PREDICTING ESG SCORES: A SENTIMENT ANALYSIS PROGRAM UTILIZING SCRAPER API, GOOGLE FIREBASE, AND FLUTTER

YongQin Zeng, Roy Chun

Woodbridge high school, Meadowbrook, Irvine, CA 92604

Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

I made this program so I can predict and deliver ESG scores of various companies to people [1]. I conducted an experiment to test out the accuracy of my sentiment determination system for posts and got about half correct. Three important systems in my program are the posting system, the authentication system, and the user interface. I used Scraper API to retrieve the posts l need for the posting system [2]. An alternative API is SmartProxy. I used Google Firebase for my authentication system [3]. Another one I could have used is Parse. I used Flutter to build the user interface system. An alternative is React Native.

KEYWORDS

ESG Scores, Sentiment Analysis, Scraper API, Flutter UI

1. INTRODUCTION

The motivation behind creating this program stemmed from the challenge of predicting a stock's future ESG (Environmental, Social, and Governance) score based on public sentiment surrounding the company [4]. My goal was to develop a tool that could help people make informed decisions about how a stock or company might perform in the future.

To achieve this, I built a posting system using the Scrape API to gather information from X and display it on my app [5]. An alternative API I considered was Smart Proxy, which supports platforms like X, TikTok, and Instagram [15]. For data storage, I utilized Google Firebase as my database, incorporating its authentication system for user security. While Firebase offers robust options, I also considered using Parse, which provides a simpler, though more limited, authentication process.

The app's user interface was developed with Flutter, a UI toolkit that uses Dart, a language based on C. An alternative to Flutter is React Native, which is based on JavaScript.

My application leverages an API and machine learning to predict changes in a company's ESG score [6]. It collects posts from X to analyze public sentiment toward a stock or company, categorizing each post as positive, negative, or neutral. The machine learning model then predicts future ESG scores based on this data.

The purpose of my experiment was to evaluate the accuracy of my sentiment analysis. I selected 20 posts discussing major companies, manually identified their sentiments, and compared these with the results generated by my program. The experiment successfully demonstrated the app's capability in accurately determining sentiment, offering valuable insights into its effectiveness.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Information

Question: What does your program post information about?
Our program posts information about a stock such as its name, ticker symbol, price ESG score, ESG score prediction, public sentiment, etc.
Question: How does your program post information?
Our application posts information about each individual stock on a single page, which you can access by looking up the stock and clicking its button.
Question: How does your program get that information?
Our program utilizes an API which gets posts from X to calculate public sentiment on a stock/company then predict the future ESG score using machine learning. We got the company names and ESG scores from refinitiv.com.

2.2. List of stocks

Question: Why did you choose to create a search bar?
We chose to make a search bar because it would be easier to look up whichever stock you're interested in instead of scrolling through a long list of stocks.
Question: Why did you make a list of stocks?
We made a list of stocks that you can scroll through on the app's home page. Users can browse through this or they can use the search bar to look for specific stocks.
Question: What does your list of stocks look like?
The list is composed of many rows, each representing a stock/company. Each row has the company logo, company name, and the stock ticker symbol. Clicking on a row will direct the user to that stock/company's page, which will display more detailed information about it.

2.3. Authentication

Question: How does your authentication work?
When our program needs to access our data in Firebase, it sends a specific key that confirms our authenticity. If the key is incorrect, Firebase will deny our request to gain access to the database.
Question: Is your authentication secure?
Anyone who can get the key can access our database. We can turn off editing permissions but they could still gain access to the data if they had the key.

Question: Why was Firebase chosen for your database solution?
We chose to utilize Firebase because it offers up to 1 GB of free storage, which made this an accessible database option.

3. SOLUTION

When you first open the application, you will see the home page, which consists of a list of stocks and their predicted Environmental Social Governance (ESG) scores [7]. There is a search button on the upper right. Using it, you can look up any stock you want. Picking a stock will send you to a page with information about it, including name, ticker symbol, price, ESG score, predicted ESG score, etc.

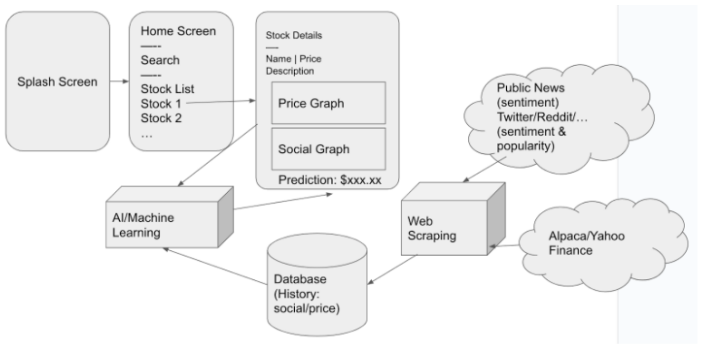
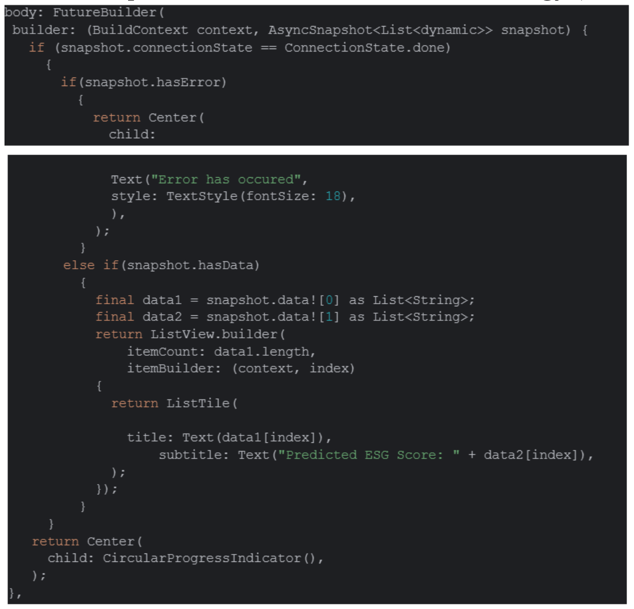


Figure 1. Overview of the solution

Our program has a posting system, which displays information about a stock. It relies on API's to retrieve posts made on X regarding the stock and uses machine learning to find public sentiment and make predictions on the future of that stock.

Figure 2. Screenshot of detail page

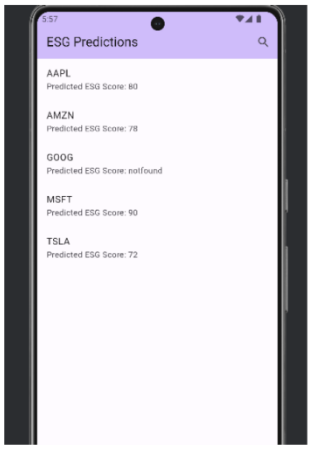


```
body: FutureBuilder(
 builder: (BuildContext context, AsyncSnapshot<List<dynamic>> snapshot) {
   if (snapshot.connectionState == ConnectionState.done)
     {
       if(snapshot.hasError)
         {
           return Center(
             child:
             Text("Error has occured",
             style: TextStyle(fontSize: 18),
             ),
           );
         }
       else if(snapshot.hasData)
         {
           final data1 = snapshot.data![0] as List<String>;
           final data2 = snapshot.data![1] as List<String>;
           return ListView.builder(
               itemCount: data1.length,
               itemBuilder: (context, index)
           {
             return ListTile(

               title: Text(data1[index]),
                   subtitle: Text("Predicted ESG Score: " + data2[index]),
             );
           });
         }
     }
   return Center(
     child: CircularProgressIndicator(),
   );
},
```

Figure 3. Screenshot of code 1

The first line creates the body. The next line creates the page/widget that will display the stock's information. After that, the program checks whether a connection to our Flask server has been established [8]. If not, the program prints a line saying: “Error has occurred". Else if there is a connection, the program sorts the data into lists. Then it builds the items on the page. While the program is loading, a circular loading icon will be displayed.

Our program has a user interface system that allows users to easily pick whichever stock/company they want. Users can use a search bar to find stocks and click a button to head to a page displaying information about that stock.



5:57

ESG Predictions

AAPL
Predicted ESG Score: 80

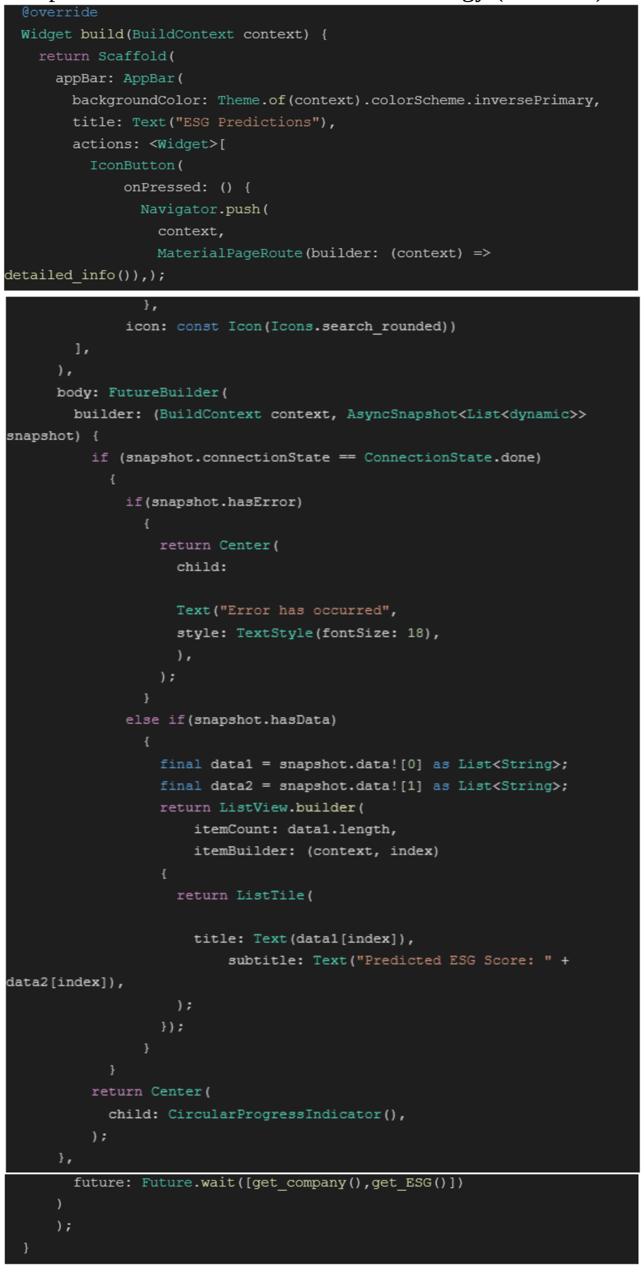
AMZN
Predicted ESG Score: 78

GOOG
Predicted ESG Score: notfound

MSFT
Predicted ESG Score: 90

TSLA
Predicted ESG Score: 72

Figure 4. Screenshot of ESG Predictions

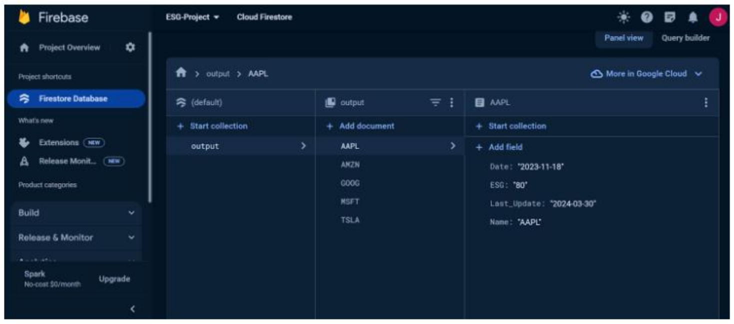


```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text("ESG Predictions"),
        actions: <Widget>[
          IconButton(
              onPressed: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) =>
detailed_info()),);
                },
              icon: const Icon(Icons.search_rounded))
        ],
      ),
      body: FutureBuilder(
        builder: (BuildContext context, AsyncSnapshot<List<dynamic>>
snapshot) {
          if (snapshot.connectionState == ConnectionState.done)
            {
              if(snapshot.hasError)
                {
                  return Center(
                    child:

                    Text("Error has occurred",
                    style: TextStyle(fontSize: 18),
                    ),
                  );
                }
              else if(snapshot.hasData)
                {
                  final data1 = snapshot.data![0] as List<String>;
                  final data2 = snapshot.data![1] as List<String>;
                  return ListView.builder(
                      itemCount: data1.length,
                      itemBuilder: (context, index)
                  {
                    return ListTile(

                      title: Text(data1[index]),
                          subtitle: Text("Predicted ESG Score: " +
data2[index]),
                    );
                  });
                }
            }
          return Center(
            child: CircularProgressIndicator(),
          );
      },
        future: Future.wait([get_company(),get_ESG()])
      )
      );
  }
```

Figure 5. Screenshot of code 2

The user interface for the app was developed in Flutter using the Dart language and completely default libraries [9]. This UI was built using Android Studio as a development environment. In the home_page.dart file, methods are defined for button functionality. The Widget class is used multiple times to define the details of the elements on the page, including background color, title, and positioning of the buttons. On each stock tile, the predicted ESG score is displayed as text. CircularProgressIndicator is used to present a loading wheel to the user in the event of loading.

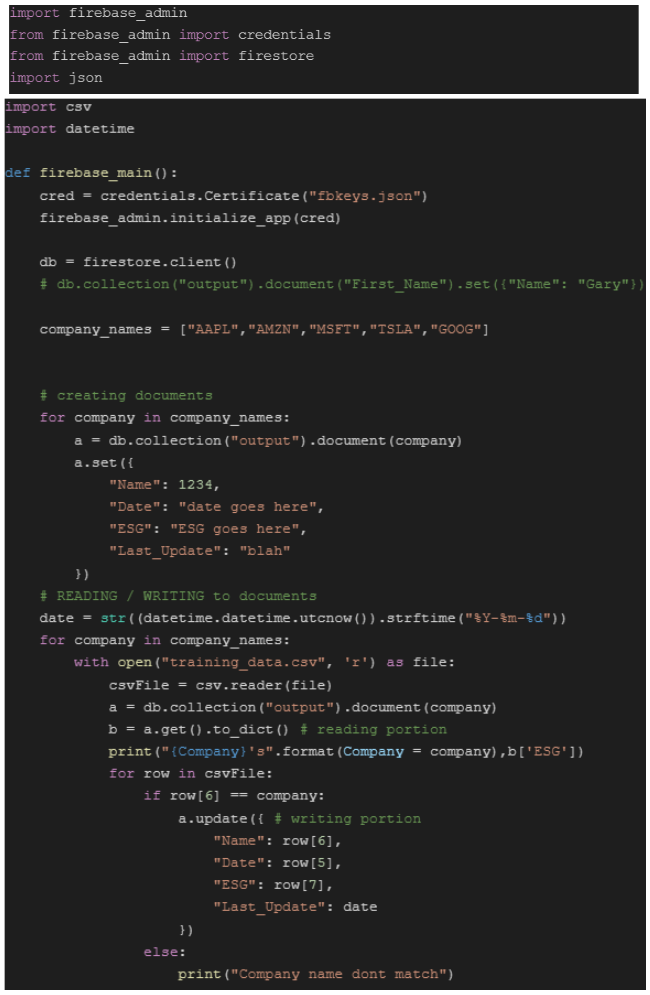
We have authentication within our program. When retrieving information from our database in Firebase, we need to provide a specific key in order to access it. Our program does this whenever it communicates with Firebase.



```
output
AAPL
AMZN
GOOG
MSFT
TSLA
```

```
Date: "2023-11-18"
ESG: "80"
Last_Update: "2024-03-30"
Name: "AAPL"
```

Figure 6. Screenshot of the firebase



```python
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore
import json
```

```python
import csv
import datetime

def firebase_main():
    cred = credentials.Certificate("fbkeys.json")
    firebase_admin.initialize_app(cred)

    db = firestore.client()
    # db.collection("output").document("First_Name").set({"Name": "Gary"})

    company_names = ["AAPL","AMZN","MSFT","TSLA","GOOG"]


    # creating documents
    for company in company_names:
        a = db.collection("output").document(company)
        a.set({
            "Name": 1234,
            "Date": "date goes here",
            "ESG": "ESG goes here",
            "Last_Update": "blah"
        })
    # READING / WRITING to documents
    date = str((datetime.datetime.utcnow()).strftime("%Y-%m-%d"))
    for company in company_names:
        with open("training_data.csv", 'r') as file:
            csvFile = csv.reader(file)
            a = db.collection("output").document(company)
            b = a.get().to_dict() # reading portion
            print("{Company}'s".format(Company = company),b['ESG'])
            for row in csvFile:
                if row[6] == company:
                    a.update({ # writing portion
                        "Name": row[6],
                        "Date": row[5],
                        "ESG": row[7],
                        "Last_Update": date
                    })
                else:
                    print("Company name dont match")
```

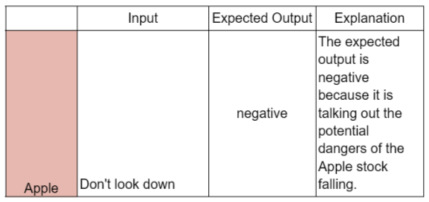
Figure 7. Screenshot of code 3

Google Firebase is the database my program uses to store data. Relevant authentication keys are stored in fbkeys.json, and are used to authenticate at the top of each file that accesses the database. Two files use the authentication for Firebase [10]. The file firebase.py is used for writing to Firebase; it sends the processed data from scrap.py to Firebase. The file flask_main.py retrieves the processed data from Firebase using a Flask server, which is then displayed to the user. The supporting file scrap.py is used to retrieve the data from X, process the data, and predict the ESG score.

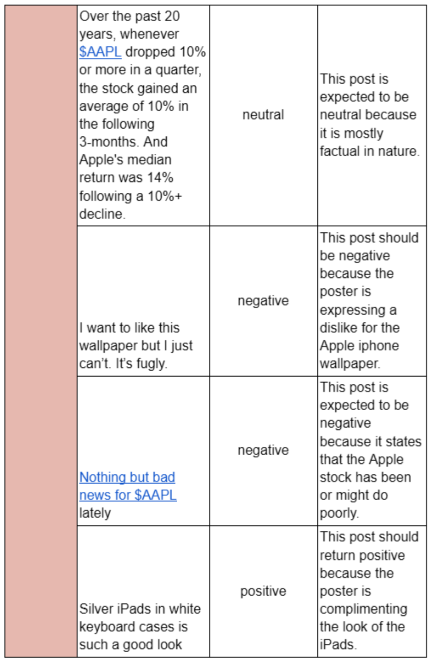
4. EXPERIMENT

I conducted an experiment to test the accuracy of the sentiment determination system in my program. l gathered a set of 20 posts from X (formerly Twitter) to use as inputs for my sentiment determination system. Then l created a chart with the expected output foreach of these posts. Lastly, l ran them through my code and recorded the outputs.

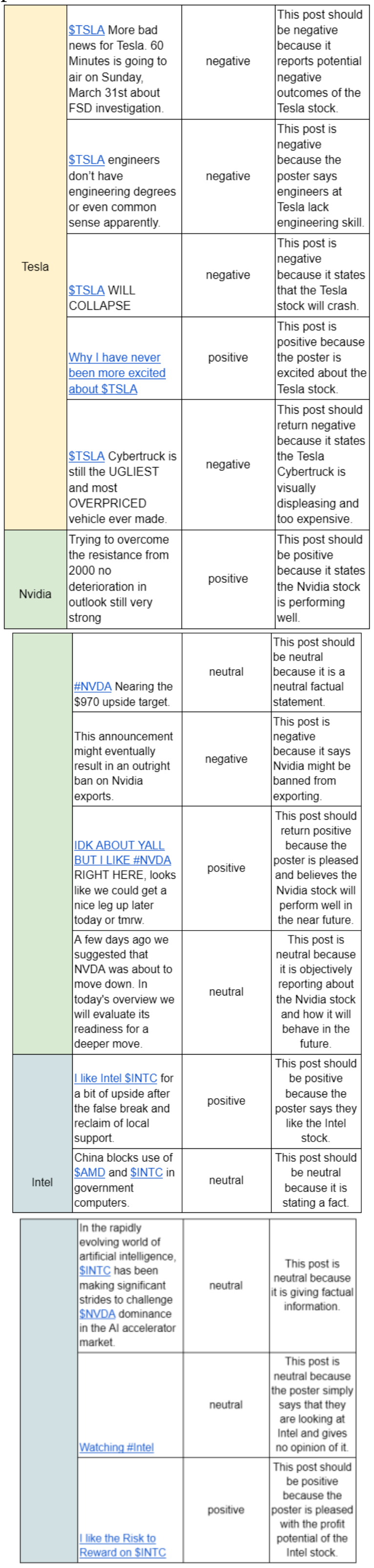
These 20 inputs are important because they are posts that my program needs to determine the sentiment of. It is important that these sentiments are predicted accurately because it allows the user to have an accurate understanding of what the public sentiments are for each stock.



	Input	Expected Output	Explanation
Apple	Don't look down	negative	The expected output is negative because it is talking out the potential dangers of the Apple stock falling.

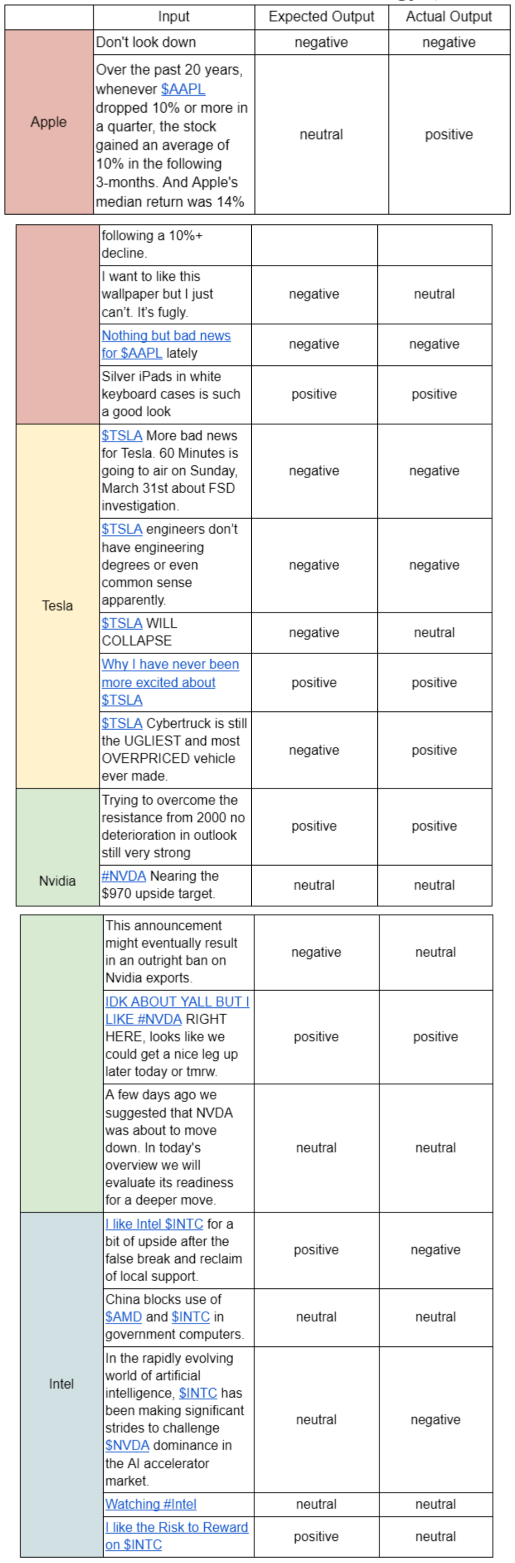


	Over the past 20 years, whenever $AAPL dropped 10% or more in a quarter, the stock gained an average of 10% in the following 3-months. And Apple's median return was 14% following a 10%+ decline.	neutral	This post is expected to be neutral because it is mostly factual in nature.
	I want to like this wallpaper but I just can't. It's fugly.	negative	This post should be negative because the poster is expressing a dislike for the Apple iphone wallpaper.
	Nothing but bad news for $AAPL lately	negative	This post is expected to be negative because it states that the Apple stock has been or might do poorly.
	Silver iPads in white keyboard cases is such a good look	positive	This post should return positive because the poster is complimenting the look of the iPads.



	$TSLA More bad news for Tesla. 60 Minutes is going to air on Sunday, March 31st about FSD investigation.	negative	This post should be negative because it reports potential negative outcomes of the Tesla stock.
	$TSLA engineers don't have engineering degrees or even common sense apparently.	negative	This post is negative because the poster says engineers at Tesla lack engineering skill.
Tesla	$TSLA WILL COLLAPSE	negative	This post is negative because it states that the Tesla stock will crash.
	Why I have never been more excited about $TSLA	positive	This post is positive because the poster is excited about the Tesla stock.
	$TSLA Cybertruck is still the UGLIEST and most OVERPRICED vehicle ever made.	negative	This post should return negative because it states the Tesla Cybertruck is visually displeasing and too expensive.
Nvidia	Trying to overcome the resistance from 2000 no deterioration in outlook still very strong	positive	This post should be positive because it states the Nvidia stock is performing well.
	#NVDA Nearing the $970 upside target.	neutral	This post should be neutral because it is a neutral factual statement.
	This announcement might eventually result in an outright ban on Nvidia exports.	negative	This post is negative because it says Nvidia might be banned from exporting.
	IDK ABOUT YALL BUT I LIKE #NVDA RIGHT HERE, looks like we could get a nice leg up later today or tmrw.	positive	This post should return positive because the poster is pleased and believes the Nvidia stock will perform well in the near future.
	A few days ago we suggested that NVDA was about to move down. In today's overview we will evaluate its readiness for a deeper move.	neutral	This post is neutral because it is objectively reporting about the Nvidia stock and how it will behave in the future.
	I like Intel $INTC for a bit of upside after the false break and reclaim of local support.	positive	This post should be positive because the poster says they like the Intel stock.
Intel	China blocks use of $AMD and $INTC in government computers.	neutral	This post should be neutral because it is stating a fact.
	In the rapidly evolving world of artificial intelligence, $INTC has been making significant strides to challenge $NVDA dominance in the AI accelerator market.	neutral	This post is neutral because it is giving factual information.
	Watching #Intel	neutral	This post is neutral because the poster simply says that they are looking at Intel and gives no opinion of it.
	I like the Risk to Reward on $INTC	positive	This post should be positive because the poster is pleased with the profit potential of the Intel stock.

Figure 8. Comparison 1



	Input	Expected Output	Actual Output
Apple	Don't look down	negative	negative
	Over the past 20 years, whenever $AAPL dropped 10% or more in a quarter, the stock gained an average of 10% in the following 3-months. And Apple's median return was 14% following a 10%+ decline.	neutral	positive
	I want to like this wallpaper but I just can't. It's fugly.	negative	neutral
	Nothing but bad news for $AAPL lately	negative	negative
	Silver iPads in white keyboard cases is such a good look	positive	positive
Tesla	$TSLA More bad news for Tesla. 60 Minutes is going to air on Sunday, March 31st about FSD investigation.	negative	negative
	$TSLA engineers don't have engineering degrees or even common sense apparently.	negative	negative
	$TSLA WILL COLLAPSE	negative	neutral
	Why I have never been more excited about $TSLA	positive	positive
	$TSLA Cybertruck is still the UGLIEST and most OVERPRICED vehicle ever made.	negative	positive
Nvidia	Trying to overcome the resistance from 2000 no deterioration in outlook still very strong	positive	positive
	#NVDA Nearing the $970 upside target.	neutral	neutral
	This announcement might eventually result in an outright ban on Nvidia exports.	negative	neutral
	IDK ABOUT YALL BUT I LIKE #NVDA RIGHT HERE, looks like we could get a nice leg up later today or tmrw.	positive	positive
	A few days ago we suggested that NVDA was about to move down. In today's overview we will evaluate its readiness for a deeper move.	neutral	neutral
Intel	I like Intel $INTC for a bit of upside after the false break and reclaim of local support.	positive	negative
	China blocks use of $AMD and $INTC in government computers.	neutral	neutral
	In the rapidly evolving world of artificial intelligence, $INTC has been making significant strides to challenge $NVDA dominance in the AI accelerator market.	neutral	negative
	Watching #Intel	neutral	neutral
	I like the Risk to Reward on $INTC	positive	neutral

Figure 9. Comparison 2

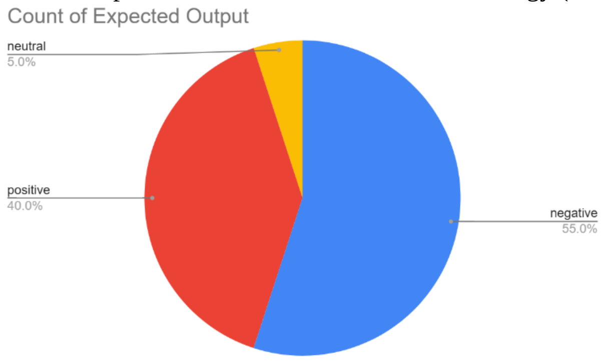


Figure 10. Count of expected output

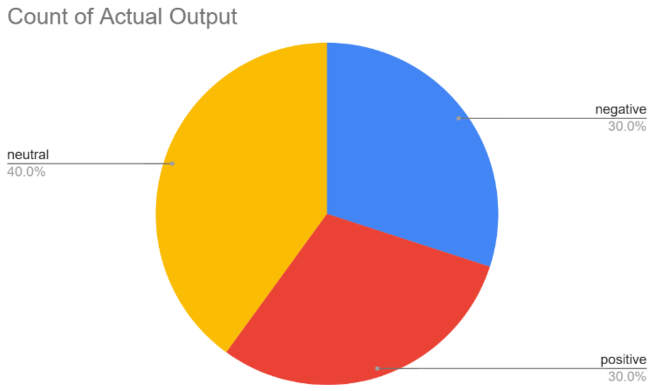


Figure 11. Count of actual output

My system got 8 out of 20 inputs incorrect. The overall accuracy of my system is 60%. Most of the mistakes were real mistakes but a few of them were worded ambiguously so that it was not easy to say what sentiment it had.

5. Related Work

Smart Proxy is a social media scraping API [11]. The creator of this project is Noah Drucker. This API works on Instagram, Tiktok, and X. It has more targets than my program, which can only perform web scraping on X. If my program were to use Smart Proxy, it would be able to get stock information from Instagram, Tiktok, or X. One of Smart Proxy's examples from the API documentation utilizes JSON code to make an API request individually. Our program finds lots of different posts that contain the desired keywords and makes decisions based on that information.

Parse is a back-end platform that is based on the database solutions MongoDB and PostgreSQL [12]. Its authentication method is more minimal than the Firebase authentication my program uses. To use Parse, on the client side, the only values that are needed for configuration are applicationId, clientKey, server, and (optionally)fileUploadController. Meanwhile, Firebase requires 11 variables in order to authenticate. Also, Firebase has a larger variety of authentication options, such as email/password, Google, Facebook, and X sign-in. Parse has email/password authentication but not as many options as Firebase.

React Native is an application developer that is similar to Flutter, which l used for my program. Both React Native and Flutter work on iOS and Android [13]. They are both free and open source. React Native uses a coding language called Typescript that is based on Javascript while Flutter has its own language named Dart that is based on C. Flutter has the same process for iOS and Android while React Native has different processes for them. This makes Flutter an easier choice for developing applications that can work both on iOS and Android. Typescript is more accessible than Dart for developing on a single platform due to its similarity to Javascript. React Native also has a wider variety of available libraries than Flutter does.

6. CONCLUSIONS

A limitation to our project is the amount of ESG score data available to us. ESG scores are only updated quarterly or annually. Also, financially viable APIs that have past data are scarce. One of the things we could improve on is making our authentication process more secure. The key we use to access our Firebase database is included inside the code of the app, which might not be very secure since anyone who could get it can access the database [14]. However, we can configure our Firebase database to be more secure for production releases.

One way we can expand our program in the future is to implement more APIs to obtain stock and company data from other sources. We might add an API named Finnhub or another similar one to acquire information regarding stocks/companies to display on the detailed pages in our posting system.

REFERENCES

[1] Clément, Alexandre, Élisabeth Robinot, and Léo Trespeuch. "The use of ESG scores in academic literature: a systematic literature review." Journal of Enterprising Communities: People and Places in the Global Economy (2023).
[2] Glez-Peña, Daniel, et al. "Web scraping technologies in an API world." Briefings in bioinformatics 15.5 (2014): 788-797.
[3] Chatterjee, Nilanjan, et al. "Real-time communication application based on android using Google firebase." Int. J. Adv. Res. Comput. Sci. Manag. Stud 6.4 (2018).
[4] Drempetic, Samuel, Christian Klein, and Bernhard Zwergel. "The influence of firm size on the ESG score: Corporate sustainability ratings under review." Journal of business ethics 167.2 (2020): 333-360.
[5] Dongo, Irvin, et al. "Web scraping versus twitter API: a comparison for a credibility analysis." Proceedings of the 22nd international conference on information integration and web-based applications & services. 2020.
[6] Mahesh, Batta. "Machine learning algorithms-a review." International Journal of Science and Research (IJSR).[Internet] 9.1 (2020): 381-386.
[7] Lagasio, Valentina, and Nicola Cucari. "Corporate governance and environmental social governance disclosure: A meta-analytical review." Corporate social responsibility and environmental management 26.4 (2019): 701-711.
[8] Relan, Kunal, and Kunal Relan. "Beginning with flask." Building REST APIs with Flask: Create Python Web Services with MySQL (2019): 1-26.
[9] Arb, Ghusoon Idan, and Kadhum Al-Majdi. "A freights status management system based on Dart and Flutter programming language." Journal of Physics: Conference Series. Vol. 1530. No. 1. IOP Publishing, 2020.
[10] Chougale, Pankaj, et al. "Firebase-overview and usage." International Research Journal of Modernization in Engineering Technology and Science 3.12 (2021): 1178-1183.

[11] Motaei, Eghbal, and Tarek Ganat. "Smart proxy models art and future directions in the oil and gas industry: A review." Geoenergy Science and Engineering 227 (2023): 211918.
[12] Makris, Antonios, et al. "MongoDB Vs PostgreSQL: A comparative study on performance aspects." GeoInformatica 25 (2021): 243-268.
[13] Wu, Wenhao. "React Native vs Flutter, Cross-platforms mobile application frameworks." (2018).
[14] Moroney, Laurence, and Laurence Moroney. "The firebase realtime database." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform (2017): 51-71.
[15] Alenezi, Faisal, and Shahab Mohaghegh. "A data-driven smart proxy model for a comprehensive reservoir simulation." 2016 4th Saudi International Conference on Information Technology (Big Data Analysis)(KACSTIT). IEEE, 2016.