and observed. We tested different usernames and passwords and confirmed that registered users are the only ones that will be allowed into the website.

2. When a user attempts to buy a given stock, they are only allowed to when they have the required amount of money, with the change being immediately reflected in their information and stock portfolio. If they have insufficient money, they will be given an error message and the transaction will fail. This will be tested by attempting two buy orders: one where the price of the ordered number of stocks is within the user's budget, and one where the user attempts to order more than they can afford. This test can be conducted on a simple docker webpage, easily confirmed and observed. We tested different amounts of different stocks that were either more or less expensive than the user's budget (default $10,000). As expected, the buy orders only worked if there was enough money to afford them.

3. When a user attempts to sell a given stock, they are only allowed to when they have the required amount of said stock, with the change being immediately reflected in their information and stock portfolio. If they have insufficient shares, they will be given an error message and the transaction will fail. This will be tested by attempting three sell orders: one where the user has sufficient shares to sell, one where the user has shares but not enough to sell, and one where the user does not hold the stock at all. This test can be conducted on a simple docker webpage, easily confirmed and observed. We tested different amounts of different stocks that the user held enough of, held too little of, and did not hold at all. As expected, the sell orders only worked if the user had enough shares to cover them.

4. When a user views their stock portfolio, including their money held in stocks and their overall net worth, the page will accurately reflect the current state of their account, updating all numbers to be in line with the current stock market values. As an example, if a user buys 10 shares of a $10 stock, their account will reflect a $100 value. If the stock then increases to $11, then the user's account will show $110 to reflect the total value of their shares. This will be tested by creating a test account and buying some amount of a stock and then waiting for the API to update. We will record the values of the account before and after the stock price updates, ensuring that they are what we expect them to be. This test can be conducted on a simple docker webpage, easily confirmed and observed. During market hours, we bought some popular stocks like AAPL while keeping an eye on the price. We waited a minute or two to sell so the price could change, and when we sold, our portfolio's balance reflected the correct calculation for the change in money.

Deployment

Must Load API Server First: https://flask-api-nhm2.onrender.com

Then the Live App: https://stocksandbox.onrender.com