Stock/Crypto Market Advisor

Coding Project Final Report

Prepared By :- Maddy Wikstrom, Moody Samkary, and Ting-Shao Lee (Fall 2021)

Implemented by :- Akash Sunda, Afnan Waseem, and Naman Jain (Spring 2022)

April 2021

Table of Contents

I Project Description

1. Project Overview

In recent years a lot of people have started to invest their money into cryptocurrencies and stocks on the stock market. Most people who are getting into it are young people who do not know much about the stock market and are just trying to make money off investments.

We believe that an app like this could serve as training wheels in a sense to help these newcomers get into the investment world and advise them with their investments. The idea is to make an application or website that will give users recommendations on what to invest in.

See the project development final report for more details on the design and goals.

2. Project Domain

This project operates as a virtual advising application for people who have little to no experience with stocks. The application also allows users to communicate with each other and help each other out.

3. Relationship to other documents

This report discusses the prototype implementation of the final design of CS 440
Fall 2021 Group 15's final development project report.

4. Naming Conventions And Definitions

4a. Definitions of Key Terms

Forum-: A medium where ideas, issues, and views on certain aspects can be exchanged between members

Preferences-: The preferences a user can choose based on which they would get their recommendations

Search-: A search bar where users can search stocks based on their knowledge, the search text has to be stock tags.

4b. UML and Other Notions Used In This Document

This report follows Fowler's Version 2 OMG UML standard from [1].

4c. Data Dictionary for Any Included Models

User = username + password
Forum Post=Username+Title+Message
Preference=Username+PreferenceList
Recommendations=Username+SearchList

II Project Deliverables

5. First Release

In our first release we created a fully functioning sign-in/sign-up page, a fully functional preferences page whose data was stored locally, and a basic UI of the home page.

6. Second Release

In our second release we updated our UI to be more user friendly in both the home page and preferences page, integrated our API to display a non-formatted version of user recommendations, stored preferences of users in our firebase database, and created a non functioning forum page.

7. Comparison with Original Project Design Document

The original design was for an app that is functional for both cryptocurrencies and stocks, whereas we created an app that's solely focused on stocks. The original design also talked about a highly efficient stock predicting algorithm, which is not included in our application, the reason for this being that we had to use a different API which had certain limitations on it.

III Testing

8 Items to be Tested

PART 1- Login-: The login component that enables users to login and use.

PART 2-Sign up-: The signup component that enables users to create an account.
PART 3-Preference-: The storage and updating of preferences on the database.
PART 4-API-call-search-: The search results based on search input.
PART 5-API-call-recommendations-: The recommendations based on user preferences.
PART 6-Forum-database-: The storage and updating of forum posts.

9. Test Specifications

ID# TS1 - Authentication

Description-: For this test we need to ensure that a user is properly authenticated and able to sign up and log in

Items covered by this test: PART-1, PART-2

Requirements addressed by this test: The requirements are that we are allowed
to create accounts without duplicates and have the user stored on our data base
.
Intercase Dependencies: We need to ensure that our accounts are being created.

Test Procedures: Create an account and retrieve account from the database, log in with the same account to see if it authenticates properly.

Input Specification: We need to input the user credentials and we need to click submit.
Output Specifications: Page redirection.

Pass/Fail Criteria: After successfully signing up we should be able to see the preference page, on successfully logging in we should be able to see the home page.

ID# TS2 - Preferences

Description-: For this test we need to ensure that a users preferences are being stored and updated on the database.

Items covered by this test: PART-3.

Requirements addressed by this test: We need to ensure that we can create
Accounts. We need to ensure that a user is able to click and submit his preferences

.
Intercase Dependencies: We need to ensure that we are passing the account
Authentication test.

Test Procedures: Create a new account and add your preferences. Then sign out and log in with that account and update your preferences.

Input Specification: We need to select our preferences (at least 3) and we need to click submit.
Output Specifications: Page redirection.

Pass/Fail Criteria: After successfully choosing the preferences the home page will display recommendations based on the preferences you have chosen.

ID# TS3 - API

Description-: For this test we need to ensure that the application displays proper stock recommendations based on the users search input or preferences

Items covered by this test: PART-4, PART-5

Requirements addressed by this test: We need to ensure that the user has already picked his preferences.
.

Intercase Dependencies: We need to ensure that we are passing the account
Authentication and preferences test.

Test Procedures: Log into an account and update your preferences. Use the search field to search for stock recommendations based on stock tag names.

Input Specification: We need to select our preferences (at least 3) and we need to click submit. We need to input our stock tag name in the search field and click submit.

Output Specifications: Page updating.

Pass/Fail Criteria: If the API is successful then the recommendations on the home page will be updated based on the stock preferences or search tag name.

ID# TS4 - Forum

Description-: For this test we need to ensure that a user is able to post and see forum posts.

Items covered by this test: PART-6

Requirements addressed by this test: We need to ensure that a user account is created
.
Intercase Dependencies: We need to ensure that we are passing the account
Authentication test.

Test Procedures: Log into an account and go start a new forum, post a title and message on the forum, then log into another account and open that forum and reply to that message..

Input Specification: We need to input the forum name, title and message from the first account in the forum page. From the second account we need to input a message in the forum page.

Output Specifications: Page updating.

Pass/Fail Criteria: After successfully creating a new forum the user will be able to see the forum in the available forums. When the second user clicks that forum he will be able to see the title and message from the first user.

10. Test Results

ID# TR1 - All Tests Run

Date(s) of Execution: We tested it every Thursday for 2 weeks.

Staff conducting tests:The staff who conducted this test were Afnan Waseem, Akash Sunda and Naman Jain
Expected Results: That our tests would run correctly and efficiently

Actual Results: We were able to see that every test has passed.

Test Status: We passed all of our tests that were stated. We haven't seen any test
cases that have brought our program to a stop.

IV Inspection

12. Items to be Inspected

1. Code by - Afnan Waseem
   Inspected by - Akash Sunda, Naman Jain

```
private void jsonParse(){
    JsonObjectRequest request= new JsonObjectRequest(Request.Method.GET, url,  jsonRequest: null, new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {

            tv.setText(response.toString());
            try {
                JSONObject fin=response.getJSONObject("finance");
                System.out.println(fin.toString());
                JSONArray res=fin.getJSONArray( name: "result");
                JSONObject our= (JSONObject) res.get(0);
                int count= (int) our.get("count");
                Hashtable<String, String>[] l=new Hashtable[5];
                JSONArray quo= (JSONArray) our.get("quotes");
                for(int i=0;i<count;i++){
                    Hashtable<String, String> my_dict = new Hashtable<String, String>();
                    JSONObject q= (JSONObject) quo.get(i);
                    String name=q.getString( name: "shortName");
                    String regularPrince=q.getString( name: "regularMarketPrice");
                    String symbol=q.getString( name: "symbol");
                    my_dict.put("name",name);
                    my_dict.put("rp",regularPrince);
                    my_dict.put("s",symbol);
                    l[i]=my_dict;
                }
```

2. Code by - Akash Sunda
   Inspected by - Afnan Waseem, Naman Jain

```
public View.OnClickListener doneListener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (pref == "" || pref_List.get(0) == "" || pref_List.get(1) == "" || pref_List.get(2) == "") {
        }
        else {

            Map<String, Object> dataToSave = new HashMap<~>();
            dataToSave.put(EMAIL_KEY, userEmail);
            dataToSave.put(PREFERENCE_TYPE_KEY, pref);
            dataToSave.put(PREFERENCE1_KEY, pref_List.get(0));
            dataToSave.put(PREFERENCE2_KEY, pref_List.get(1));
            dataToSave.put(PREFERENCE3_KEY, pref_List.get(2));
            mDocRef.set(dataToSave).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) { Log.d( tag: "Tag",  msg: "Preferences Saved"); }
            });

            startActivity(new Intent(getApplicationContext(),HomePage.class));
        }

    }
};
```

3. Code by - Naman Jain
   Inspected by - Afnan Waseem, Afnan Waseem

```
mDocRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            DocumentSnapshot document = task.getResult();
            if (document.exists()) {
                Map<String, Object> re=document.getData();

                System.out.println( "DocumentSnapshot data: " + re.get("title"));
                title =  re.get("title").toString();
                question = re.get("question").toString();
                response = re.get("response").toString();

                tv.setText(title);

                tv3.setText(question);

                if(response.equals("")) {
                    tv2.setText("Responses:" + "\n" +response);
                }else{
                    tv2.setText(response);
                }

            } else {
                System.out.println( "No such document");
            }
        } else {
            System.out.println( "get failed with : "+ task.getException());
        }
    }
});
```

13. Inspection Procedure

For the purpose of inspection, we followed the checklist provided by the proandrioddev website (derived from Michaela Greiler's Code Review Checklist). Our team used to meet once a week on Saturdays over Zoom to discuss the progress and roadblocks that we were facing. We also used a discord channel to communicate throughout the week. All the members used to test their code before the meeting so we can save each others time from going over the same cases again. We would share our results through discord on a text document. During the meeting, we used to go through each others code to find possible optimatizations, edge case error, bugs, runtime errors and the time code took during execution. Some part of codes were inspected by all teammates together (outside the meeting time) as it contained code which concerned us all. For example, database updates and data extraction from the database.

14. Inspection Results

Inspection of API parse function - This code was written by Afnan Waseem and inspected by Akash Sunda and Naman Jain. This part of the code gets a json object from the Yahoo API. Then we are extracting the required data from that JSON object like name of the stock, symbol of the stock and the price of the stock. We went through many test cases for this function as there are some stocks which may not have a part of the information on the API. So through some testing we figured out ways in which it can be avoided. For that purpose, we made some filter code beforehand. We also included try and catch blocks for unseen exceptions.

Inspection of Database update function - This code was written by Akash Sunda and inspected by Afnan Waseem and Naman Jain. This part of the code saves the users preferences for the types of stocks they want in their recommendation. They have an option of type of trade (long term and short term) and 10 different sectors of stock market. The database saved the data based on particular user email address as the identifier. If their is already an instance of it, the database simply updates the entry rather then creating a new one. For this functions, we tested to check if their are not any duplicate entries in the database and the data is updating perfectly for preexisting users. The function also checked that the user have the required number of choices (3) to generate the recommendations.

Inspection of Database getter function - This function was written by Naman Jain and inspected by Akash Sunda and Afnan Waseem. This function is responsible for getting data from database regarding the forum questions. This database saves the data takes the question number as a token. This function retrieves title, question and preexisting responses to that question. When a user clicks on a particular question on the forum page, all this data is extracted from database and shown to the user. We to test this rigorously as their was problem with storing the data to local variable on the same thread. Also, we had to make sure that new response from a user are not overwriting their previous ones.

V Recommendations and Conclusion

The testing we have done on this application is satisfactory as we checked all the bugs and edge case errors we could think of occurring. Some of the parts of the code were hard and proved to be challenge but our teamwork got us through. The API function was one of the most challenging part as we had to try multiple API to get the one which works

with the platform which we were using (Andriod Studio). Even then, getting the data from the json object wasn't easy. One other hard part was the data extraction from the database. It was a creating a separate thread instead of storing the data on the required thread. We eventually got through all the errors. As the end of the term, our team is really happy with the prototype we developed.

VI Project Issues

15 Open Issues

Currently, there is no way of differentiating spam posts or false information from the really helpful ones. This could be harmful to new investors who don't have a great deal of information regarding trading. Also in the given time, we were only able to develop an android application but in the future, we will like to expand the application to IOS and PC.

Another issue we found was that there was limited flexibility in the endpoints of the results of API. The results for a given query were very limited. In the future, we might consider using a new API.

16 Waiting Room

One of the features that we wanted to add to the product but weren't due to the lack of time was giving the ability to users to invest in crypto, the stock market right through the application. With this feature users could have simply opened the app, seen the recommendations, and invested in the stock - crypto market simply within a few clicks on the application. Another feature that we wanted to add to the application is to make the app an ever-evolving platform that will learn and improve its recommendations, using user's activities like user searches, investments, profits-losses, etc.

17 Ideas for Solutions

We recommend that we allow users to upvote or downvote the posts and then let the application organize the posts accordingly. With this, the most beneficial post will be displayed on the top and the least helpful post on the

bottom. Another recommendation we have is that the application is expanded to IOS, Windows as well as PC to increase the user base.

18 Project Retrospective

Our final prototype really delivered on our expectations and worked exactly how we wanted it to. But there were some big hurdles along the way that could have been avoided through more planning and organization. During our integration of api with our application we wasted a few weeks, as the suggested apis in the development report weren't compatible with java and had to waste few weeks testing new apis. Furthermore the functions for traversing the api in java were out of date and now no longer compatible. This again took us a week to resolve. But I believe that this issue could have been avoided if we had planned and researched a bit better.

One more suggestion that I have is that we could have collaborated more in our application. In the beginning of this project we divided the application in small parts that were handled by each member individually and if that member faced any problems, it was discussed in the group. But I believe that if we would have discussed our portion of the code a bit more we could have come with new, unconventional solutions or methods that could have further enhanced the application.

VII Glossary

Forum: Which forum page does the user want to access.

Question: A question asked by a user related to crypto/stock market.

Response: It is reply to the question posted on the forum.

Recommendations: Stock/Crypto suggestions generated according to user's preferences using the yahoo api.

Preferences: What kind of industry does the user want to trade in.

VIII References / Bibliography

1. Wikstrom, Maddy et al. Crypto/Stock Market Advisor. Software Engineering Development Project Report. CS 440 Fall 2021 Group 15, December 2021.
2. Java Platform SE 7, https://docs.oracle.com/javase/7/docs/api/.