Spring 4-15-2020

Stock Investment Helper

Mingrui Xu

Stock Investment Helper

by
Mingrui Xu

Applied Project report submitted to the Faculty of the Graduate School of the

in fulfillment of the requirements for the degree of
Master of Science in Computer and Information Sciences

Supervised by: Abrar Qureshi, Ph.D.
Spring 2020

Table of Contents

1. Background of this project

My project is to design an android based application focusing on stock investment for individual users. The name of this project is Stock Investment Helper. People are willing to participate in the stock market as a way to invest their spare money. Experienced users have mastered skills. But some newbies don't know which stock to invest in. They do not understand the vast amount of information from complicated stock applications.

In modern society, with the increase of wealth, many people want to invest the money they have saved, the stock market is one of them. There are many experienced investors who already have some financial sense and can make a lot of investment decisions. There are also some investors who leave their idle money in the hands of professional institutions.

However, many novice users want to make their own investments, and they cannot understand and digest the huge amount of information provided by professional stock apps[1]. Therefore, it is impossible to make good and correct judgment. The stock app on the market was too complicated for the first-time stock market users, so I decided to make a stock app for the entry-level investors. Customers can get the most direct and useful information without having to analyze complex models themselves[2]. They can also communicate with other users of the app, get their Suggestions and make their own decisions. This is a stock investment assistance app.

2. Project Introduction

This project includes the Android application interface in the front end, data fetching in the back end, database storage, data analysis, algorithm support, and connection between front and back end. I'm going to develop everything from the front end to the back end. In my design, first of all, users can get the relevant information of stocks they want to pay attention to with one click, including real-time price and the trend of the day. Secondly, users can see other users' comments and suggestions on the stock and make relevant investment decisions after their own judgment. Finally, some deliberately misleading information needs to be removed to protect other novice investors from being misled.

The first step is to grab stock data from a financial website. The yahoo finance API is the only stock API on the market that is free and open to the public[3-5]. I will use it to grab stock data first, and it will be replaced by another commercial API if the embedded investment function is needed in the future. After fetching the data, I will simply process it and send it to Amazon DynamoDB database for analysis.

For the other two functions, I'll embed a small forum behind each stock. The forum allows users to share their views and opinions on common shares they are concerned with. Moreover, I solved problems that I was facing in 695 course such as performing all computing and querying on the end device which may consume more wireless traffic and may not be very battery friendly. Implement a real-time plotted chart that looks more straight-forward to beginner users.

This mobile application is user-friendly, intuitively looking with real-time chart, recommendation system (small forum inside each stock). It also has an amazing function of stock price prediction. I will use data analysis tools to set up (train) various models, including linear/cubic regression, etc. After that, I will compare the deviation among these prediction results. I will use the latest past 30-day close prices and daily volume to predict close price of the next trade day. I will choose the algorithm with least deviation.

3. System requirements

This project includes the front-end Android user interface interaction and back-end database design, algorithm, and forum.

For the front end, I will develop an Android application using the Intellij IDE Android SDK as a tool[6-7]. In addition to the design and interaction of the Android user interface, another major module is connecting Amazon AWS service to the Android application. Because Android's main threads are usually dealing with user interface issues. It does not provide any Internet service itself. To do this, I need to implement multithreading through the AWS Android SDK to implement all the functions provided by AWS on an Android phone.

For the back-end design, this project uses the DynamoDB database provided by Amazon Cloud services. It can provide NoSQL database. This design is suitable for stock data storage and processing data analysis of this project. To get the stock data I needed, I had to first use the yahoo finance API to grab it from the yahoo site. It is the only free finance API on the market that is open to public. It is quite simple and easy to use. For data analysis part, I will use RStudio to draw real-time chart of stocks and make prediction [8]. R provides many useful abstractions, is an easy tool for data analysis, and has tons of useful packages. RStudio is a good IDE and interface for beginners to edit/write codes.

4. Software Specification

Software

Operating System	Windows 10 Professional
Programming Language used	Java Version 8
IDE	IntelliJ Latest version
Cloud service	Amazon AWS
Front-end Technology	Android SDK
Database	Amazon DynamoDB
Supported Algorithm	RStudio

Hardware

Architecture	64 Bit windows
CPU	Intel I7 9700K
RAM	32 GB 3200 MHz

5. System Design

1. Back-end Implementation:

1) Stock data fetching and storage

In order to present the stock information to the user, I first need to get the stock data. This step is done through the yahoo finance API. After I grab the data from the API, I need to store and analyze it. Therefore, I need to setup a database. Here I use DynamoDB to implement it.

In my app, I need not only display the current prices of the stocks, but also draw a chart of real-time price of today. Besides, I also provide prediction for the next day's stock close prices. To do this, I need to not only get the current price of the stock from the yahoo finance API, but also the closing prices and trading volume of the stocks over the past 30 days. They are sent to the data analysis section. For the stock selection, I chose the most famous big companies stocks. They are selected from 150 stocks in the S&P500-stock index, such as Apple, IBM, Amazon, etc.

For each stock in the database, I created a table called “realtime table”. It is used to record the real-time price of a stock. It has a five minutes delay. Each time the user clicks to see the price of a stock, it will call a function called “update” to update the price. The updated results will be reloaded into the corresponding table. After that, the updated data in the table is fed into the user interface via the AWS Android SDK and displayed to the user. I needed to achieve this deceptively simple function by consuming additional system resources. This is because I usually need data throughout the day for some other function, such as a real-time chart of the day, rather than the price at a certain point of the time. Therefore, it is worth maintaining a “real time table” to achieve this functionality.

Moreover, the price prediction feature is one of the features of this project. To do this, I didn't create 30 history tables for each stock to record their historical closing prices. Instead, I used the yahoo finance API to dynamically search for closing prices over the last 30 days. After getting the data from the API, I send it to the R programming analyzer. They are used to build mathematical models that predict stock prices. Because the data changes dynamically, the data is disposable, it is discarded after use each time. I don't need to create a static table to maintain them.

2) Forum implementation inside each stock:

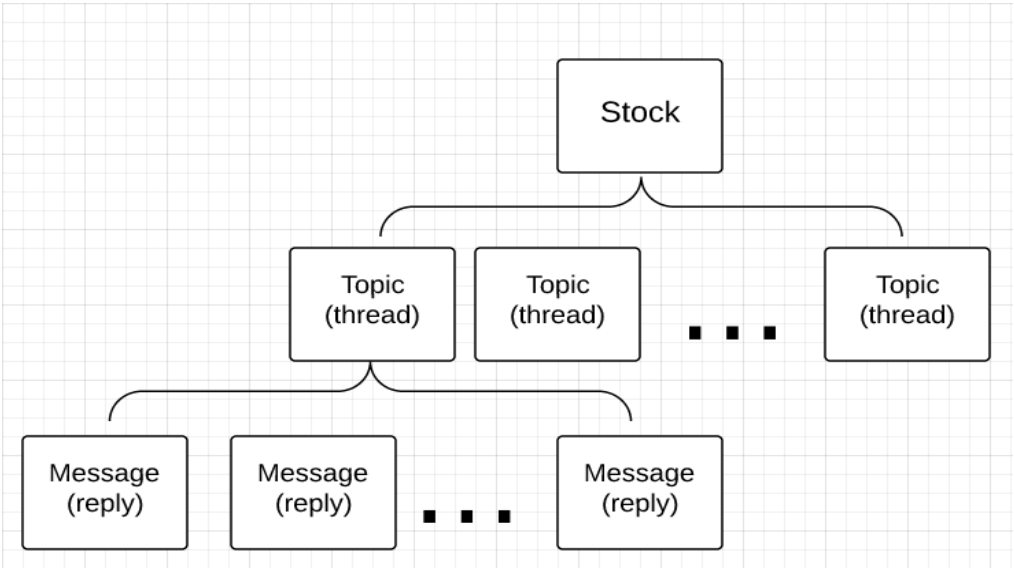


Figure 1: Forum schema behind each stock

Figure 1 shows the main structure of the stock forum. As you can see from the diagram, each stock is structurally a mini forum. Under mini forum, chat content related to the stock is embedded, and each chat topic is a thread. Under each chat topic thread, there are several child threads of reply threads.

Here, I'm still using DynamoDB to store each stock, the thread embedded in the stock, and the reply sub-threads. As shown in the figure, I defined a class in each layer of structure. Each class contains some required attributes and functions that need to be implemented.

Pseudo-structure of the three classes that constitutes a forum:

Stock:

```
 String stock_name; //one-to-one mapping to a known stock
 int number_of_threads; //number of children threads

 public Stock findStock(String stock_name) {...}
//find the instance in the database and return a copy for manipulation

 public addThreadToStock(String thread_title, String thread_text, String
poster_name) {...}
//add a thread to this stock, increase number_of_thread by 1 and recorded
as the thread_id. The number_of_threads is freshly fetched in this
function to avoid error during parallel manipulation

 public List<Thread> listThreads() {...}
 //query the database and list all children threads
```

There is a challenge here. The database is not allowed to store a list of objects. To implement the forum function, I had to scan the entire DynamoDB table. This is to look for the topic thread and the reply child thread under the stock. This step can be optimized. I do this by creating a dedicated thread table for each stock. However, due to the small amount of data acquired by my application, this method is not efficient in the use of space. I did not implement this method here. It is suitable for larger forums.

Thread:

```
String stock_name; //parent stock name
int thread_id;           //uniquely identify a thread with stock_name
String title;      //title of the thread
String text;       //body(message) of the thread
String usr;        //user that posted this thread
int number_of_relies;

public addReplyToThread(String reply_text, String poster_name, int
reply_target_floor) {...}
```

```
//add a reply to this thread, increase number_of_relies by 1 and recorded
as the reply_floor. The number_of_relies is freshly fetched in this
function to avoid error during parallel manipulation

public List<Reply> listReplies() {...}
//query the database and list all children replies
```

The two paragraphs of pseudo code above enumerate the structure of two classes. As a stock forum, it is important to eliminate misleading responses. Because our customers are novice investors, misleading responses can greatly influence their decisions. I used the thread.listreplies () method to sort replies to minimize the impact of misleading replies. Here's the strategy:

(1) the replies with the highest number of likes have the highest priority, if several replies have different likes.
(2) the latest reply has the highest priority, if several replies have the same number of likes.

In this way, I try my best to present useful replies to the user and lower the impact of useless and misleading one. Users can get effective information as quickly as possible.

2. Front-end implementation:

Because my ultimate goal is to develop a user-friendly Android app, the design of the front-end user interface is very important. Users want to get the information they need as easily as possible.

The main challenge here is how to integrate the functionality that Amazon AWS implemented in this project into the Android application. Because Android's main threads are usually dealing with user interface issues. It does not provide any Internet service itself. To do this, I need to implement multithreading through the AWS Android SDK to implement all the functions provided by AWS on an Android phone. The AWS Java SDK and the AWS Android SDK look very similar. Most of the APIs are the same. However, depending on jar size, number of methods, and memory usage, the AWS Android SDK is optimized for the Android platform. Although very similar, I cannot use the AWS Java SDK directly on Android.

Most of the time, the project depends on the third party library, the carrier may be jar, so, aar, if it is jar package that is easy to do, just drop it into the libs directory; So library is not simple, it only provides a few abi so library, then I need to choose according to the requirements, either all copies, or only keep a directory, otherwise it will report an error; An aar is a collection of code, resources (images, layouts, etc.), so libraries, and so on. My SDK requires users to inherit my Application, so I should provide a solution of multiple inheritance of it. After all, users may

not only use one SDK, which is troublesome when there are multiple SDKs with this requirement.

Java multiple inheritance, where the requirement is to inherit two classes, does not "inherit" method declarations on multiple interfaces by implementing interfaces. The solution is:
The proxy Application object is created by means of interface implementation + reflection, and it implements multiple inheritance of application indirectly.

Besides, value transmitting is necessary for each activity from front-end since the database need to interact with users in real-time. For example, the ID and password, which is typed by a user need to be transmitted to the database to identify the user, etc.

3. Supported Algorithm:

The implementation of this algorithm is mainly to predict the stock closing price of the next day. To do this, I used several different models. Their main sources of data are consistent. All of them use the known stock data to train the model and obtain the parameters required by the formula, so as to predict the stock price of the next day. In my project, I used the stock closing price, trading volume, opening price, high price and low price in the last 30 days as raw data for training.

(1) Linear regression

Linear regression is a regression analysis that uses the least square function called linear regression equation to model the relationship between one or more independent variables and dependent variables. This function is a linear combination of one or more model parameters called regression coefficients. The case where there is only one independent variable is called simple regression, and the case where there is more than one independent variable is called multiple regression. This, in turn, should be distinguished by multiple linear regressions predicted by multiple dependent variables, rather than a single scalar variable.

In linear regression, data is modeled using linear prediction functions, and unknown model parameters are estimated by data. These models are called linear models. The most common linear regression model is that the conditional mean of y given an X value is an affine function of X. Less generally, a linear regression model can be a median or some other quantile of the conditional distribution of y given X as a linear function of X. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of y for a given value of X, not on the joint probability distribution of X and y.

Linear regression models often use least-square approximation to fit, but they may also use other method to fitting, such as minimizing the fitting of defect in some other specifications (such as minimum absolute error regression), or in the bridge back to minimize the least square loss function of punishment. On the contrary, the least squares approximation can be used for fitting the nonlinear model. Linear regression is the main empirical tool in economics. For example, it is used to predict consumer spending, fixed investment spending, inventory investment, the purchase of a country's exports, import spending, requirements to hold liquid assets, labor demand, labor supply.

In this project, xi in formula (1) below represents the number of days backtracking from current day, and yi represents the stock-related information of that day (as described above). After the parameters are obtained through training, formula (2) is used to predict the stock price.

$$\hat{\beta}_j = \arg\min \sum_{i=1}^{n} (y_i - \beta_3 x_{3i} - \beta_2 x_{2i} - \beta_1 x_{1i} - \beta_0)^2 \quad (1)$$

$$y = \beta_3 x_3 + \beta_2 x_2 + \beta_1 x_1 + \beta_0 \quad (2)$$

(2) Random forest

Random forest refers to a classifier that uses multiple trees to train and predict samples. In machine learning, a random forest is a classifier that contains multiple decision trees, and its output category is determined by the mode of the category output from an individual tree.

Each tree is built according to the following algorithm:

N is used to represent the number of training use cases (samples), and M is used to represent the number of features.The number of input features m is used to determine the decision result of a node in the decision tree; Where m should be much less than m.

A training set (i.e., bootstrap sampling) was formed by sampling N times from N training cases (samples) with the method of sampling back, and the unselected use cases (samples) were used for prediction and error assessment.

For each node, m features are randomly selected, and the decision of each node in the decision tree is determined based on these features. According to these m characteristics, the optimal splitting mode is calculated.Each tree will grow intact without pruning, which may be adopted after the construction of a normal tree classifier.

Let me visualize the construction of the decision tree using the process of selecting the quantitative tool. Suppose we want to choose a good quantitative tool to help us better choose a stock.

Step 1: see if the data provided by the tool is very comprehensive.
Step 2: see if the API provided by the tool works and if it doesn't work, don't use it.
Step 3: check to see if the tool's backtracking process is reliable.
Step 4: see if the tool supports simulated trading. Backtracking only allows me to determine if the strategy has historically been useful.

In this way, the quantitative tools on the stock market are labeled as "use" and "don't use" by "whether the data is comprehensive", "whether the API is easy to use", "whether the back-test is reliable" and "whether the simulated transaction is supported". Therefore, after the stock hit a particular price, it will display whether to buy, hold or sell a price.

There are two aspects to build a random forest model: random selection of data and random selection of characteristics to be selected.

1) Random selection of data:

First, take the sample with the return from the original data set and construct the sub-data set. The data volume of the sub-data set is the same as that of the original data set. Elements in different sub-data sets can be repeated, as can elements in the same sub-data set. Second, the sub-data set is used to build the sub-decision tree, and this data is put into each sub-decision tree, and each sub-decision tree outputs a result. Finally, if new data are needed to obtain the classification results through the random forest, the output results of the random forest can be obtained by voting on the judgment results of the sub-decision tree. As shown in the following figure, suppose there are 3 sub-decision trees in the random forest, 2 sub-trees are classified as class A, and 1 sub-tree is classified as class B, then the classification result of the random forest is class A.

2) Random selection of characteristics to be selected

Similar to the random selection of data sets, each sub-tree splitting process in a random forest does not use all the selected features, but randomly selects certain features from all the selected features, and then selects the optimal feature from the randomly selected features. This can make the decision trees in the random forest different from each other, improve the diversity of the system, and thus improve the classification performance.

(3) Naive Bayes

The naive Bayes method is based on the bayes algorithm to simplify, that is, it is assumed that the attributes are mutually conditional independent when the target value is given. That is to say, no attribute variable has a large proportion for the decision result, and no attribute variable has a small proportion for the decision result.

Naive Bayesian classification (NBC) is based on Bayes' theorem and characteristics of hypothesis conditions between independent method, based on given training set, with key between independent as a premise, learning from the input to the output of joint probability distribution, and then based on the study to the model, the input x and make the output of the a posteriori probability maximum Y.

With sample data set D = {d1, d2, … dn}, the corresponding feature attribute set of sample data is X = {x1, x2, …, xn}. Class variable is Y = {y1, y2, … yn}, that is D, it can be divided into ym categories. Where, the prior probability P = P(Y) is independent and random, and the posterior probability Post = P(Y|X), is obtained by the naive Bayesian algorithm. The posterior probability can be calculated by the prior probability, evidence and conditional probability:

$$P(y_i|x_1, x_2, \cdots, x_d) = \frac{P(y_i)\prod_{j=1}^{d} P(x_j|y_i)}{\prod_{j=1}^{d} P(x_j)}$$

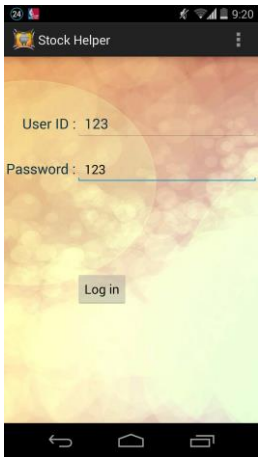
For each model, I backtrack 30 days from the latest stock price of the day. The function is recalculated daily. This ensures that the predicted results are relatively accurate.

After the error comparison of the prediction results of the three methods, I finally chose linear regression as the final method used in this project. Of course, there are problems with the predictions of all of these models relative to stock prices, a variable that is affected by many factors. This is because I am not a financial professional, which knows enough knowledge about the price trend of a stock. Moreover, severe condition and special event that has big impact on stock market is clearly not taken into account by these models. For example, COVID-19 this time has a greater impact on stock prices than any the company's fundamentals and financial condition. These outliers in the statistical analysis were not included in the prediction model. Improvements are needed in the future.

6. User Interface

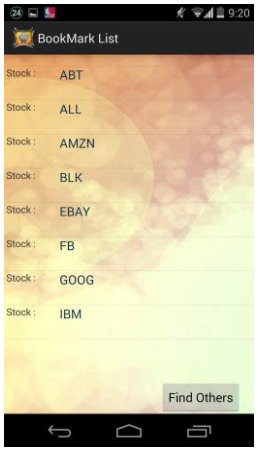
The following image shows the user interface in this project. As an Android app, it has various functions from logging in to browsing, posting comments and replying to comments. The following figures show all the visual interface to clients.

(1) on Page 1, the user logs in with his own account and password.



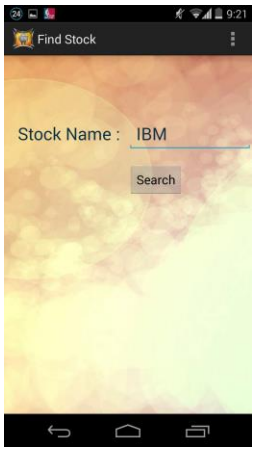
Page 1

(2) Page two displays the list of stocks that the user is concerned about. These stocks are added by users themselves, and they can add any stock they want to follow. This is the main page after login. This page is implemented through the ListView function of the Android SDK. This feature can quickly display a variety of lists.



Page 2

(3) Page 3. Click “Find Others” on Page 2, it will jump to Page 3. It's a search engine that has 150 stocks from the S&P 500 that are stored in my database. As long as the correct stock symbol is entered, the user can search the corresponding stock information here. If I want to expand the stock pool in the future, I just need to expand the corresponding back-end stock database.



Page 3

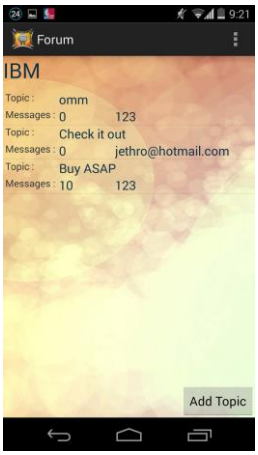
(4) Page 4. Click a certain stock, it will jump to the detailed information of the stock. The information contains the opening price of the day, the real-time price, the updated time, and the forecast of the closing price for tomorrow. The closing price prediction is based on the algorithm mentioned above. On this page, users can choose to click the "Add" or "Delete" button to edit the watchlist.

Page 4

(5) Page 5 shows the real-time graph of the stock from the opening of the day to the present.

Page 5

(6) Page 6. Click "Find Topic" on Page 4 to enter the mini forum of the stock. There are topics related to the stock from other users. At this page, users can post comments (Add Topic) themselves.



Page 6

(7) Page 7. Click a topic on Page 6, and you will enter the reply page related to the topic. It displays the details under the theme page. Notice here that I set up a vote function. It is used to prevent false information from being used maliciously. Here users can see other people's views on the stock. Buy or sell? I limit the length of each message to 20 words. The idea is to make it easier for newbies to get useful information and make decisions.

7. Social effect

This project is a stock market auxiliary analysis software based on mobile terminal. It is focused on investors who are new to the stock market. For these novice investors, the temptation of the stock market is great, their own investment experience is insufficient, the mature stock investment software is too complex for them. The meaning of this app is to help them better understand the stock market. Through simple interface, simple operation, it can achieve the tracking of the stock market. At the same time, due to the existence of the price prediction function, people can clearly feel the volatility of the stock market by the difference between the prediction price and the actual price. That could set off alarm bells for investors. Tell them to be cautious about their investments and that entering the market is risky. Be more rational with each investment you make.

8. Summary

To sum up, this project is a stock auxiliary analysis application based on android mobile terminal, which mainly realizes the following functions.

(1) Look for and add the share to the watchlist that users are interested in at any time
(2) View the opinions of other users who jointly follow the stock, and also express their own opinions.
(3) check the price forecast of the stocks concerned by users at any time.

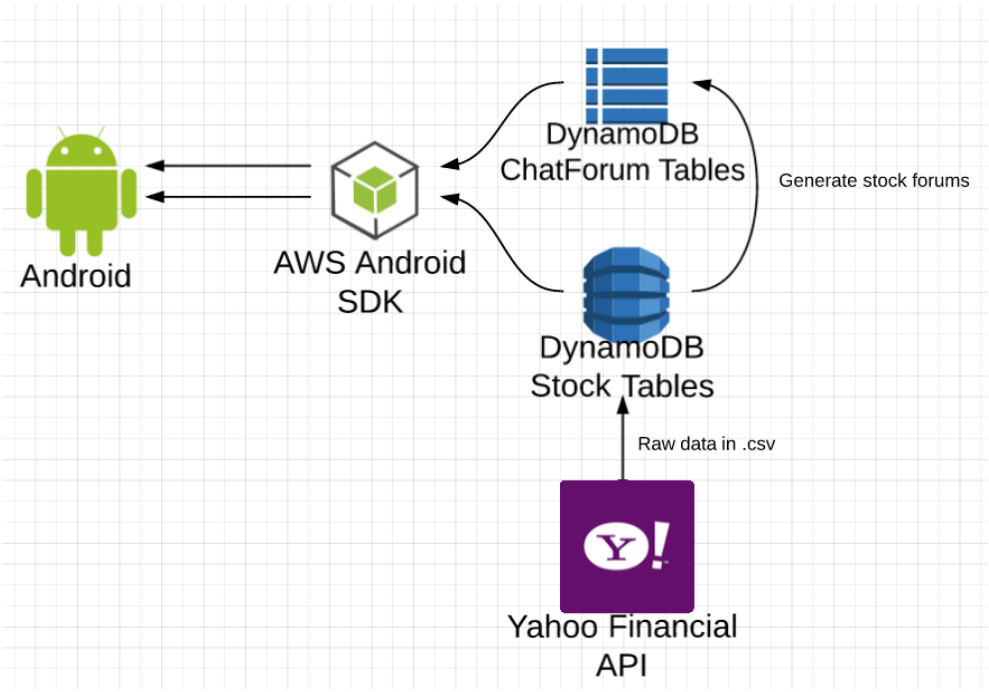


Figure 3: Blueprint of the application

The picture above is the frame of the project. As can be clearly seen from the figure above, the data is first read from the Yahoo Financial API into the DynamoDB database. Create various tables in the database, process and analyze them, and send the analyzed data and prediction information to the Android client through AWS Android SDK, and display them on the user interface.

The user forum and the stock price forecast are the innovative functions of this app, which can better help the rookie investors understand the uncertain situation of the stock market, warn investors from two aspects of data and people that investment should be cautious.

9. Future work

In order to become a real financial assistance software that can be put into the market, this project still has some areas for optimization and improvement. For example, real-time pricing is not really real-time, but has a five-minute delay because of the time consumed by front-end computing and back-end communication. And putting a lot of computing on Android phones not only consumes the phone's system resources, it also makes software performance unstable.

Moreover, severe condition and special event that has big impact on stock market is clearly not taken into account by these models. For example, COVID-19 this time has a greater impact on stock prices than any the company's fundamentals and financial condition. These outliers in the

statistical analysis were not included in the prediction model. Improvements are needed in the future.

10. References

[1] Ceyhun Ozgur, Sanjeev Jha, Elyse " Bennie, David Eugene Booth (2018). The Usage of R Programming in Finance and Banking Research. Journal of Accounting and Finance. DOI:10.33423/jaf.v18i3.412

[2] KEN KELLEY, KEKE LAI, AND PO-JU WU (2008). Using R for Data Analysis: A Best Practice for Research. Best Practices in Quantitative Methods, Chapter 34. DOI: https://dx.doi.org/10.4135/9781412995627.d40

[3] Rstudio Team (2011). RStudio IDE Features and RStudio desktop services.
Retrieved from https://rstudio.com/products/rstudio/features/

[4] William Leigh, Cheryl J. Frohlich etc (2007).Trading With a Stock Chart Heuristic. IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans 38(1):93-104

[5] Ignacio Bermudez, Stefano Traverso, Marco Mellia & Maurizio Munafò (2013). Exploring the cloud from passive measurements: The Amazon AWS case. IEEE INFOCOM, DOI: 10.1109/INFCOM.2013.6566769.

[6]Amazon AWS team (2013). Amazon DynamoDB Documentation. Retrieved from https://docs.aws.amazon.com/dynamodb/?id=docs_gateway

[7] Amazon AWS team (2013). Amazon Elastic Compute Cloud Documentation. Retrieved from https://docs.aws.amazon.com/ec2/?id=docs_gateway

[8] AP Felt, E Chin, S Hanna & D Song (2011). Android permissions demystified. Chicago, Illinois, USA. ISBN: 978-1-4503-0948-6 doi>10.1145/2046707.2046779.

[9] Android Developer Team (2009). Documentation for app developers: App data and files. Retrieved from https://developer.android.com/guide/topics/data/
Android Developer Team (2009). Documentation for app developers: User Interface & Navigation. Retrieved from https://developer.android.com/guide/topics/ui/

[10] Android Developer Team (2009). Documentation for app developers: Test apps on Android. Retrieved from https://developer.android.com/training/testing/

[11] RapidAPI Staff (2018). How To Use the Yahoo Finance API in 2019. Retrieved from https://blog.rapidapi.com/how-to-use-the-yahoo-finance-api/

[12] Abidatul Izzah, Yuita Arum Sari, Ratna Widyastuti & Toga Aldila Cinderatama (2017). Mobile app for stock prediction using Improved Multiple Linear Regression. 2017 International Conference on Sustainable Information Engineering and Technology (SIET), 978-1. DOI: 10.1109/SIET.2017.8304126.

11. Appendices

Testcase of my project related functions

Number of project tests: FJ001

Revision: No. 1

Author: Chris Xu

Test Conductor: Chris Xu

Date Conducted: 02/16/2020

Customer Representative: Bob Birth

Description: the purpose of this test is mainly to register, log in, log out, stock search and other related functions related to the program. The corresponding case codes are FFJ001, FFJ002, FFJ003.

Pre-Test Setup:

1. The whole program of my project should be running correctly, including front end android application and back end database services.

2. The Wi-Fi network is with good quality and cell-phone is in good operation status.

FFJ001.

User Action	Expected Results	Pass/Fail	Comment
1. An ID and password is registered by a new user	User id existed? Yes: return wrong user id.	Pass	

	No: register passed and the home page appears		
2. An ID and password was input by a existed user	User id existed? Yes: the home page appears No: return user not existed	Pass	Database will be searched to see if the user info is in it
3. The app is logged out by a user	Initial page is displayed again	Pass	
4. A stock code is clicked by a user on the home page	This particular stock home page is displayed	Pass	

FFJ002.

User Action	Expected Results	Pass/Fail	Comment
1. A user uses the search function by clicking find other stocks	Stock search page is displayed.	Pass	Only if the user inputs right stock code, will the result be displayed. Otherwise, error will return
2. A user uses bookmark function by clicking “Add” on the stock or search page	This particular stock is added to watchlists. It will be displayed on the home page when this user login.	Pass	Only if the user inputs right stock code, will the result be displayed. Otherwise, error will return
3. A user want to delete a stock from his/her watchlist	By clicking delete, this particular stock will be removed from the watchlist. It will not be	Pass	

	displayed on the home page of this user.		
4. A user want to see other's discussion about a stock by clicking "find the topic"		Fail	

FFJ003.

User Action	Expected Results	Pass/Fail	Comment
1. A user wants to see the historical price of a particular stock by clicking "history"	Stock history page is displayed	Pass	Only if the history data is stored in the database, will the result be displayed. Otherwise, it will show "no history"
2. A user wants to see the real-time curve of a particular stock by clicking "chart"	This particular stock's real-time chart will be displayed on the screen	Pass	There will be some delays, depending on the network quality and the android device performance