Error404

Stock Metrics

Semester: Fall 2017

Team Members: Jaynish Shah, Monark Modi, Yanjie Gao, Wenbo Xie, Bin Yu

GitHub Repository: https://github.com/monarkmodi/Stock-Metrics

Heroku Link: https://immense-brook-24952.herokuapp.com/StockMetrics/

Overview: Our web application is a cloud-based stock analysis tool designed for the stockholders, which is going to help them manage their stocks and keep a track of their net profit/loss. It uses machine learning techniques to analyze the existing metrics and tries to predict the future value of the stocks. It fetches from a list of 8000 stocks across Nasdaq, NYSE exchanges and performs SVM-based machine learning technique on historical values like open, close, high, low and volume to forecast the next day price changes. The web application could also work as a simple watch list and portfolio application for users to supervise their own stocks and compare their performances.

User Interface: We have implemented five different views for the user as shown in the screenshots section. A pre registered user will be required to login(Fig1) with their credentials. In case if the user forgets the password, the user is allowed to reset it by filling out the password reset form (future requirement)(Fig6). As soon as the user has logged in, they will be redirected to the home page (Fig2) which shows details about the user's portfolio including their net profit/loss, portfolio value and available funds. The user is able to check the real-time stock prices by using the portfolio page (Fig3). Specifically, the portfolio page could show five most recent search results, and the user is allowed to input the stock symbol via the input box at the bottom of the screen then hit submit. Besides, the stock prices will be shown in a table which includes the open, high, low, close, and volume of different stocks. More importantly, the stock metrics page (Fig4) is able to calculate different values automatically based on the technical indicators used. A user can add stocks to their portfolio using the order page(Fig5). In detail, the user is supposed to input the amount of stocks they bought and the corresponding stock symbol, then the total amount will be calculated directly. In the end, the user could sign out of their account clicking the logout button. After that, the user can also jump to the login page with the link on the logout page (Fig7).

Data Model: In our web application, we have defined four types of data models (Stock, User, StockMetrics, and Portfolio). The Stock model is going to store all the information about a specific stock, i.e. its title, stock ID, buy price, sell price and volume. The StockMetrics model is mapped directly from the previous one, which has properties including the stock name, metric title and metrics. The User model contains all the details about a specific registered user such as the name, email, password and their account balance. Additionally, the Portfolio model is designed to save the information about each user's stock account such as the username, portfolio value, buying power, withdrawable cash, cash balance and invested funds. All these models are related to each other in the sense that each user could have multiple stocks but only one portfolio and each portfolio could contain multiple stocks but the stock and Stockmetrics are one-to-one matching.

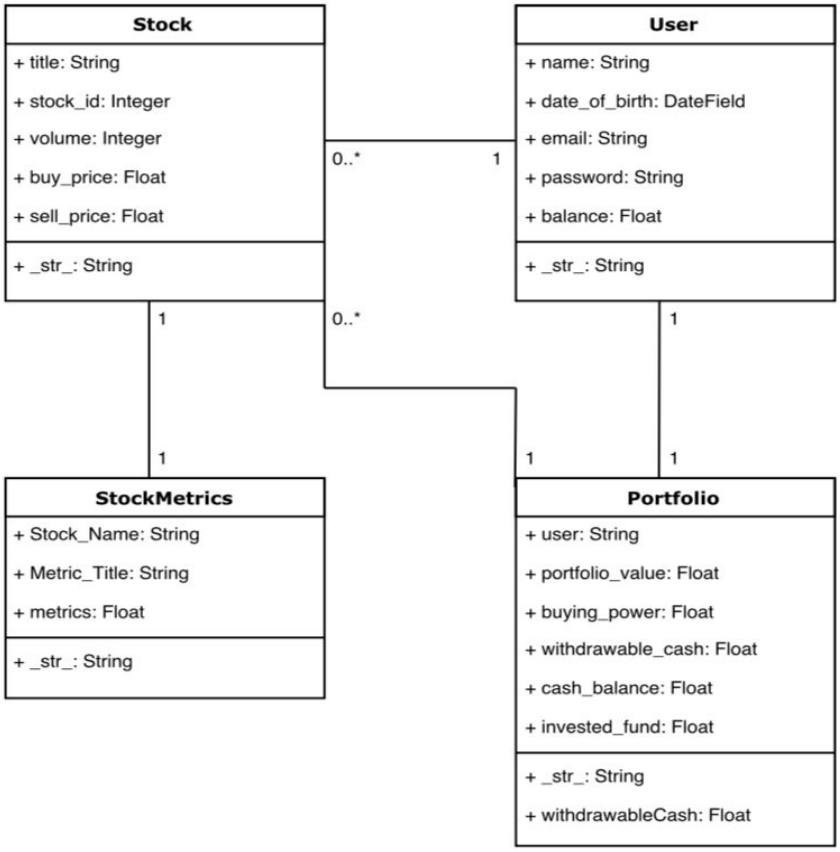

Fig: Data Model

URL Routes/Mappings: We have included some basic URL routes. Some of the URL routes are as soon as the user opens of the website, they will be required to login/create an account. Without creating an account/logging in they won't be able to view other pages of the website. Once they successfully login, they will be directed to their home page. url(r'^$', views.index, name='index') shows a user's home page(Fig2). url(r'^portfolio/?', views.portfolio, name='portfolio') directed the user to the portfolio page(Fig3). url(r'^stockMetric/?', views.stockMetric, name='stockMetric') directs the user to their "stockmetric" page.(Fig4). url(r'^order/?', views.order, name='order'), directed the user to the order page.

Authentication/Authorization: In our project, the superuser we created has all permissions including inserting, deleting, editing and querying all instances in the system. Apart from that, we used the admin site to build a user group called StockHolders which consist of several non admin users. The normal users are required to fill up the login form with their credentials for entering the system. In case a user forgets their password, the user is allowed to use the password reset form to reset the new passwords with the email they used for registration. As soon as they have successfully logged in, they will be redirected to the home page (Fig2) which shows details about the user's portfolio including their net profit/loss, portfolio value and available funds. Besides, the user is able to check the real-time stock prices and look at their five most recent search results by using the portfolio page (Fig3). On the stock metrics page (Fig4), the user could look into the automatic calculation results based on the technical

indicators used. Moreover, the user is allowed to add any stocks to their portfolios anytime by using the order page (Fig5). Finally, the user could quit the system by directly clicking the logout button. In general, the single user is prohibited to view other users' data or portfolios. For the future improvement, we are going to enhance the web security and the privacy of user data.

Team Choice: For the additional components, our team wishes to implement the complex stock metric indicators such as RSI (Relative Strength Index), OBV (On Balance Volume), SMA (Simple Moving Average) and Aroon Indicator. These indicators are complex mathematical calculations that assist in understand the stock price movement.

Conclusion: We had a great experience as a team working on this project especially we have learned a bunch of techniques and how to make a website using Django. We have learned UI implementation, data models, CSS, and django. Besides, the advantages of the agile development are really impressive for us and we will try to apply such kind of software engineering development method to our future projects. Meanwhile, we also learned about stocks and stock management, which gave us an extra interdisciplinary learning experience. However, we also faced a certain amount of challenges. At the very beginning, a few of our teammates were not familiar with using the version control system, so it was a difficult for our collaboration and project management. Besides, we planned to use Yahoo Finance for the real-time data collection but their service was terminated this year. Then, we switched to the API from Google Finance but we found out that their API was under maintenance and remodelling three days before our presentation. Therefore it was difficult for us to get the real time data of stocks because of the unstable API. Meanwhile, we also had some technical difficulties while working with Django forms such that we were not able to store the stock data in the view as a table and edit the properties of Django form elements. Finally, we would say even though our team collaboration was not that good at the beginning, it's getting much better with the pushing forward of our project, and our communication is also becoming more and more effective.

Screenshots:

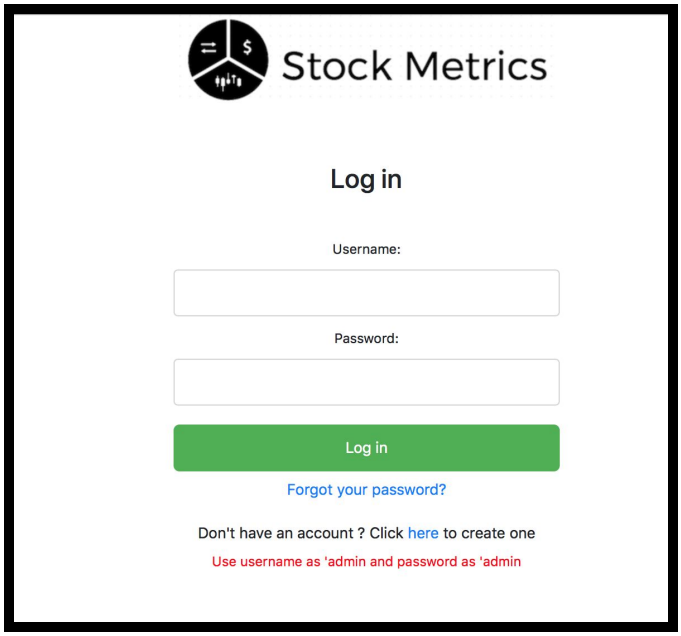

Fig1: Login Page

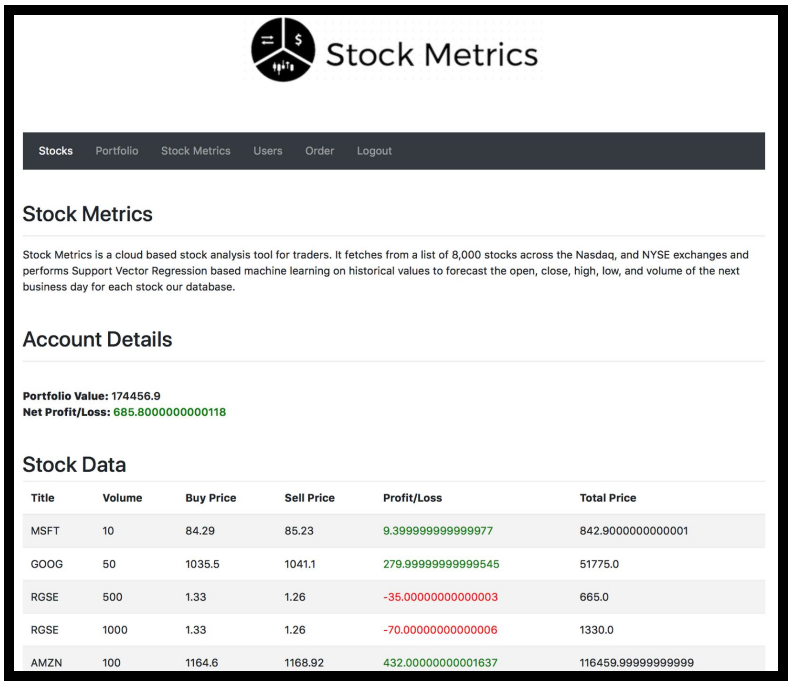

Fig2: User Home Page

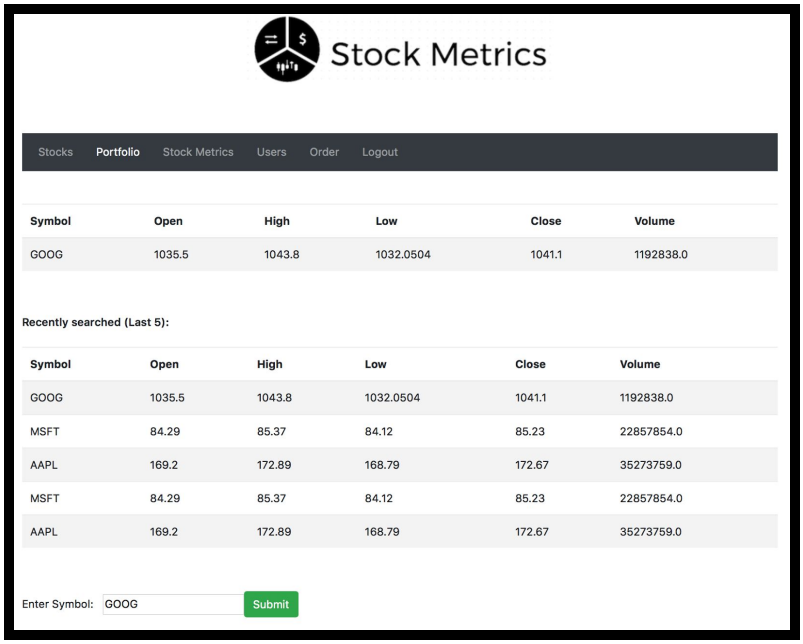

Fig3: Portfolio Page

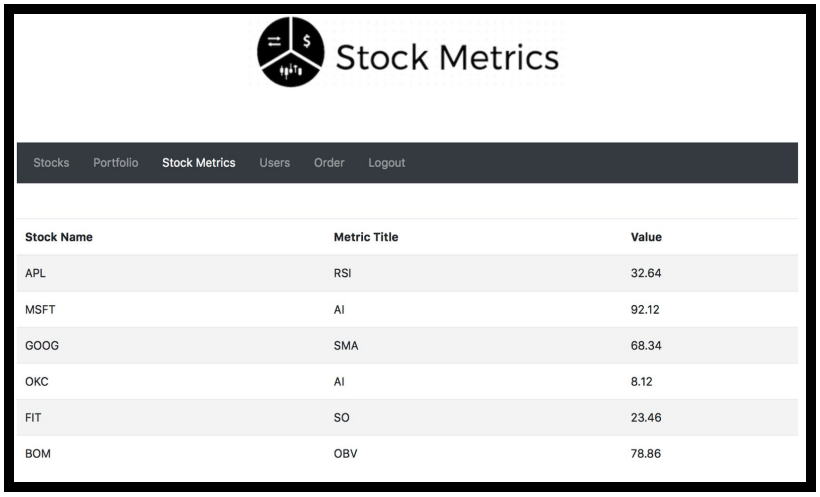

Fig4: Stock Metrics Page

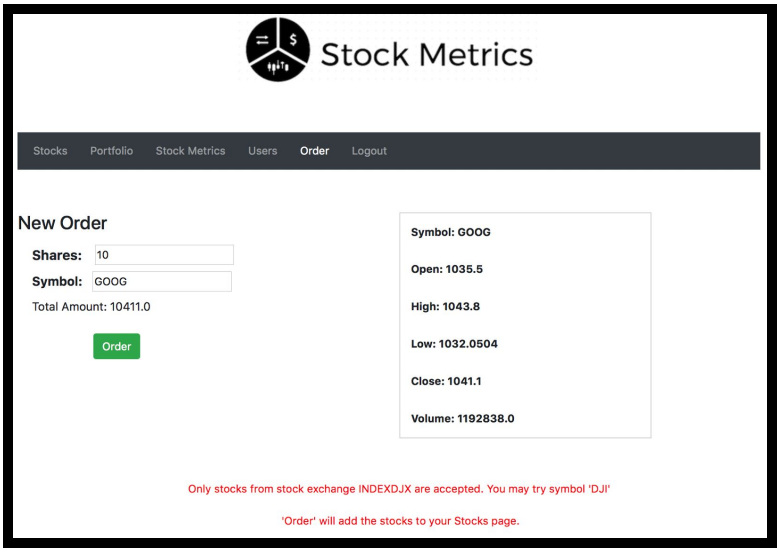

Fig5: Order Page

Fig6: Password Reset Form

Fig7: Logout Page