Team 6: Hannah Nguyen, Puskar Dev, Safi Ullah

Stock Analysis App:

Employing AI for real-time stock market analysis

We are planning to use react-native on the front-end and Python for the backend. We will use python libraries like keras, scikit-learn and NLTK to build the model. We will use data like stock info, news/posts through APIs, specific ones are listed below.

Why?

With the rise of online brokerage like Robinhood, Webull many people are into buying/selling/shorting stocks. There are many people, especially in social media, so called “influencers" who give stock advice to people and people blindly follow their advice and most of the time lose their money.

The goal is to provide the users with stock suggestions that are based on real-time analysis using AI powered models.

Approach for building the app

Data Points

Tweets about a company
Stock Data (Date, Open price, close price, high, low, volume/float)

Get the data from respective APIs, perform data preprocessing.

Sentiment Analysis will be a classification model.
It classifies whether the overall content is positive/negative.

For a given company on a given date,
We will compile all these twitter posts and run sentiment analysis (**Using Flair Library**)
We will take the majority vote of sentiments for that stock on that day.
Assign a numerical value or insight which is useful to the user.

Stock Price prediction

This will be a regression model (i.e. it will give a prediction on the price of stock for a date).
For prediction, Use a regression model from **(sci-kit learn library**) or Use **LSTM neural net** to build the model. (**keras library on python).**

Use Yahoo finance API to get stock data for a given company,
Data includes all sorts of financial metadata (dividends, financial reports, etc) extracted from Yahoo finance API.
Useful data for a stock-> Date, Open price, close price, high, low, volume/float

Data Analysis

Iteration 1: Sentiment Analysis on tweets pulled from Twitter API.

Twitter API will be used for searching tweets, and improving requests.

We will be using Twitter Developer Portal to monitor the use of API.
The parameters for the API call will be stock ticker (eg. TSLA for Tesla) and language as english. When the response from API is received in json format. Each tweet returned by API contains three fields ‘id_str’, ‘created_at’ and ‘full_text’. A get function will be written to extract these.

We will be using python to do the sentiment analysis on this data we just collected. Specifically, we will use flair, a simple natural language processing (NLP) library developed and open-sourced by Zalando Research. Flair has a pre-trained sentiment analysis model. It is a powerful model, which uses pytorch and keras tensorflow.

Data preprocessing is needed since tweets may contain excessive whitespaces, web addresses and Twitter users which is not really important. We will use regular expressions to identify them and remove from the tweets. We will use these modified tweets to predict the sentiment. The prediction results in a *score* to get the confidence/probability (0-1), and a *value* (NEGATIVE/POSITIVE) . One API request gives back 100 tweets, we will try to improve our requests in further iterations.

```
PS C:\Users\Puskar\Desktop\sentiment> python twitter.py
2021-03-01 22:33:44.860819: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.
dll not found
2021-03-01 22:33:44.862459: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Enter the stock ticker or company name: TSLA
2021-03-01 22:33:59,537 loading file C:\Users\Puskar\.flair\models\sentiment-en-mix-distillbert_3.1.pt
                    created_at                 id                                                text  probability sentiment
0   Tue Mar 02 04:33:53 +0000 2021  1366607669587443713  RT : Warning🚨 to everyone out there... this is...     0.999601  NEGATIVE
1   Tue Mar 02 04:33:42 +0000 2021  1366607622305050626  RT : So the Buffalo plant will now be a FSD de...     0.999843  NEGATIVE
2   Tue Mar 02 04:33:35 +0000 2021  1366607592949248001  You didn't think I'd really end the day red di...     0.862924  NEGATIVE
3   Tue Mar 02 04:33:16 +0000 2021  1366607513232244738  RT : I don’t have a dog in the $RKT fight... U...     0.997127  NEGATIVE
4   Tue Mar 02 04:32:49 +0000 2021  1366607402980814855  RT : Best chatroom, Up to 2200% success rate !...     0.997023  POSITIVE
..                             ...                  ...                                               ...          ...       ...
95  Tue Mar 02 04:14:32 +0000 2021  1366602799274856457       I wouldn’t dwell further until $LI $TSLA...     0.999969  NEGATIVE
96  Tue Mar 02 04:14:25 +0000 2021  1366602772037193729  RT : 🚨TRADE ALERT🚨  and  purchased a whopping ...     0.999829  NEGATIVE
97  Tue Mar 02 04:14:21 +0000 2021  1366602756023279619   I wanted to but Tesla back in 13 but I bought...     0.995284  POSITIVE
98  Tue Mar 02 04:14:08 +0000 2021  1366602700201238532    funny thing is that high capital intensity i...     0.722895  NEGATIVE
99  Tue Mar 02 04:14:02 +0000 2021  1366602673668067329  Well if today wasn't the day the tomorrow will...     0.975520  POSITIVE

[100 rows x 5 columns]
The result from sentiment analysis is total count: 100 positive count: 28 negative count: 72
```

Iteration 2: No Change in Data Analysis implemented.

Plan for Iteration 2

The detailed delivery schedule is provided at the end of this document. For the upcoming iteration, we plan to implement the following features:

1) Add a login/registration feature to our app. We plan to use Firebase to store the information of our users and Firebase authentication will allow users to sign in to our account by entering email address and password.
2) We will connect the front-end with the backend. We plan to use django or flask rest framework in python to serve the API backend with react-native for the front-end.
3) We will add a disclaimer and a user agreement contract.

Plan for Iteration 3

1) Add signup and present the user agreement form to the user before creating an account.
2) Have front-end and back-end connected.
3) Add a feature of customizable watch-lists users can add-remove stocks from the list.
4) Perform analysis on data from Yahoo finance.

Final Goal

We will combine results from sentiment analysis and the prediction model and provide useful feedback for a given stock and deliver it to the user.

Customers/Users and Feedback

Any active or non-active traders or anyone interested in investing.
Users can be recruited by appealing to traders who are having trouble deciding on what stock to invest in or whether or not to keep or sell a stock that they are currently investing in. Users can also be people who are new to investing.

We can attract users through our UI, since the app's UI is similar to Robinhood and easy to navigate.

Google form for Customer feedback: https://forms.gle/nAjV5nnZGZk8gn766

Feedback	Our Solution	Status
On the homepage, there is confusion between the different lists, "My Stocks" and "Watch List"	When the user opens the app for the first time, there will only be only one list. And from there, they can create and add on their own lists and customize them however they like.	In-Progress
The wording in the "Based on our analysis" section in the stock information page should be chosen carefully, since it is illegal to suggest or give advice on selling stocks.	We will say "Our predictions say that this stock will…(go up or down)" instead of giving advice on what to do with their stock.	Complete
For users who are new to investing in stocks, most information on the app may not make any sense to them since they are technical.	We will add the necessary information in the "HELP" section which will help the new-user to navigate through the app.	In-Progress

Competitors

Competitors:

- Tradingview.com
    - This competitor differs from us because this is just a website, not a mobile app.
    - Stock analysis is not their main feature, so their focus is on all their other features too. Stock analysis is our main feature, so we will have more focus on making sure our analysis is advanced and our accuracy is the best it can be at all times.
    - This website is not easy to navigate, as there is so much information on the screen which can overwhelm a user, while our app will be more user-friendly.
    - This website has some locked features that can only be accessed if you pay. Our app will be free to use.

We are different from our competitors because we will be performing a sentiment analysis on the stocks. We also have an easily navigable UI, which attracts users who are familiar with Robinhood or are newer to trading and don't know how to start. Our stock analysis is also through a cross-platform mobile app, rather than a web application. This way, users can use our app on the go and can easily and quickly navigate to our app.

Projected Risks

1) Challenges faced on data access and preprocessing through various APIs.
    - **Risk Exposure:**
        - Probability: **Pr** = 20%
            - Our team has experience in accessing APIs and already have an idea of what APIs to use, however, we have never used these specific APIs before. For this reason, our probability of this risk would be around 20%.
        - Effect: **E** = -41 (Lose 41 points / get a failing grade)
            - Accessing APIs is very important for our app and failing to do so could give us a failing grade, so the effect would be -41 points from our grade.
        - Risk Exposure: **RE** = **Pr** x **E** = .20 x -41 = **-8.2**
    - We can mitigate this risk by trying our best to access and process the data from the APIs in advance. This way, we are prepared to make any changes before any iteration deadlines.
2) Most stock APIs are not free, cost for the project can be an issue.
    - **Risk Exposure:**
        - Probability: **Pr** = 5%
        - Effect: **E** = -21 (Lose 21 points / get a B)
        - Risk Exposure: **RE** = **Pr** x **E** = .05 x -21 = **-1.05**
    - We can mitigate this risk by researching the cost of different APIs we could use and finding which ones would be the best to use in advance.This way, we are

prepared to make any changes, such as finding different APIs or altering any features in our app, before any iteration deadlines.

3) Achieving a satisfactory test accuracy on the model can be a challenge.
    - **Risk Exposure:**
        - Probability: **Pr** = 30%
            - It could be a challenge to find a good measure of accuracy within our code, so there is a 30% chance of us possibly having a bad accuracy.
        - Effect: **E** = -31 (Lose 31 points / get a C)
            - Accuracy is hard to measure, but we could get as close an accuracy as possible. Getting an accuracy that is not as good could get us a C, so our effect is -31 points from our grade.
        - Risk Exposure: **RE** = **Pr** x **E** = .30 x -31 = **-9.3**
    - We can mitigate this risk by all working together on our code to plan and develop the best accuracy. We can also designate developers to testing as many cases as we can as we go.
4) Falling behind in app development, since we have little to no prior knowledge of developing in React Native or working with APIs
    - **Risk Exposure:**
        - Probability: **Pr** = 20%
            - Our team is learning React Native, researching the APIs ahead of time and have tried starting development in advance, so the probability of us falling behind at this rate is 20%.
        - Effect: **E** = -31 (Lose 31 points / get a C)
            - The scenario of our team falling behind in app development due to not having knowledge in React Native or working with APIs could result in us having a mediocre front-end design and smaller use in APIs. This could still result in a functional app, so the effect could be -31 points from our grade.
        - Risk Exposure: **RE** = **Pr** x **E** = .20 x -31 = **-6.2**
    - We can mitigate this risk by searching for ways to learn react native or searching for how to work with API's ahead of time. We'll need to dedicate more time to learning and starting off our project when we are first planning the app, then we'll set specific deadlines so that we are aware of what we need to prioritize and what our plans are for the rest of the iteration.
5) Running out of time to complete our app, since we have to complete this project in a limited time frame
    - **Risk Exposure:**
        - Probability: **Pr** = 40%
            - The chance of us running out of time to complete our app is somewhat likely, as we only have a semester to complete this project and have to balance this on top of our other classes.
        - Effect: **E** = -31 (Lose 31 points / get a C)

        - Us running out of time to complete our project could end with us having unfinished features within our app. We would most likely have other functioning features, so we would pass, but would not get the best grade. (-31 points off our grade)
    - Risk Exposure: **RE = Pr** x **E =** .40 x -31 **= -12.4**
- We can mitigate this risk by planning future deadlines for high-priority implementations and setting our own personal deadlines for specific features. We will categorize our planned features by high priority and low priority, so that we can finish our high-priority features first and implement our low-priority features when we have time. This way, we can have a functioning project with our intended features if we end up running out of time at the end of the semester.

Delivery Plan and Schedule:

Task/Milestone Description	Anticipated Start Date	Anticipated End Date	Status	Comments
Prepare Inception Deliverable	01/29/2021	02/01/2021	Completed	
Prepare prototype sketch for app UI	01/29/2021	02/01/2021	Completed	Sketched in OneNote
Research APIs for app use	01/29/2021	02/01/2021	Completed	
Inception Presentation and Deliverables Due	**02/01/2021**	**02/01/2021**	**Completed**	Presented prototype sketch and project idea.
Access API data through python in backend	02/01/2021	03/01/2021	Completed	Implemented sentiment analysis on Twitter API. We may add APIs later as we go as needed.
Implement basic front-end code for app UI	02/01/2021	03/01/2021	Completed	Completed most basic UI so far in React-native.

Implement user registration and login	02/11/2021	03/23/2021	In-progress	Using firebase
Connect front-end with back-end code	02/11/2021	04/05/2021	In-progress	Use django/flask rest framework to create a server and host.
Iteration 1 Presentation and Deliverables Due	**03/01/2021**	**03/01/2021**	**Completed**	Will mainly present front-end design so far and some of the code we have so far in the back-end (like how we're accessing analyzing data through Twitter API).
Implement a help page	03/01/2021	04/05/2021	In-progress	Still in progress because we haven't figured out all the specific analysis yet for the app (Will figure out after we connect front-end and back-end)
Add feature where users can create/customize new lists in the homepage	03/09/2021	04/05/2021	In-progress	Users will only have an empty "Stock List" on their homepage when they first create an account. Under this, users can create their own lists. (Customize the list name, add any stocks, etc.)

Iteration 2 Presentation and Deliverables Due	**03/22/2021**	**03/22/2021**	**In-progress**	Present user registration and login? Present any changes in back-end code?
Incorporate data from back-end (stock information from Yahoo Finance API) into front-end.	03/22/2021	04/05/2021	TBC	This date may change, depending on when the front-end and back-end is connected.
Iteration 3 Presentation and Deliverables Due	**04/05/2021**	**04/05/2021**	**TBC (To Be Completed)**	
Implement stock analysis in back-end code.	03/22/2021	04/26/2021	TBC	This date may change, depending on when the front-end and back-end is connected.
Final Presentation and Deliverables Due	**04/26/2021**	**04/26/2021**	**TBC (To Be Completed)**	

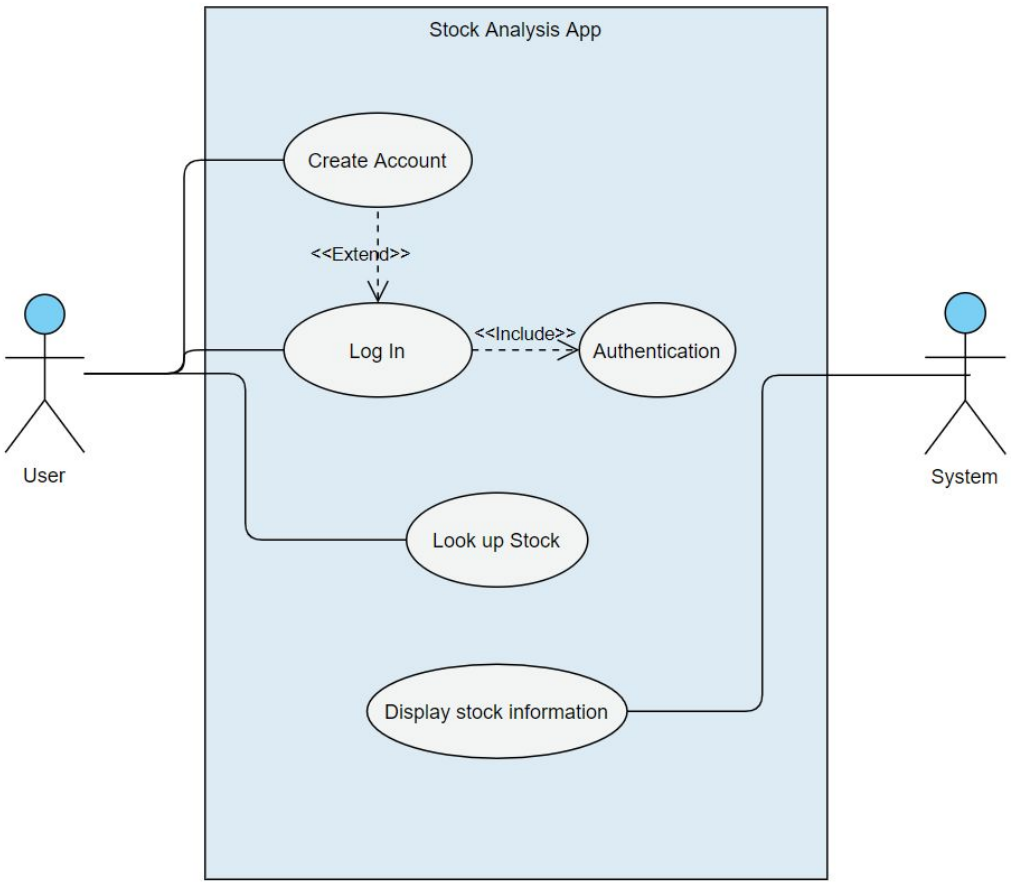
How to Compile and Run App through Expo:

1. Download Expo Go Client from App Store or Google Play Store on the device or emulator being used to run the app.
2. Use the command prompt to cd into the “StockAnalysisApp_ReactNative” folder.

3. Type “expo start” into the command line. This will open the Expo Developer Tools browser and start the Metro Bundler.
4. Run app differently depending on what device is being used:
    a. If wanting to run on an android or iOS emulator, open the emulator and click on either the “Run on Android device/emulator” or “Run on iOS simulator” button in the Metro Bundler browser.
    b. If wanting to run on a physical device, use the device’s camera to scan the QR code in the bottom left of the Metro Bundler browser. Scanning this will open up the app within the Expo Go Client app on the physical device.

App can also be run using this link: https://expo.io/@hannahhn/projects/stockanalysisapp

Use Case



Test Cases

Feature: User Login

Test Case #	Test Case Description	Test Data	Expected Result	Actual result	Pass/Fail
1	Check response when valid email and password is entered	Email: example@gmail.com Password: Stockapp123	Login should be successful	Login was successful	Pass
2	Check response when invalid email and response s entered	Email: abc@gmail.com Password: yahoo567	Login should not be successful	Login was not successful	Pass

Data Structures

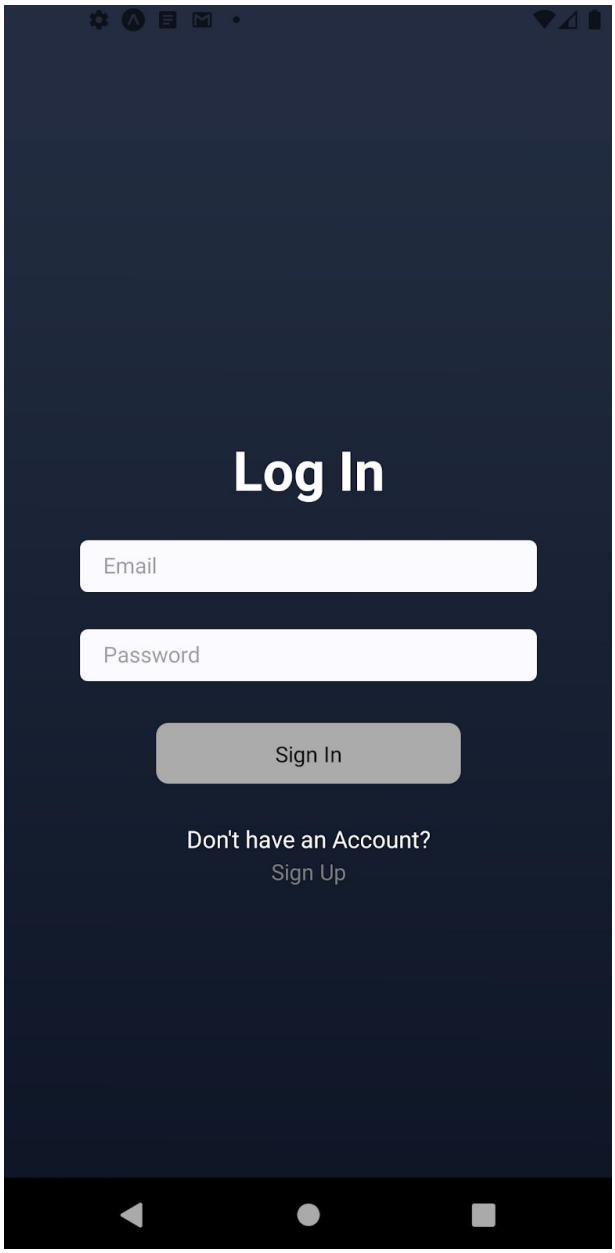
Backend - Python

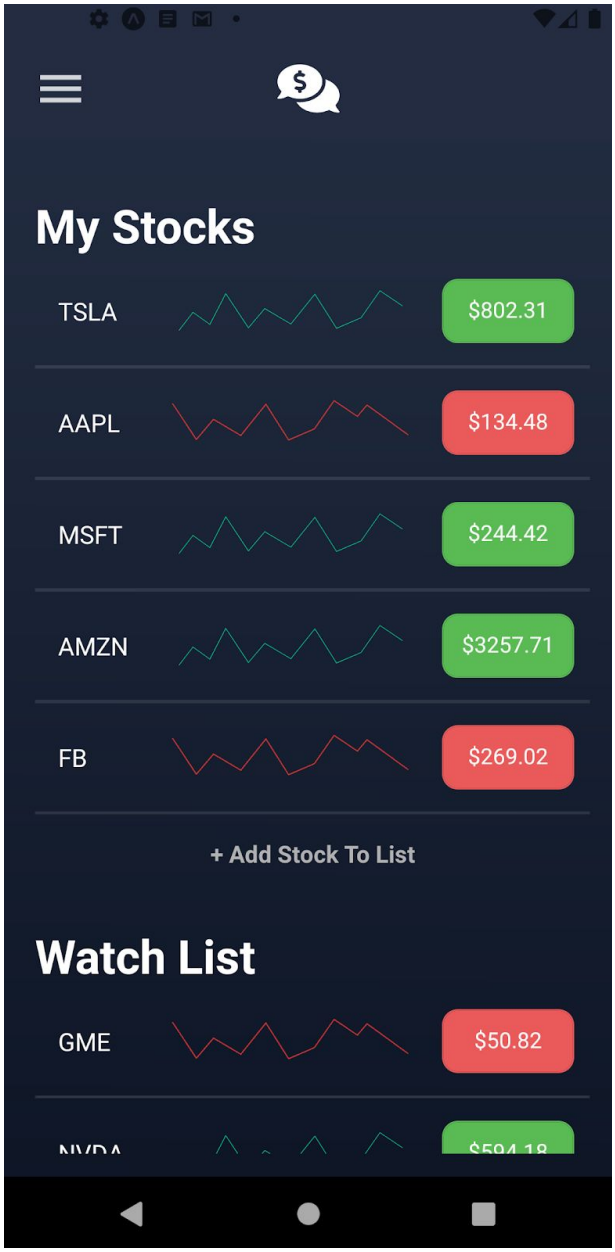
twitter.py

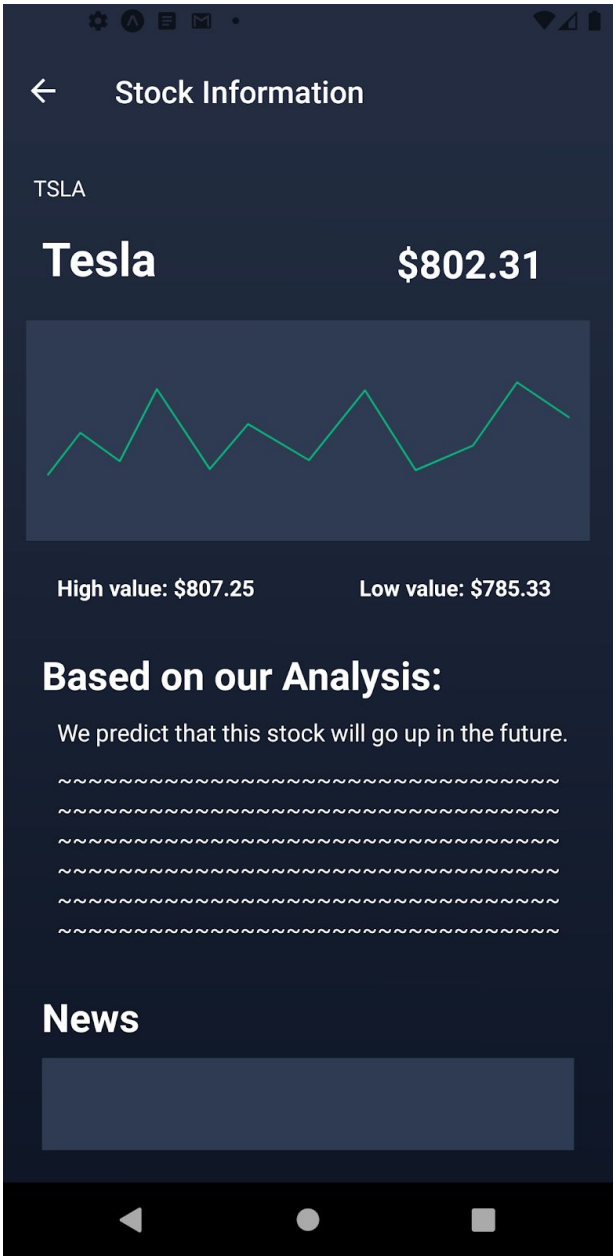
Variable/Type Name	Data Structure Type	Description
params	Python dictionary	Parameters for the twitter API call
response	JSON object	Data/response of the API call is stored here
data	Python Dictionary	Stores individual tweet data. The keys are: id -> tweet's ID created_at -> tweet's date text -> tweet's full text
count_tweets	Type int	For counting total numbers of tweets accessed
df	pandas.DataFrame (2-dimensional labeled data structure with columns of potentially different types)	Stores all the tweet data. Also, includes columns for sentiment type and probability on each tweet.
sentence	Python Dictionary	Stores text as a string type and prediction in a list named *labels* with keys *score* and *value*.

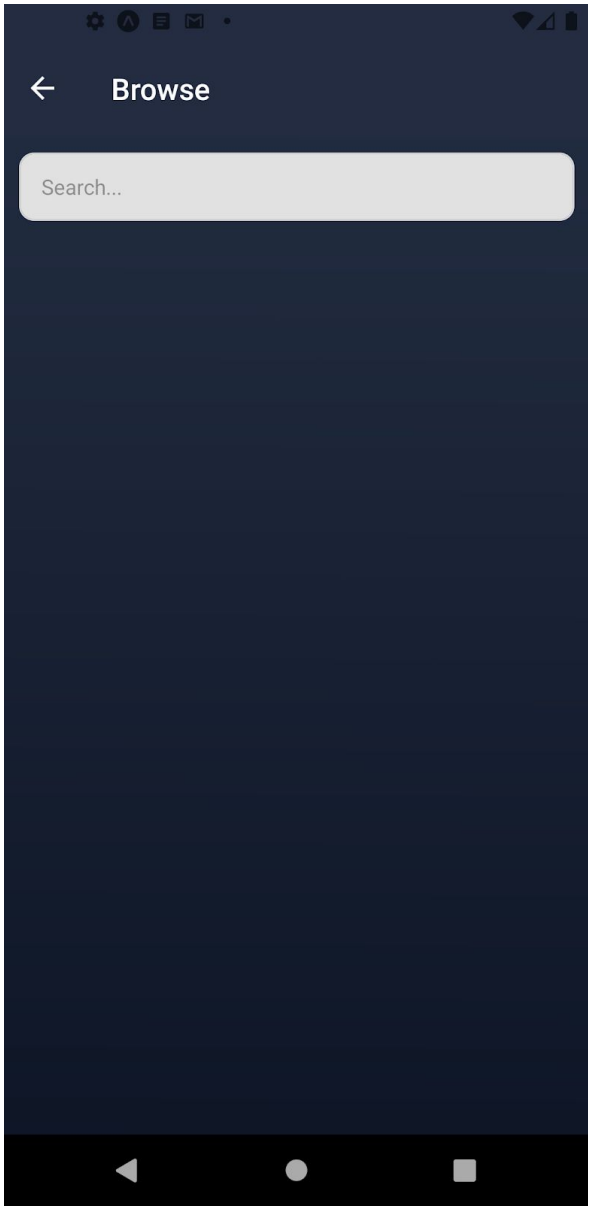
User Interface 2.0

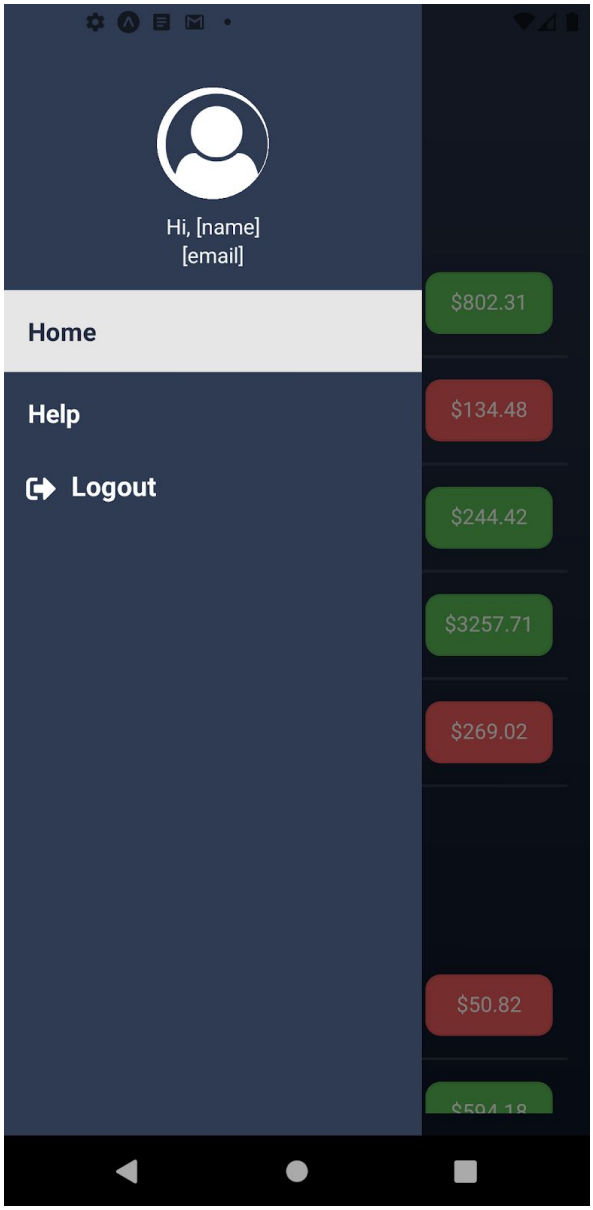
Android Screenshots:



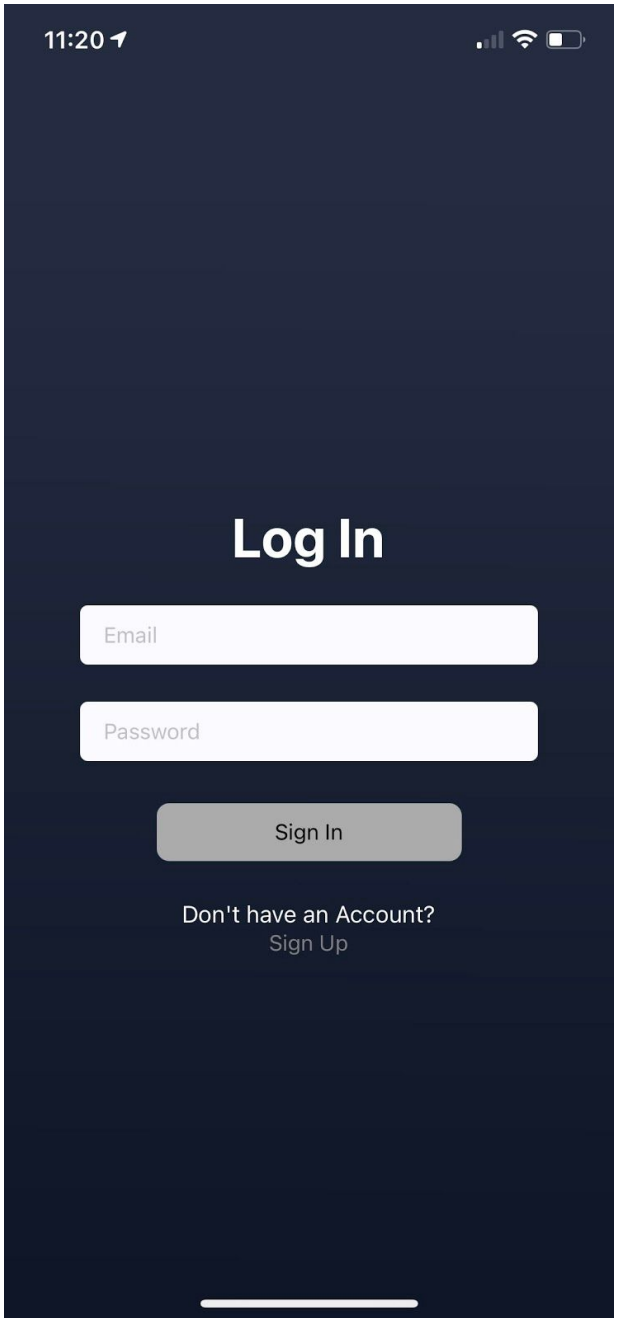

My Stocks
TSLA
$802.31
AAPL
$134.48
MSFT
$244.42
AMZN
$3257.71
FB
$269.02
+ Add Stock To List
Watch List
GME
$50.82

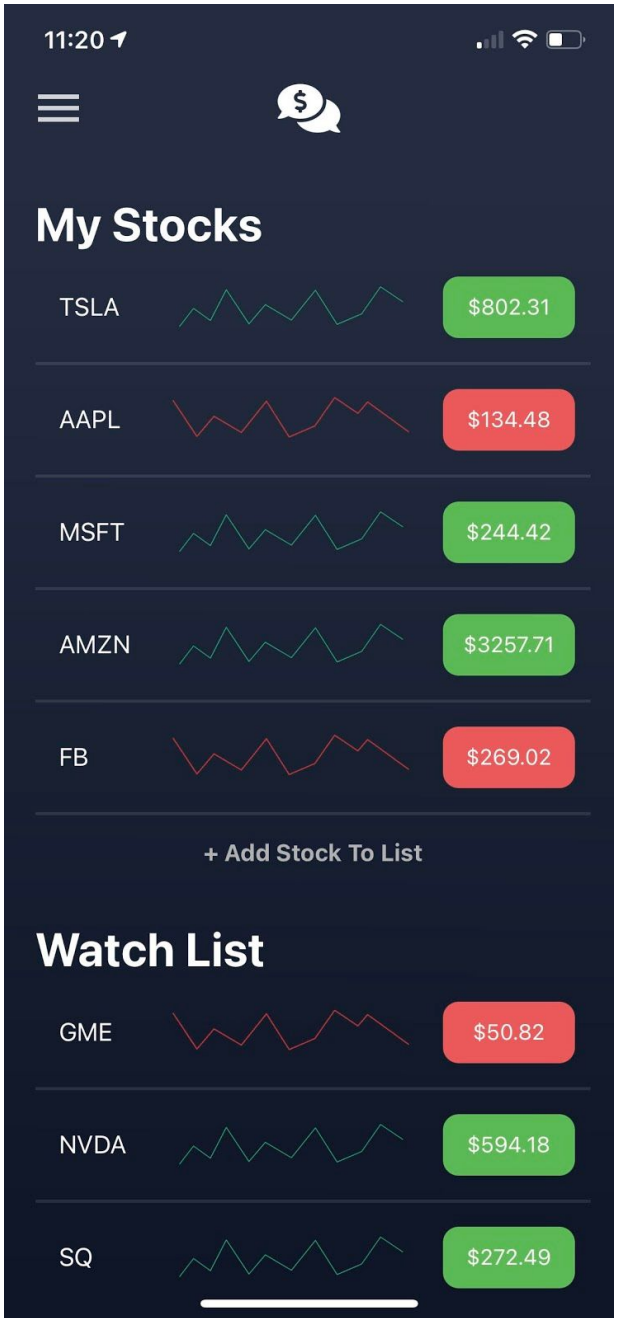

Stock Information
TSLA
Tesla
$802.31
High value: $807.25
Low value: $785.33
Based on our Analysis:
We predict that this stock will go up in the future.
News

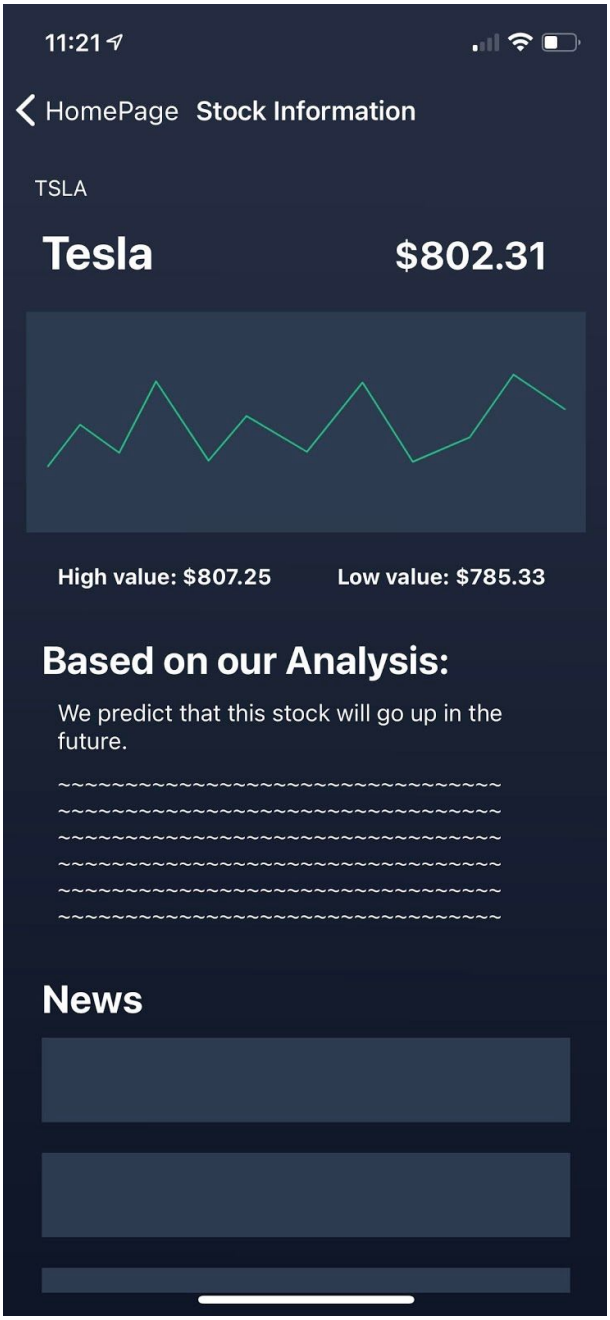

Browse
Search...

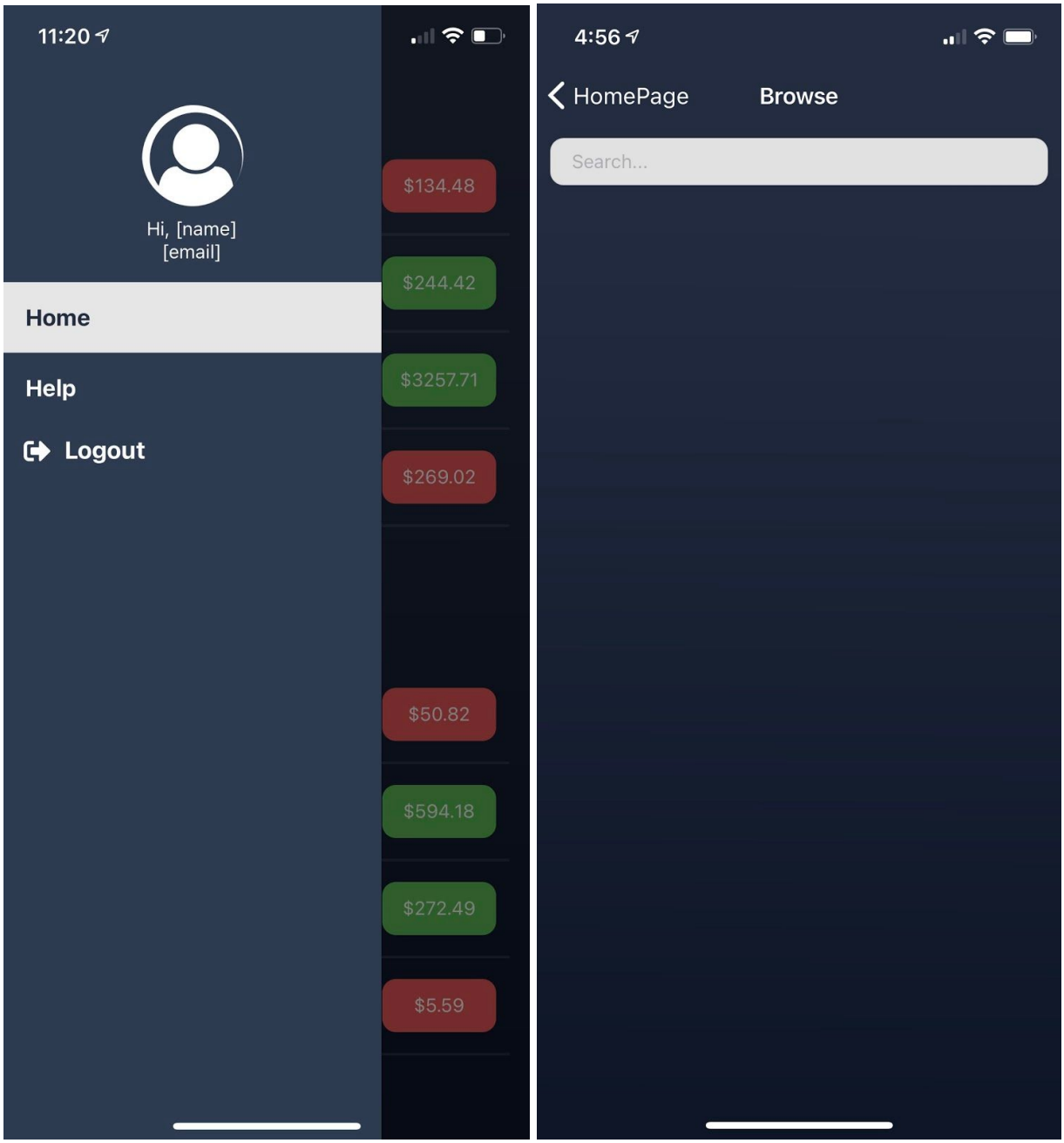

Hi, [name]
[email]
Home
Help
Logout
$802.31
$134.48
$244.42
$3257.71
$269.02
$50.82

iOS Screenshots:

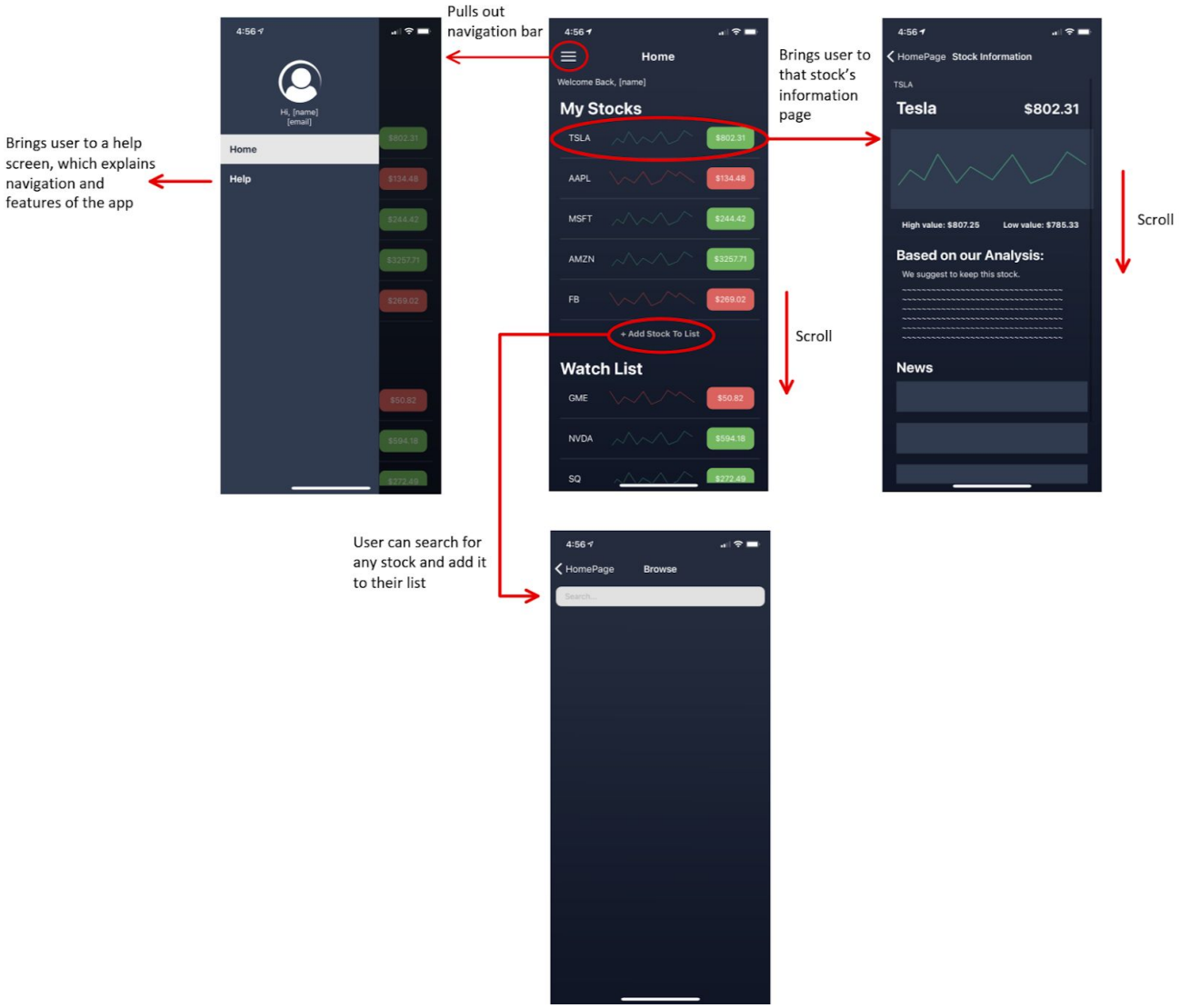



11:20
My Stocks
TSLA
$802.31
AAPL
$134.48
MSFT
$244.42
AMZN
$3257.71
FB
$269.02
+ Add Stock To List
Watch List
GME
$50.82
NVDA
$594.18
SQ
$272.49


11:21
HomePage
Stock Information
TSLA
Tesla
$802.31
High value: $807.25
Low value: $785.33
Based on our Analysis:
We predict that this stock will go up in the future.
News


11:20
Hi, [name]
[email]
Home
Help
Logout
$134.48
$244.42
$3257.71
$269.02
$50.82
$594.18
$272.49
$5.59
4:56
HomePage
Browse
Search...

UI Navigation/Functionality:



Github link: https://github.com/puskardev/StockApp