University of New South Wales

STOCK PORTFOLIO MANAGEMENT SYSTEM REPORT(IT CAPSTONE PROJECT)

Group: Running Man

Name	Role	sID	Email
Rui Liang	Scrum Master	z5174445	rui.liang1@student.unsw.edu.au
Zhicheng Tao	Developer	z5148738	z5148738@unsw.edu.au
Chuguan Tian	Developer	z5145006	z5145006@unsw.edu.au
Hao Jiang	Developer	z5146830	z5146830@ad.unsw.edu.au

A project report submitted for COMP9900/19T3

November 26, 2019

Abstract

Since the living standards have improved so much in recent years, more and more people tend to spend their spare money on stock investment as an extra income source. However, due to a lack of financial background and techniques about stock data analysis, many people buy stocks blindly which causes great loss. There are a lot of stock analysis software on the market, however, they don't have the ability to help users with intelligent analysis or predict the trend of stock price, and most of them are hard to use. Therefore, we propose to develop a smart stock portfolio management system that equipped with modern machine learning prediction tools as well as factor analysis tools to help investors to buy stock wisely. In this report, we first introduce the system architecture of the overall system and functionalities when comparing with other financial systems, then we elaborate the detail of each functionality developed by the team as well as challenges.

Contents

1 Overview **1**
- 1.1 Aim and objectives . . . 1
- 1.2 Application architecture . . . 1
- 1.3 Functionality Summary . . . 3

2 Functionalities **4**
- 2.1 User administration . . . 4
- 2.2 Portfolio management console . . . 5
- 2.3 Stock search . . . 6
- 2.4 Stock detail information . . . 6
- 2.5 Stock price prediction . . . 7
- 2.6 Factor analysis . . . 7
- 2.7 Comparison with other system . . . 8

3 Frontend Implementation and Challenges **15**
- 3.1 System framework . . . 15
- 3.2 Challenges . . . 16

4 Backend Implementation and Challenges **17**
- 4.1 Backend Architecture . . . 17
- 4.2 Database design . . . 18
- 4.3 Backend feature description . . . 19
  - 4.3.1 User engine . . . 19
  - 4.3.2 User profile . . . 21
  - 4.3.3 Stock engine . . . 22
  - 4.3.4 Stock price prediction . . . 24
  - 4.3.5 Stock Recommendation . . . 28

5 Third party libraries **30**

6 User Documentation **31**

7 Test Documentation **32**

Bibliography **33**

Chapter 1

Overview

1.1 Aim and objectives

Nowadays, smart phones, tablets and smart TV are becoming more and more popular and powerful, it will be beneficial and convenient if we can access the stock application management console through these devices anytime at anywhere. Our goal for this project is to create a web stock portfolio management application that support many modern electronic devices via web browser.

The reason why we choose to develop a web application rather than a computer/smartphone software is that different platform tend to have different framework for running the software and it is hard to maintain software consistence, in addition, it is time consuming and costly to develop application in different platforms. For example, an IOS application that runs on Apply devices cannot be run directly in Android devices or computers, we have to develop them separately using different techniques and libraries/packages. On the other hand, a web application can run on any device that has a web browser, so that we do not need to bother developing multiple software that support different electronic device platforms.

Moreover, we aim to tackle the existing system drawbacks and develop a web application that is easy to use, and do not need much finance background to easily choose a set of stocks to create a portfolio with built in analytic tools such as factor analysis tools and the state-of-the-art machine learning price prediction tools.

In a nutshell, our objectives are:

- Easy to use
- Support multiple electronic devices (IOS, Android, Mac, Windows, Linux, Smart TV, etc)
- Responsive and fast
- Rich functions and equipped with state-of-the-art financial analysis tools

1.2 Application architecture

The high level application architecture can be illustrate as the following figure:

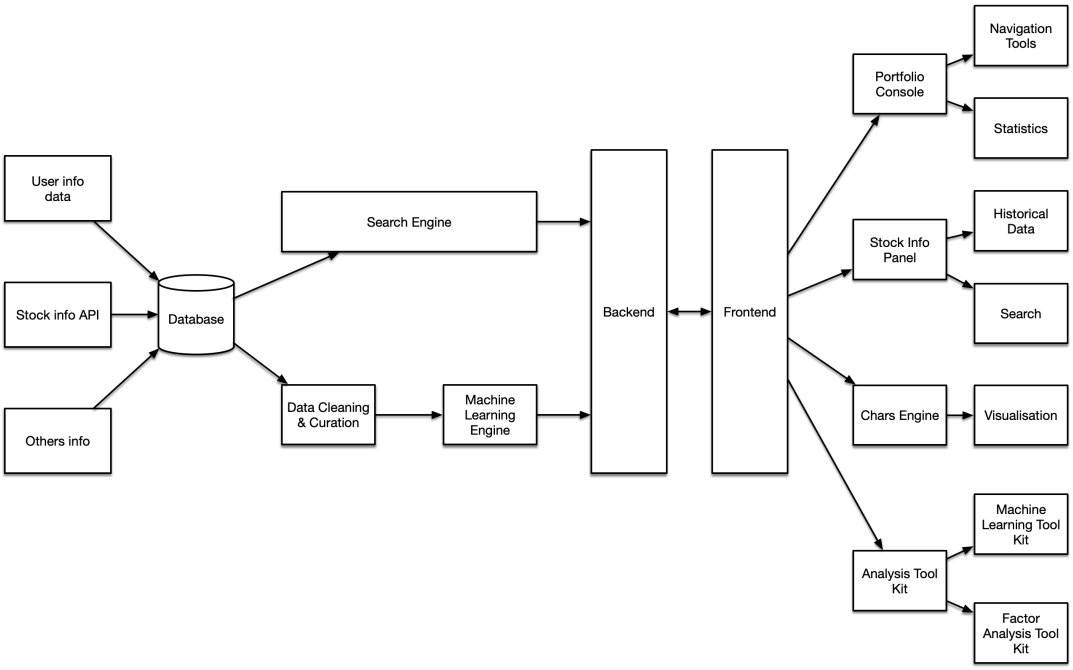


Figure 1.1: *High level application architecture*

As the graph shows, the web portfolio management application can be divided into two major parts, namely backend and frontend. We developped backend using Django[1] REST web framework and use React[2] as the frontend framework. Backend consists of the logic part behind the scene, when it receives a query or request from the frontend, it will pull the data from the main database, then go through different information process engine and finally give a response in commonly used json format. The application can periodically update the database from different APIs from the internet and fetch stock information as well as news data etc. Database also acts as a warehouse for user information such as user name, password as well as stock portfolio data.

As for the application frontend, it is the GUI (Graphical User Interfaces) for the web app which can interact with normal users. The frontend has been designed in four major components, namely Portfolio Console, Stock Info Panel, Charts Engine and Analysis Tool Kit. Each of which acts as a key feature of the application. Users can navigate through different components through the portfolio management console and easily see the statistics in the dashboard. If we are interested in a particular company, we can perform a fuzzy search on the company name. The Charts Engine will then render all the historical data for the stock info of that company. Moreover, a stock analysis tool kit which uses the-state-of-the-art machine learning algorithms will help investors to perform deep analysis of the stock price and portfolio combination.

1.3 Functionality Summary

The overall functionalities are fulfilled as the project proposed and can be summarised as the following:

- User administration: login/logout user, registration, authentication, administration, user profile.
- Portfolio management: statistical panel, responsive charts, stocking holding list, transaction list.
- Stock list and search: list of stocks, search, fuzzy search, dynamic columns.
- Stock detail information: interactive charts, news feed.
- Stock price prediction: predict each stock price.
- Factor analysis: apply factor model and recommend stocks.

The detail of each feature will be elaborated in the coming chapters.

Chapter 2

Functionalities

2.1 User administration

User administration includes user registration, authentication, administration and user profile editing. It is the basic component of the portfolio management application. A user can only use the application after authentication and edit his profile. Admin user are in charge of managing all the user data, which can be implemented in the backend. The difficulty score for this epic can be shown as follows:

Component	Feature	Major Challenge	Difficulty Score
Registration	Register a user.	Frontend design/implementation and update backend database.	5/10
Authentication	Authenticate a user.	Difficult to resolve cross-platform authentication.	7/10
Administration	Backend admin console.	Need to implement backend admin page.	7/10
User Profile	Edit user profile.	Frontend design/implementation and update backend data	6/10
Total Difficulty Score	6.25/10		

Table 2.1: *User administration*

2.2 Portfolio management console

Portfolio management console is the major feature of the web application, it consists of three major components: statistic panel, charts component and stock holding list. The statistics panel mainly shows the user portfolio summary in order to be easily checked by users. Charts component acts as a visualisation tool for users to analysis portfolio components and historical profit. Stock holding list shows current holding list of shares of a company. Users can add or delete a stock, add or delete transactions and synchronise the stocks with the current prices.

We can break down this epic into several features together with individual difficulty score as follows:

Component	Feature	Major Challenge	Difficulty Score
Statistic Panel	Portfolio summary	Fetch historical portfolio data, calculation as well as frontend design/implementation.	7/10
Charts	Draw radar chart, pie char and composed graph for historical portfolio data.	Not easy to draw graph and calculate data in frontend/backend.	10/10
Stock Holding List	Add/delete stock	Difficult in implementing interactive frontend.	10/10
	Add/delete transaction	Difficult in implementing interactive web pages.	10/10
	Synchronise with current stock data.	Difficult in fetching data and calculate statistics.	10/10
Total Difficulty Score	9.4/10		

Table 2.2: *Portfolio management console*

2.3 Stock search

Searching a stock is an essential part of the portfolio management application in order to help users to add a particular stock into the portfolio. Users can input the full name (case insensitive) of the stock or its financial code in order to perform a search. Then the result would be sorted by the probability to be matched in ascending order and returned to users, if a user click on a particular stock, the application would be redirected to the stock information page. Apart from basic full text search, fuzzy search is enabled in this project. Since the search can be done by using "material-table" library for frontend, so that the difficulty score is not high.

Component	Feature	Major Challenge	Difficulty Score
Search	Search a stock.	Need to find an effective external package to handle the search.	3/10
Fuzzy Search	Search a stock.	Need to find an effective external package to handle the search.	3/10
Total Difficulty Score	3/10		

Table 2.3: *Stock search*

2.4 Stock detail information

Stock detail information is also an essential part of the application which enable users to see the historical price movement and technical charts of a stock.

The stock detail view consists of two major components, namely stock price movement/technical charts and news feed. Users can interactively see the historical price movement of a particular stock in interactive view. In addition, users can also see different technical lines in the graph. We have used "react-stockcharts" library to build the frontend in order to achieve high performance. News feed is enabled by our project and we can also search news by keywords.

Component	Feature	Major Challenge	Difficulty Score
Interactive Chart	Draw historical stock data and technical index charts.	Need to learn how to use the external library and configure for our application.	8/10
News Feed	Display news for a particular stock.	Difficult to acquire free news feed and implementation in frontend.	7/10
Total Difficulty Score	7.5/10		

Table 2.4: *Stock detail information*

2.5 Stock price prediction

Nowadays, machine learning and AI becomes more and more popular due to the ground breaking deep learning algorithm [3] and much faster cpu and gpu than the last few decades. Machine learning technics can be utilised on various applications, forecasting time-series data is one of them. However, we can rarely see stock price prediction products in the market. One of the reason for this is that stock data is generated by non-stationary time-series "windows" so that it is difficult to forecast a feature "window" according to past data [4]. Having said that, when using the right model and factor, we can still generate a probability that whether a stock price may go up or down to some degree by applying machine learning on time-series data. Another reason is that when a strategy which is making profit is used by most of the people, then this strategy will lose its ability to gain profit in the near future [5] because financial market can only be robust when people are trading using different strategies. This is why many stock price prediction products can only be accessed by premium users or be kept secrete (like a secrete recipe in a restaurant).

We have tried to use traditional machine learning model such as linear model, KNN and external library such as "prophet" to predict the stock price. Since selecting, training and analysing different models as well as tuning parameters are time-consuming tasks and cannot guarantee the effectiveness of the algorithm. The development of our project only lasts for around 5 weeks and we cannot develop the project in full time (we have other courses in this term), so that we choose to use external library to predict the stock price.

Component	Feature	Major Challenge	Difficulty Score
Stock price prediction	Developing, training, testing, analysing and evaluating different machine learning models on stock data and final prediction result demo.	Difficult to develop an effective machine learning model for stock price prediction and store them in the database.	8/10
Total Difficulty Score	8/10		

Table 2.5: *Stock price prediction*

2.6 Factor analysis

Factor analysis is a commonly used technic in stock selection and portfolio evaluation [6] [7] [8]. We can often see a variety of technical and financial indices of a particular stock or a financial market such as PE (Price-Earnings), PB (Price-to-Earning Ratio), EPS (Earnings Per Share) and ROA (Return on Asset), etc. These indices are correlated with the stock price by some means or other, with factor analysis, we can explore the most correlated factors (indices) and select them as the final stock selection factors. Then we can score each stock according to these factors and rank them in descending order, the stocks gets higher score then can be added into the stock portfolio.

Factor analysis requires a lot of concepts, formula and technical details to be digested, all the members in our team do not have a financial background, however, it is very exciting

to learn something new and try to apply them in real world financial market. We decided to use Fama-French Three Factor model to rank all the stock in our database in order to recommend a stock to users.

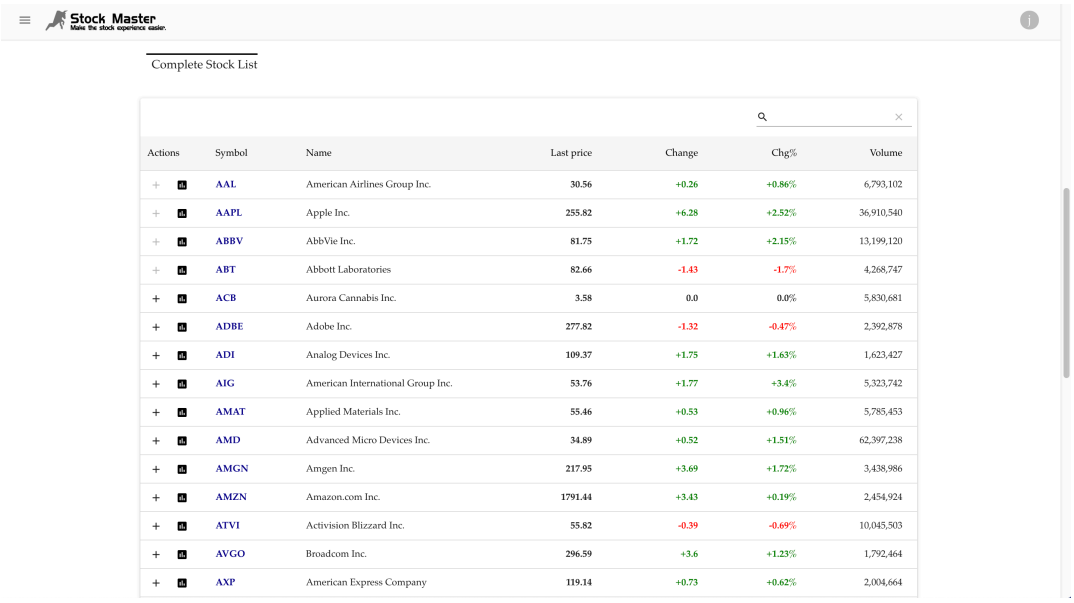
Component	Feature	Major Challenge	Difficulty Score
Stock Recommendation	Applying Fama-French three Factor model on stock data and generate stock recommendation.	Difficult to digest factor analysis concepts and mathematical formula.	10/10
Total Difficulty Score	10/10		

Table 2.6: *Factor analysis*

2.7 Comparison with other system

For stock list, we have the following innovation:

- The stock list is searchable and elastic.
- Symbol name and basic statistics can be easily identified.
- Adding stock into portfolio is convenient.
- Layout is clean and easy to navigate.



Actions	Symbol	Name	Last price	Change	Chg%	Volume
+	AAL	American Airlines Group Inc.	30.56	+0.26	+0.86%	6,793,102
+	AAPL	Apple Inc.	255.82	+6.28	+2.52%	36,910,540
+	ABBV	AbbVie Inc.	81.75	+1.72	+2.15%	13,199,120
+	ABT	Abbott Laboratories	82.66	-1.43	-1.7%	4,268,747
+	ACB	Aurora Cannabis Inc.	3.58	0.0	0.0%	5,830,681
+	ADBE	Adobe Inc.	277.82	-1.32	-0.47%	2,392,878
+	ADI	Analog Devices Inc.	109.37	+1.75	+1.63%	1,623,427
+	AIG	American International Group Inc.	53.76	+1.77	+3.4%	5,323,742
+	AMAT	Applied Materials Inc.	55.46	+0.53	+0.96%	5,785,453
+	AMD	Advanced Micro Devices Inc.	34.89	+0.52	+1.51%	62,397,238
+	AMGN	Amgen Inc.	217.95	+3.69	+1.72%	3,438,986
+	AMZN	Amazon.com Inc.	1791.44	+3.43	+0.19%	2,454,924
+	ATVI	Activision Blizzard Inc.	55.82	-0.39	-0.69%	10,045,503
+	AVGO	Broadcom Inc.	296.59	+3.6	+1.23%	1,792,464
+	AXP	American Express Company	119.14	+0.73	+0.62%	2,004,664

Figure 2.1: *Stock list of our application*

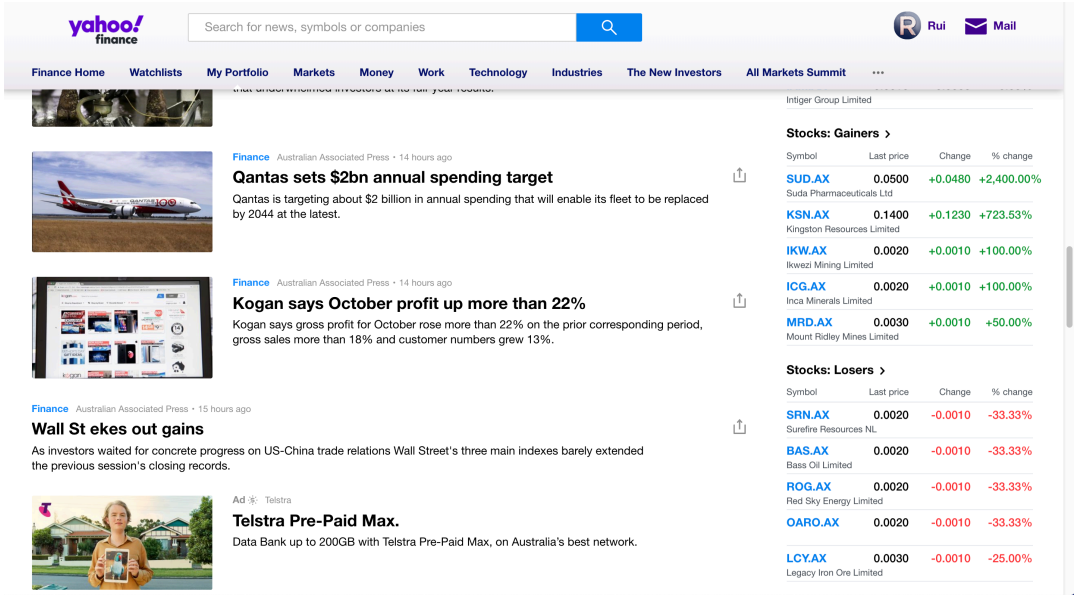


Figure 2.2: *Stock list of Yahoo Finance*

For stock recommendation, we have the following innovation and Yahoo Finance does not provide such functionality:

- Our powerful system provides user stock recommendation, which is based on Fama-French Three Factor Model.
- Rank the most profitable stock among all the other stocks.

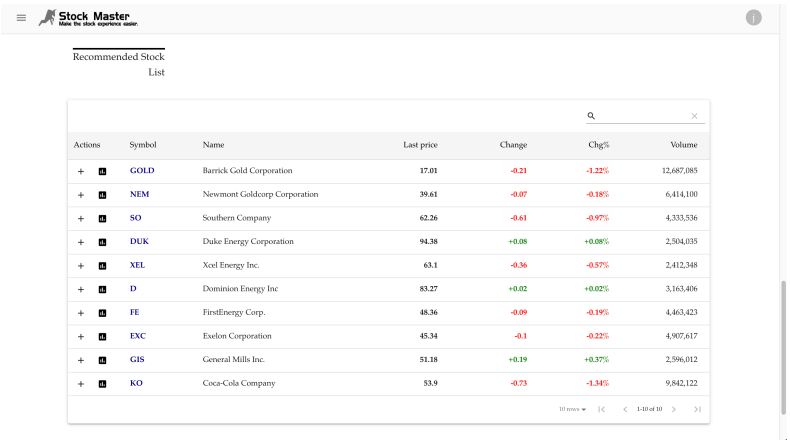


Figure 2.3: *Recommend stock list of our appliaction*

For stock detail, we have the following innovation and Yahoo Finance does not provide such functionality:

- Our system can predict stock price and shows how our model fits historical data. You can decide whether or not to trust the prediction by looking at the validation charts.
- The stock detail view also includes technical indicators so that you can easily analyse the historical data.
- The stock news is searchable.

For portfolio management console, we have the following innovation and Yahoo Finance does not provide such functionality:

- User can easily visualise the portfolio data in a dynamic and interactive way.
- Holding list can be expand and elastic.
- Each symbols transaction data can be visualise and we can compare stock price with cost basis.

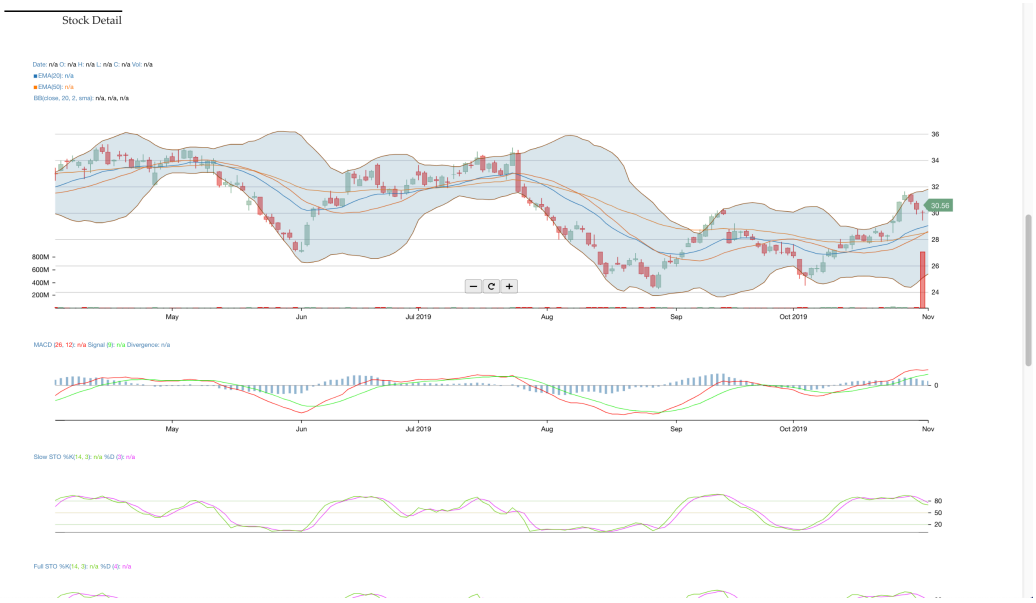


Figure 2.4: *Stock detail of our appliaction*

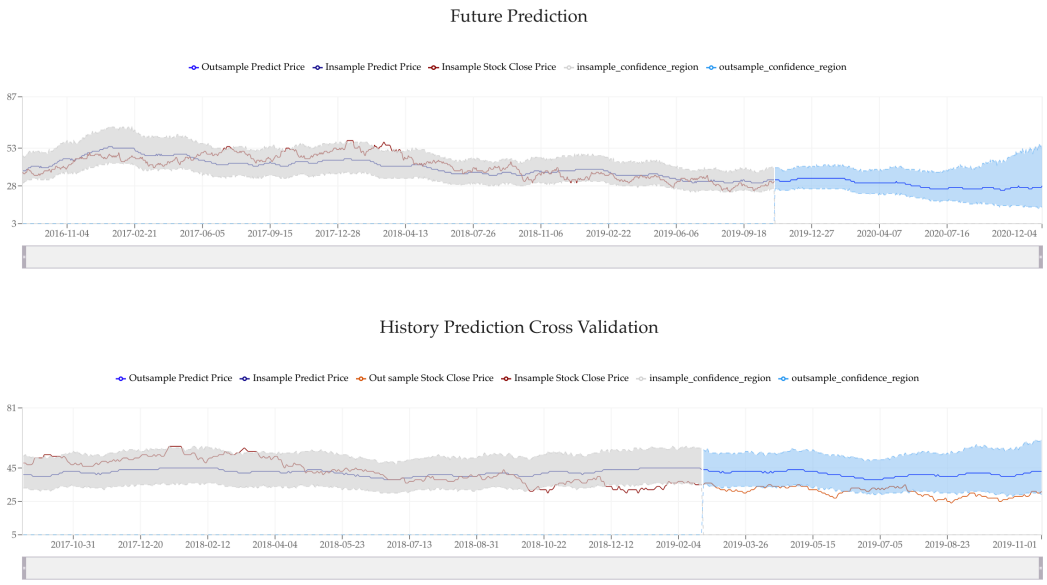


Figure 2.5: *Stock price prediction of our appliaction*

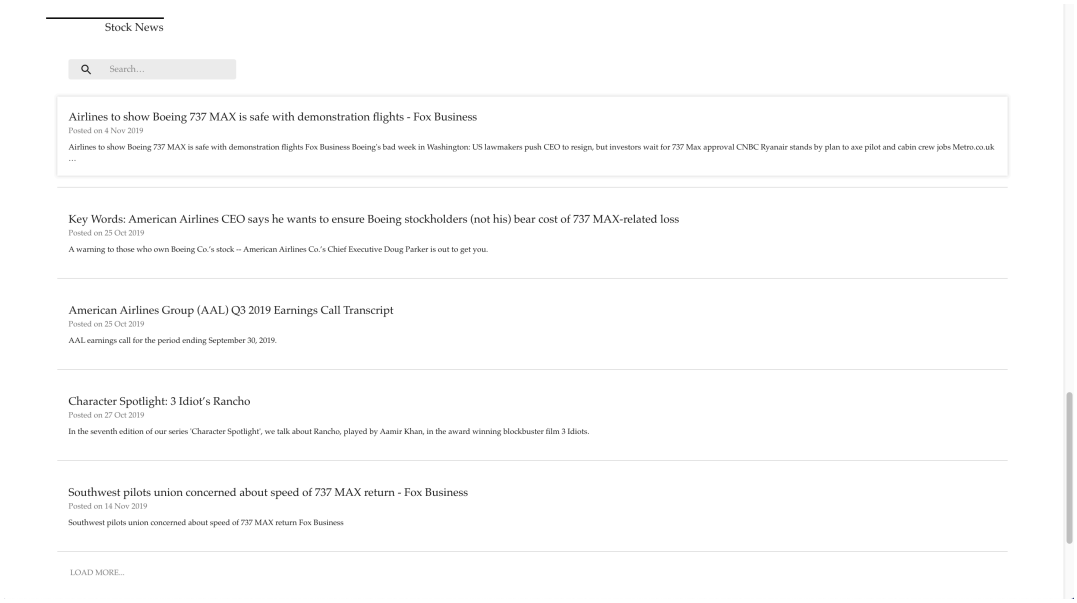


Figure 2.6: *Stock news of our appliaction*

Figure 2.7: *Yahoo Finance stock detail*

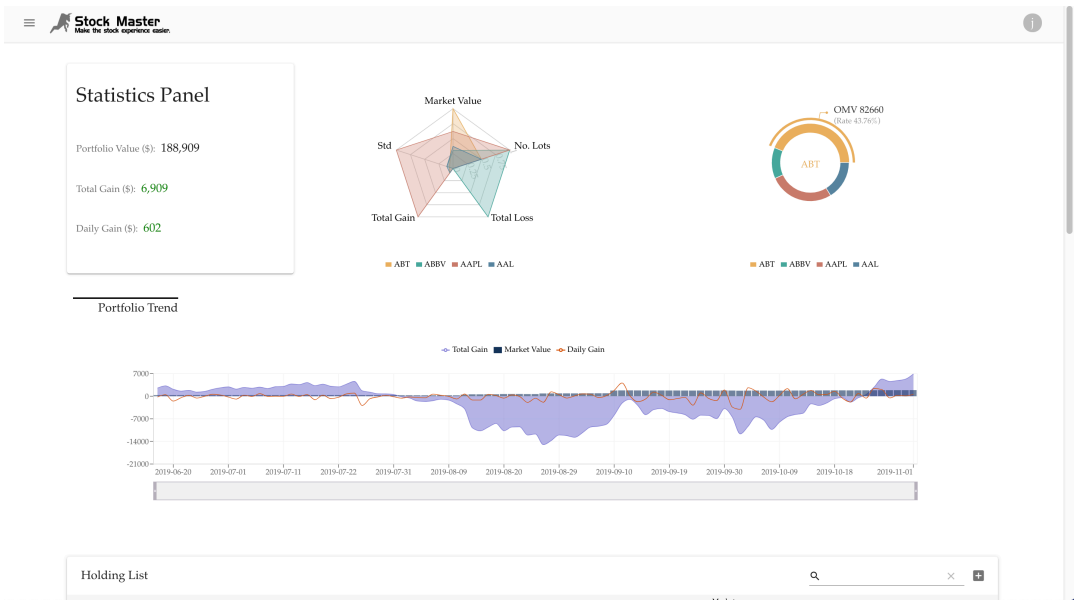


Figure 2.8: *Portfolio interactive chart of our appliaction*

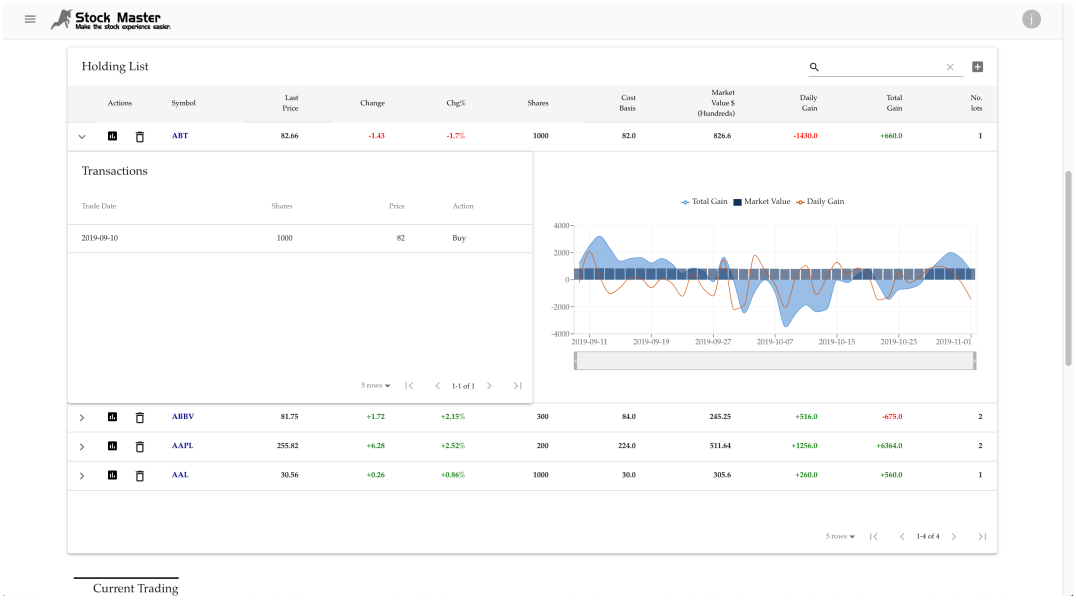


Figure 2.9: *Portfolio interactive chart of our appliaction*

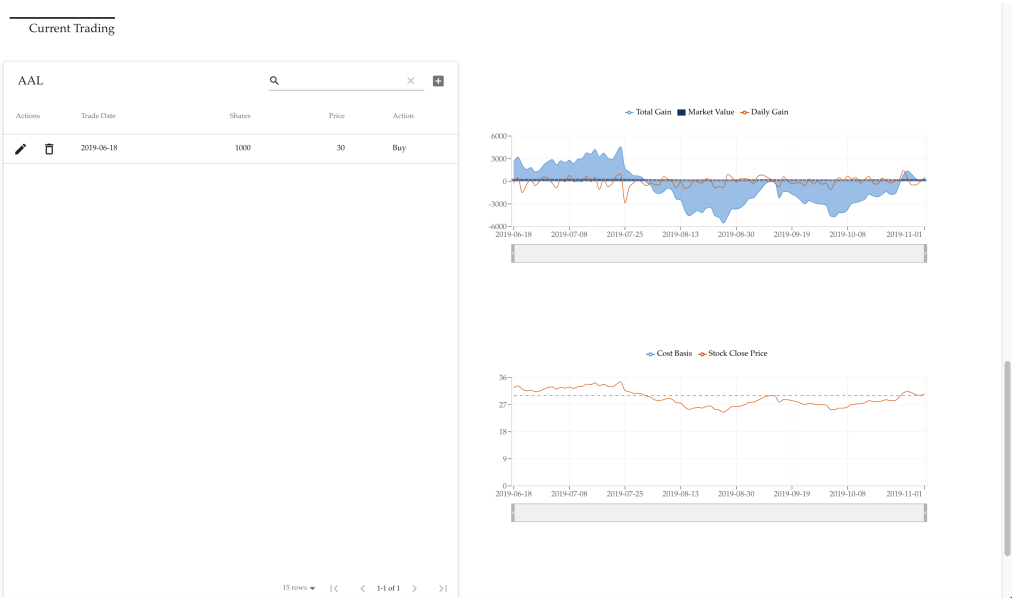


Figure 2.10: *Portfolio interactive chart of our appliaction*

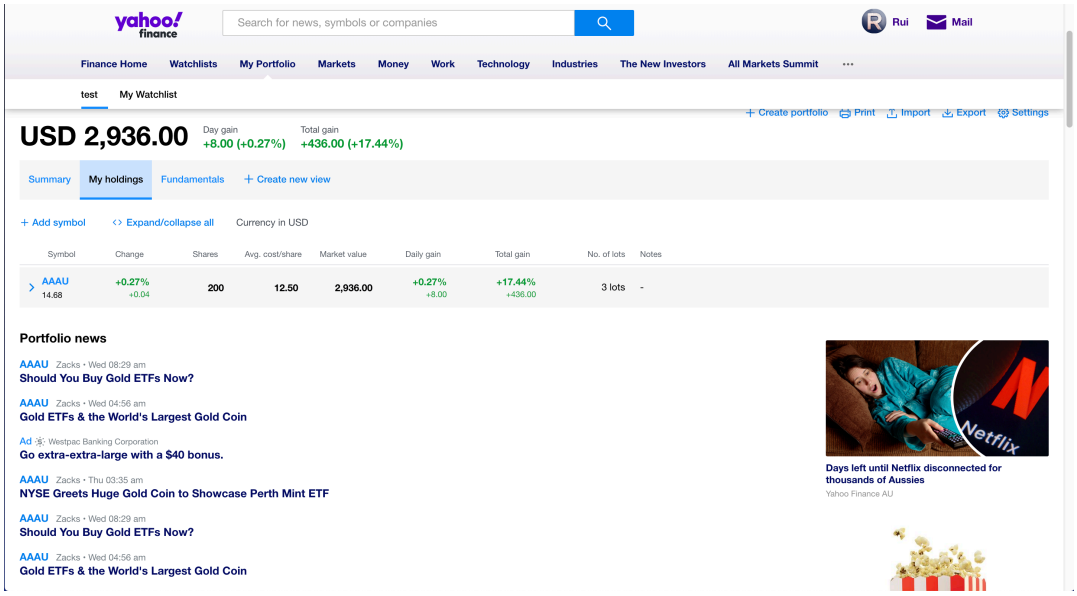


Figure 2.11: *Yahoo Finance portfolio*

Chapter 3

Frontend Implementation and Challenges

3.1 System framework

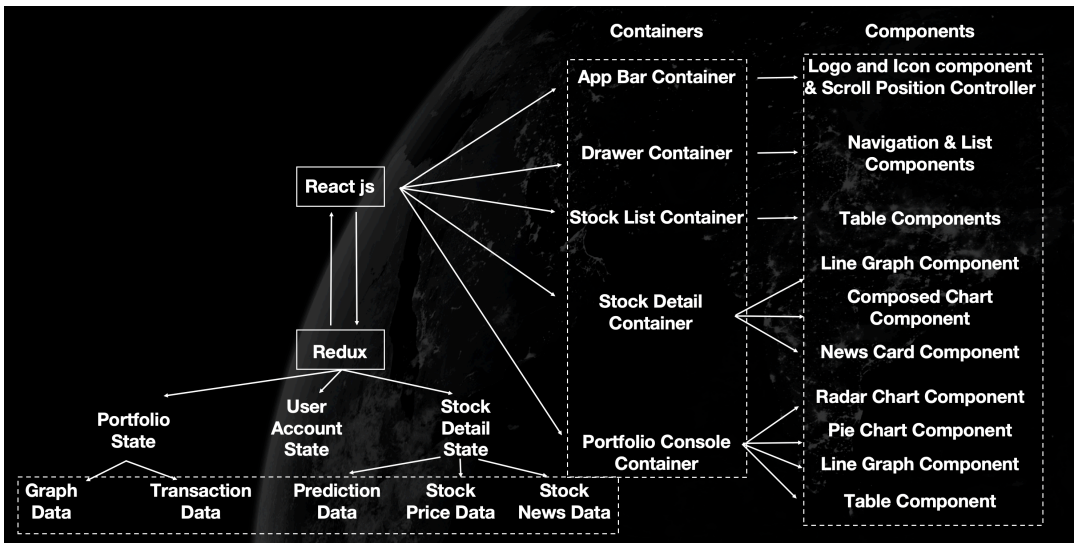


Figure 3.1: *Frontend system framework*

The frontend framework consist of the several parts as the above figure described, we mainly use "React.js", "Redux" and "Material-UI" to construct the frontend. "React" is a javascript library for building interactive and responsive web applications. Our application is written fully in javascript rather than html, so that it can dynamically change the content according to the frontend state. "Rudux" is the frontend state management library, which acts as the "database" for frontend, storing all the data to be rendered for the frontend.

The work flow can be described as follows: first of all, when the application starts, "react" will tell "redux" to sent API calls to the backend, when "redux" receives the returned data, it will update the database and check whether the database has been changed, then it will

tell “react” that which part of the database has been changed and “react” will re-render that components which uses that part of the data. In this way, the web application will perform much better than re-rendering all the components if only a small part of the data has been changed.

3.2 Challenges

The major challenge of implementing the frontend is that all the component needs to be written in JSX rather than HTML and each component in “Material-UI” has its own property, so that we need to frequently check the official documentation and experiment with those properties. For example, if I want to create a “TextField” component, then I need to specify “variant=”outlined“, margin=”normal“, required, fullWidth, id=”loginUserName“, label=”User Name“, name=”loginUserName“, autoComplete=”user name“, autoFocus, onChange=this.onChange”. This is just one simple input text field, if we want to build a complete component, we need to specify hundreds of properties then we also need to write API request functions called “actions” as well as update frontend database functions which called “reducers”. In order to implement a simple login component, we need to write 400 to 500 lines of code. The complete implementation of the frontend takes more than 5000 lines of code. This is the nature of “React” and “Redux” development combination, however, only then we can achieve high performance web application.

Another challenge is that when implementing the charts in portfolio, I have to experiment with different data to adjust for the appearance of the chart, and I also have to calculate many figures for all the charts in the “reducer” so that they can match the data format of the chart component.

When implementing the transaction component, I have to make sure user has inputed the correct data when adding/editing/deleting transactions. For example, the transaction date should not greater than the last stock price date in the database, the stock shares you sell should not greater than the shares you buy.

There are many other challenges when implementing the frontend, such as making the charts responsive, making sure the state change match the news feed change between search and each stock, using customised sorting algorithm to sort tables, adjust chart width according to different screen sizes etc.

Chapter 4

Backend Implementation and Challenges

4.1 Backend Architecture

The system architecture graph can be illustrated as the following figure:

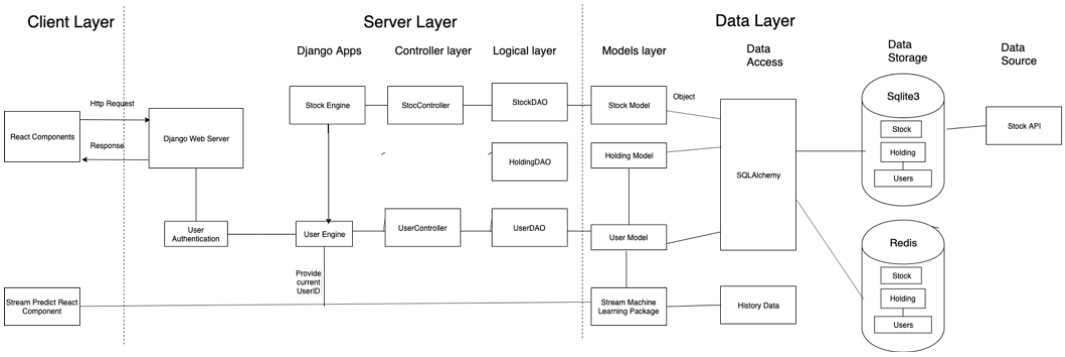


Figure 4.1: *Backend system framework*

As you can see, we designed our project into three layers.

- The first layer is the client layer. It is responsible for interacting with users. We choose React as our front-end framework. React is a popular JavaScript library which is used for building user interface. It is an efficient and flexible tools for development. Http requests will be sent to the back-end by the React.
- The next layer is the server layer which is used to handle http requests, retrieve data from database and finally return results to the front-end where data are displayed appropriately. Django is our back-end framework. It is a powerful framework that used to build Web APIs which help us development in a standard way. There are two main Apps in our project. The first one is the user engine that used for user registration, user authentication and user login. The another Apps is stock engine which enables users to purchase stocks and sell stocks. We also divided the server layer into three sub-layers.

  - The first sub-layer is the controller layer which is used to interact with the front-end. It will use different DAO functions to get data according to the http url and the http verb that are contained in the http requests. And then returning data back to the front-end
  - The next layer is logical layer which is an important layer as well. It is associated with the data access, data storage and data updates. In other word, how we manipulate the database. In this layer, SQLAlchemy is our object relational mapper which help us access database efficiently.
  - There are 5 models in this layer which are used to store stock information. Each time when users purchase or sell stocks, transaction details, like when does the transaction happens or how many shares does the user purchase. will be record by models.

- Data layer: SQLite is our database tool which is a lightweight database and always apply on small data size. Besides, because there are existing unrelational data structure in our project, we decided to use Redis to cache these kind of data. Redis is a key-value database which means it can store many kinds of data structures, like string, list, and set. The key benefit of Redis is that it stores all data in memory. So the speed of accessing Redis is quicker than accessing database. Thats the reason why we choose to use Redis as a middleware. Stock APIs are the data source of our project. We downloaded all stocks price information as csv files. Then after data cleaning, data transformation, data are inserted into its corresponding table in the database.

4.2 Database design

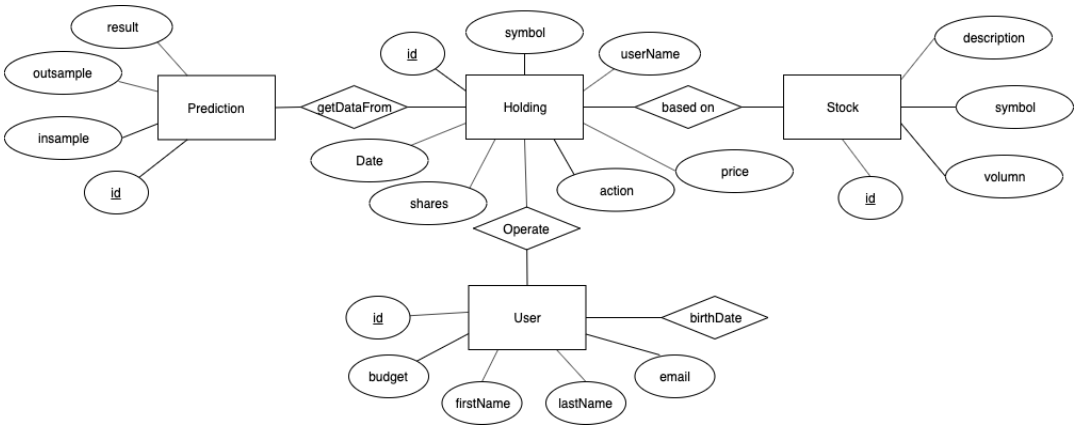


Figure 4.2: *Database system framework*

For database architecture part, it includes four part, which are user model, holding model, stock model and prediction model.

For user model, after user login, it will get the authentication user id from the authentication token as foreign key, then the user could create his profile, the attributes include: budget, first name, last name, email and birthdate, and the user id is automatically generated.

For holding model, it connected the user model and stock model. The data that user operate on the stock will be stored in holding model as history records which also be provided to prediction model to prediction useful information after being analyzed by strong machine learning models for users. In this way, in holding model, it not record the history of users operation, it also has the foreign key of user and stock.

For stock model, it is the place where stock data stored. It includes the stock id, stock volume and description.

For prediction model, it stored the prediction result from fully analyzing the history data of holding model record, which supply the data for machine learning results display.

4.3 Backend feature description

4.3.1 User engine

User engine is an important part of the architecture, it ensures the security of the system and is also the foundation stone of the whole project, which includes two parts: authentication and user profile.

Authentication

The authentication part of user engine fulfilled these following functions, which ensure the basic requirement of user service for whole project. Authentication part includes five functions: retrieve information, register, login, change password and logout. The authentication flow chart is the following:

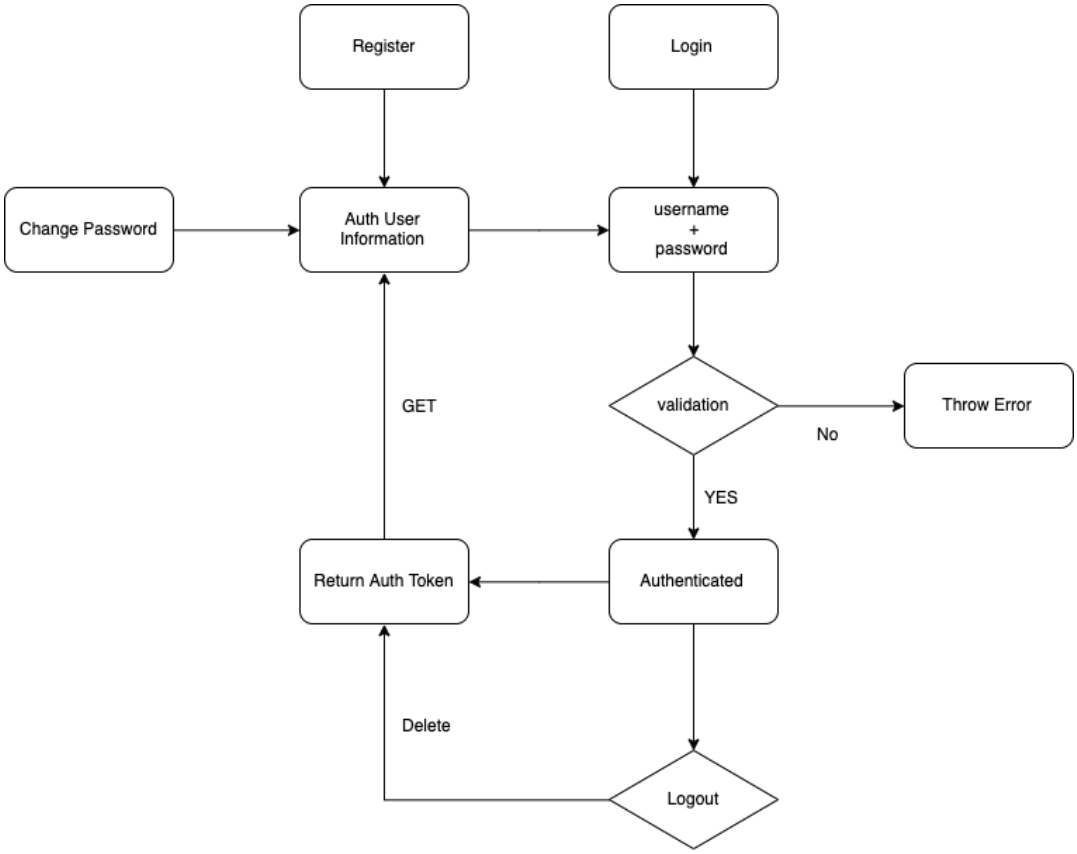


Figure 4.3: *Authentication Flow Chart*

- Register: User could register his information, input its username, password, then an account is created, and the auth-user information will be returned as well as a register authentication token.
- Login: After registering finished, user could use his account to login, after successfully login, auth-user information and login authentication token will be returned.
- Retrieve information: If user has successfully registered, then user could retrieve its information by sending the get request with an either register or login token, the information include users account and email (only authentication profile which is for login).
- Change password: The system offered the service for user to change password after login successfully, if the validation of the old password is validated, then the new password will be granted.
- Log out: Each time user login, it will get a specific token, if expiration is not set, then user could login to the system with token for multiple times. In this way, log out function could delete this token in database, then this token become invalid, so user need to login again to grantee his later operation in system.

Implementation challenges

Authentication always runs at the very beginning of the view, before the permission and restriction checks occur, and before any other code is allowed to continue. It is the first step of the whole project, so guarantee the security of the system is significant, only in this way, users could user other functions of system safely. Our challenge is to find a practical and efficient authentication way to achieve this goal, so lots of research is done.

Authentication is the mechanism by which an incoming request is associated with a set of identifying credentials, such as the requesting user or its signed token. The permissions and restrictions policies can then use these credentials to determine if the request should be allowed.

The REST framework provides some out-of-the-box authentication schemes and also allows to implement custom scenarios. TokenAuthentication is one way of the authentication which offered by the rest framework of Django. This authentication scheme uses a simple token-based HTTP authentication scheme. Token authentication applies to client-server settings such as local desktop and mobile clients. Similar to the TokenAuthentication built in to DRF, Knox authentication is found by us, it is token based, however, it overcomes some problems present in the default implementation. Knox provides easy-to-use authentication for the Django REST framework, allowing for a common pattern in REST-based applications without additional effort and ensuring that connections remain secure. The RestFramework token of Django is limited to one per user, which makes it difficult to securely log in from multiple devices since the token is shared. If a server-side logout is required, the DRF token also requires all devices to be logged off (i.e. the token has been deleted). Each time it is called to the login view, Knox provides a token to allow each client to have its own token, which will be deleted on the server side when the client logs out. Knox also provides a way for the logged-in user to remove all tokens from the server, forcing all clients to re-authenticate.

In contrast, DRF tokens are stored unencrypted in the database, and in case the database were compromised, the intruder can get into the account with the token without restriction. However, knox tokens are stored only in encrypted form, and even if someone stolen the database in some method, it will not be able to use the stolen credentials to log in. What is more, knox tokens are able to be configured with expiration time in the application settings (default is 10 hours), while DRF tokens can track their creation time, but there is no built-in token expiration mechanism. So, after fully research, we decide to use knox token authentication as our authentication method.

4.3.2 User profile

The user-profile part of user engine fulfilled these following functions, which ensure the user to manage its profile in order to provide user information for other model engine. User-profile part includes five functions: create information, update information and retrieve information.

- Create user profile: After user login, user could use the login token to create his profile, input its first-name, last-name, email, budget, birthdate, then the profile is registered, and the user profile information will be returned.
- Update user profile: After user login, user could use the login token to create his profile, input its first-name, last-name, email, budget, birthdate, then the profile is registered,

and the user profile information will be returned.

- Retrieve user information: If user has successfully registered his profile, then user could retrieve its information by sending the get request to the user profile path.

Implementation challenges

After login, system need to know which user is it, and if user operate on the user-profile, system need to handle the specific data for this user, this needs to connect the user-profile model to user model. So, it will be a challenge to manage this. First, the foreign key is added to ensure the user-profile model has the id of auth-user. First, the foreign key is added to ensure the user-profile model has the id of auth-user. In addition, when user update profile, system will only fetch the record whose owner id equals his authentication model id, in this way, the problem which user handle his profile may operate on other users is solved. In addition, when user update profile, system will only fetch the record whose owner id equals his authentication model id, in this way, the problem which user handle his profile may operate on other users is solved.

4.3.3 Stock engine

Stock detail information

Stock detail information is a necessary and important feature in the stock portfolio management service. Except showing the boring data, we also display a vivid chart in a stock detail page. As shown in the chart, there are different technical lines in the graph that enable users to see the historical price floatation in daily/ monthly/ yearly view. When users click a stock, which want to have a full understanding of its price flotation trending, database will search that stock according to its name that is sent to back-end by front-end. After sorting the result by date, the stocks detail information would be displayed on the front page where data are drawn into graphs.

Stock transaction

Stock transaction is one of the main function in our project. It stimulates the real stock transaction process that enables users to purchase or sell stocks. The process of stock transaction has two steps. At first, each time when a user want to purchase a stock, he has to adding that stock into his holding lists firstly and this will create a holding model in the database which contains users private information. Then user needs to input the amount of shares that he want to purchase for this stock. At last all transaction details, like when does the transaction happen, and the current price of that stock, will be automatically added to the corresponding holding model. For those stocks that are currently in the holding list, users have two options to update it. Users can either choose to sell stocks or purchase more shares for stocks.

Stock search

Stock transaction is one of the main function in our project. It stimulates the real stock transaction process that enables users to purchase or sell stocks. The process of stock trans-

action has two steps. At first, each time when a user want to purchase a stock, he has to adding that stock into his holding lists firstly and this will create a holding model in the database which contains users private information. Then user needs to input the amount of shares that he want to purchase for this stock. At last all transaction details, like when does the transaction happen, and the current price of that stock, will be automatically added to the corresponding holding model. For those stocks that are currently in the holding list, users have two options to update it. Users can either choose to sell stocks or purchase more shares for stocks.

Stock holding list

Sometimes users have the intent to purchase some stocks, but they could not make up their mind to do that. Instead, they hope there is a way for them to keep tracking of these stocks information. In order to solve this problem, we use Redis to build the stock watching list. Users could add any stock to the watching list where they could only see interested stocks information or remove it. We use the "zset" data structure in Redis to build the watching list because this kind of data structure can be automatically sorted by the score of value. So we set the current "datetime" as the score for each value when users add stocks into watching list. Although there is one drawback for using Redis, which is that Redis is not as reliable as SQLite. But it is tolerable to lost one stock or more in the watching list. Besides, Redis is an in-memory database which means accessing Redis is faster than accessing SQLite. According to these reasons, we think Redis is a more appropriate database for building the stock watching list.

Financial News

Sometimes users have the intent to purchase some stocks, but they could not make up their mind to do that. Instead, they hope there is a way for them to keep tracking of these stocks information. In order to solve this problem, we use Redis to build the stock watching list. Users could add any stock to the watching list where they could only see interested stocks information or remove it. We use the zset data structure in Redis to build the watching list because this kind of data structure can be automatically sorted by the score of value. So we set the current datetime as the score for each value when users add stocks into watching list. Although there is one drawback for using Redis, which is that Redis is not as reliable as SQLite. But it is tolerable to lost one stock or more in the watching list. Besides, Redis is an in-memory database which means accessing Redis is faster than accessing SQLite. According to these reasons, we think Redis is a more appropriate database for building the stock watching list.

Implementation challenge

The first challenge is the choices of development tools. There are two backend framework and several database that are popular in development. While building the back-end system, Studying Django is one of the main challenge I have met. The reason why we choose Django as our back-end framework is because it is a powerful framework for building APIs. Although Flask is also a tool for building backend, it is too flexible that needs to import a lot of packages. In order to ensure the quality of the project, we could not just know how to use

these packages because when we meets problems relating to that, we may not know where to find help. While for Django, It can provide a more standard way to build backend than Flask. However, I have not used Django before, I need to watch many tutorials and videos to learn how to use it. Another challenge is the implementation of Redis. Because I have read a book that talk about Redis, I have a little understanding of it. I know in some cases it is a better choice for data storage than SQLite because accessing Redis is faster than accessing SQLite. So learning how to use Redis in our project and choose which kind of data structure in Redis is a higher challenge for me as well. The biggest challenge in our project is the framework design. Because our team members do not have previous experiences that worked in a groups of four members to build a big web project, we met many problems, like, how to design functionalities, how to distribute tasks fairly, and how to solve merge problems when using git. Another problem is that because we did not design our database tables at first, we changed models several times, which could be avoid. Although these problems took us some time to overtake, we learned many valuable teamwork experiences which could help us work better in future job.

4.3.4 Stock price prediction

Prepossessing

For Linear Regression model and KNN model, we use the same prepossessing, the y of these two models should difintely be every days' close prices. And for x, we should create some features for x. We create 12 features for x, they are "year, month, week, day, day_of_week, day_of_year, is_month_start, is_month_end, is_quarter_start, is_quarter_end, is_year_start and is_year_end". We use fastai's add_datepart function to do this kind of prepossessing.

Year	Month	Week	Day	Dayofwee	Dayofyear	Is_month_end	Is_month_start	Is_quarter_end	Is_quarter_start	Is_year_end
2015	11	46	12	3	316	FALSE	FALSE	FALSE	FALSE	FALSE
2015	11	46	13	4	317	FALSE	FALSE	FALSE	FALSE	FALSE
2015	11	47	16	0	320	FALSE	FALSE	FALSE	FALSE	FALSE
2015	11	47	17	1	321	FALSE	FALSE	FALSE	FALSE	FALSE
2015	11	47	18	2	322	FALSE	FALSE	FALSE	FALSE	FALSE
.....										
2018	10	44	29	0	302	FALSE	FALSE	FALSE	FALSE	FALSE
2018	10	44	30	1	303	FALSE	FALSE	FALSE	FALSE	FALSE
2018	10	44	31	2	304	TRUE	FALSE	FALSE	FALSE	FALSE
2018	11	44	1	3	305	FALSE	TRUE	FALSE	FALSE	FALSE
2018	11	44	2	4	306	FALSE	FALSE	FALSE	FALSE	FALSE

Figure 4.4: *Data prepossessing*

Above is the data after prepossessing, for example, see the last third line, it is Oct 31st, and it is in week 44, its the second day of week, the 304th day of the year, and it is the month end(the last day of this month), it is not the month start, not the quarter end or start, the year end or start.

Linear regression

The linear regression model attempts to learn the linear model to predict the output marker Y of the real value X as accurately as possible. In this model, the dependent variable Y is continuous, and the independent variable X can be continuous or discrete. In regression

analysis, if only one independent variable and one dependent variable are included, and the relationship between the two can be approximated by a straight line, it is called linear regression analysis; if the regression analysis includes two or more independent variables, and the variable is The independent variable is a linear relationship and is called multiple linear regression analysis. For two-dimensional space, linearity is a straight line. For three-dimensional space, linearity is a plane. For multidimensional space, linearity is a hyperplane.

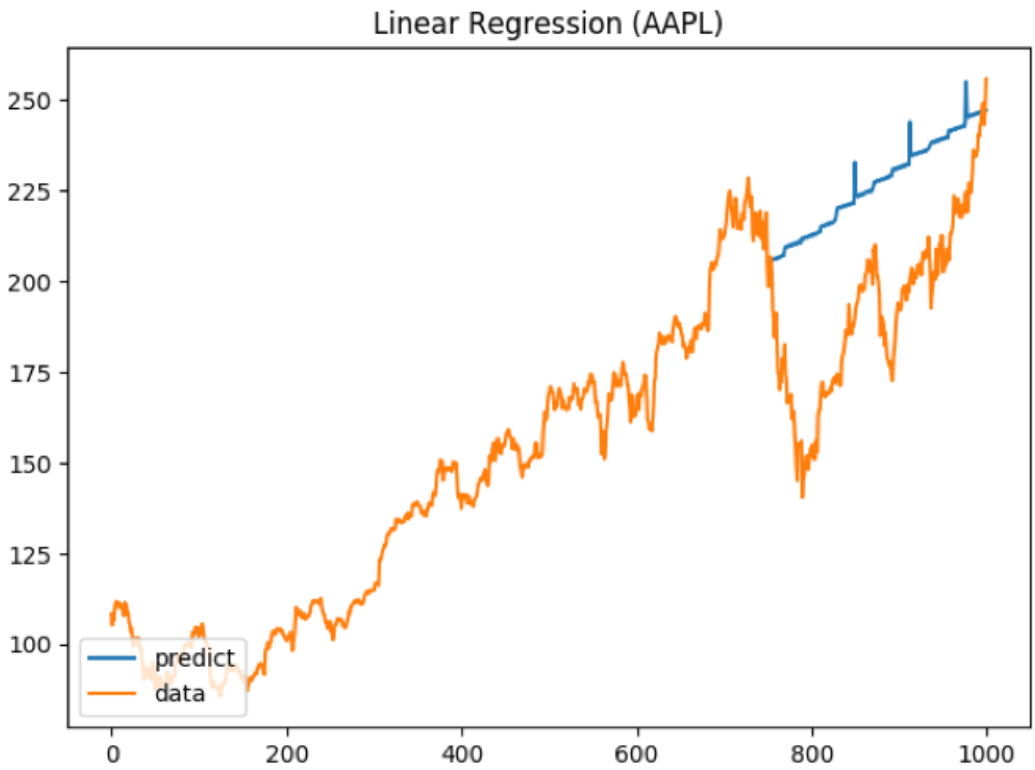


Figure 4.5: *Linear regression*

We used apples stock price data for Linear Regression validation. We used previous 4 year to 1 year data for training and use the trained model to predict previous 1 years data, and compare them with actual data, we can see the linear model predicts the right trend and there is some outliers may be because the training data is not big enough.

KNN

The proximity algorithm, or K-Nearest Neighbour classification algorithm, is one of the simplest methods in data mining classification technology. The so-called K nearest neighbour is the meaning of k nearest neighbours, saying that each sample can be represented by its nearest k neighbours. The core idea of the kNN algorithm is that if the majority of the k most neighbouring samples in a feature space belong to a certain category, the sample also belongs to this category and has the characteristics of the samples on this category. The

method determines the category to which the sample to be classified belongs based on only the category of the nearest one or several samples in determining the classification decision. The kNN method is only relevant to a very small number of adjacent samples in class decision making. Since the kNN method mainly relies on the surrounding limited samples, rather than relying on the discriminant domain method to determine the category, the kNN method is more than the other methods for the crossover or overlapping sample set of the domain.

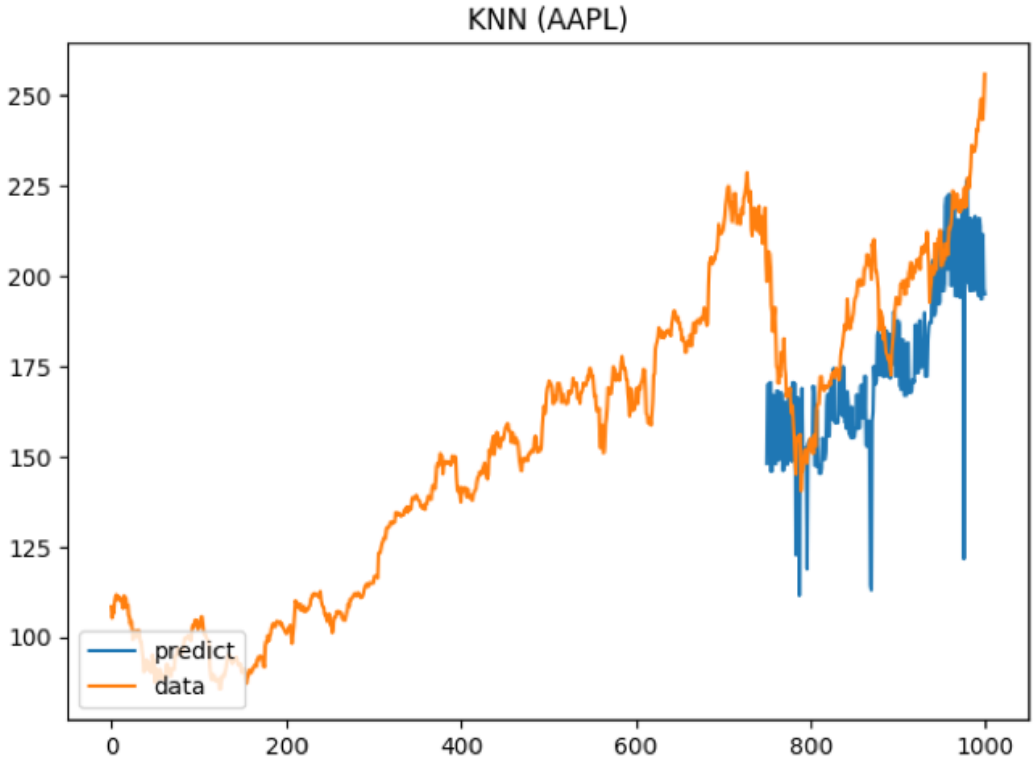


Figure 4.6: *KNN*

Above is the graph of KNN models prediction of apple stock price. We can see the predicted price is very close to the actual price.

Prophet

prophet is an API provided by facebook, it has good performance on time series prediction. The input of prophet is very simple, just date and close price, and it will outputs dates and predict close price, upper bound of predict close price and lower bound of predict close price.

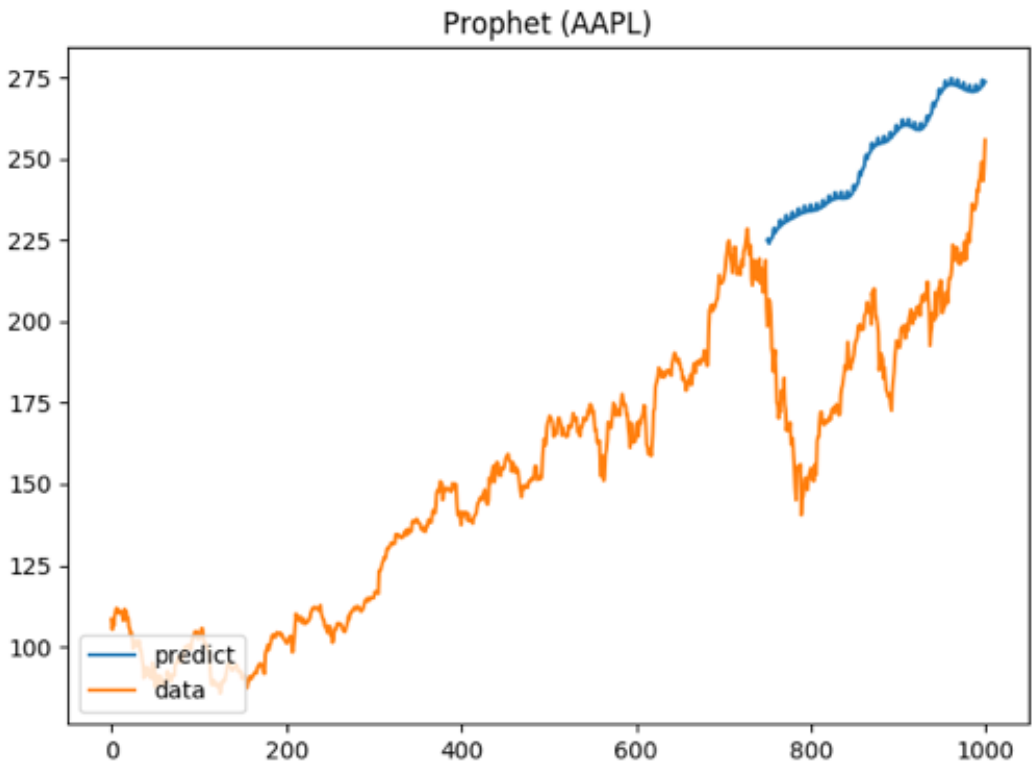


Figure 4.7: *Prophet prediction example*

Model evaluation

The KNN models has the smallest MAE and MSE. Prophets MAE and MSE is slightly higher than KNN. However, in our project, we still choose prophet for price prediction function. This is because, prophet is more reliable, we can see in prophets graph there is no outliers while knns graph has. Moreover, prophet can not only predict close price, but also provide lower bound and upper bound. These can be used in the graph in frontend.

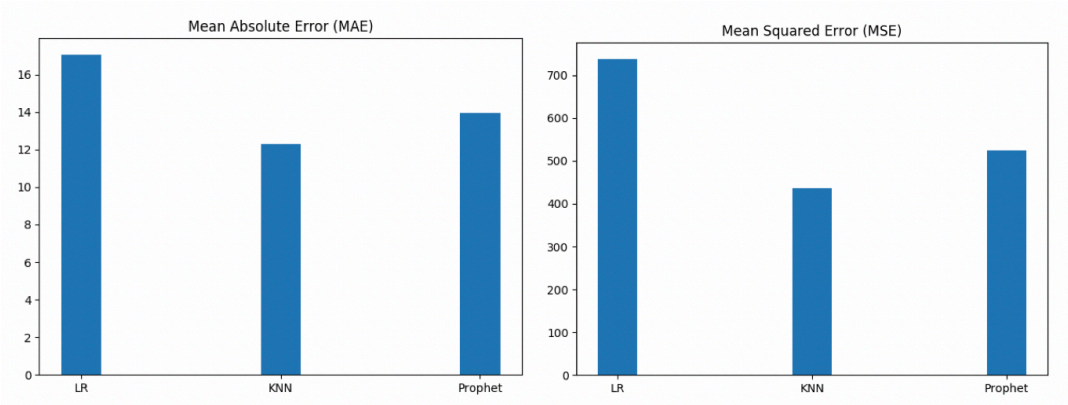


Figure 4.8: *Model evaluation*

Implementation challenge

This function is very challenging. It is very challenge to build machine learning model, because initially the data do not have many features, just data and close price. We wanted to use news as a feature, but it is too complicated. First it is impossible for us to find a news which is coincidentally happened on that date, even if we find them, we need to do emotion analysis to the news, it is also very difficult. After consideration, we give up this idea. We decide to just used date to build model. We built two traditional machine learning model, Linear Regression and K nearest neighbour, and compare them with the facebook Prophet API, to see which model performs better.

4.3.5 Stock Recommendation

Fama-French Three factor model

The Fama and French Three-Factor Model (or the Fama French Model for short) is an asset pricing model developed in 1992 that expands on the capital asset pricing model (CAPM) by adding size risk and value risk factors to the market risk factor in CAPM. This model considers the fact that value and small-cap stocks outperform markets on a regular basis. By including these two additional factors, the model adjusts for this outperforming tendency, which is thought to make it a better tool for evaluating manager performance.

We get SMB and HML values from an official website and do regression base on this formula. The key point of this model is the alpha value, when the alpha value is smaller than zero, this means the market is underestimating this stock, so the stock price may likely to

$$R_{it} - R_{ft} = \alpha_{it} + \beta_1(R_{Mt} - R_{ft}) + \beta_2 SMB_t + \beta_3 HML_t + \epsilon_{it}$$

- R_{it} is the total return of a stock or portfolio, i at time t;
- R_{ft} is the risk free rate of return at time t;
- R_{Mt} is the total market portfolio return at time tl
- R_{it} - R_{ft} is expected excess return;
- R_{Mt} - R_{ft} is the excess return on the market portfolio (index);
- SMB_t is the size premium (small minus big); and
- HML_t is the value premium (high minus low).
- $\beta_{1,2,3}$ refer to the factor coefficients.

Figure 4.9: *Fama-French three factor model*

increase in the future. In our price recommendation function, every week we calculate every stocks alpha value and recommend the stocks with lowest alpha value to users.

Evaluation of the model

To evaluate the effect of this model, we do a simulation based on the previous 1 years data. We choose 170 typical stocks from us stock market. Every 10 trading days, we calculate the alpha value for every stock. We sell all the stock we bought last time, and buy 10 stocks with lowest alpha value this week. We start from 10000 dollar, and after 250 trading days, we reach 13184 dollar while the average of market is just 10078 dollar.

Figure 4.10: *Fama-French three factor model evaluation*

Implementation challenge

This function is very challenging because there is almost no such function on any commercial stock website. The most challenging point is getting the economic data of the listed company. We wanted to try some models and compare their effect, however we the models need various data, and we cannot get these data in an appropriate price, some kinds of data are charged of very high price and we cannot afford it. At last, we just used Fama-French model, this model need SML and HML data, and get these data for free on an official website.

Chapter 5

Third party libraries

We have use a list of third party libraries to build our application in order to achieve higher performance, some of them are:

- django-rest-knox (MIT License): used for user authentication.
- fbprophet (MIT License): used for stock prediction.
- react-stockcharts (MIT License): used for react.js to draw stock detail chart.
- recharts (MIT License): used for react.js to draw stock different type of charts.
- material-table (MIT License): used for react.js to build tables.
- Alpha Vantage: used for stock data API.
- newsapi.org: used for news API.

Since all the third party libraries are MIT open source license, so that we can use them for free. Alpha Vantage is external stock data API which is not free, and we only use the free account for limited use, so that we have to manually download the stock data we need (we have download 170 stocks) and insert them into database for testing and demo. The news api "newsapi.org" is not free and with limitations for free account, but we can use it for testing and demo.

Chapter 6

User Documentation

Since the user documentation includes many pictures, so that we includes them in the Appendix A.

Chapter 7

Test Documentation

Since the test documentation includes many pictures, so that we includes them in the Appendix B.

Bibliography

[1] Django. https://www.djangoproject.com, 2019. [Online; accessed 06-Oct-2019].

[2] React - a javascript library for building user interfaces. https://reactjs.org/, 2019. [Online; accessed 06-Oct-2019].

[3] Ritika Singh and Shashi Srivastava. Stock prediction using deep learning. *Multimedia Tools and Applications 18569*, 76:18, 2017.

[4] Can machine learning techniques be used to predict stock prices? https://www.researchgate.net/post/Can_Machine_Learning_Techniques_Be_Used_To_Predict_Stock_Prices, 2019. [Online; accessed 29-Sep-2019].

[5] Lecture 16: Portfolio management. https://ocw.mit.edu/courses/mathematics/18-s096-topics-in-mathematics-with-applications-in-finance-fall-2013/video-lectures/lecture-16-portfolio-management/, 2019. [Online; accessed 27-Sep-2019].

[6] Jeffrey Donaldson and Marcus Allan Ingram. Applying multi-factor models of stock returns: Student exercises and applications. *Journal of Financial Education 1*, 40(3/4):1–21, 2014.

[7] Christophe Morel. Stock selection using a multi-factor model - empirical evidence from the french stock market. *The European Journal of Finance*, 7(4):312–334, 2001.

[8] Lecture 15: Factor modeling. https://ocw.mit.edu/courses/mathematics/18-s096-topics-in-mathematics-with-applications-in-finance-fall-2013/video-lectures/lecture-15-factor-modeling/, 2019. [Online; accessed 01-Oct-2019].

Appendix A

1. Major system requirements

- MacOS
- Python >= 3.6
- npm
- node
- redis
- Homebrew
- Chrome web browser

2. System setup

Setup database

First you need to download the databases for our project, **make sure the project folder is on the desktop on your mac** because the command I use assumes you have put the project on the desktop.

1. Download `db.sqlite3` from "https://drive.google.com/open?id=1lGICmbDGhoVapLBg-pnXoCA-cYAGcjhH" and `stockDB.zip` from "https://drive.google.com/open?id=1QKZbwWsxqr1iQ9Ht9bQOqqka_lSNp2Nu" in share google drive and then put them in `project/backend/src` folder, unzip `stockDB.zip` into that directory.
2. Download `node_modules.zip` from share google drive "https://drive.google.com/open?id=1NGG4-y6Y4K_Sud-ovLI7nBpDPbcW2SUl" and then put it in `project/frontend` folder, unzip `node_modules.zip` into that directory.
3. Download `stock_ml.zip` from share google drive "https://drive.google.com/open?id=1mgtLtxOHt5YXAriqupscx3J51N7TIew5" and then put it in `project` folder, unzip `stock_ml.zip` into that directory.

Setup backend server

1. First we need to install `homebrew` on mac, open your mac `terminal` and input the following command (since the pdf file cannot display the full length of the command, you can either type each character in the terminal or open readme.md file on the project root folder and copy the text):

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

During installation you need to press `enter` and may need to enter your mac login password.

2. Then we need to setup `redis` cache server. Type the following command:

```
brew install redis
```

3. After that, start `redis` server:

```
redis-server /usr/local/etc/redis.conf
```

When the server is running, **do not close the terminal window!**

4. Then open **another terminal**, you need to use `homebrew` to install `python3` in the terminal:

```
brew install python3
```

After installation, check the python version using the following command:

```
python3 --version
```

which give you

```
Python 3.7.4 or some version >= 3.6
```

Then check whether you are using the correct python package from homebrew (this is required because if you have install anaconda then the default python will be using anaconda rather than homebrew):

```
which python3
```

which should give you

```
/usr/local/bin/python3
```

5. Now you can use `pip3` to install `virtualenv` to setup virtual environment:

```
/usr/local/bin/pip3 install virtualenv
```

6. Then go to `project/backend` folder, and setup virtual environment:

```
cd ~/Desktop/project/backend
/usr/local/bin/virtualenv -p /usr/local/bin/python3 env
```

7. Activate virtual environment:

```
source env/bin/activate
```

Now you can type the following command to check whether you have activated successfully:

```
which pip3
which python3
```

which gives

```
.../project/backend/env/bin/pip3
.../project/backend/env/bin/python3
```

`...` means your upper folder directory.

8. Then you need to install python packages using the following command:

```
pip3 install -r requirements.txt
```

9. After all the packages are installed, you can start the server in the `project/backend/src` folder:

```
cd ~/Desktop/project/backend/src
python3 manage.py runserver
```

When the server is running, **do not close the terminal window!**

Setup frontend server

1. Now we need to open **another terminal window**, and install `node js` and `npm`:

```
brew install node
```

2. Then go to frontend directory and start frontend server:

```
cd ~/Desktop/project/frontend
npm start
```

Now you can see the application is running on `localhost:3000`. Note that the default size of fronts in different computer may vary, in order to have better user experience, we recommend that you can shrink your browser to 90% using the following command: `command` + `-` key on your keyboard.

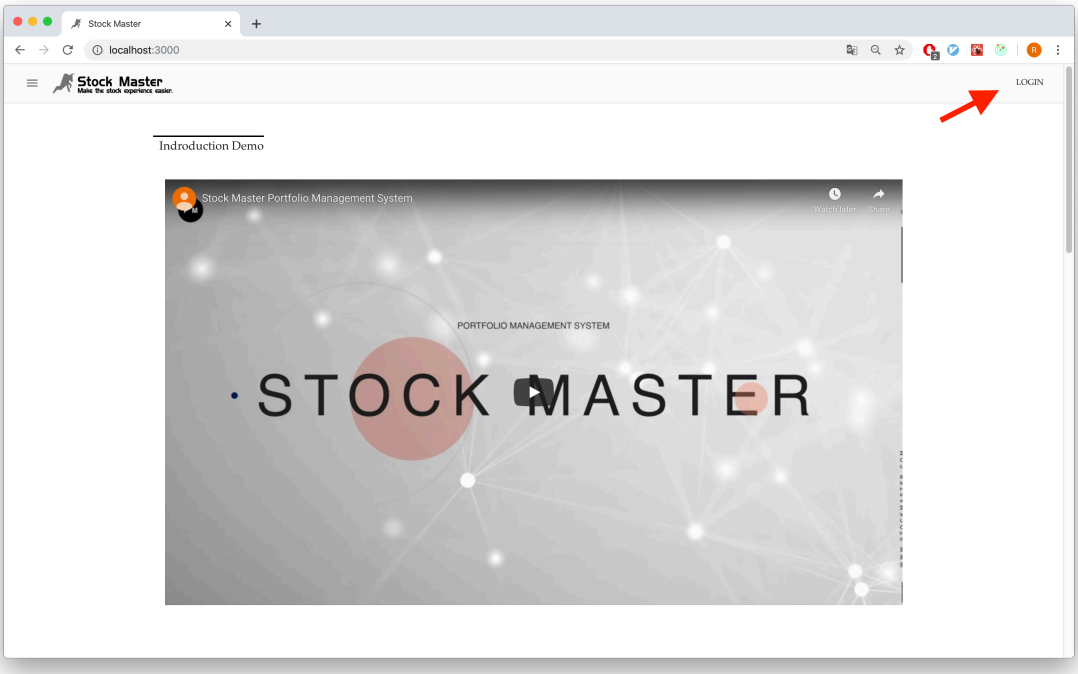
If there are any issues, please email me: `liangrui00@gmail.com` or `rui.liang1@student.unsw.edu.au`, or you can text me or call me using `0451238789`.

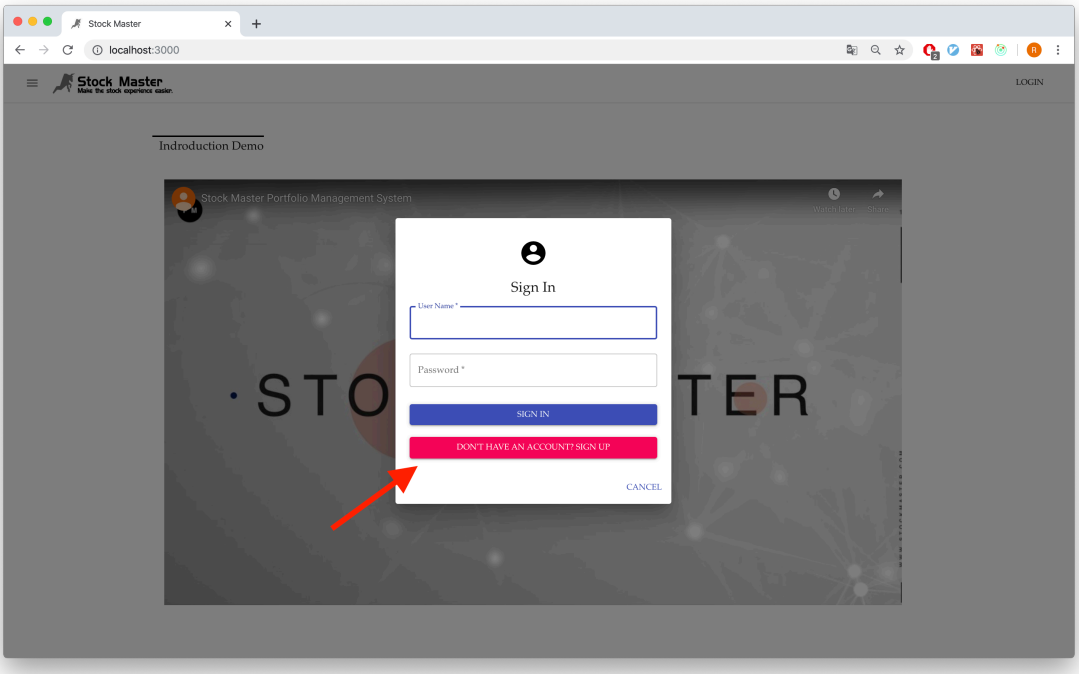
After system setup, you can go to homepage `http://localhost:3000/` and click the video to see the short demo of some features.

3. User administration features

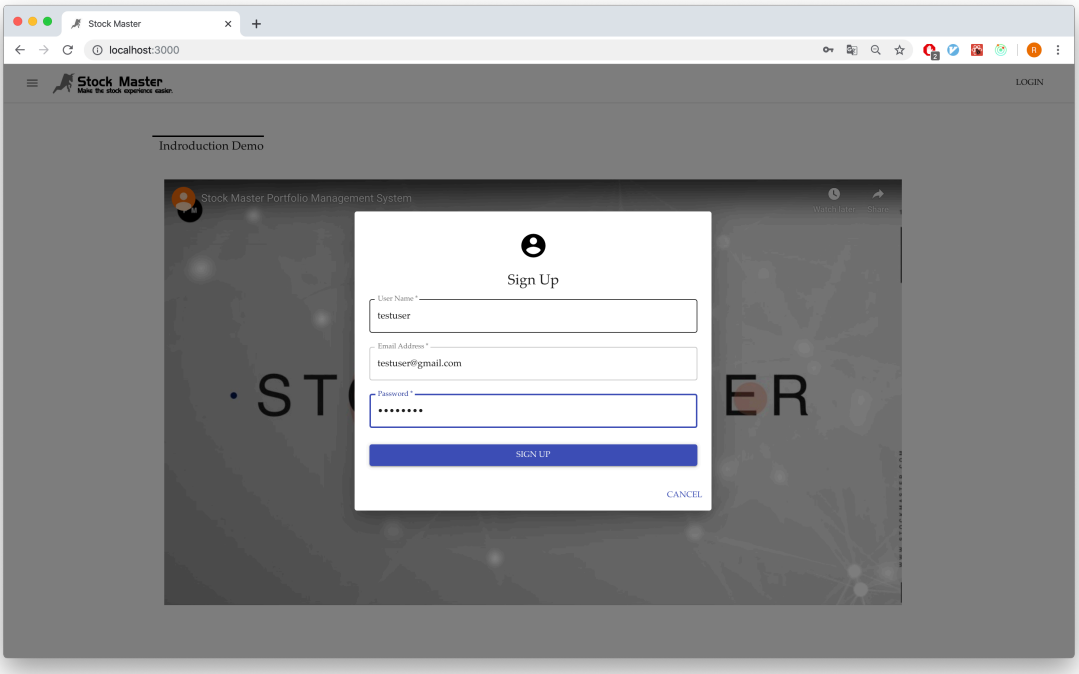
(3.1) Register

In order to register, user need to click `login` button on the upper right corner, then click `DON'T HAVE AN ACCOUNT?SIGN UP`.



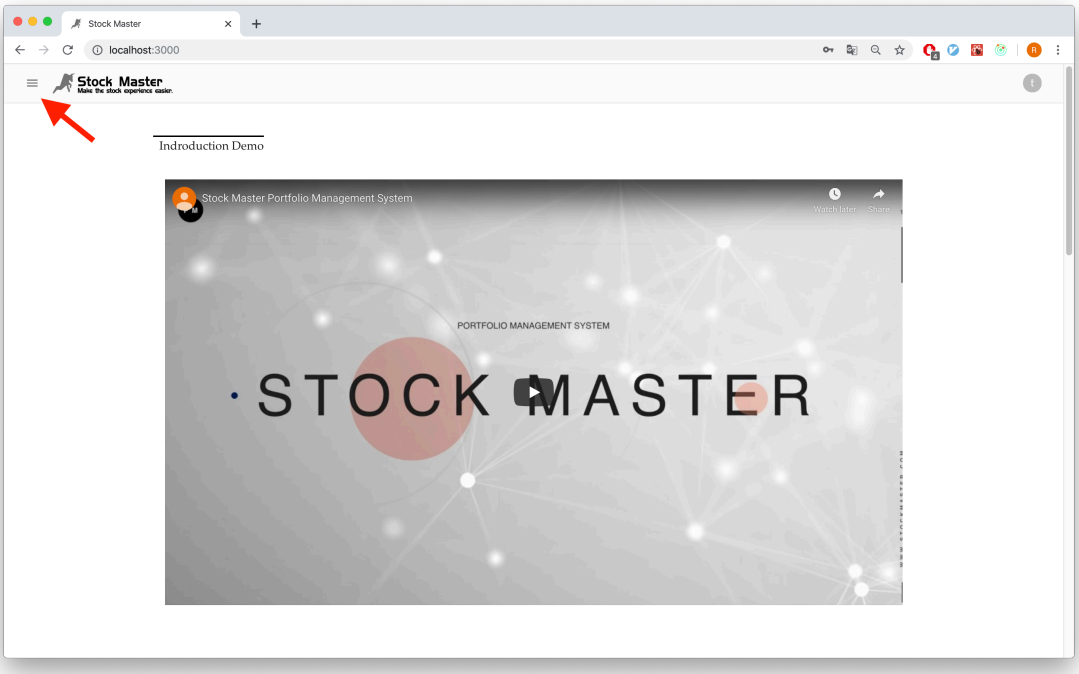


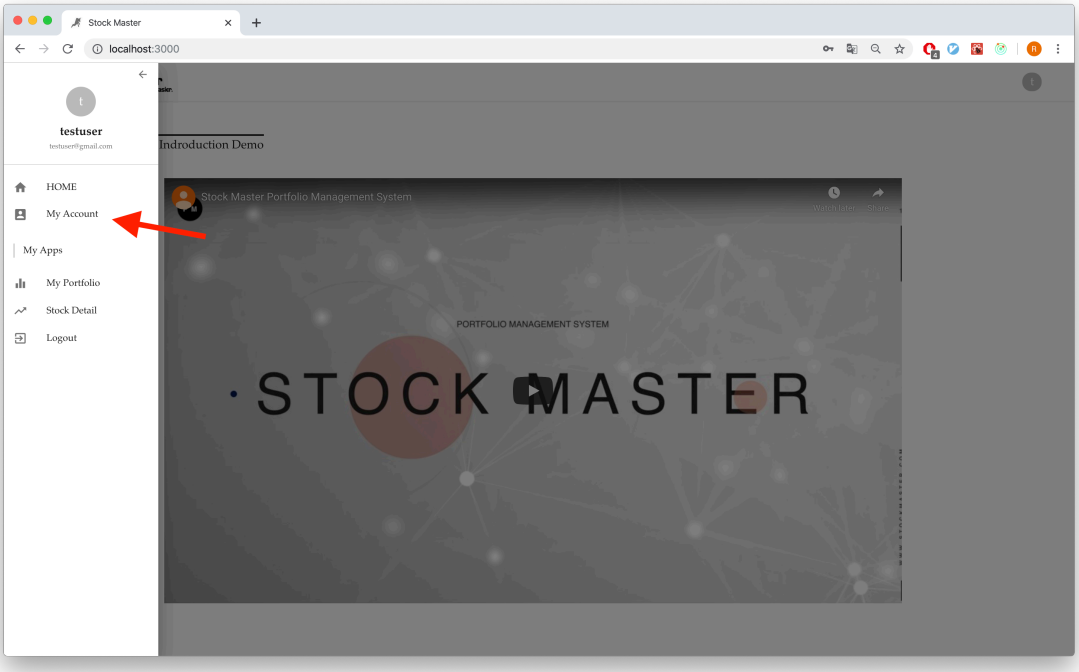
Then for example input the user name as `testuser`, email as `testuser@gmail.com` and password `testuser`:

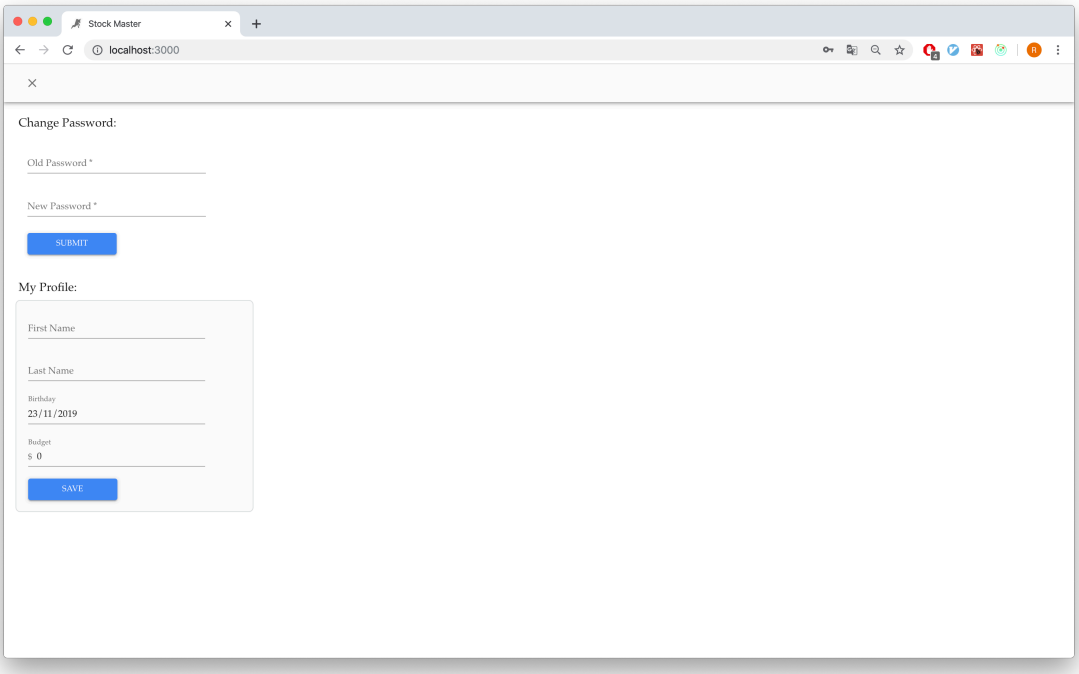


(3.2) User profile

After sign up, system will automatically login. You can see the user profile by expanding the left drawer and select `My Account`:


Stock Master
localhost:3000
Stock Master
Make the stock experience easier.
Indroduction Demo
Stock Master Portfolio Management System
Watch later
Share
PORTFOLIO MANAGEMENT SYSTEM
STOCK MASTER

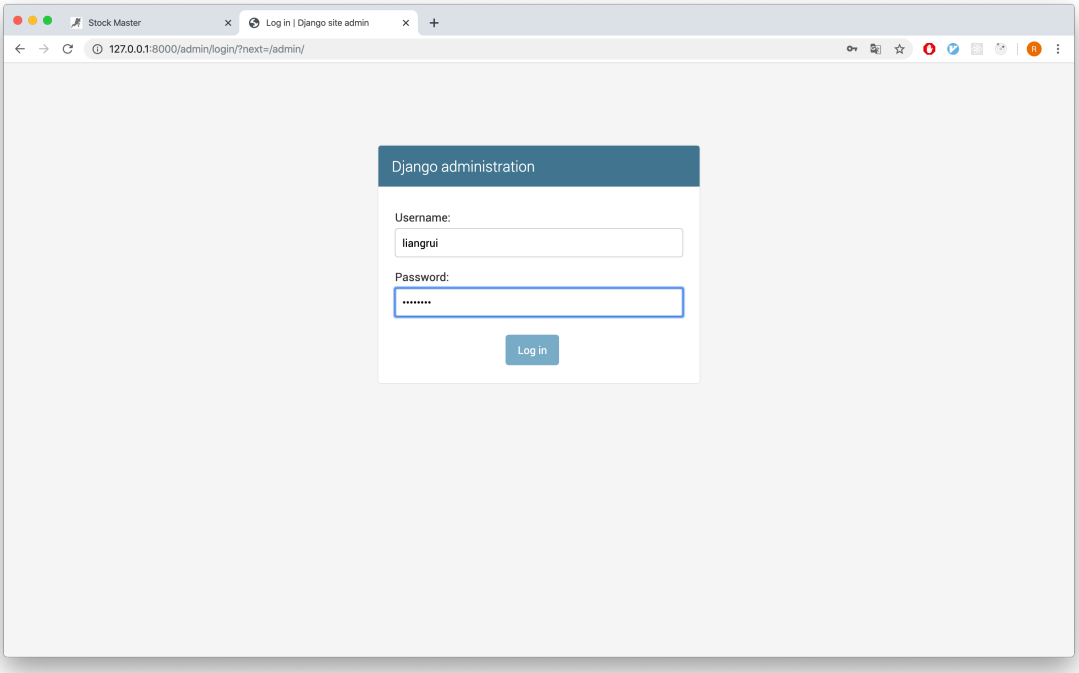

Stock Master
localhost:3000
testuser
testuser@gmail.com
HOME
My Account
My Apps
My Portfolio
Stock Detail
Logout
Indroduction Demo
Stock Master Portfolio Management System
Watch later
Share
PORTFOLIO MANAGEMENT SYSTEM
STOCK MASTER

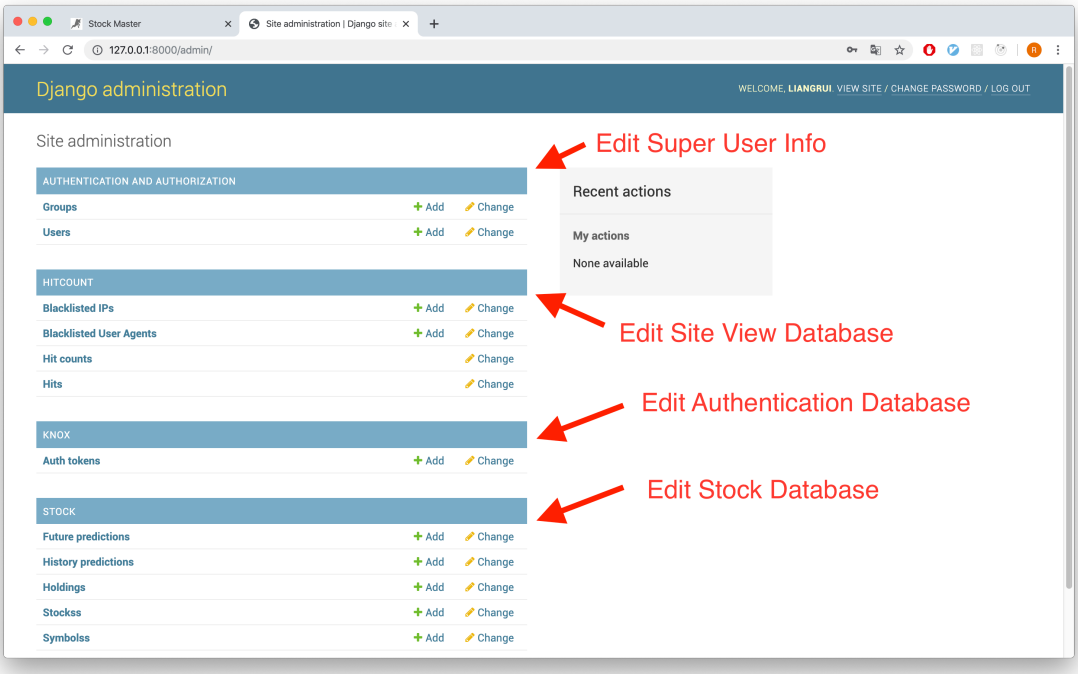


Now you can change the user profile by editing it.

(3.3) Admin page

In order to login to admin page, you need to make sure you have already create a superuser using `django`. Then you can go to `http://127.0.0.1:8000/admin/`. We have already create a superuser for you: user name is `liangrui` and password is `liangrui`.

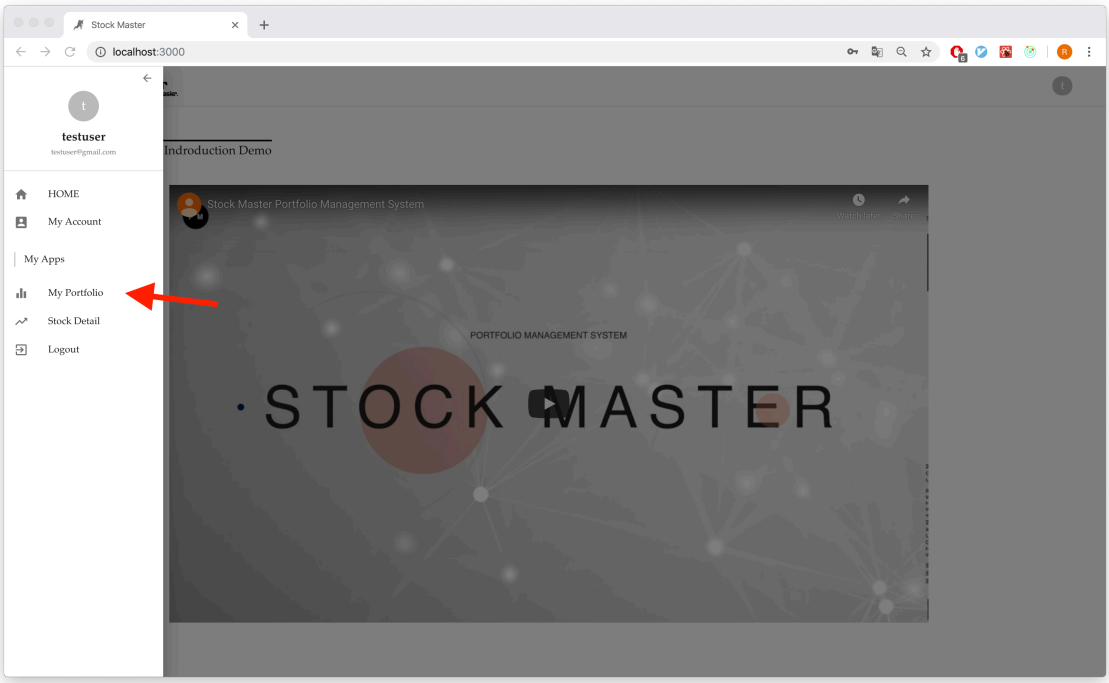


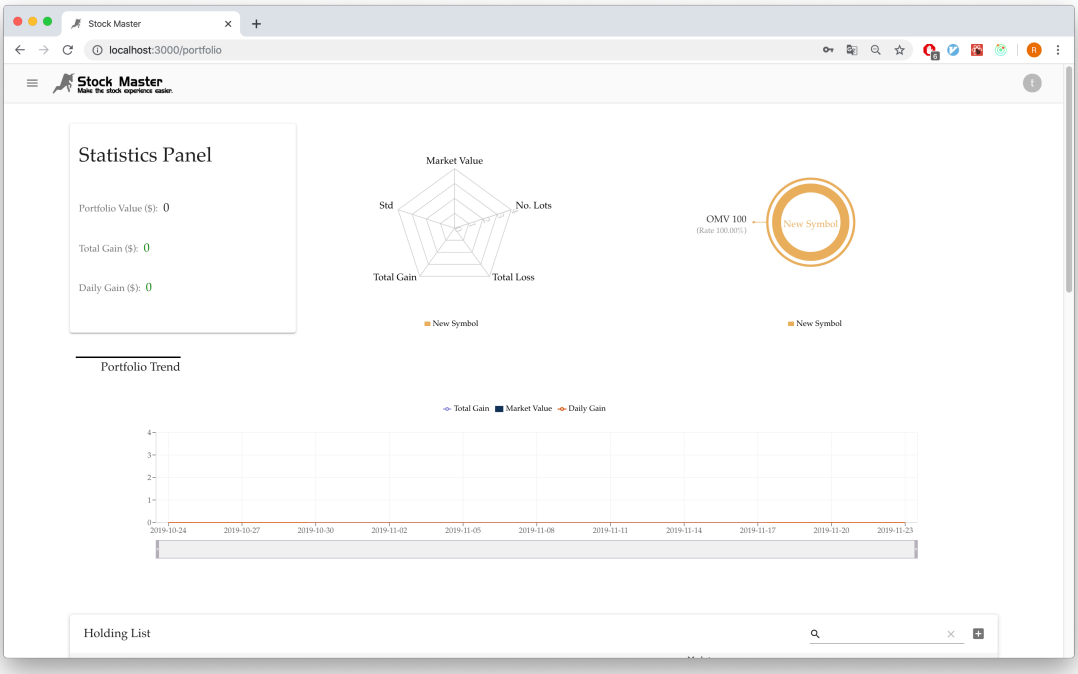


After login to admin page, you can see all the database data and edit them. **Note that editing the database is dangerous! Which will cause inconsistent issues!**

4. Portfolio management console

In order to see the portfolio management console, you need to expand the left side drawer, then select `My Portfolio`, you can see that at the very beginning, there is nothing there.

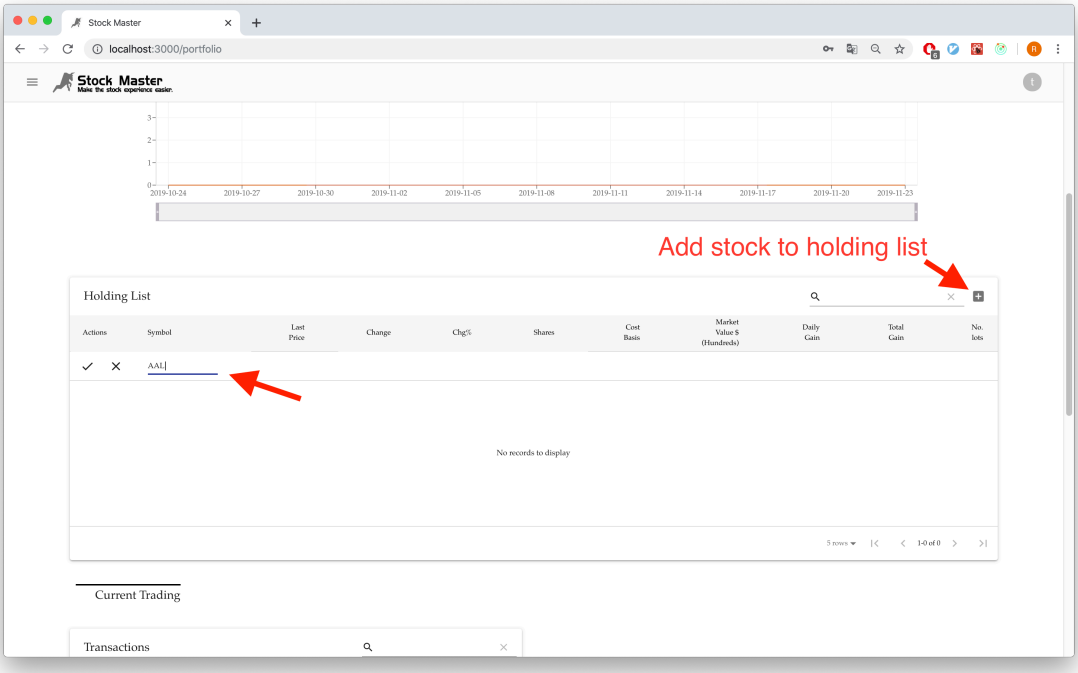




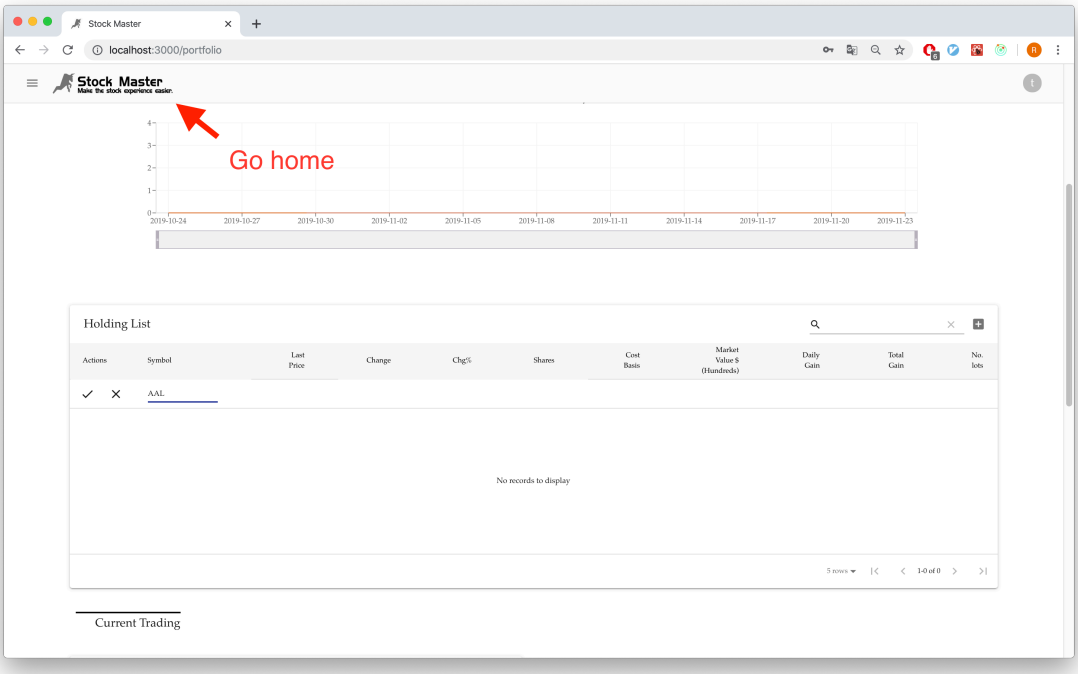
(4.1) Add stock to holding list

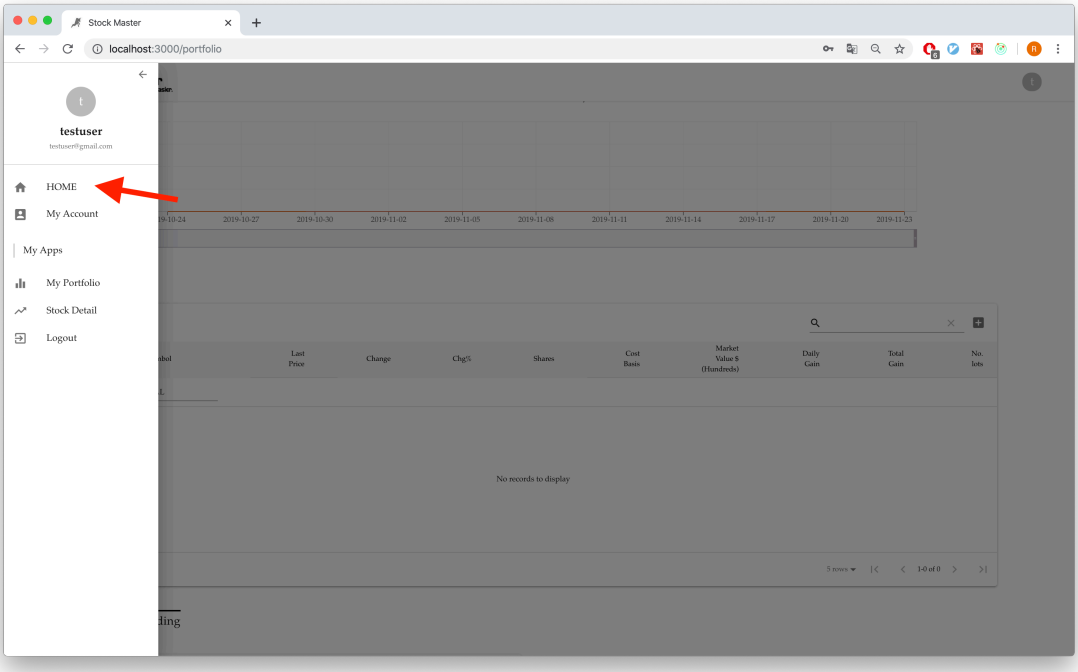
In order to add stock to holding list, there are two methods that you can use:

- Add the symbol in the holding list table, but in this case, you need to make sure the symbol should appear in the database.

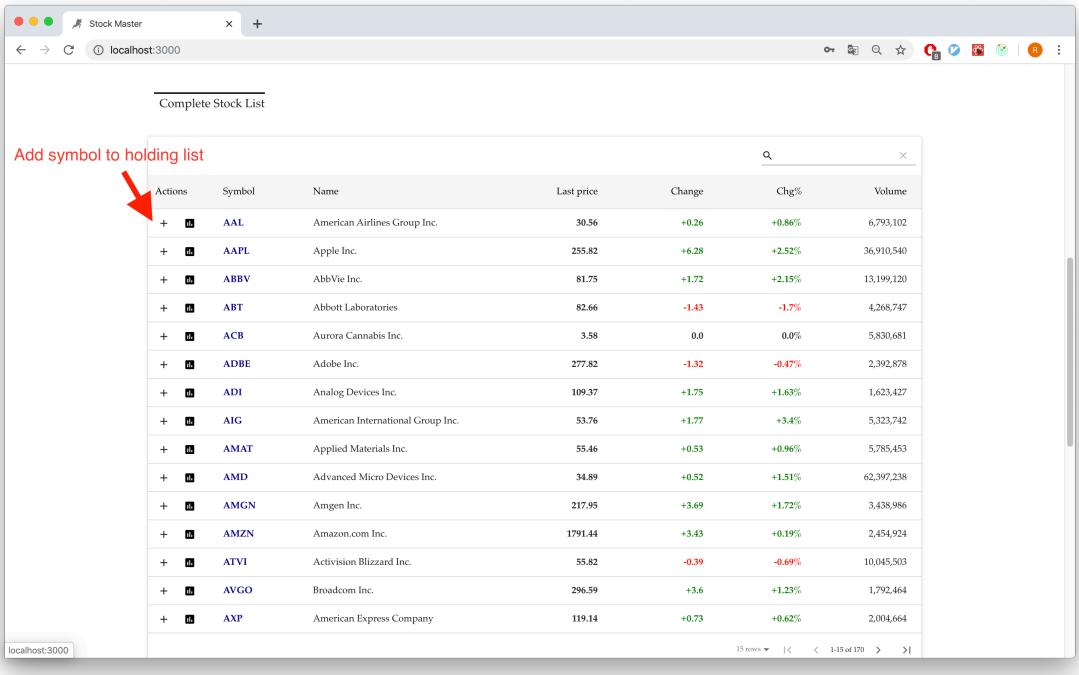


- Add the symbol in the complete stock list (the complete stock list is in the home page, you can go home using either of the following two methods)



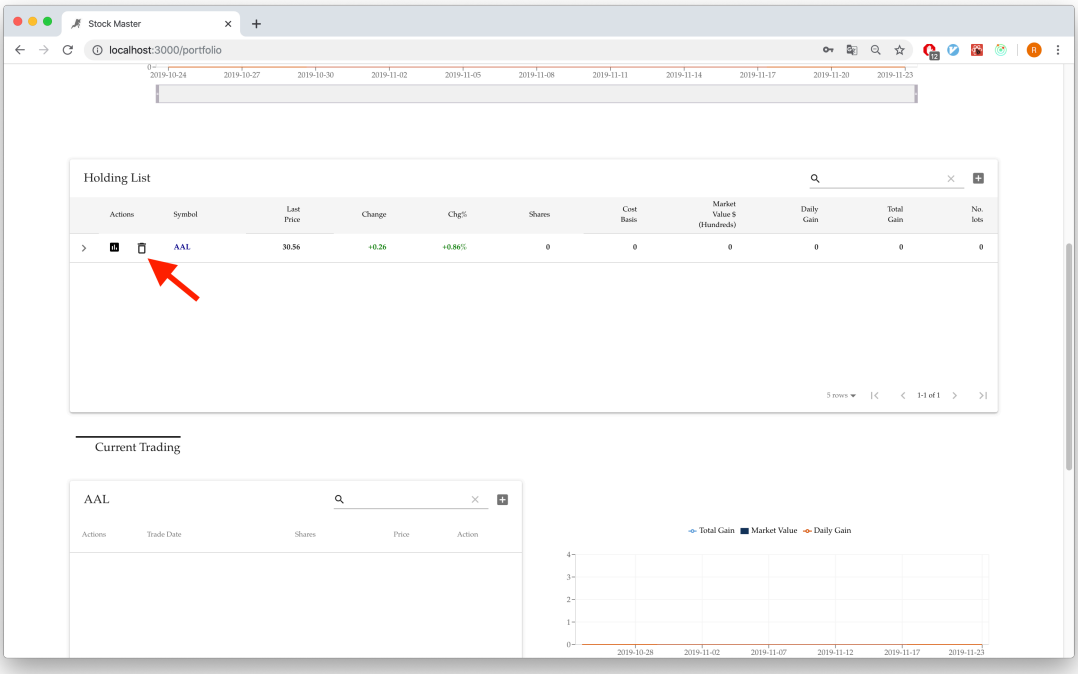


After go home, you can scroll down the list and add the symbol using the `+` sign in front of the symbol.



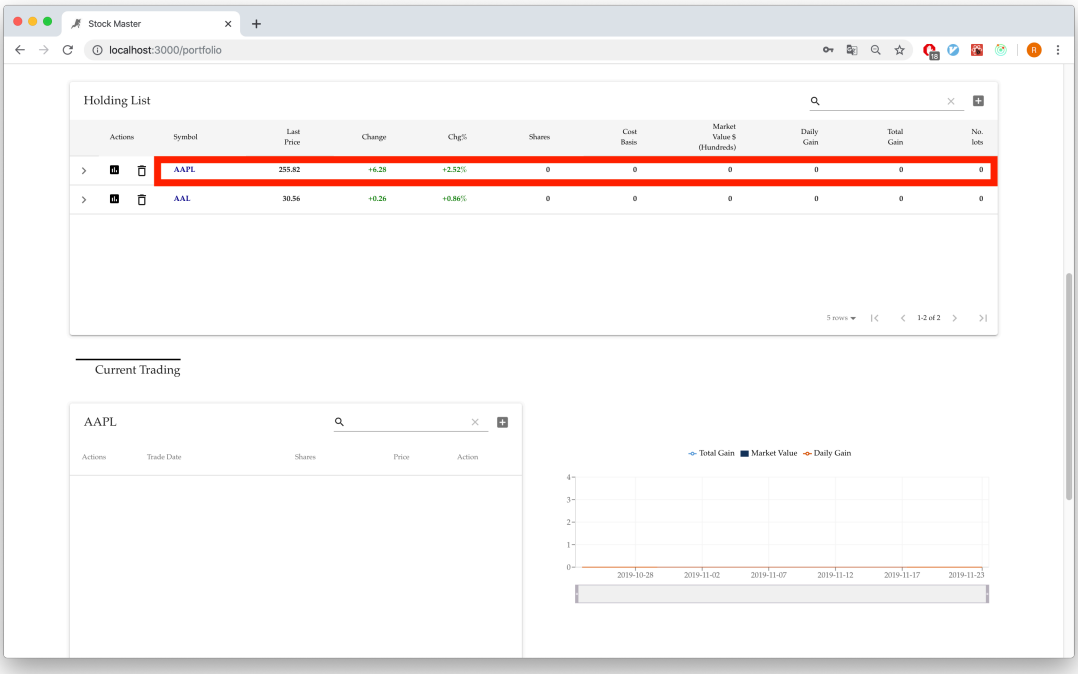
(4.2) Delete a stock in the holding list

Deleting a symbol is straightforward, you can simply click the bin button.

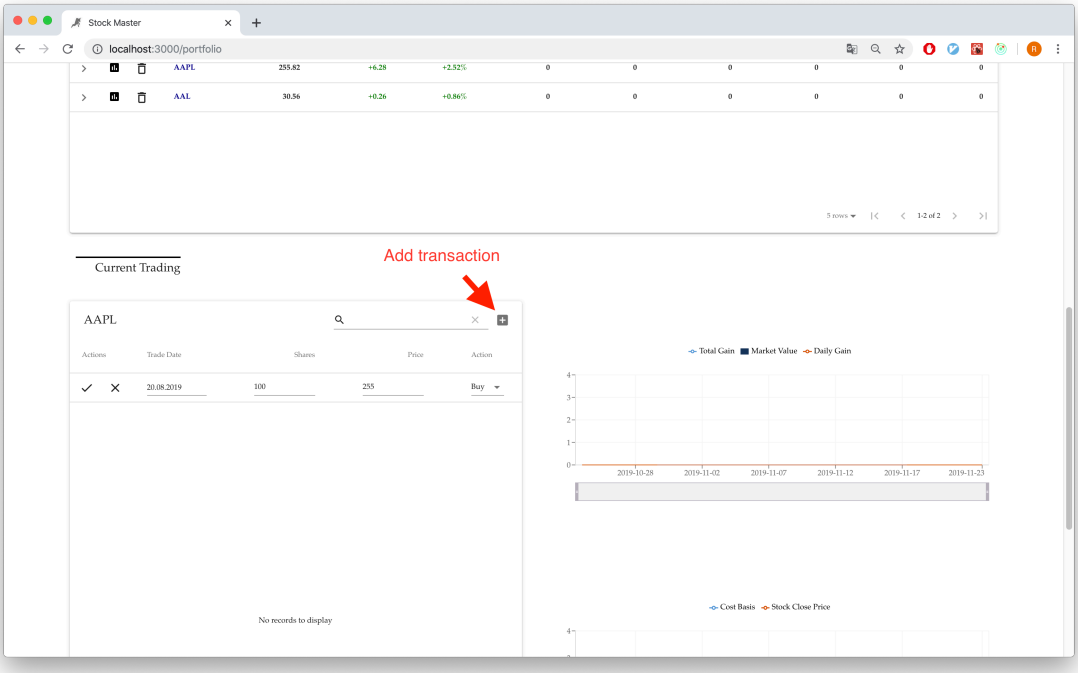


(4.3) Add/delete/edit a transaction

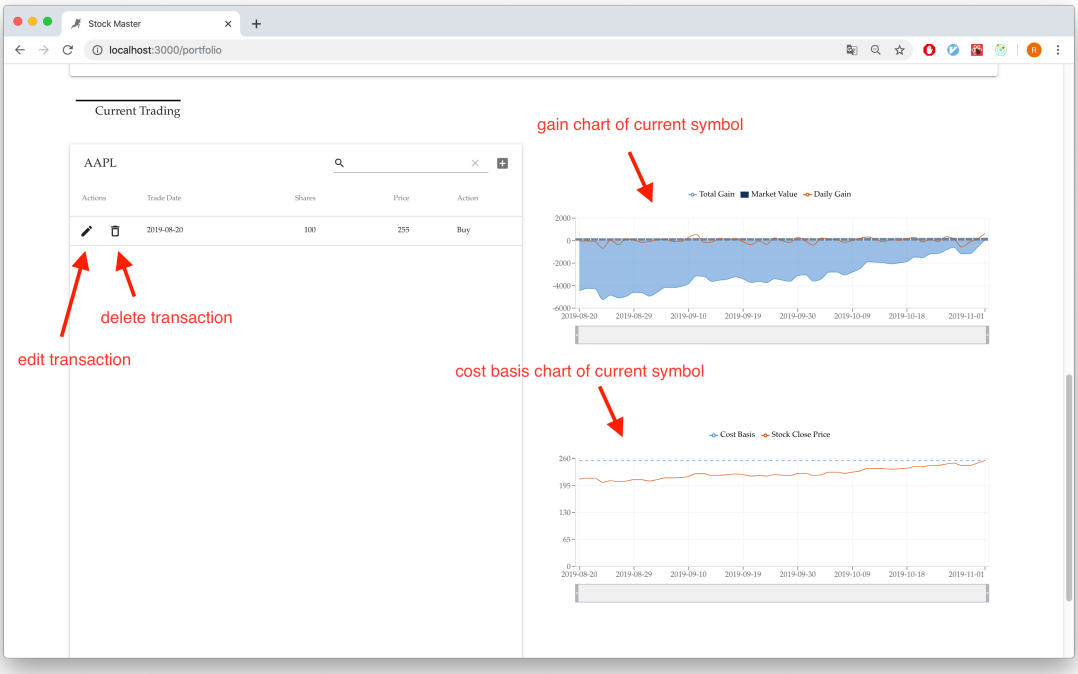
In order to add a transaction, you need to click the row to activate current transaction. For example, if we want to add transaction to symbol `AAPL`, then we need to place the mouse within the following mouse position in the red square area and left click the mouse.



Then the current trading will show the `AAPL` symbol table list which indicates the current trading is activated for `AAPL`. If you want to add a transaction, you need to click the `+` sign and input the data. **Important! Note that our database have only been inserted data before 2019/10/31, so that you can only insert transaction data before that!**



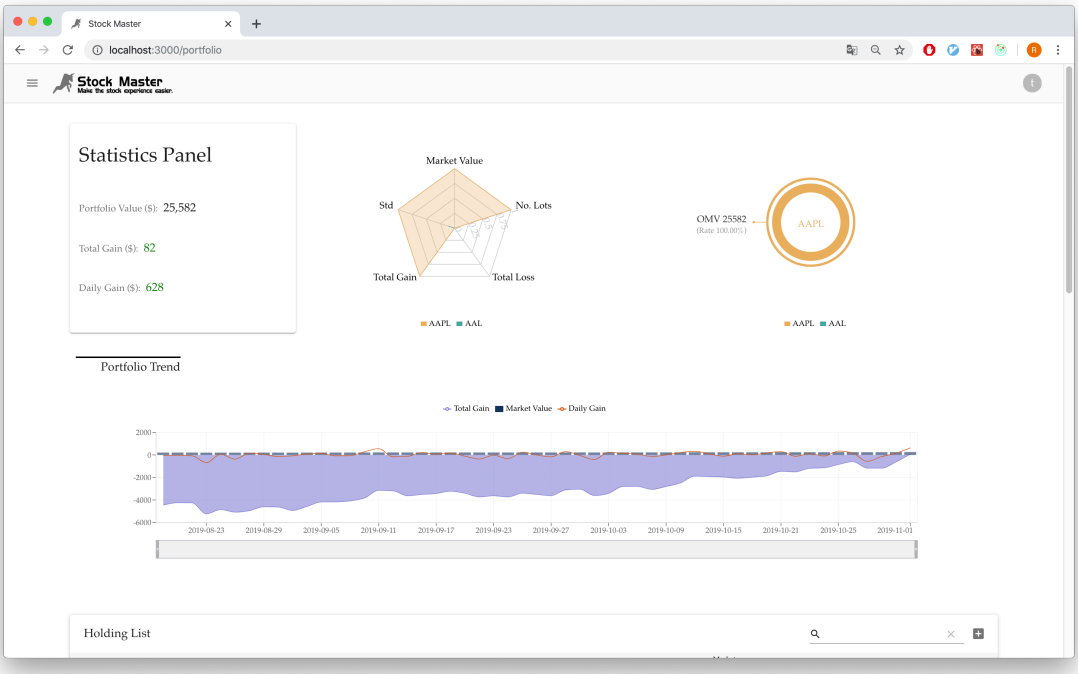
After that, you can see the current stock performance chart on the right hand side. You can also delete and edit the transaction as the figure indicates.



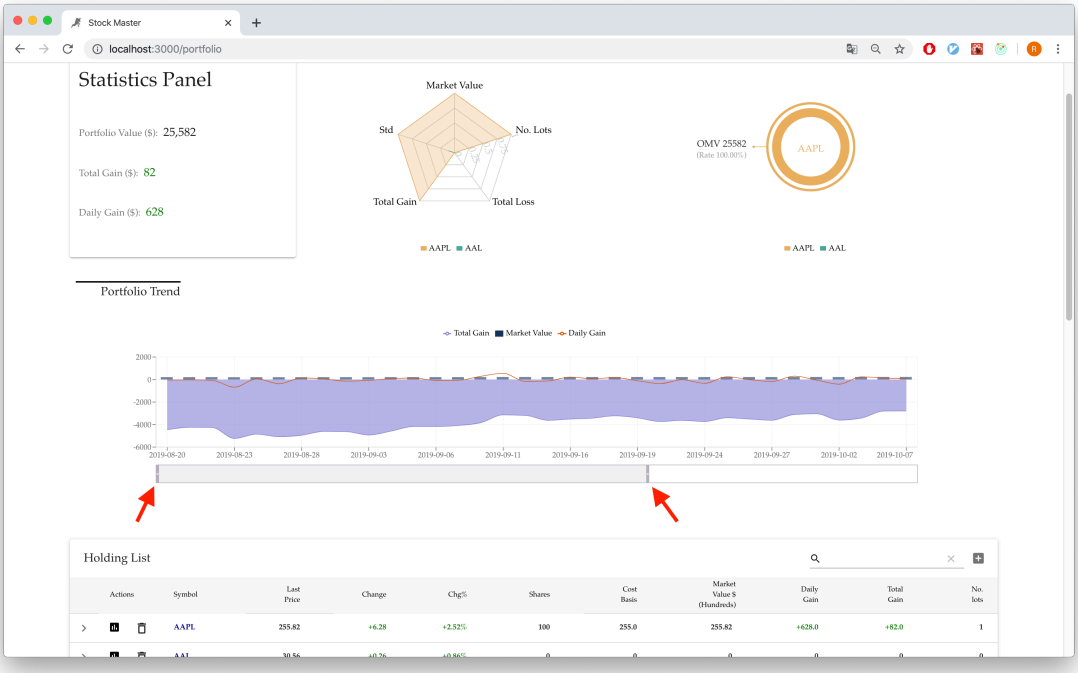
(4.4) Portfolio charts

The portfolio performance is shown using the statistical panel, radar chart, pie chart and composed chart.

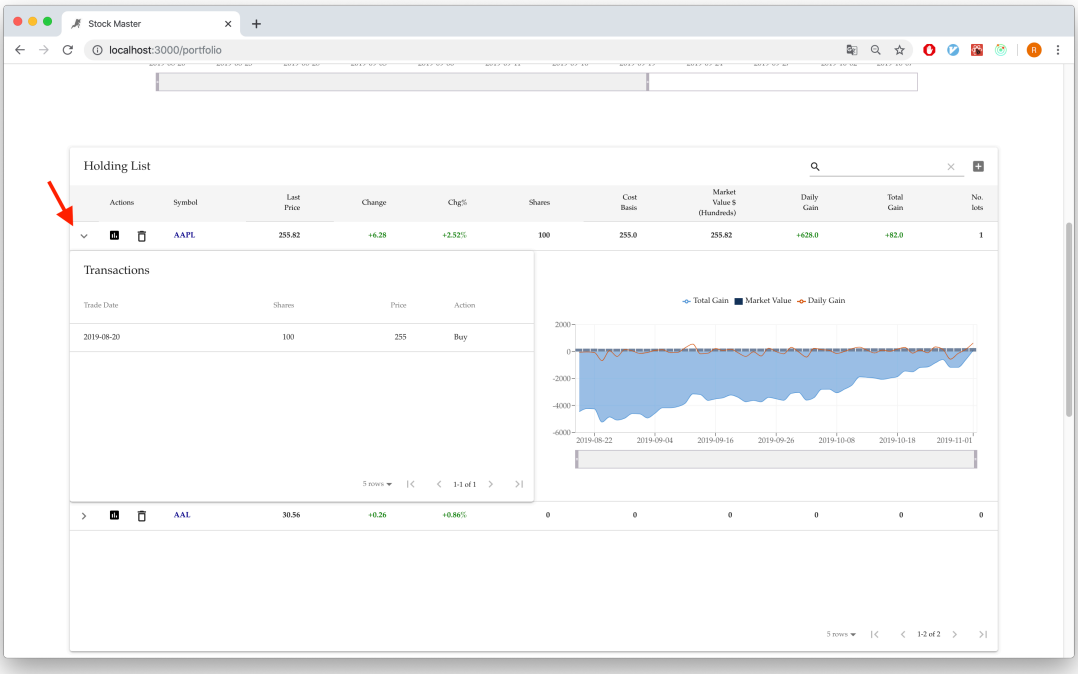
- The statistics panel shows the total portfolio value, total gain and daily gain.
- The radar chart shows the stocks in the holding list comparing with each other in range 0 to 1, for example, if `AAPL` has market value 100 and `AAL` has market value 200, then in the market value dimension, `AAPL` will be $\frac{100}{100+200} = 0.333$.
- The pie chart shows the proportion of market value of each stock.
- The composed chart shows the portfolio total gain, total market value and daily gain.



You can also drag the grey bar below the composed chart to shrunk the range.

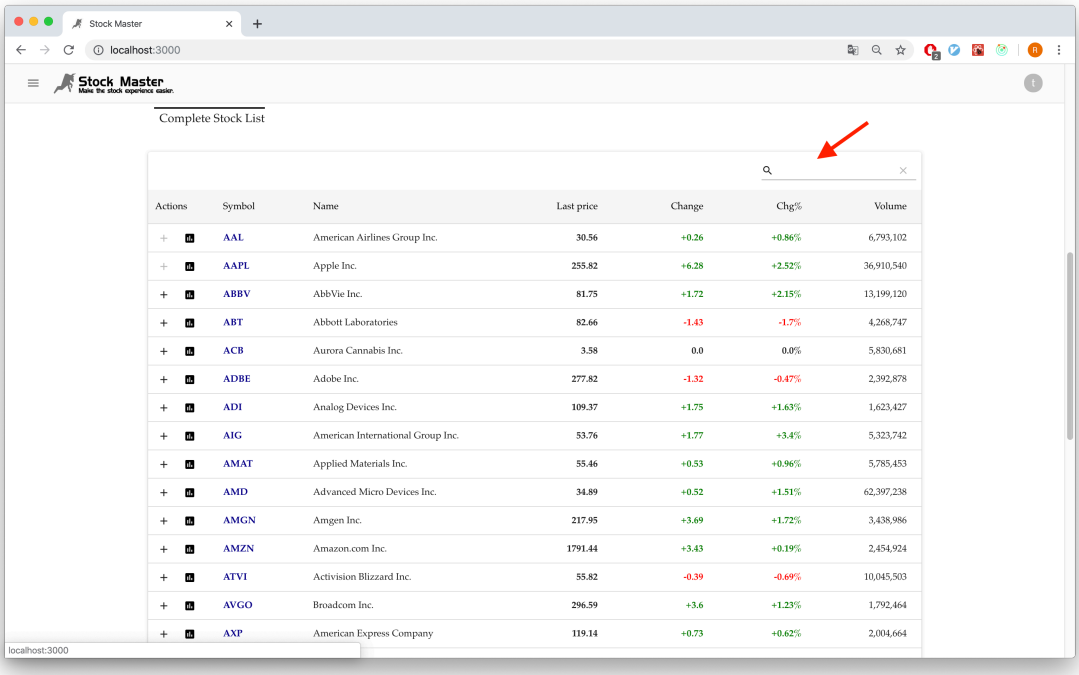


You can also see each stock performance by expanding the row.



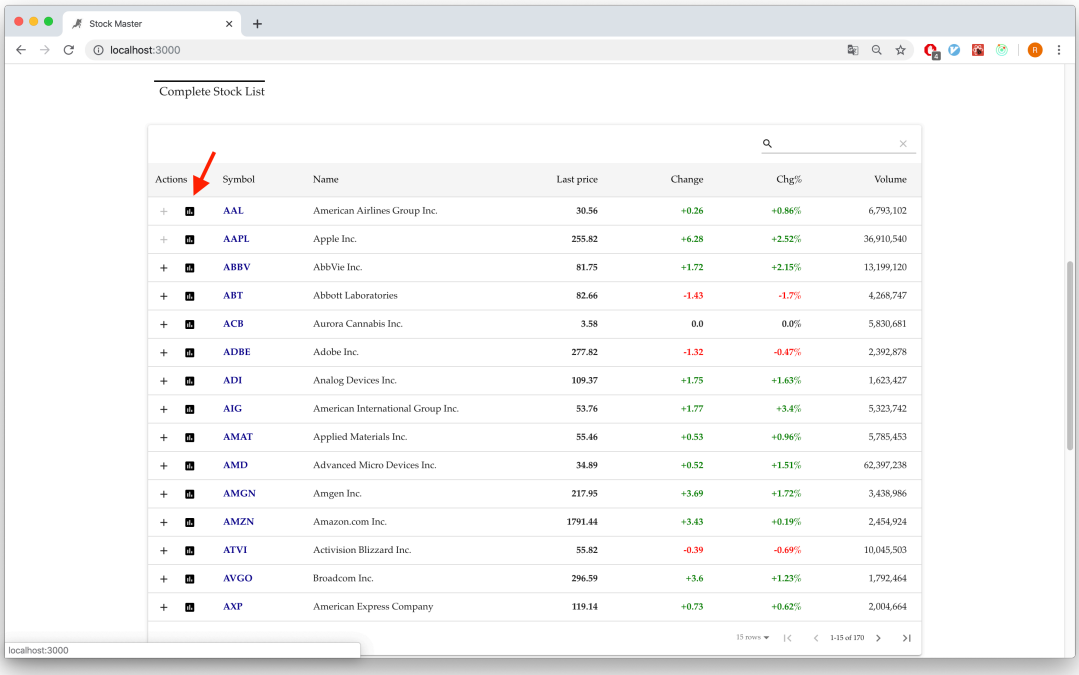
5. Stock search

You can search a stock by entering the stock symbol name or company name in the search bar on complete stock list in homepage.



6. Stock detail information

You can see stock detail by clicking the small icon on each row of the symbol list as follows:

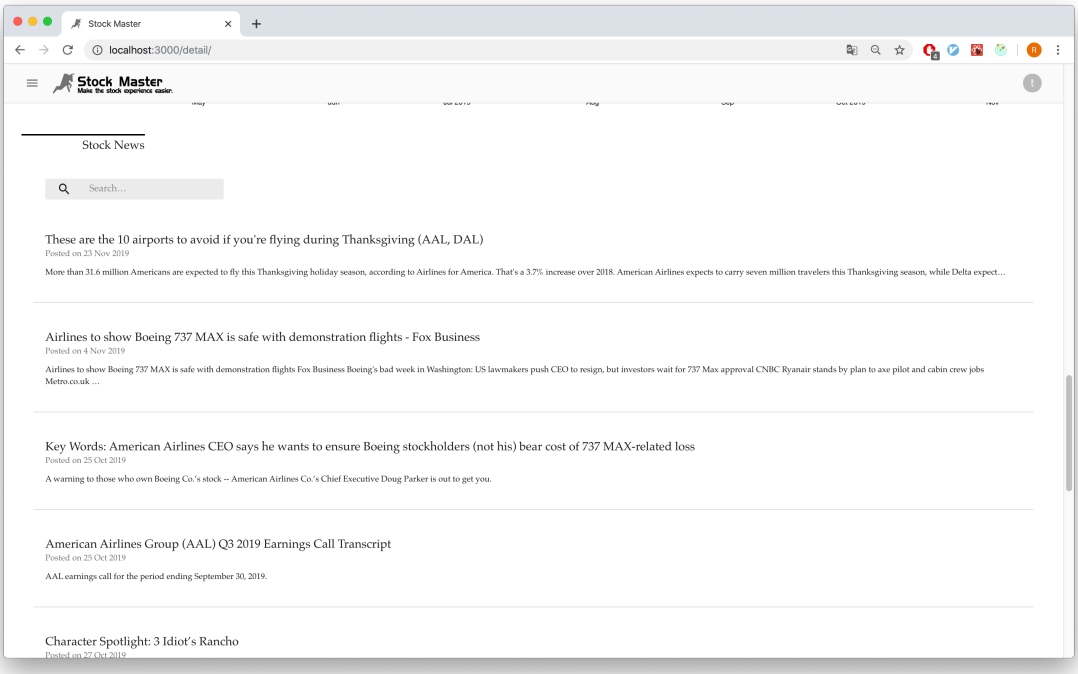


(6.1) Stock detail chart

Then you can see the stock detail chart when scrolling down the page. You can shrink the chart by clicking the `–` icon and zoom by clicking the `+` icon, you can also reset the position by clicking the reset icon. The statistics is shown on the upper left corner of the chart and the technical indicators are shown on the bottom of the detail chart.

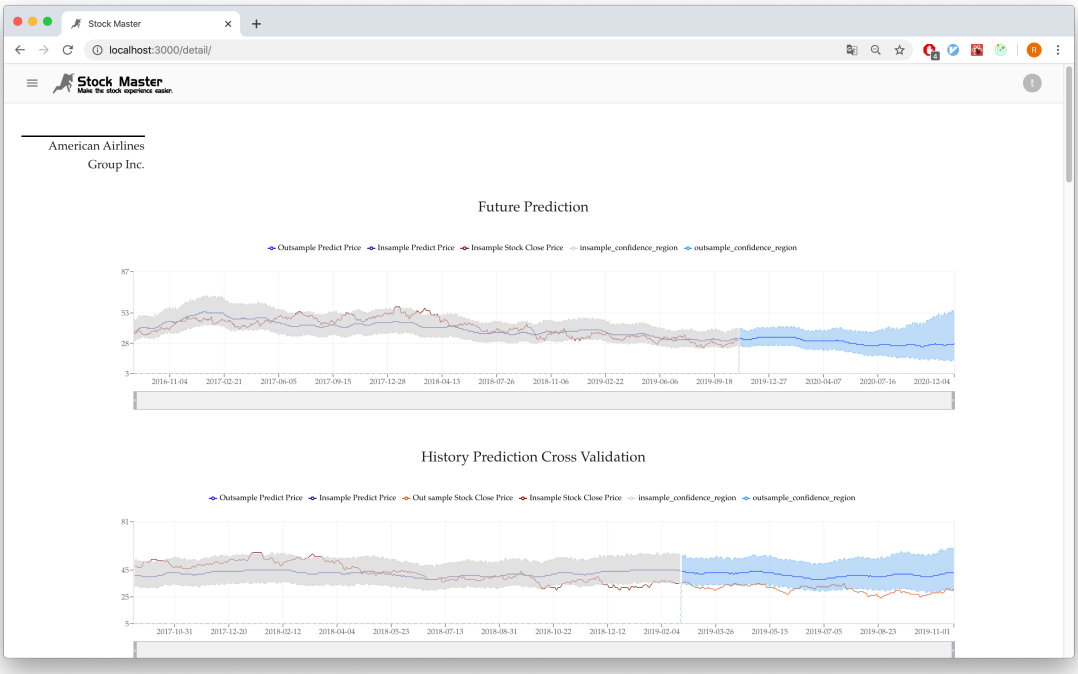
(6.2) Stock news feed

You can see the stock news when scrolling down the page, and you can also search news by entering the keywords in the search bar.

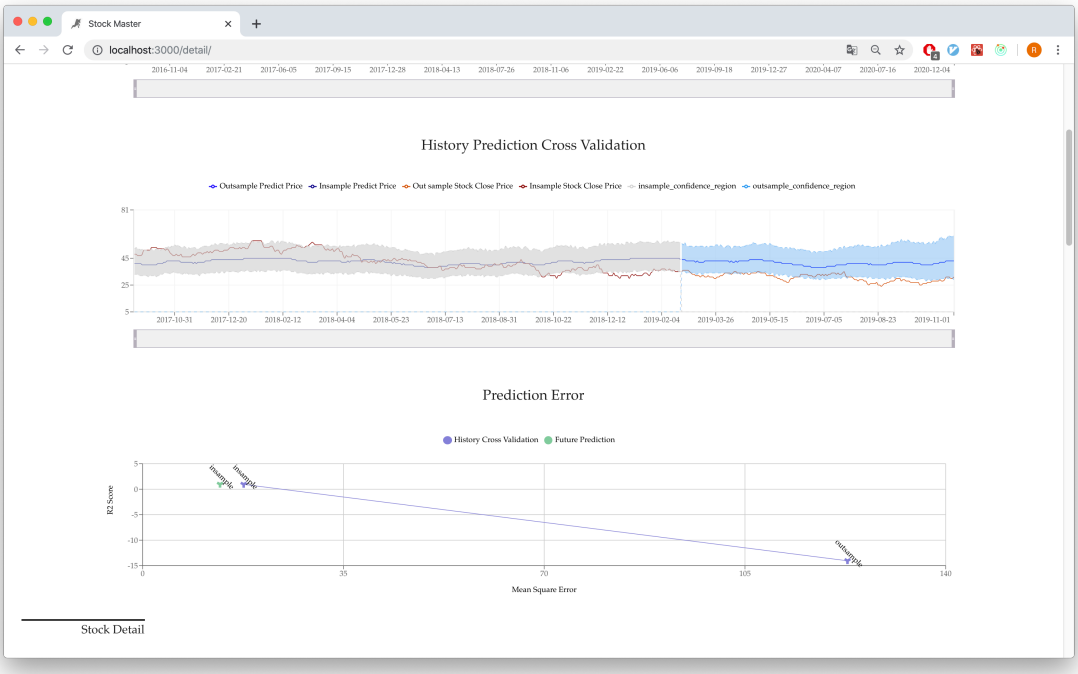


7. Stock price prediction

The stock price prediction section is in the same page as the stock detail page, you can see the future prediction on the first chart, the grey and blue area shows the confidence region which means the upper bound and lower bound of the price prediction. The red line is the actual stock price, the blue line the predicted price.

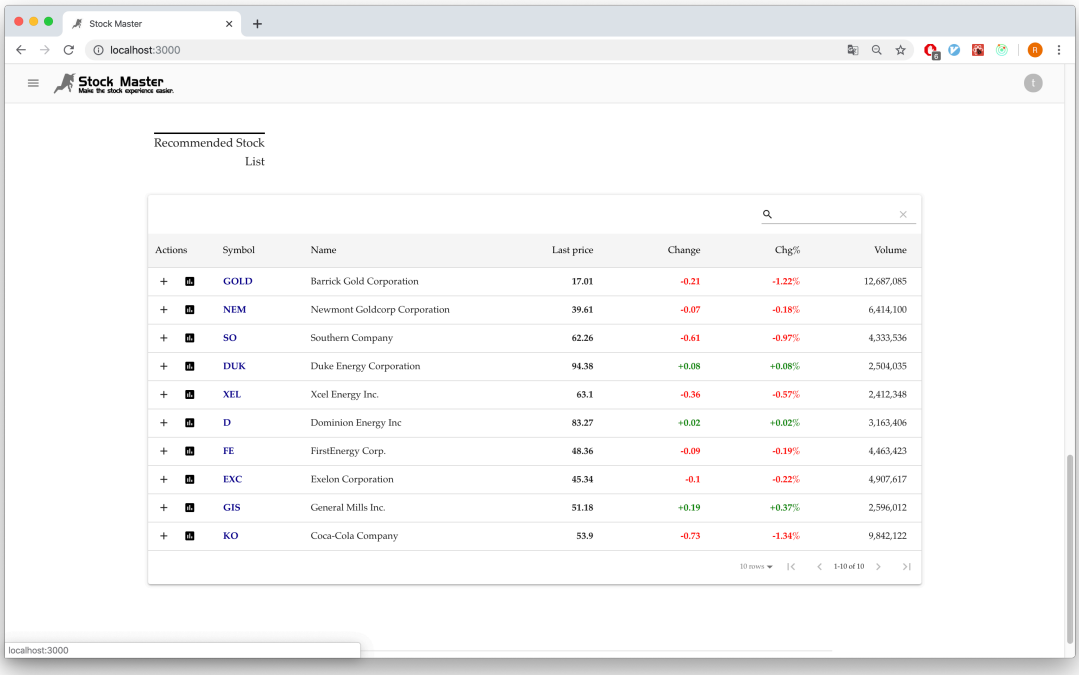


We also shows the in-sample error and out-sample error of the model, the model is better when R2 score is close to 1, the mean square error is close to 0.



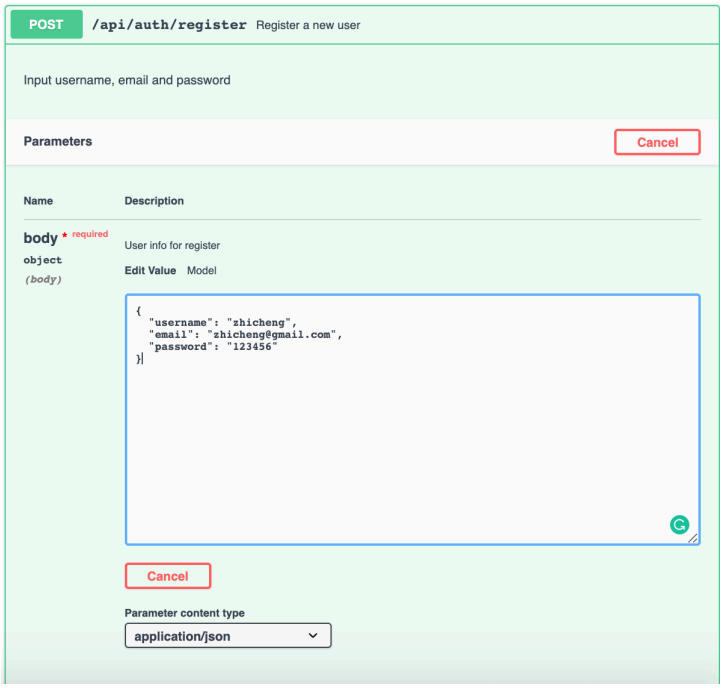
8. Factor analysis and stock recommendation

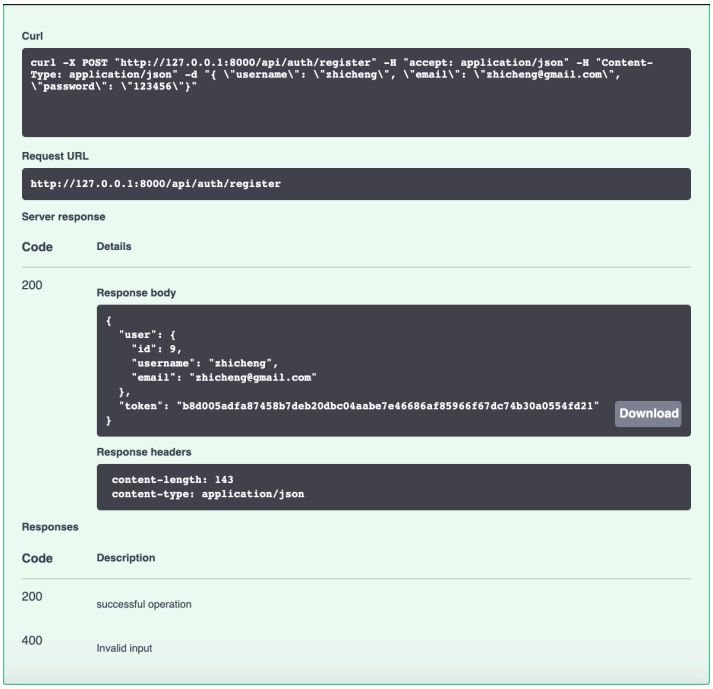
We use Fama-French three factor model to rank the stock and choose the top 10 stock to recommend to users, you can see the recommended stock when go to homepage and scroll down to the bottom.


Stock Master
localhost:3000
Stock Master
Make the stock experience easier.
Recommended Stock List
Actions
Symbol
Name
Last price
Change
Chg%
Volume
GOLD Barrick Gold Corporation 17.01 -0.21 -1.22% 12,687,085
NEM Newmont Goldcorp Corporation 39.61 -0.07 -0.18% 6,414,100
SO Southern Company 62.26 -0.61 -0.97% 4,333,536
DUK Duke Energy Corporation 94.38 +0.08 +0.08% 2,504,035
XEL Xcel Energy Inc. 63.1 -0.36 -0.57% 2,412,348
D Dominion Energy Inc 83.27 +0.02 +0.02% 3,163,406
FE FirstEnergy Corp. 48.36 -0.09 -0.19% 4,463,423
EXC Exelon Corporation 45.34 -0.1 -0.22% 4,907,617
GIS General Mills Inc. 51.18 +0.19 +0.37% 2,596,012
KO Coca-Cola Company 53.9 -0.73 -1.34% 9,842,122
10 rows
1-10 of 10
localhost:3000

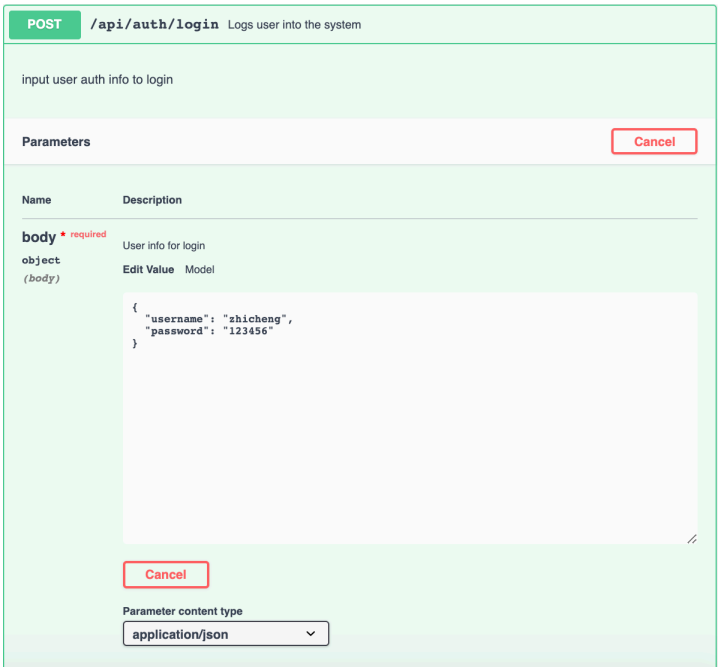
Appendix B

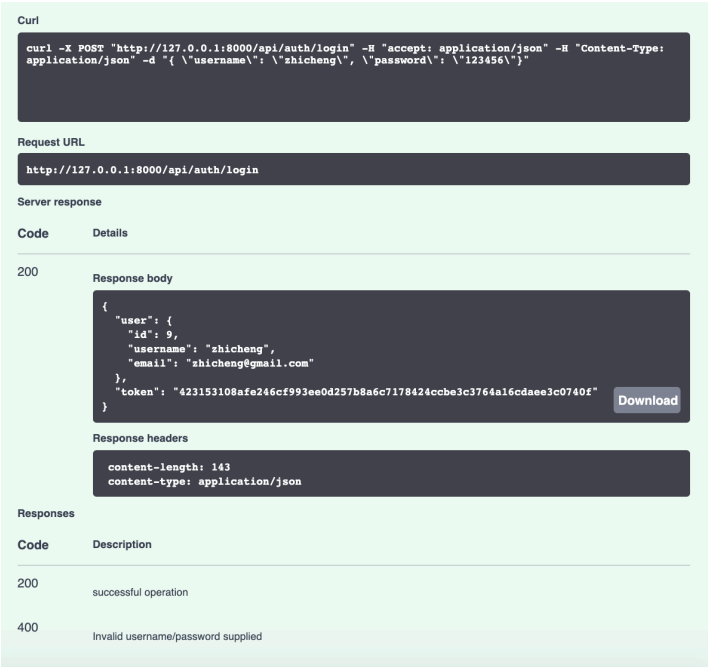
Column Names	Test Case ID	Test Case Summary	Related Requirement	Prerequisites	Test Procedure	Test Data	Expected Result	Actual Result	Status	Remarks	Executed By	Date of Execution
Authentication-Register (Backend)	AR001	To verify if user is able to register after user input the information in swagger UI.		1.User has input its information on swagger UI.	1.Input the information in json format in swagger UI. 2.Send the post request	1.username:conlin, eric, ted, hao 2.password:123456 , 654321, fine, docker	1.The swagger UI will response with 200, and return the user information in json format with auth token.	1.The swagger UI will response with 200, and return the user information in json format with auth token.	Pass	This function works.	Chuguan Tian	12/10/19



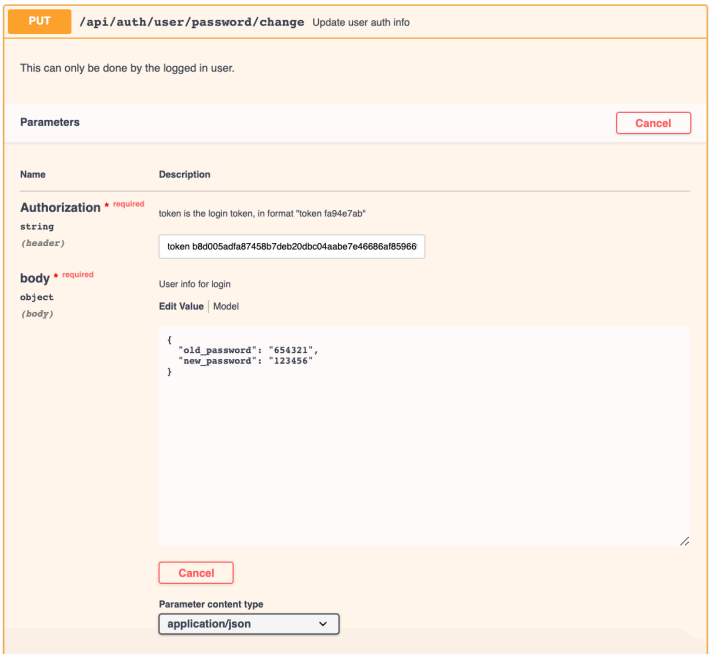


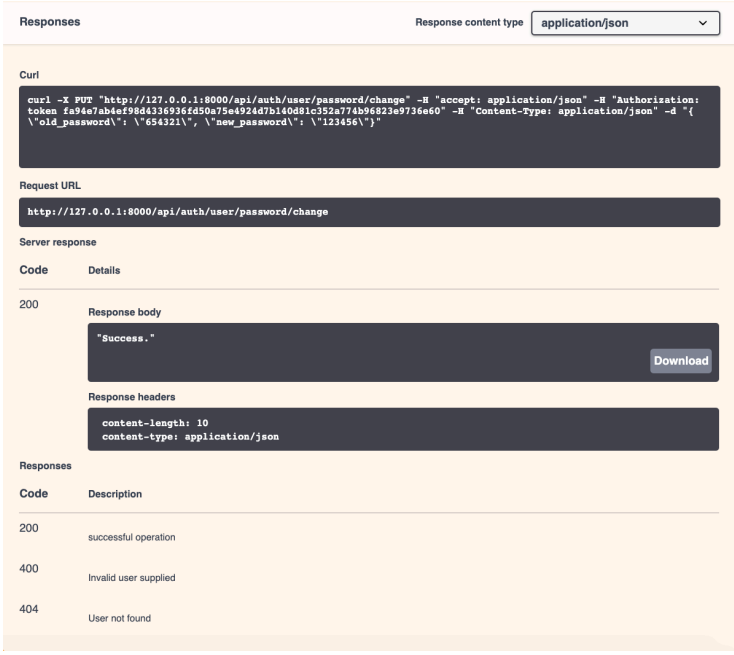
Column Names	Test Case ID	Test Case Summary	Related Requirement	Prerequisites	Test Procedure	Test Data	Expected Result	Actual Result	Status	Remarks	Executed By	Date of Execution
Authentication-Login (Backend)	AL001	To verify if user is able to login after user input the information on swagger UI.	AR001	1.User has registered. 2.User has input its information on swagger UI.	1.Input the information in json format in swagger UI. 2.Send the post request	1.username:conlin, eric, ted, hao 2.password:123456 , 654321, fine, docker	1.The swagger UI will response with 200, and return the user information in json format with auth token.	1.The swagger UI will response with 200, and return the user information in json format with auth token.	Pass	This function works.	Chuguan Tian	12/10/19



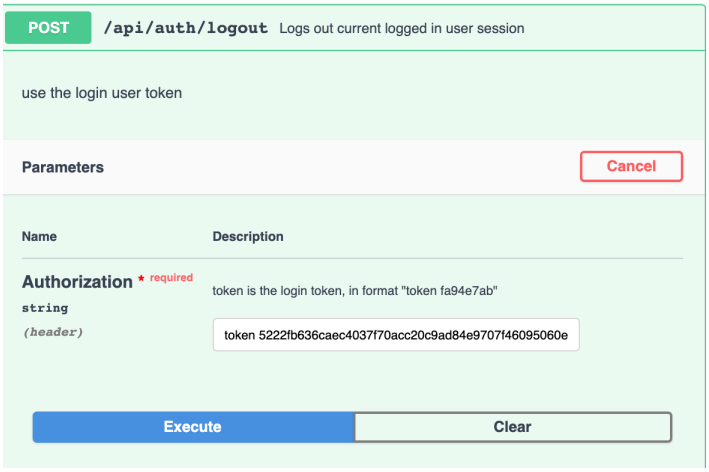


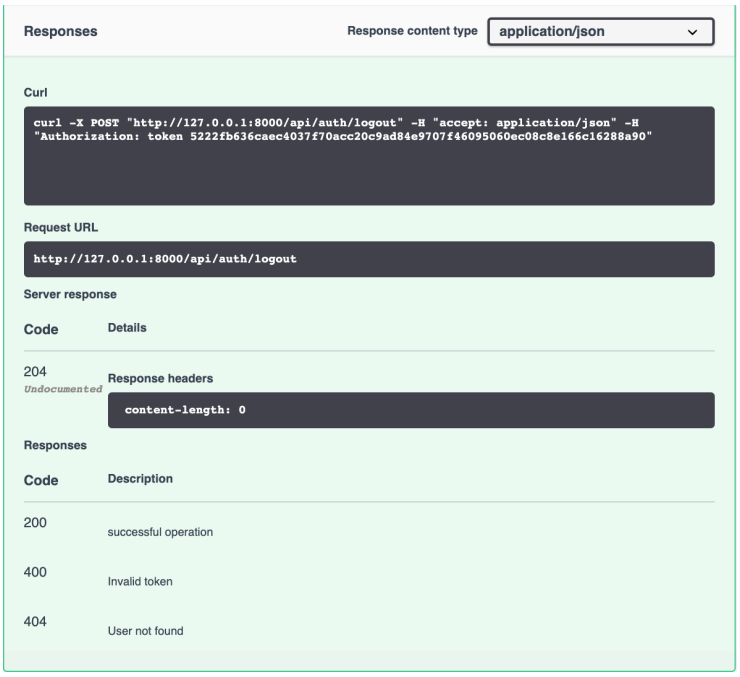
Column Names	Test Case ID	Test Case Summary	Related Requirement	Prerequisites	Test Procedure	Test Data	Expected Result	Actual Result	Status	Remarks	Executed By	Date of Execution
Authentication-ChangePassword (Backend)	AC001	To verify if user is able to change his password after user input the old password and new password on swagger UI.	AR001, AL001	1.User has registered. 2.User has login.	1.Input the information in json format in swagger UI and fill the login token in the header. 2.Send the put request	1.old_password:123456, 654321, fine, docker 2.new_password: fine, docker,123456, 654321,	1.The swagger UI will response with 200, and return the message "success"	1.The swagger UI will response with 200, and return the message "success"	Pass	This function works.	Chuguan Tian	12/10/19



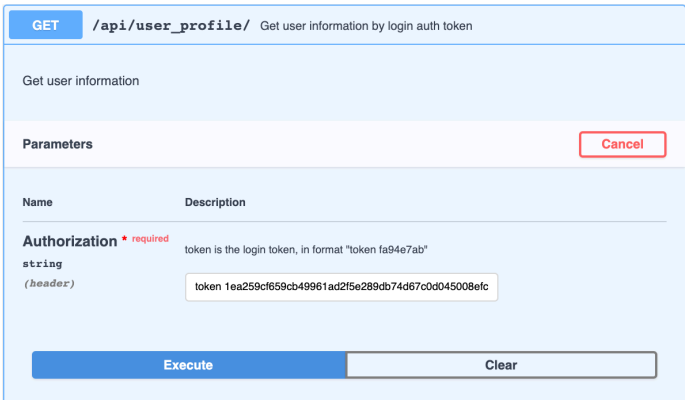


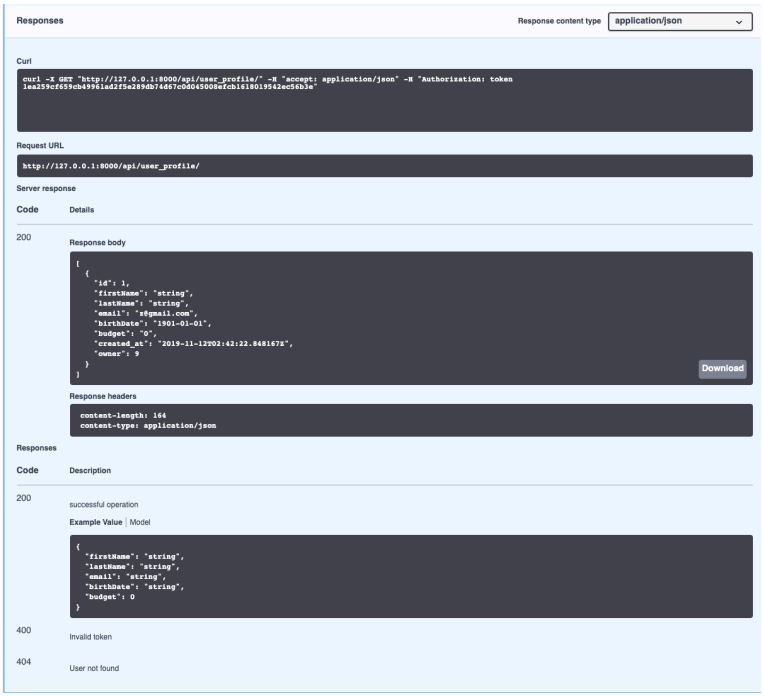
Column Names	Test Case ID	Test Case Summary	Related Requirement	Prerequisites	Test Procedure	Test Data	Expected Result	Actual Result	Status	Remarks	Executed By	Date of Execution
Authentication-Logout (Backend)	AC001	To verify if user is able to change his password after user input the old password and new password on swagger UI.	AR001, AL001	1.User has registered. 2.User has login.	1.Input the information in json format in swagger UI and fill the login token in the header. 2.Send the put request	1.old_password:123456, 654321, fine, docker 2.new_password: fine, docker,123456, 654321,	1.The swagger UI will response with 200, and return the message "success"	1.The swagger UI will response with 200, and return the message "success"	Pass	This function works.	Chuguan Tian	12/10/19



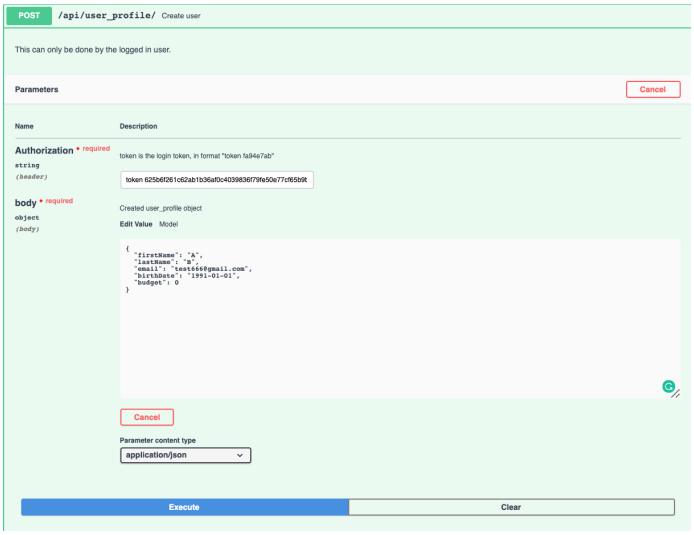


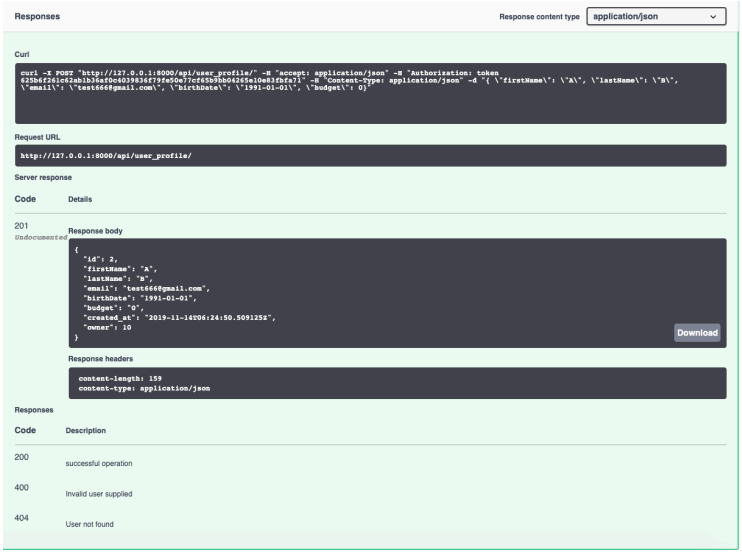
Column Names	Test Case ID	Test Case Summary	Related Requirement	Prerequisites	Test Procedure	Test Data	Expected Result	Actual Result	Status	Remarks	Executed By	Date of Execution
UserProfile-Get (Backend)	UG001	To verify if user is able to login after user input the information on swagger UI.	AR001, AL001, UC001	1.User has registered. 2.User has login 3.User has create his profile information 4.User has input its information on swagger UI.	1.Input the information in json format in swagger UI. 2.Send the Get request	1.username:conlin, eric, ted, hao	1.The swagger UI will response with 200, and return the user information in json format.	1.The swagger UI will response with 200, and return the user information in json format.	Pass	This function works.	Chuguan Tian	30/10/19



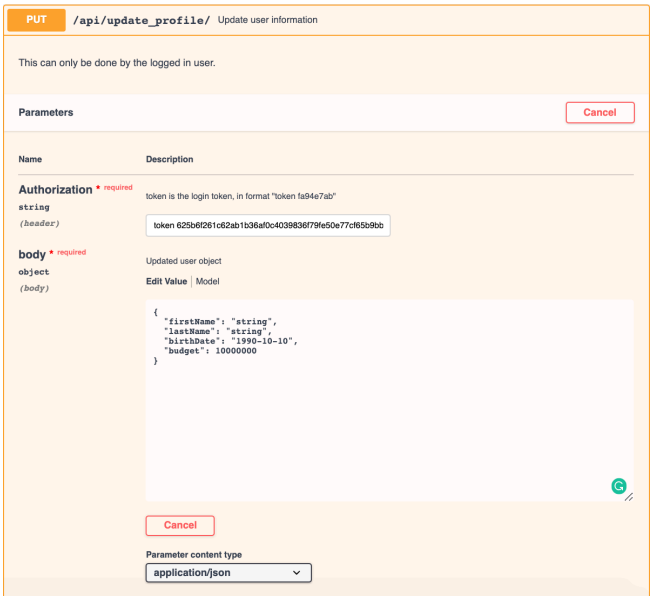


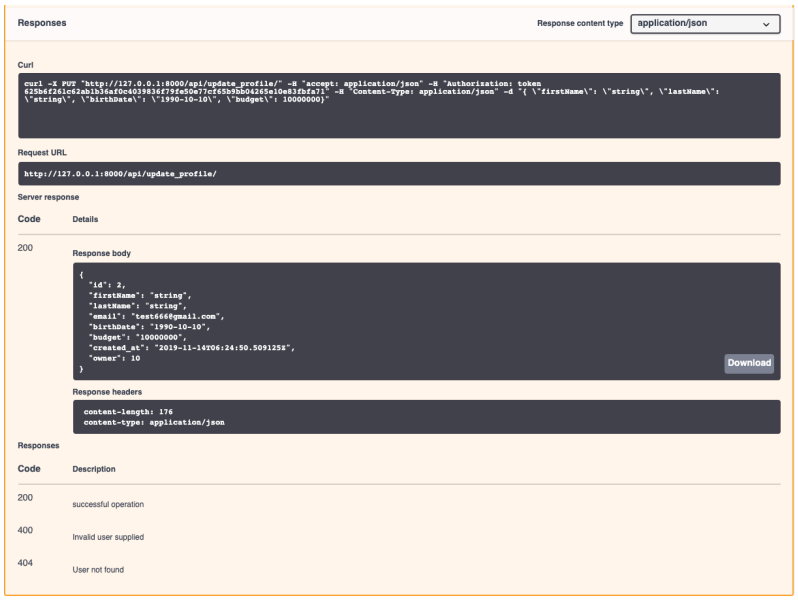
Column Names	Test Case ID	Test Case Summary	Related Requirement	Prerequisites	Test Procedure	Test Data	Expected Result	Actual Result	Status	Remarks	Executed By	Date of Execution
UserProfile-Create (Backend)	UC001	To verify if user is able to create its information after user login swagger UI.	AL001, AR001	1.User has registered. 2.User has login 3.User has input its information on swagger UI.	1.Input the information in json format in swagger UI. 2.Send the post request	1.firstName: conlin, eric, ted, hao 2.lastName: tian, liang, tao, jiang 3.birthDate: 1991-01-01 4.budget: 0, 10000, 111000	1.The swagger UI will response with 200, and return the user information in json format.	1.The swagger UI will response with 200, and return the user information in json format.	Pass	This function works.	Chuguan Tian	30/10/19



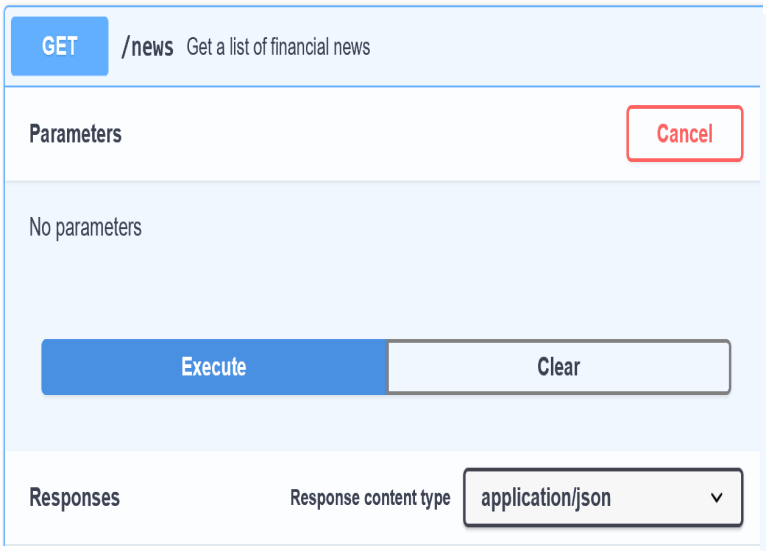


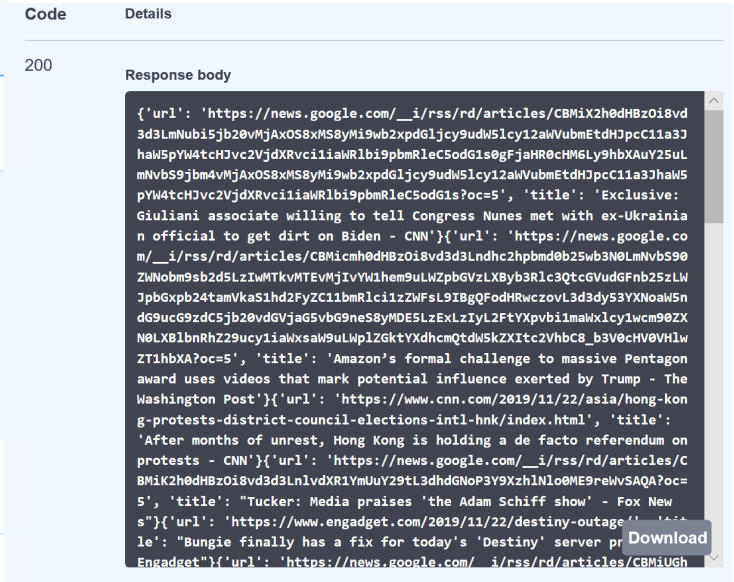
Column Names	Test Case ID	Test Case Summary	Related Requirement	Prerequisites	Test Procedure	Test Data	Expected Result	Actual Result	Status	Remarks	Executed By	Date of Execution
UserProfile-Update (Backend)	UU001	To verify if user is able to change his password after user input the old password and new password on swagger UI.	AR001, AL001, UC001	1.User has registered. 2.User has login 3.User has create his profile information 4.User has input its information on swagger UI.	1.Input the information in json format in swagger UI and fill the login token in the header. 2.Send the put request	1.firstName: conlin, eric, ted, hao 2.lastName: tian, liang, tao, jiang 3.birthDate: 1991-01-01 4.budget: 0, 10000, 111000	1.The swagger UI will response with 200, and return the user information in json format.	1.The swagger UI will response with 200, and return the user information in json format.	Pass	This function works.	Chuguan Tian	30/10/19



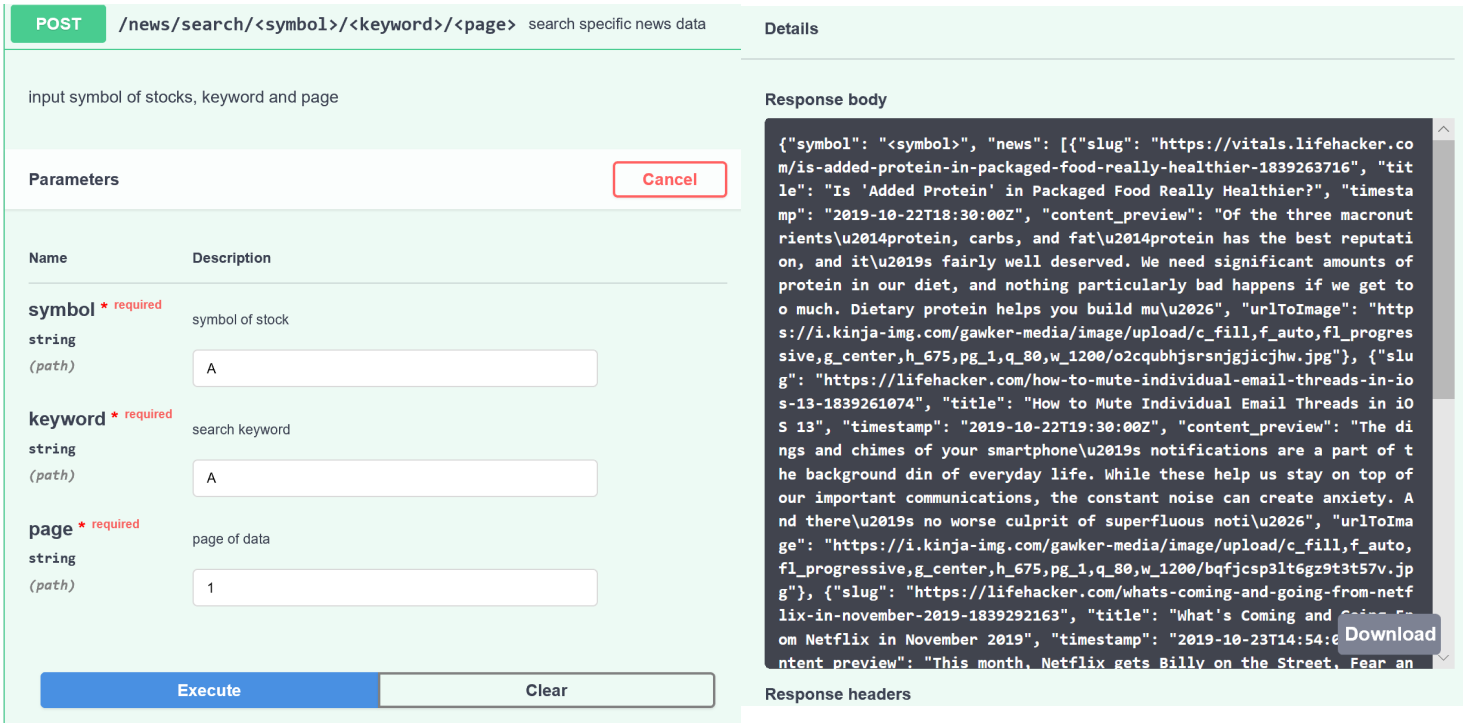


Tags	URL	Description	Requirement	Inputs	Output
News	/news/	Get a list of news	None	None	The swagger UI will response with 200, and return a list of news

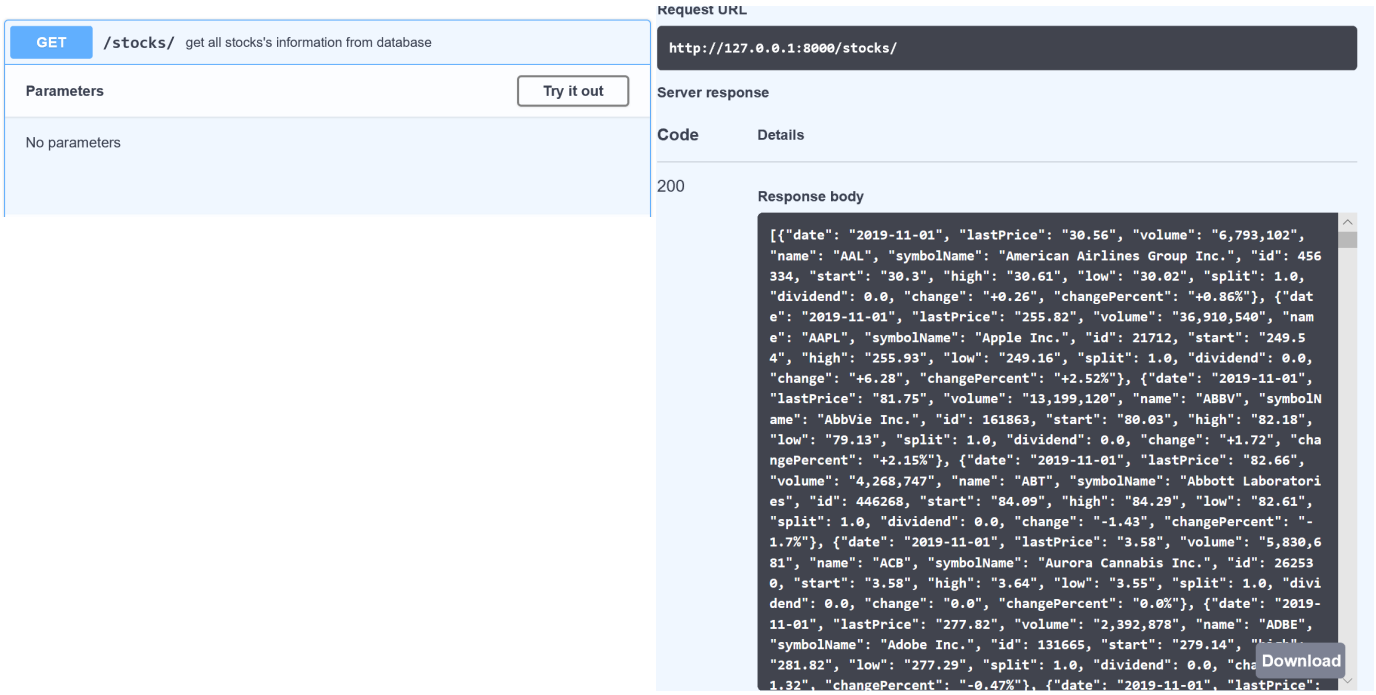




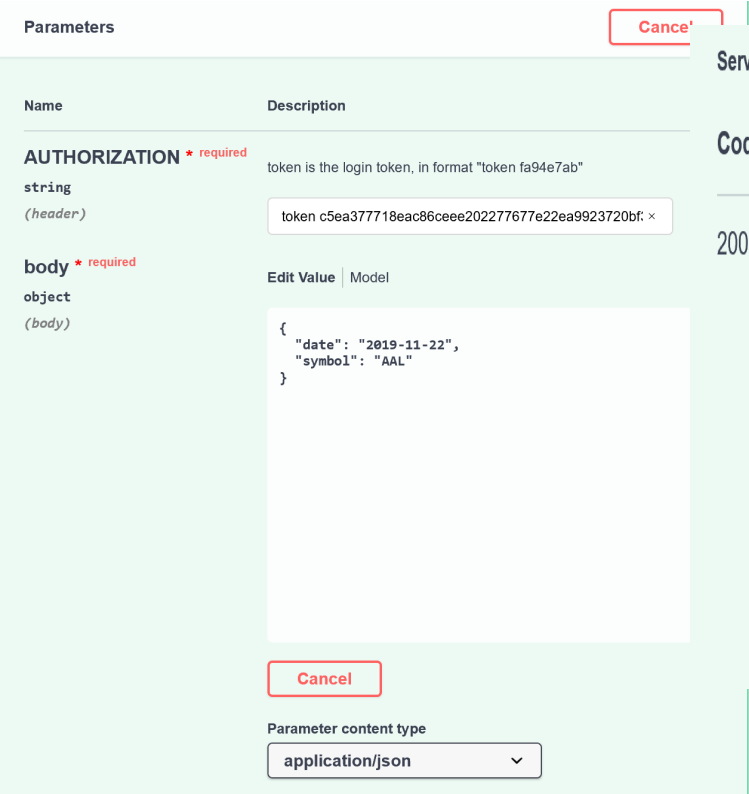
Tags	URL	Description	Requirement	Inputs	Output
News	/news/search/<symbol>/<keyword>/<page>	search specific news data	None	“symbol” : ” A” , “keyword” : “A” , “page” :1,	The swagger UI will response with 200, and return news that related to symbol and keyword in json format

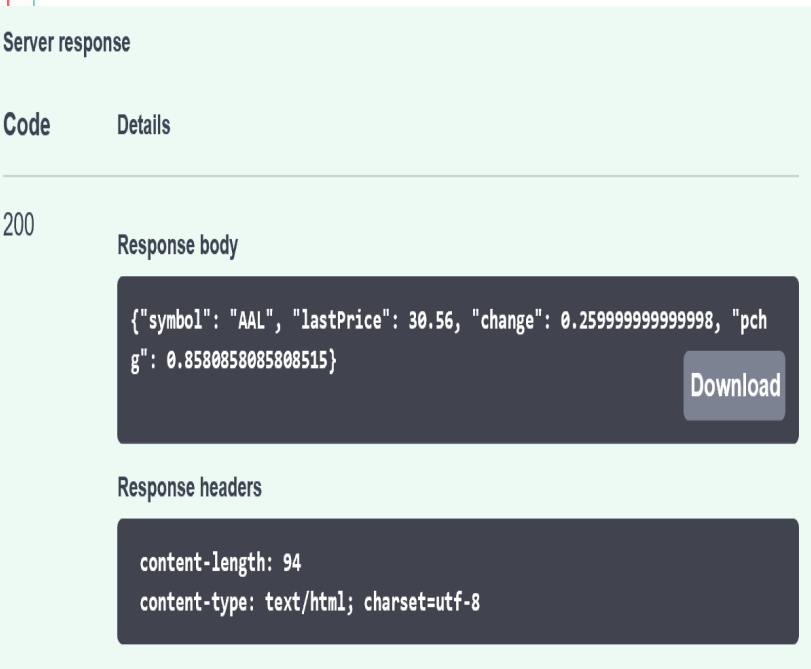


Tags	URL	Description	Requirement	Inputs	Output
Stock	/stock	get all stocks information from database	None	None	The swagger UI will response with 200, and return all stocks information in json format

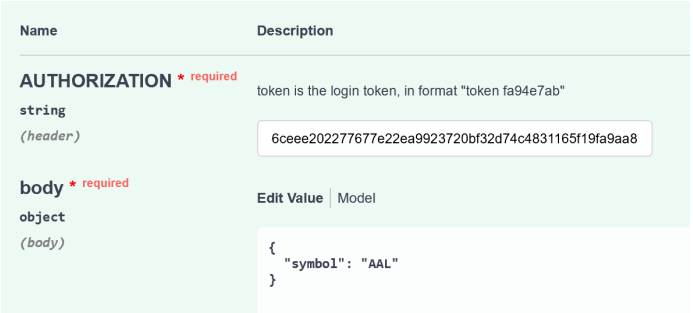


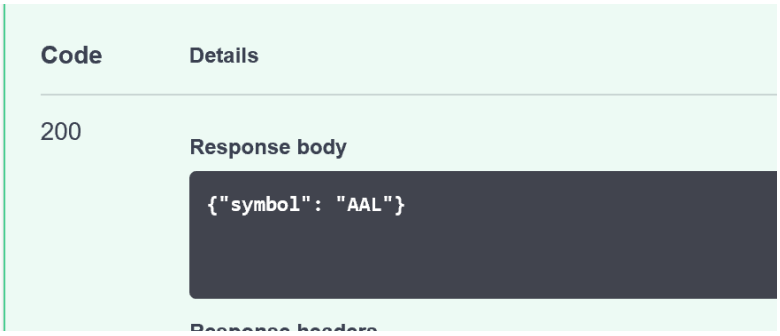
Tags	URL	Description	Requirement	Inputs	Output
Stock	/add/	Add stock to user' s watching list	Need to login in	"date" :" 2019-11-22" , "symbol" : "AAL"	The swagger UI will response with 200, and return symbol' s price information in json format



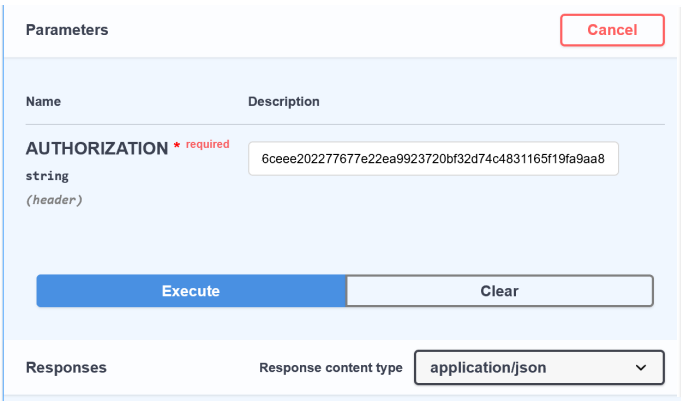


Tags	URL	Description	Requirement	Inputs	Output
Stock	/remove/	Remove stock from user' s watching list	Need to login in	"symbol" : "AAL"	The swagger UI will response with 200, and return a string in json format





Tags	URL	Description	Requirement	Inputs	Output
Stock	/holdings/	Used to show all stocks that user holds	Need to login in	None	The swagger UI will response with 200, and return symbol' s price information in holding list in json format



Tags	URL	Description	Requirement	Inputs	Output
Stock	/purchase.	Used to purchase stock	Need to login in	Symbol: “AAL” , Shares.:” 10, Price: 10 Date: ” 2019-11-20” Action: 0	The swagger UI will response with 200, and return transaction detail in json format

Parameters Cancel

Name	Description
AUTHORIZATION * required string *(header)*	token is the login token, in format "token fa94e7ab" 6ceee202277677e22ea9923720bf32d74c4831165f19fa9aa8

body * required
object
(body)
	Edit Value | Model
	{
	"symbol": "AAL",
	"shares": 10,
	"price": 10,
	"date": "2019-11-20",
	"action": 0
	}

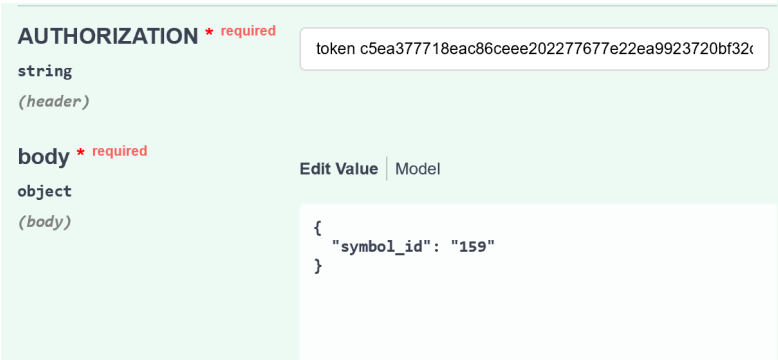
Code	Details
200	Response body {"id": 159, "symbol": "AAL", "tradeDate": "2019-11-20", "shares": 10.0, "price": 10.0, "action": 0} Download

Tags	URL	Description	Requirement	Inputs	Output
Stock	/update/	Used to update holdings	Need to login in	Symbol_id: “159” , Shares: “20” , Price,: “10” , Date: ” 2019-11-21” , Action: “0”	The swagger UI will response with 200, and return a string in json format

Name	Description
AUTHORIZATION * required `string` *(header)*	6ceee202277677e22ea9923720bf32d74c4831165f19fa9aa8
body * required `object` *(body)*	Edit Value | Model `{ "symbol_id": "159", "shares": 20, "price": 10, "date": "2019-11-21", "action": 0 }`

Code	Details
200	**Response body** `update success`

Tags	URL	Description	Requirement	Inputs	Output
Stock	/delete/	Used to sell stocks that user has already holds	Need to login	“Symbol_id”:”159”	The swagger UI will response with 200, and return a string in json format



Tags	URL	Description	Requirement	Inputs	Output
Stock	/profolio/	Used to show user' s portfolio	Need to login in	None	The swagger UI will response with 200, and return three components in json format

Parameters Cancel

Name	Description
AUTHORIZATION * required string *(header)*	6ceee202277677e22ea9923720bf32d74c4831165f19fa9aa8

Execute Clear

Code **Details**

200

Response body


```
{"holdingList": [{"name": "AAL", "lastPrice": "30.56", "change": "+0.2
6", "changePercent": "+0.86%", "shares": 0.0, "costBasis": "0", "marke
tValue": "0.0", "dailyGain": "0.0", "totalGain": "0", "noLots": "1",
"std": 16.87}], "transactions": {"AAL": [{"tradeDate": "2019-10-02",
"shares": 10.0, "price": 10.0, "action": 0, "id": 162}]}, "symbolDat
a": {"AAL": [{"date": "2019-10-02", "dg": -0.0, "tg": 0, "mv": 0.0, "o
pen": 26.5, "close": 25.37, "dailyCostBasis": 0}, {"date": "2019-10-0
3", "dg": -0.0, "tg": 0, "mv": 0.0, "open": 25.47, "close": 25.27, "da
ilyCostBasis": 0}, {"date": "2019-10-04", "dg": 0.0, "tg": 0, "mv": 0.
0, "open": 25.33, "close": 25.83, "dailyCostBasis": 0}, {"date": "201
9-10-07", "dg": 0.0, "tg": 0, "mv": 0.0, "open": 25.8, "close": 25.83,
"dailyCostBasis": 0}, {"date": "2019-10-08", "dg": 0.0, "tg": 0, "mv":
0.0, "open": 25.56, "close": 26.26, "dailyCostBasis": 0}, {"date": "20
19-10-09", "dg": 0.0, "tg": 0, "mv": 0.0, "open": 26.78, "close": 27.0
7, "dailyCostBasis": 0}, {"date": "2019-10-10", "dg": 0.0, "tg": 0, "m
v": 0.0, "open": 26.68, "close": 27.18, "dailyCostBasis": 0}, {"date":
"2019-10-11", "dg": -0.0, "tg": 0, "mv": 0.0, "open": 27.63, "close":
27.36, "dailyCostBasis": 0}, {"date": "2019-10-14", "dg": 0.0, "tg":
0, "mv": 0.0, "open": 27.37, "close": 27.62, "dailyCostBasis": 0}, {"d
ate": "2019-10-15", "dg": 0.0, "tg": 0, "mv": 0.0, "open": 27.63, "clo
se": 28.27, "dailyCostBasis": 0}, {"date": "2019-10-16", "dg":
"tg": 0, "mv": 0.0, "open": 28.33, "close": 28.05, "dailyCostB
0}, {"date": "2019-10-17", "dg": -0.0, "tg": 0, "mv": 0.0, "open": 28.
```


Download

Response headers

```
content-length: 4120
content-type: text/html; charset=utf-8
```

Tags	URL	Description	Requirement	Inputs	Output
Stock	/stock/{symbol}/chart/	Used to show chart	Need to login in	Symbol: AAL	The swagger UI will response with 200, and return historical data in json format

Parameters Cancel

Name	Description
AUTHORIZATION * required string *(header)*	6ceee202277677e22ea9923720bf32d74c4831165f19fa9aa8
symbol * required string *(path)*	AAL

Execute Clear

Code **Details**

200

Response body


```
[{"date": "2007-11-27", "close": 19.51, "volume": 4106600, "symbol":
"AAL", "id": 11649, "open": 19.81, "high": 20.13, "low": 18.75, "spli
t": 1.0, "dividend": 0.0, "aChange": 1.379999999999999, "pChange": 0.0
7359999999999994}, {"date": "2007-11-28", "close": 20.97, "volume": 37
40900, "symbol": "AAL", "id": 11648, "open": 19.97, "high": 21.38, "lo
w": 19.5, "split": 1.0, "dividend": 0.0, "aChange": 1.879999999999999,
"pChange": 0.09641025641025636}, {"date": "2007-11-29", "close": 20.0
6, "volume": 3034100, "symbol": "AAL", "id": 11647, "open": 20.6, "hig
h": 20.6, "low": 19.83, "split": 1.0, "dividend": 0.0, "aChange": 0.77
00000000000031, "pChange": 0.03883005547150798}, {"date": "2007-11-3
0", "close": 20.77, "volume": 1968600, "symbol": "AAL", "id": 11646,
"open": 20.6, "high": 21.44, "low": 20.54, "split": 1.0, "dividend":
0.0, "aChange": 0.9000000000000021, "pChange": 0.04381694255111987},
{"date": "2007-12-03", "close": 19.93, "volume": 2127800, "symbol": "A
AL", "id": 11645, "open": 21.21, "high": 21.21, "low": 19.82, "split":
1.0, "dividend": 0.0, "aChange": 1.3900000000000006, "pChange": 0.0701
311806256307}, {"date": "2007-12-04", "close": 19.62, "volume": 166890
0, "symbol": "AAL", "id": 11644, "open": 19.47, "high": 20.0, "low": 1
9.1, "split": 1.0, "dividend": 0.0, "aChange": 0.8999999999999986, "pC
hange": 0.047120418848167464}, {"date": "2007-12-05", "close": 18.04,
"volume": 4279900, "symbol": "AAL", "id": 11643, "open": 19.48,
h": 19.48, "low": 17.86, "split": 1.0, "dividend": 0.0, "aChar
20000000000001, "pChange": 0.09070548712206053}, {"date": "2007-12-0
```


Download

Tags	URL	Description	Requirement	Inputs	Output
Prediction	/stocks/{symbol}/predict/history	Used to predict stock' s historical trending	Need to login in	"Symbol" : "AAL"	The swagger UI will response with 200, and return historical

Parameters Cancel

Name	Description
AUTHORIZATION * required string *(header)*	token c5ea377718eac86ceee202277677e22ea9923720bf32
symbol * required string *(path)*	AAL

Code **Details**

200

Response body


```
{"result": [{"insample_y": 47.7, "insample_yhat": 41.43299809819489,
"insample_yhat_upper": 51.572613599163056, "insample_yhat_lower": 32.9
1334199268088, "date": "2017-09-25"}, {"insample_y": 47.58000000000000
5, "insample_yhat": 41.41274223017505, "insample_yhat_upper": 52.85654
825411647, "insample_yhat_lower": 32.66357692592949, "date": "2017-09-
26"}, {"insample_y": 46.85, "insample_yhat": 41.20689934966671, "insam
ple_yhat_upper": 52.15455877732278, "insample_yhat_lower": 32.55251821
8896154, "date": "2017-09-27"}, {"insample_y": 47.49, "insample_yhat":
40.98520317240715, "insample_yhat_upper": 51.66636345403625, "insample
_yhat_lower": 32.396926495445534, "date": "2017-09-28"}, {"insample_
y": 47.49, "insample_yhat": 40.94944115607748, "insample_yhat_upper":
51.36044966499805, "insample_yhat_lower": 32.297467836750826, "date":
"2017-09-29"}, {"insample_y": 47.72999999999999, "insample_yhat": 40.3
1161939619135, "insample_yhat_upper": 51.86762884106986, "insample_yha
t_lower": 32.14342532706021, "date": "2017-10-02"}, {"insample_y": 50.
510000000000005, "insample_yhat": 40.33125712270468, "insample_yhat_up
per": 50.87099047698621, "insample_yhat_lower": 32.43901559601617, "da
te": "2017-10-03"}, {"insample_y": 50.73, "insample_yhat": 40.19486199
475321, "insample_yhat_upper": 50.680490481481016, "insample_yhat_lowe
r": 31.78582118279127, "date": "2017-10-04"}, {"insample_y": 50.56, "i
nsample_yhat": 40.0661270969469, "insample_yhat_upper": 50.67
781, "insample_yhat_lower": 31.699313660032498, "date": "2017-
{"insample_y": 51.300000000000004, "insample_yhat": 40.14085564839858
```


Download

Tags	URL	Description	Requirement	Inputs	Output
Prediction	/stocks/{symbol}/predict/future	Used to predict stock' s future trending	Need to login in	"Symbol" : "AAL"	The swagger UI will response with 200, and return historical data in json format

Parameters Cancel

Name	Description
AUTHORIZATION * required string *(header)*	7585472a93abfdf4c47660cce3fb1c418c103e070bf7da8edc3
symbol * required string *(path)*	AAL

Code **Details**

200

Response body


```
{"result": [{"insample_y": 37.04999999999999, "insample_yhat": 37.8150
99296697696, "insample_yhat_upper": 47.67330976681675, "insample_yhat_
lower": 30.485073657473365, "date": "2016-08-30"}, {"insample_y": 36.
3, "insample_yhat": 37.96042134821719, "insample_yhat_upper": 47.11065
563672341, "insample_yhat_lower": 29.61543817076287, "date": "2016-08-
31"}, {"insample_y": 36.24, "insample_yhat": 38.10980297354566, "insam
ple_yhat_upper": 47.34439034272937, "insample_yhat_lower": 30.66909699
168023, "date": "2016-09-01"}, {"insample_y": 36.53, "insample_yhat":
38.39975638216179, "insample_yhat_upper": 48.199525474123774, "insampl
e_yhat_lower": 30.711151688649167, "date": "2016-09-02"}, {"insample_
y": 36.96, "insample_yhat": 39.209749158685845, "insample_yhat_upper":
48.588923896785076, "insample_yhat_lower": 31.051273602319657, "date":
"2016-09-06"}, {"insample_y": 38.75, "insample_yhat": 39.4002446705947
6, "insample_yhat_upper": 48.666136559452625, "insample_yhat_lower": 3
2.00932800103188, "date": "2016-09-07"}, {"insample_y": 39.35, "insamp
le_yhat": 39.57752173056948, "insample_yhat_upper": 49.87346128268377,
"insample_yhat_lower": 31.784813359497473, "date": "2016-09-08"}, {"in
sample_y": 38.489999999999995, "insample_yhat": 39.88218230459382, "in
sample_yhat_upper": 51.01206510517324, "insample_yhat_lower": 32.38953
929594864, "date": "2016-09-09"}, {"insample_y": 38.34000000000001, "i
nsample_yhat": 40.26633545116423, "insample_yhat_upper": 50.71
0616, "insample_yhat_lower": 32.16680640797914, "date": "2016-
{"insample_y": 37.370000000000005, "insample_yhat": 40.54021937127707,
```


Download