



**RUTGERS, THE STATE UNIVERSITY OF NEW JERSEY**

# **StockUp Web Application**

**Final Project**

**ECE-568: Software Engineering Web Applications**

**Professor: Mr. Yinglung Liang**

## **GROUP 6:**

Pranav Shivkumar (ps1029)  
Hari Priya Ponnakanti (hp467)  
Naga Venkata V Vinnakota (nvv13)  
Swapnil Shashikant Kamate (sk2181)

# Table of Contents

<b>1. Introduction</b>	4
<b>2. Requirements</b>	4
<b>3. Software Environment</b>	6
3.1. MySQL Database Management System	6
3.2. Python	6
3.3. Django	7
3.4. HTML	7
3.5. CSS	8
<b>4. Use Cases</b>	8
4.1. Admin Use Cases	9
4.2. User Use Cases	9
<b>5. Implementation</b>	27
5.1. Data Collection and Updation	27
5.1.1. Stocks Considered	27
5.1.2. Data and Database Tables	27
5.1.3. Database Schema for Storing Stock Data	29
5.1.4. MySQL Commands	29
5.1.5. Update Database with Data from Web Source	31
5.1.6. Connection to MySQL Database from Python	32
5.1.7. Yahoo Finance API	32
a. Fetching Historical Data:	33
b. Fetching Real-Time data:	33
5.1.8. Scheduling Data Collection Module	34
5.1.9. Screenshots for Stock Data Tables	34
5.2. Prediction Algorithms and Indicators Used	36
5.2.1. Short Term Prediction	36
5.2.1.1. Inputs and Outputs for Short Term Prediction	36
5.2.1.2. Polynomial Curve Fitting for Short Term Predictions	36
5.2.1.3. Input and Output to the Curve Fitting algorithm:	39
5.2.2. Long Term Prediction	39
5.2.2.1. Inputs and Outputs for Long Term Prediction	39
5.2.2.2. Artificial Neural Networks for Long Term Predictions	40
5.2.2.3. Input and Output to the Neural Network:	42

5.2.3. Technical Indicators Used	43
1. Exponential Moving Average (EMA)	44
2. Relative Strength Index (RSI)	44
3. Moving Average Convergence/Divergence (MACD)	45
5.3. Web Service	46
5.3.1. Implementation	46
5.4. User Interface	52
<b>6. Conclusion</b>	<b>70</b>
<b>7. Future Work</b>	<b>71</b>
<b>8. Contribution Breakdown</b>	<b>71</b>
<b>9. References</b>	<b>71</b>

# 1. Introduction

The goal of this project is to develop a web application which assists users in making investment decisions in the stock market. This assistance is provided by performing stock price predictions using Machine Learning algorithms in the application background. Stock market prediction is the act of trying to determine the future value of a company's stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. Predicting the stock market with 100% precision is impossible, but we can still manage to develop some analysis techniques to improve our understanding of the market and make better decisions in our investments.

In our application, we store the stock data for 10 stocks and provide predictions for these stocks. Though the main aim of the project is to provide stock price predictions and buy/sell suggestions to users, we also implemented several other functionalities to let the user have an informative and personalized experience. We will describe these in detail in the later sections of this document.

## 2. Requirements

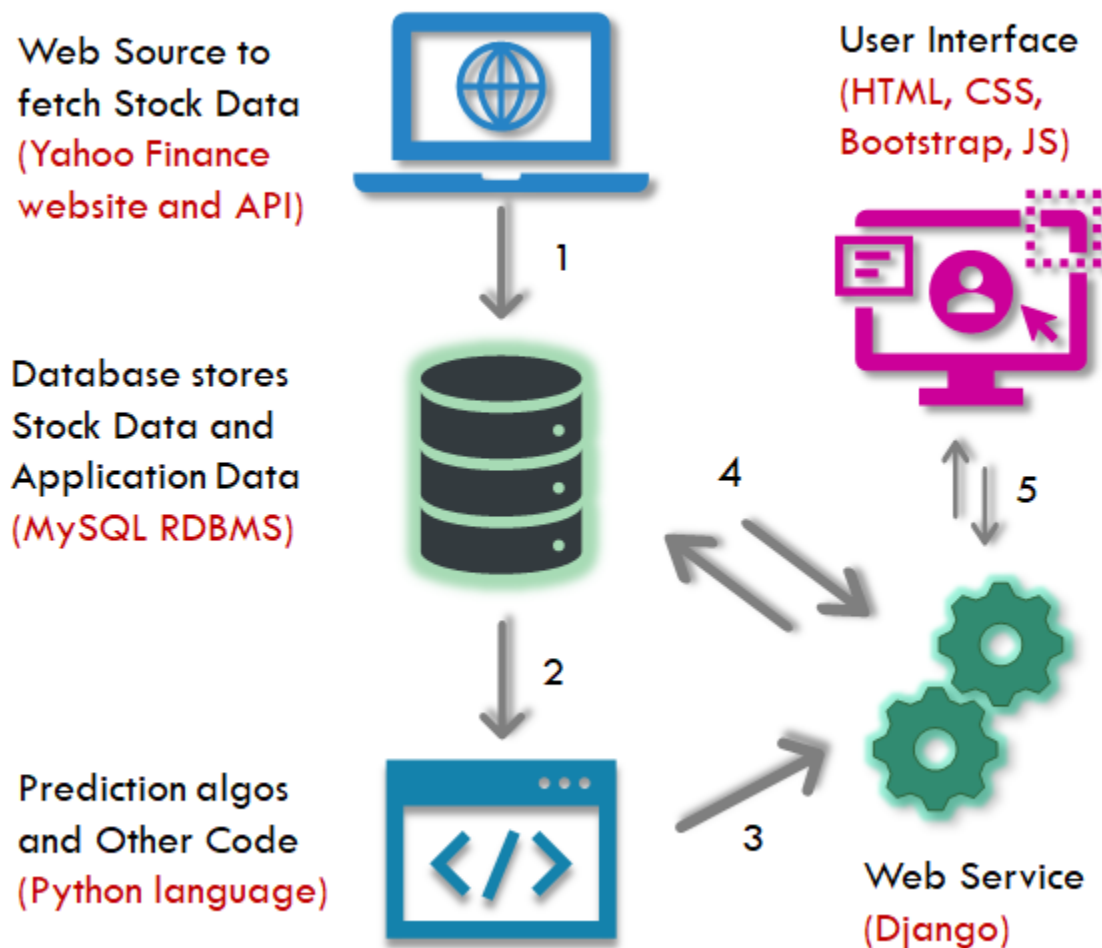
The main requirement of this project is to build an end to end full-stack web application which helps users in making stock investment decisions. A web application or web app is a client-server computer program that the client (including the user interface and client-side logic) runs in a web browser. The web application which we built for this project is a collection of the following segments:

1. Web source to gather data
2. Database to store the required data
3. Database Language to perform queries on data
4. Scripting Language to communicate with the database
5. Prediction Algorithms which run in the application background
6. Web Service to facilitate the communication between the backend and front-end
7. User Interface

The technologies which we have used for each of these segments are as follows:

1. Web source for stock data: **Yahoo Finance**
2. Database: **MySQL Database Management System**
3. Database Query Language: **Structured Query Language (SQL)**
4. Communication with the database: **Python Language**
5. Prediction Algorithms: **Python Language**
6. Web Service: **Django REST Framework**
7. User Interface: **HTML, CSS, Bootstrap**

The high level architecture of our web application can be seen below.



### **Explanation:**

1. Keep updating the database daily with stock data from the web source.
2. Use the data for Prediction Algorithms and other functionalities.
3. Send results from Prediction algorithms and other functionalities to Django Web Service.
4. Use data from the database to display to users and input User data into the database.
5. Render the results from the backend code and database to the User Interface and read data from the user interface to insert into the database.

We are required to use prediction algorithms for both short-term and long-term predictions. We picked two Machine Learning algorithms for this purpose. We have used

Polynomial Curve Fitting for short-term predictions while we have used Artificial Neural Networks for long-term predictions. Besides these, we have also made use of technical indicators like EMA, RSI and MACD to observe the market trends and then provide a buy/sell suggestion to the users accordingly.

We are required to build a User Interface for the application. We have provided various analysis techniques to the user by providing all the necessary short-term and long-term data for each stock and we have also provided various graphs all throughout our interface to let the user observe the various movements and trends in the market.

### **3. Software Environment**

This section is to provide a brief introduction and background about the frameworks and technologies we have used in our application.

#### **3.1. MySQL Database Management System**

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. Nowadays, we use relational database management systems (RDBMS) to store and manage huge volumes of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys. MySQL is one such open-source, fast, reliable, and flexible relational database management system. Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The SQL part of "MySQL" stands for "Structured Query Language". SQL is the most common standardized language used to access databases.

MySQL creates a database for storing and manipulating data, defining the relationship of each table. It is used for developing web-based software applications. MySQL handles large databases and works on many different platforms. A large number of web developers worldwide are using MySQL to develop web applications.

#### **3.2. Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective. Python also supports a lot of libraries for scientific and numeric computing. It provides simple and efficient tools for data mining and data analysis.

### **3.3. Django**

Django REST framework is a powerful and flexible toolkit for building Web APIs. We use the Django REST Framework application to easily integrate a REST API into existing Django functionality. Django applications are a Python package that provide some set of features like reusability and they are a combination of models, views, templates, URLs, etc.

Django REST Framework is a third Party application that will be used in conjunction with our own application to create a REST API. It provides us with many features that connect deeply with Django's existing structures, helping us create RESTful HTTP resources that correspond to the models we create. We will use this framework to handle the creation of URLs, perform data manipulation with generic views, authenticate users, handle request permissions and so on.

### **3.4. HTML**

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript). HTML elements form the building blocks of all

websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages. An HTML Application is a Microsoft Windows program whose source code consists of HTML, Dynamic HTML, and one or more scripting languages supported by Internet Explorer, such as VBScript or JScript. The HTML is used to generate the user interface, and a scripting language is used for the program logic.

### **3.5. CSS**

CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content. CSS is a cornerstone technology used by most websites to create visually engaging web pages, user interfaces for web applications for many mobile applications.



## 4. Use Cases

There are two major roles in our application. These are:

- Admin
- User

### 4.1. Admin Use Cases

Admin is the role who takes care of the website and maintaining the users and list of stocks in our application. Use Cases provided for Admin are:

#### **1. *Create an admin account***

The Admin should be able to create an account for himself using the service. He can do this by executing the `createsuperuser` command from the Django web service where he can provide a username and password for the admin account.

#### **2. *Login to the admin account***

The Admin should be able to login to his account using the credentials with which he created the account. The url for the admin login page will be `localhost:8100/admin`.

#### **3. *Change account password***

The Admin should be able to update the account password from the interface provided to him.

#### **4. *Add or delete user accounts***

The Admin should be able to add or delete users from the interface provided to him.

#### **5. *Add, delete and modify the list of stocks***

The Admin can add, modify or delete any stock from the list of stocks present in the database through the interface.

#### **6. *Logout of the admin account***

The Admin should be able to logout of his account.

## 4.2. User Use Cases

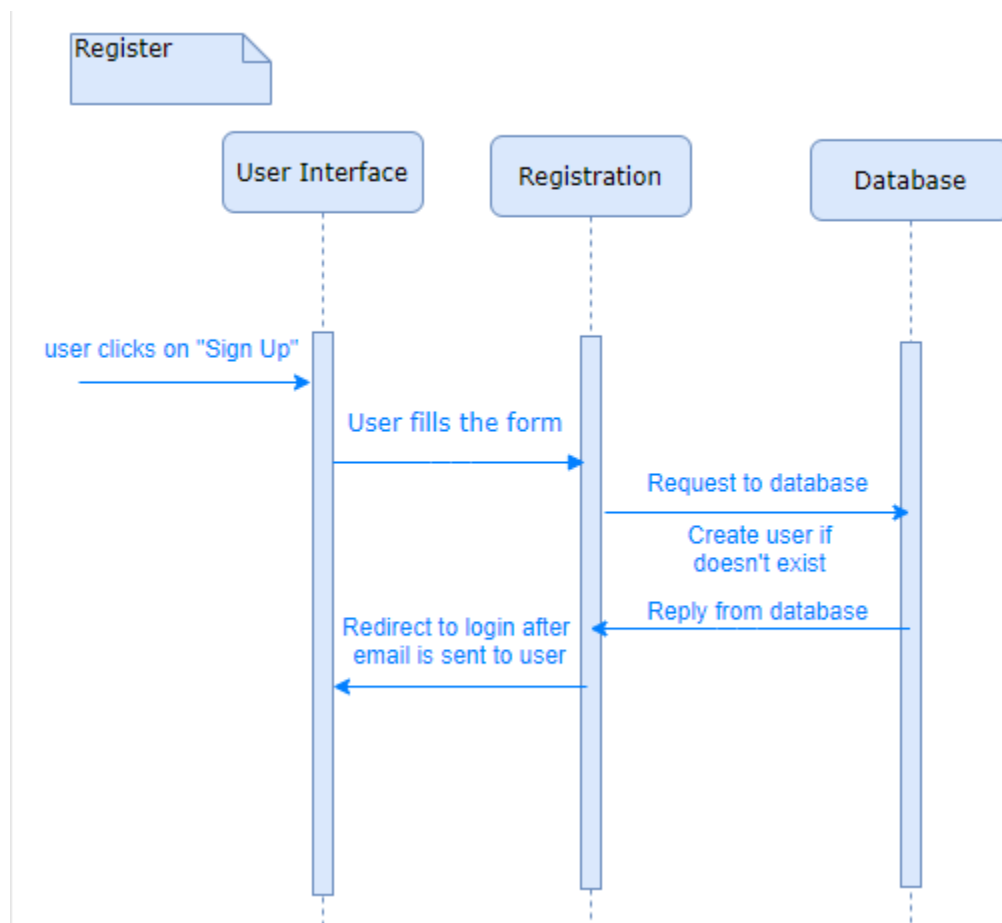
User is the role who makes use of all the application services by registering on the application and logging in to the website. He has lesser privileges than the admin regarding the application. The following are the list of use cases and the corresponding descriptions in brief along with their associated interaction diagrams for all our application users.

### 1. Create a user account

The users must be able to register on the website and create an account for themselves by signing up with details like Username, FirstName, LastName, Email and Password.

### 2. Registration Notification

Once a user creates an account on our application, he gets notified via email informing him that a StockUp account has been successfully created for him. The interaction diagram depicting this use case is shown below:

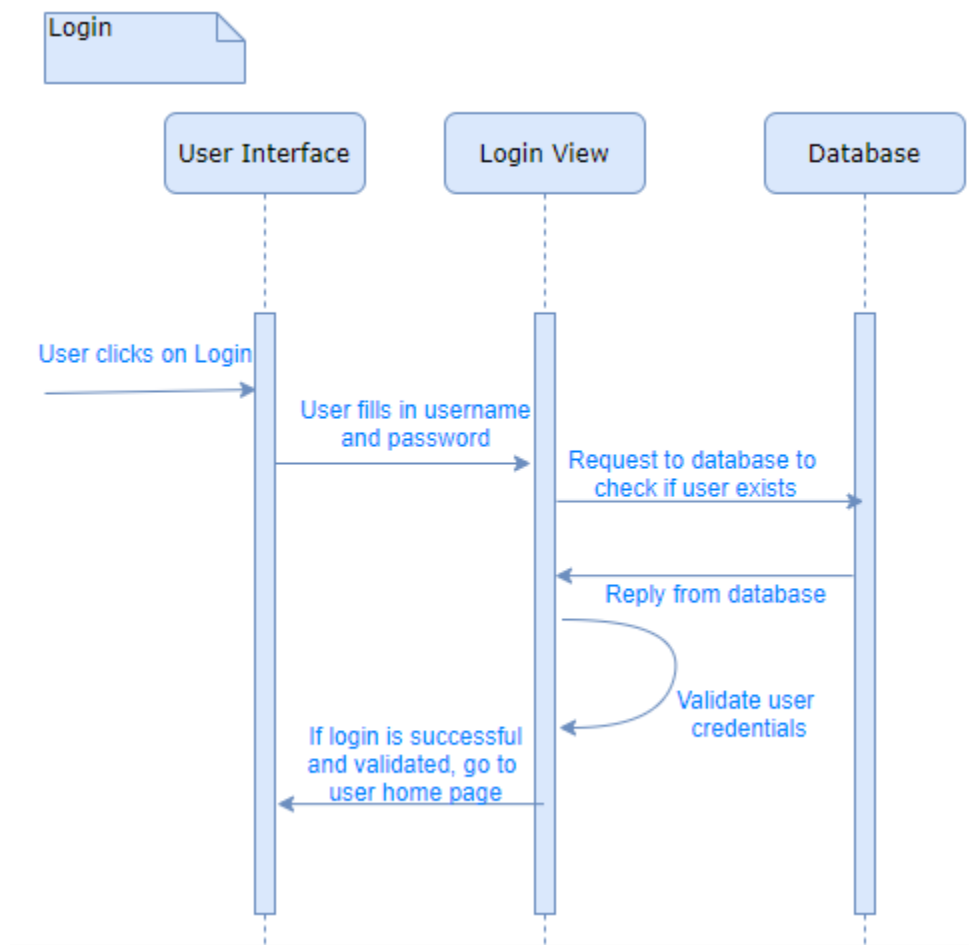


### 3. Login into the user account

The users must be able to login to their account by providing the correct username and password.

The input form is validated once the user enters the login credentials, that is username and password. The user will be logged in only if these details are correct, otherwise, we display him an error message that his username/password is incorrect.

The interaction diagram depicting this use case is shown below:

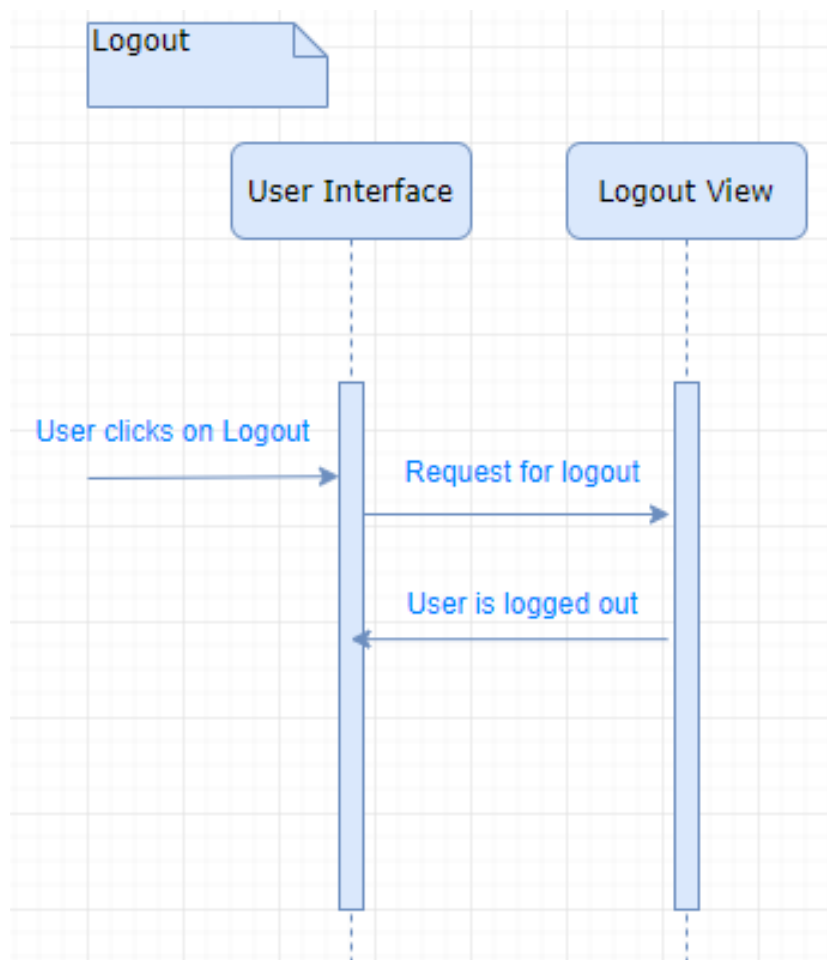


#### 4. Logout of the user account

The users must be able to logout of their account.

Using the logout, the user will be logged out and their session will end. They need to login again to access their respective application accounts.

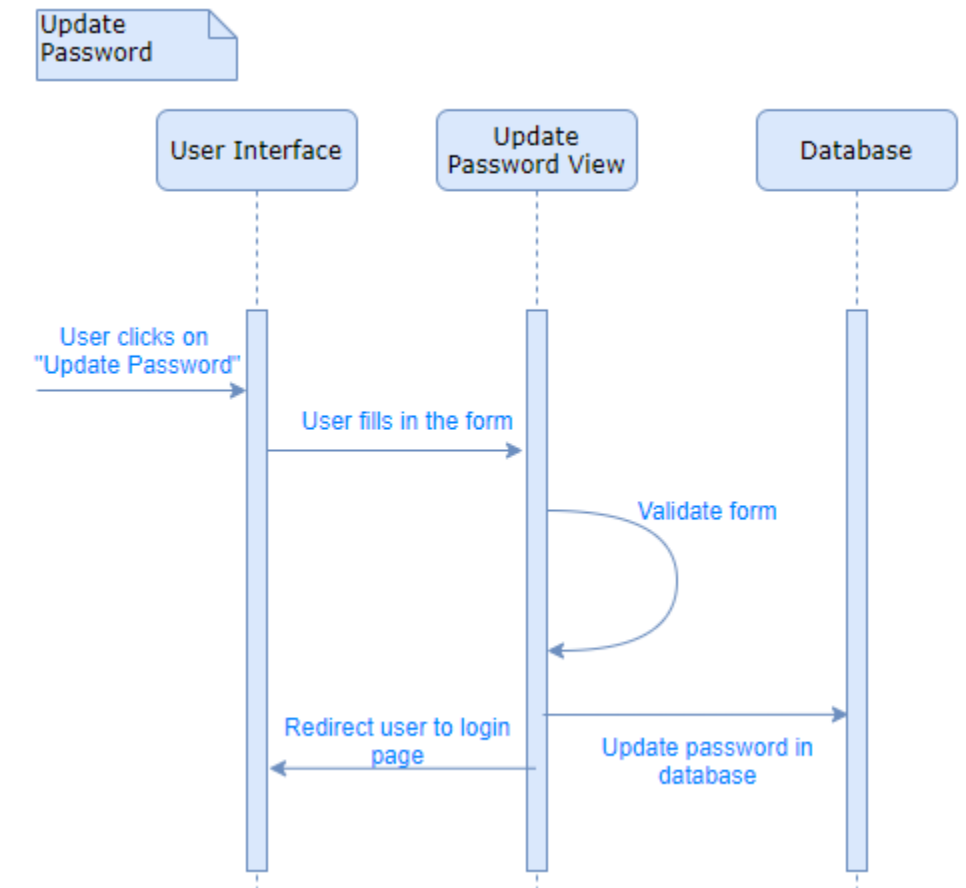
The interaction diagram for this use case is shown below:



## 5. Change account password

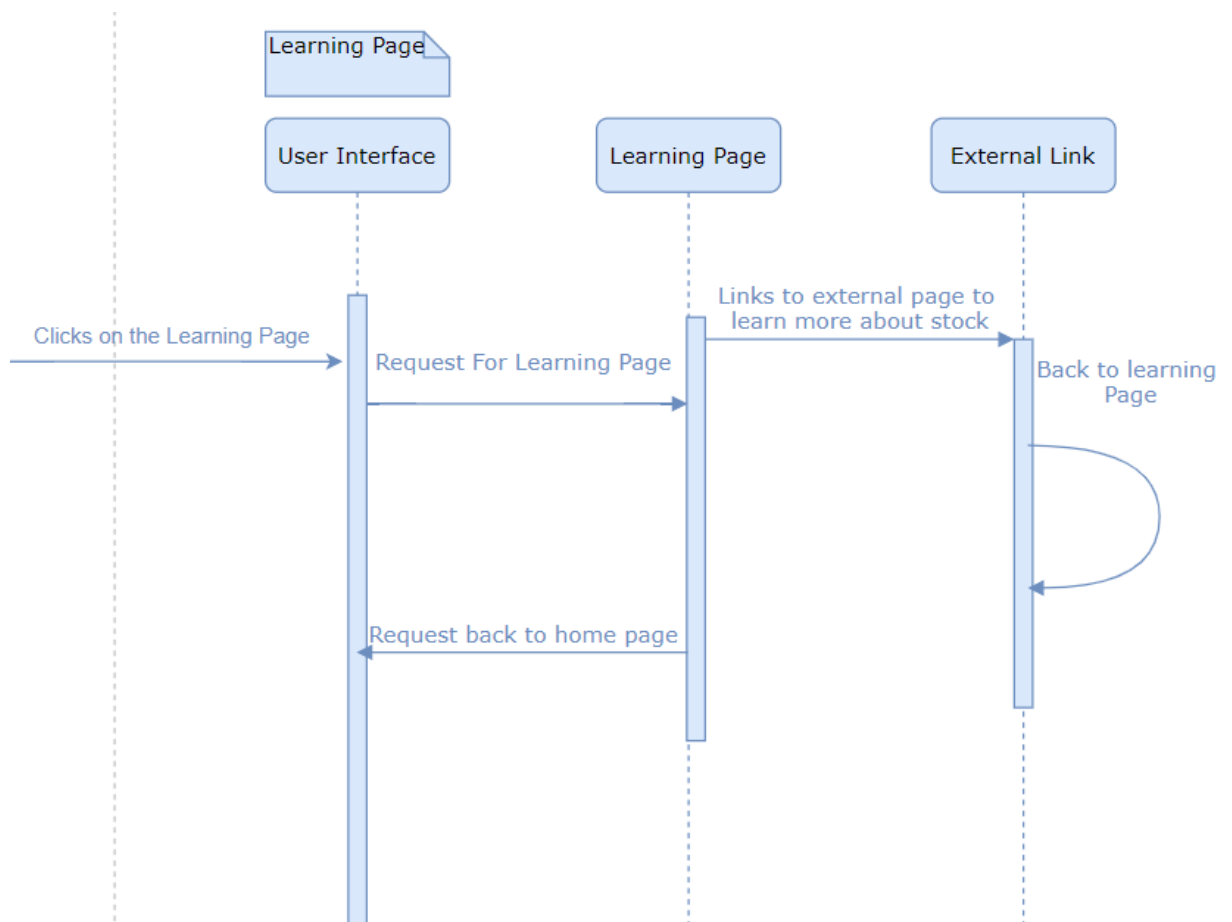
The users must be able to update their account password by providing their old password and new password.

The interaction diagram depicting this use case is shown below:



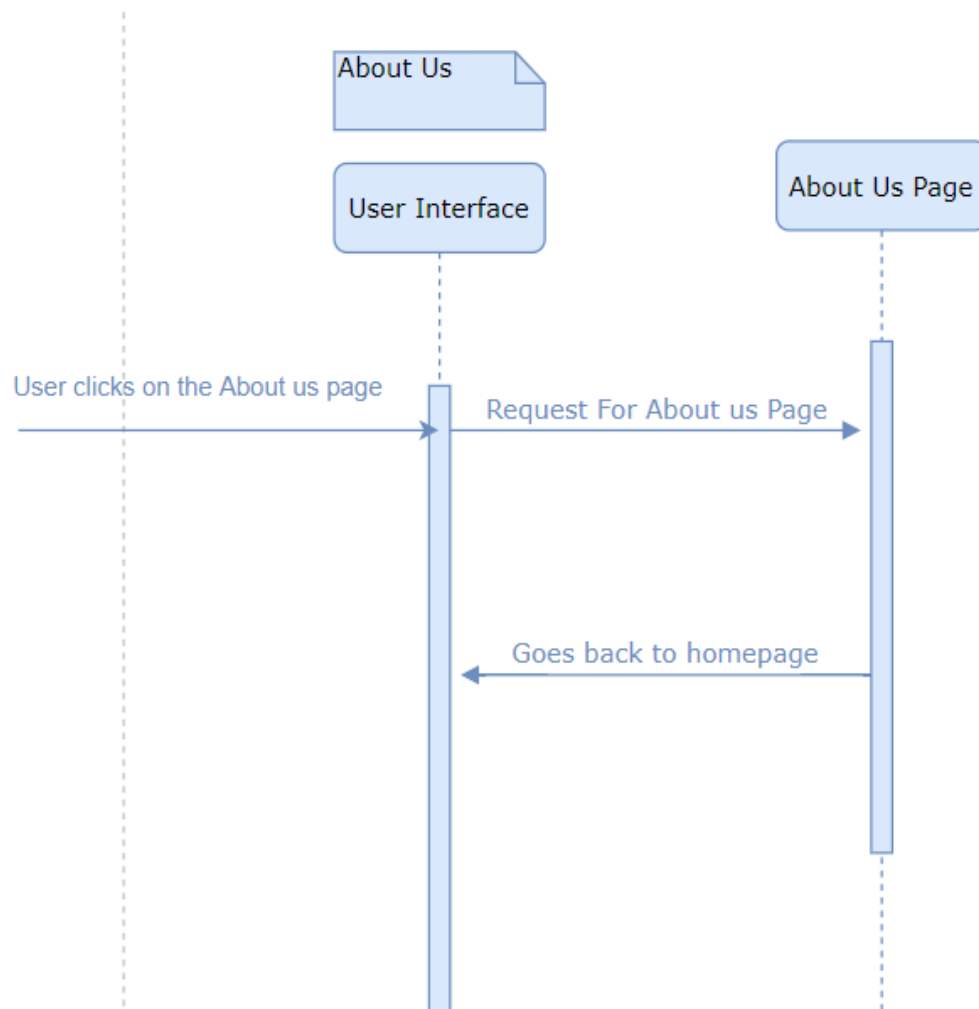
## 6. Beginner's Guide Page

If a user is a beginner to the stock market trading, then he can refer to the Beginner's Guide page to get acquainted with the stock market trading and basic concepts related to the stock market. We provide some basic knowledge on this page. Besides that, we also provide links to external websites and YouTube tutorials regarding stocks and trading.



## 7. About Us page

The About Us page contains the overview of our application and the wide range of services the user can make use of if he registers on our application. This also serves as a brief advertisement for our application.



## 8. Get long-term and short-term predictions

The users can receive short-term and long-term price predictions for any stock of their choice. We gave an option to the user to enter the number of minutes/days he wants the predictions for.

### 9. Get buy or hold or sell suggestions

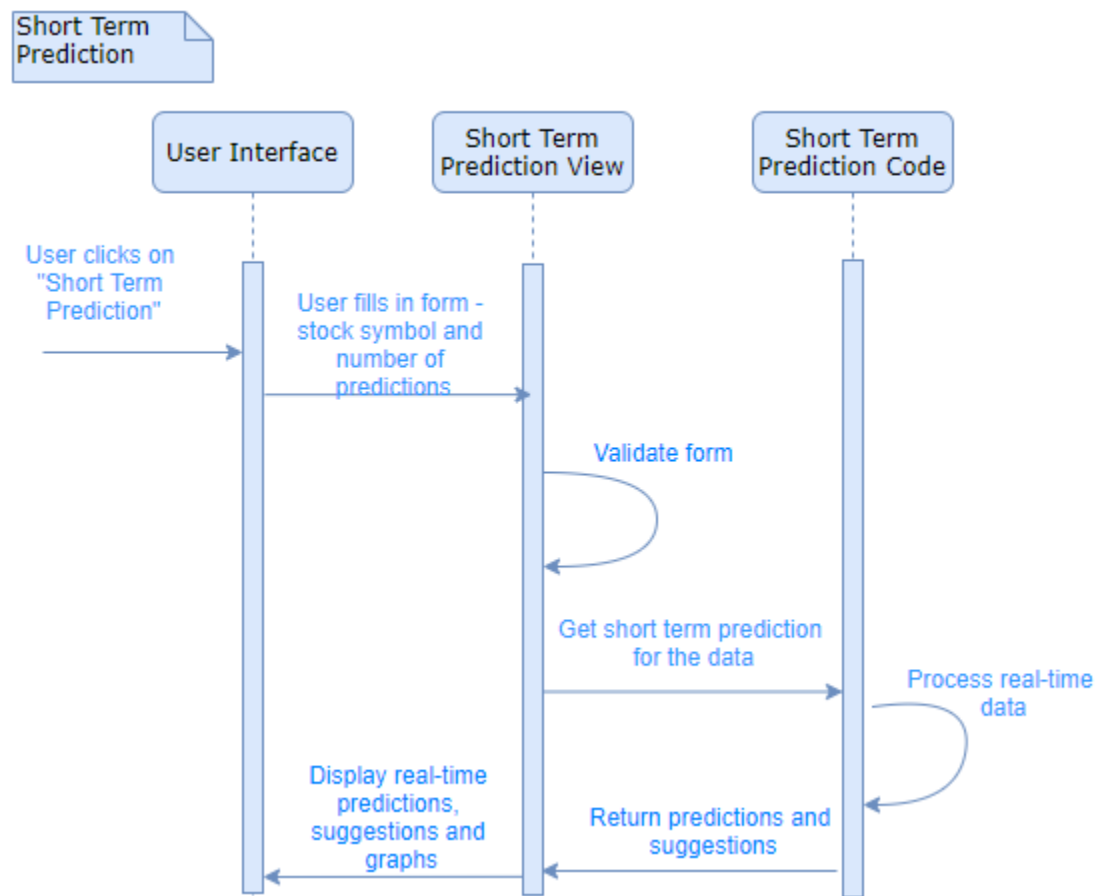
The users also receive buy or hold or sell suggestions along with the predictions for any stock of their choice in either of the short-term and long-term cases.

### 10. View graph plots of the price and the indicators used

The users will also be able to visualize the price trends and the indicator curves through graphs when they use the prediction feature of the application.

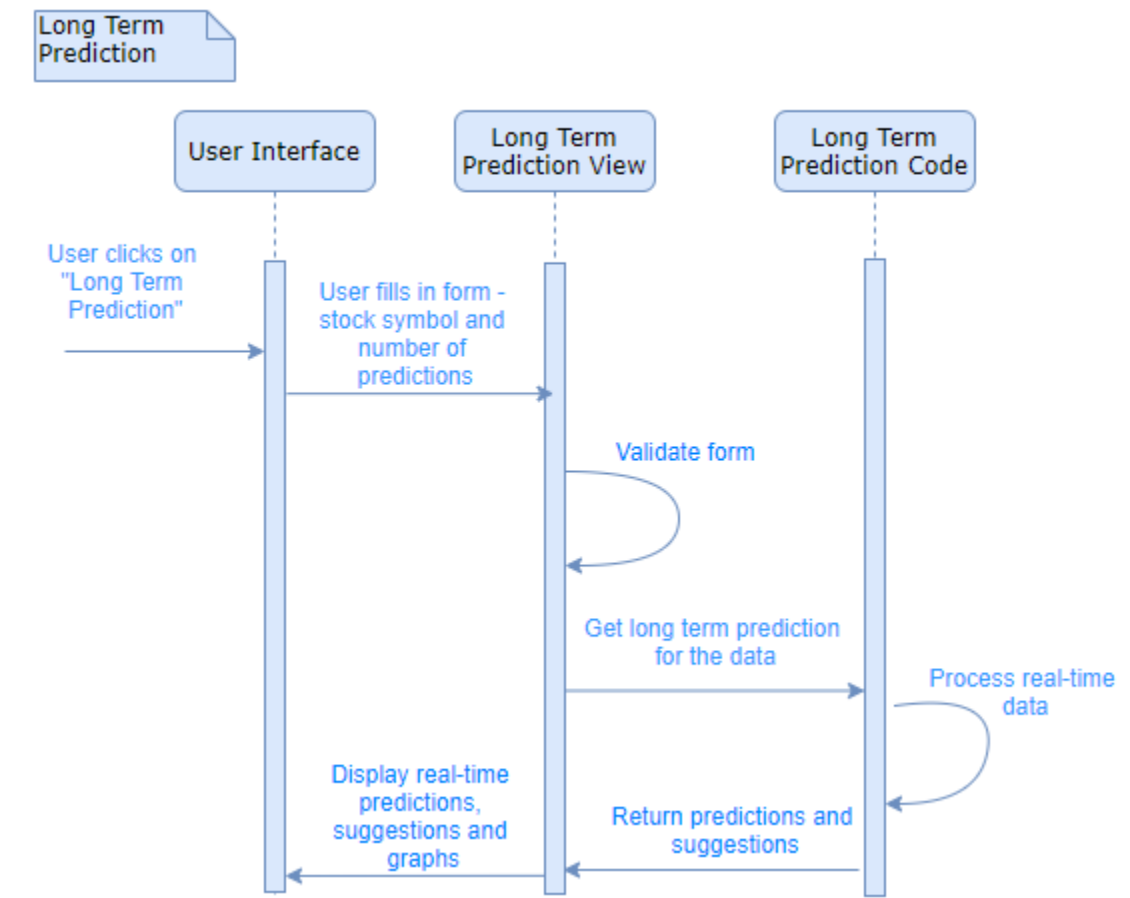
The interaction diagram for the above use cases is shown below for:

#### a. Short Term





## b. Long Term

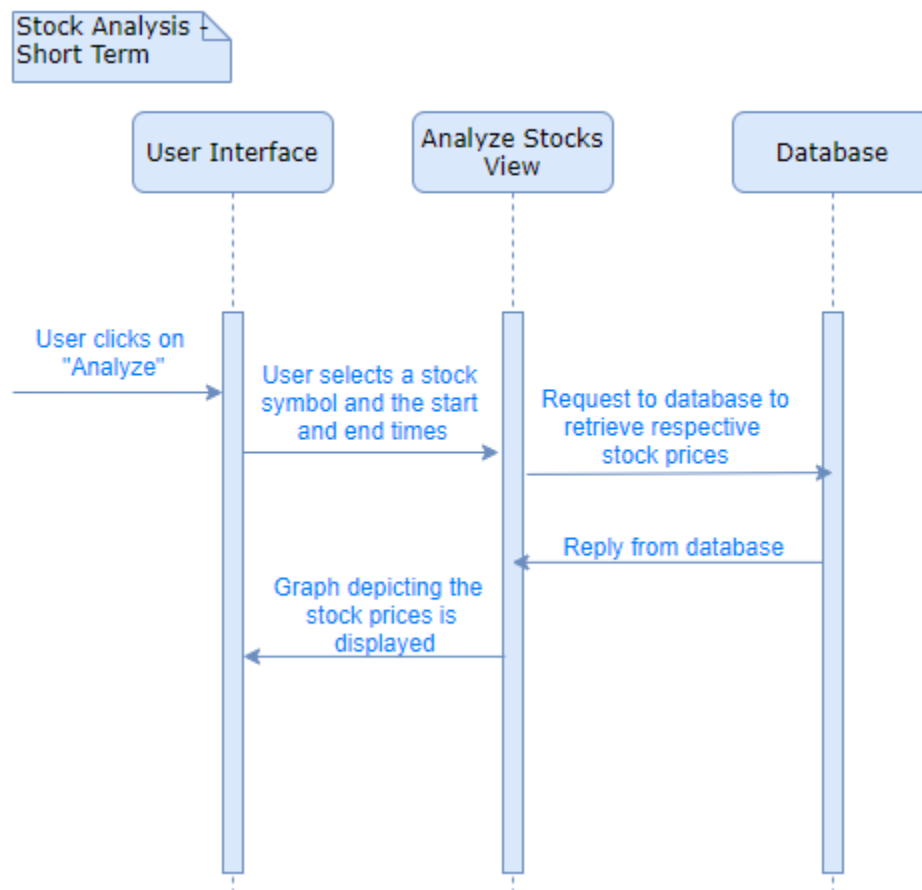


### 11. Able to analyze Stock Trends for both short and long term

The user will be able to analyze both historical and real-time price trends for any stock he/she chooses and for the selected period of time.

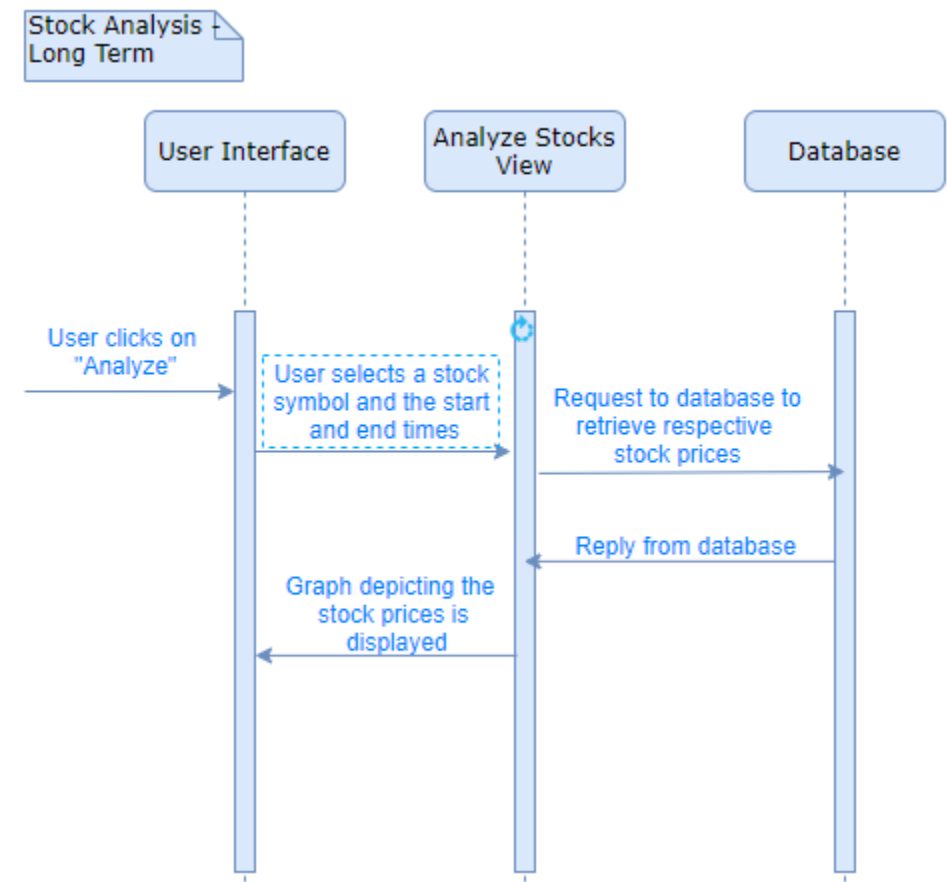
#### a. Short Term Stock Analysis:

A user can receive short-term analysis (using real-time data) for any stock he chooses. He can view a graph and get additional details apart from the basic queries asked like Previous Close, Percentage Change in Price on Current day from Previous Close, lowest stock price in 10 days and highest stock price in 1 year.



## b. Long Term Stock Analysis:

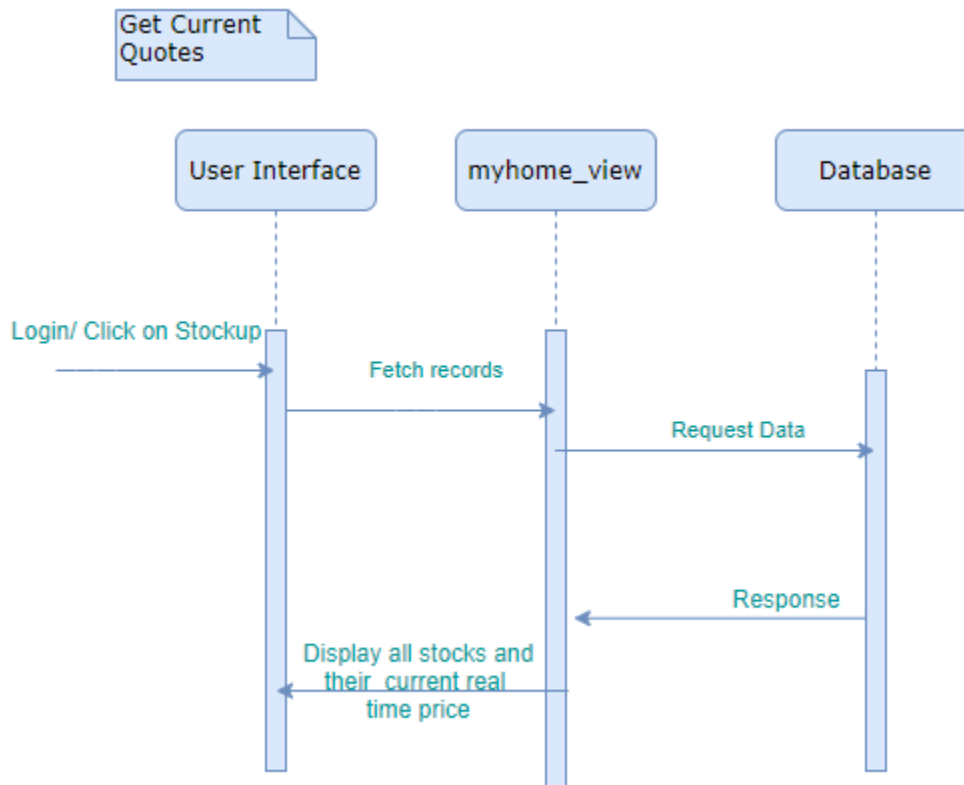
A user can get long-term analysis (using historical data) for any stock he chooses. In this case, besides the basic requirement for the graph where the user can choose the date range, we also implemented two additional cases where the user can choose the metric for which he wants the graph for (eg: open price, close price, low, high and volume) and he can also choose the frequency of data for the graph (eg: daily, weekly and monthly).



## 12. Get Current Quotes of stocks

The user will be able to view the list of all stocks on the website with their most recent prices.

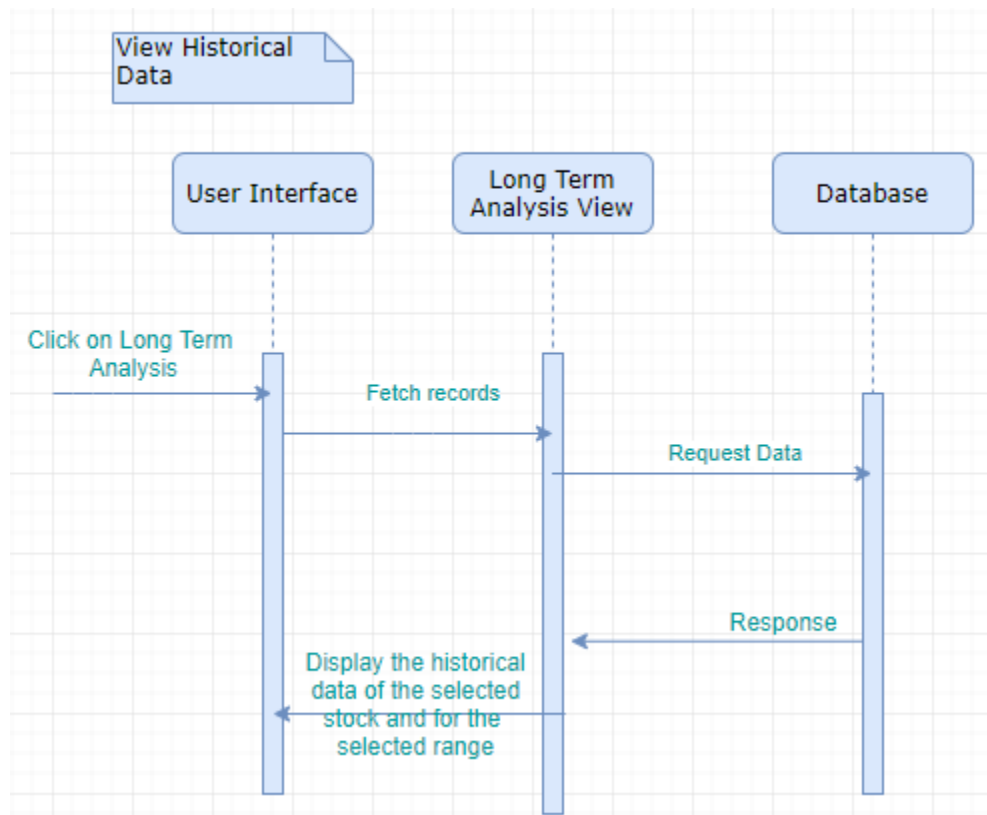
The interaction diagram for this use case is shown below:



### 13. View Historical Data

The user can view all the historical data records for any stock he chooses, and he can also choose the start date and end date between which he wants the records to be displayed.

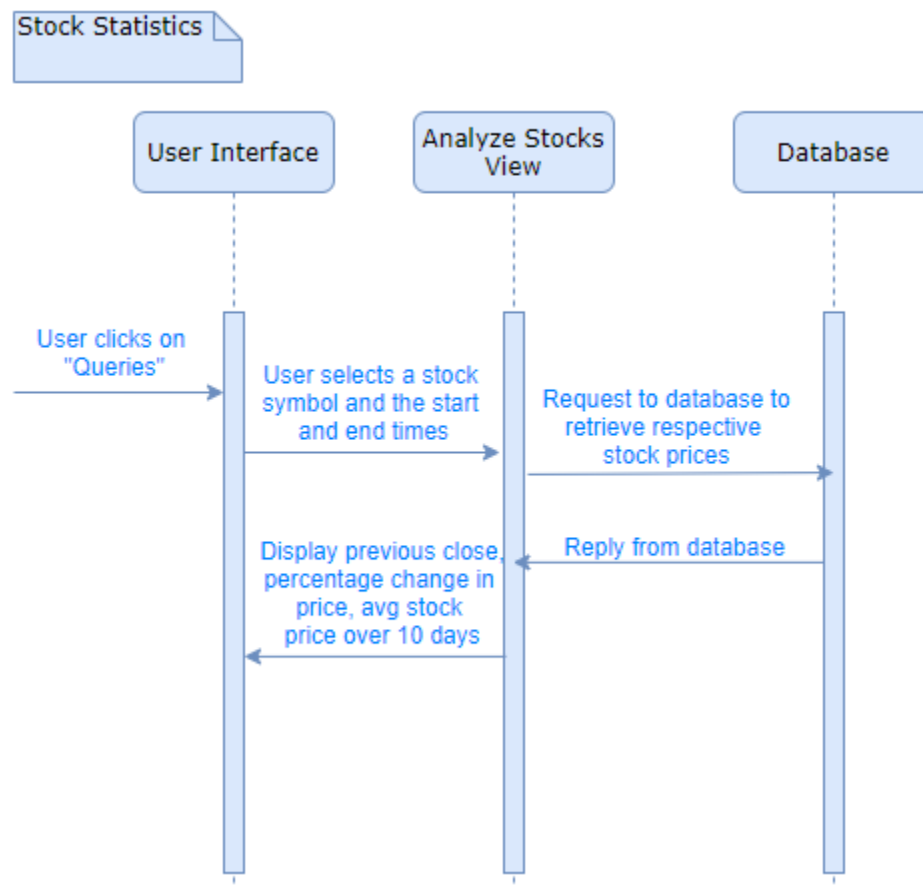
The interaction diagram for this use case is shown below:



## 14. View Stock Statistics

The user will be able to view the statistics of any chosen stock like the previous close, percentage change in price, average stock price over 10 days, one year and so on.

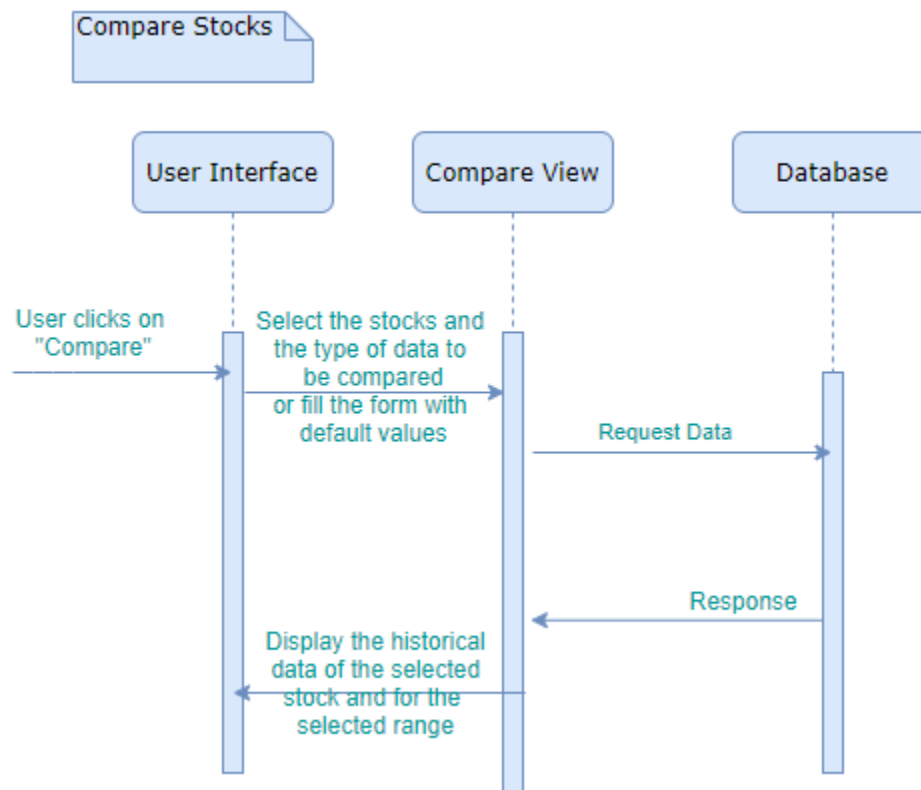
The interaction diagram for this use case is shown below:



## 15. Compare Stocks

The user can choose two or more stocks from a list of stocks and he can choose the short-term or long-term option to compare these stocks either using real-time data or historical data respectively. He then gets two plots, one plotting the price movements of all stocks and another plotting the volumes traded for all stocks. He can also view the average values of all the attributes for the stocks either over the short-term or long-term.

The interaction diagram for this use case is shown below:



## 16. Specify Price Percentages

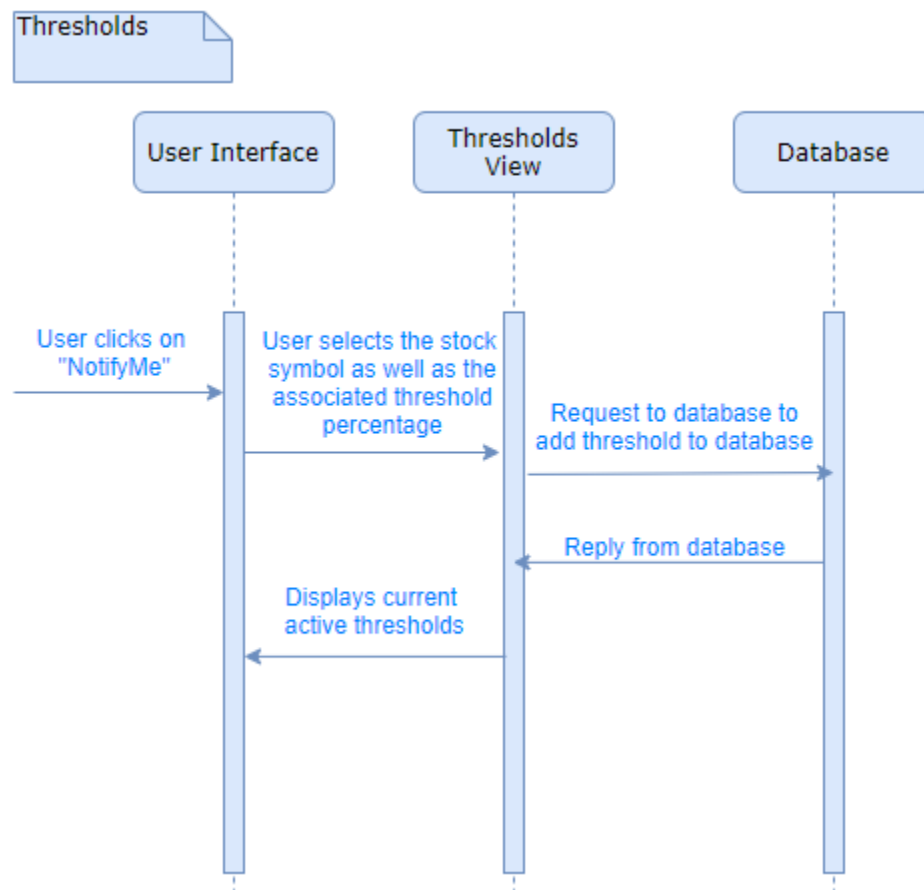
The user will be able to set percentage change in prices of any of the stocks he is interested in. This can be useful in two cases:

- When the user wants to buy a stock and he wants to wait until the stock price falls by a certain amount.
- When the user wants to sell a stock and he wants to wait until the stock price rises by a certain amount.

## 17. Receive Notifications for Price Changes

The user will receive email notifications whenever any of the percentages they set are reached so that they don't miss out on a good trading opportunity.

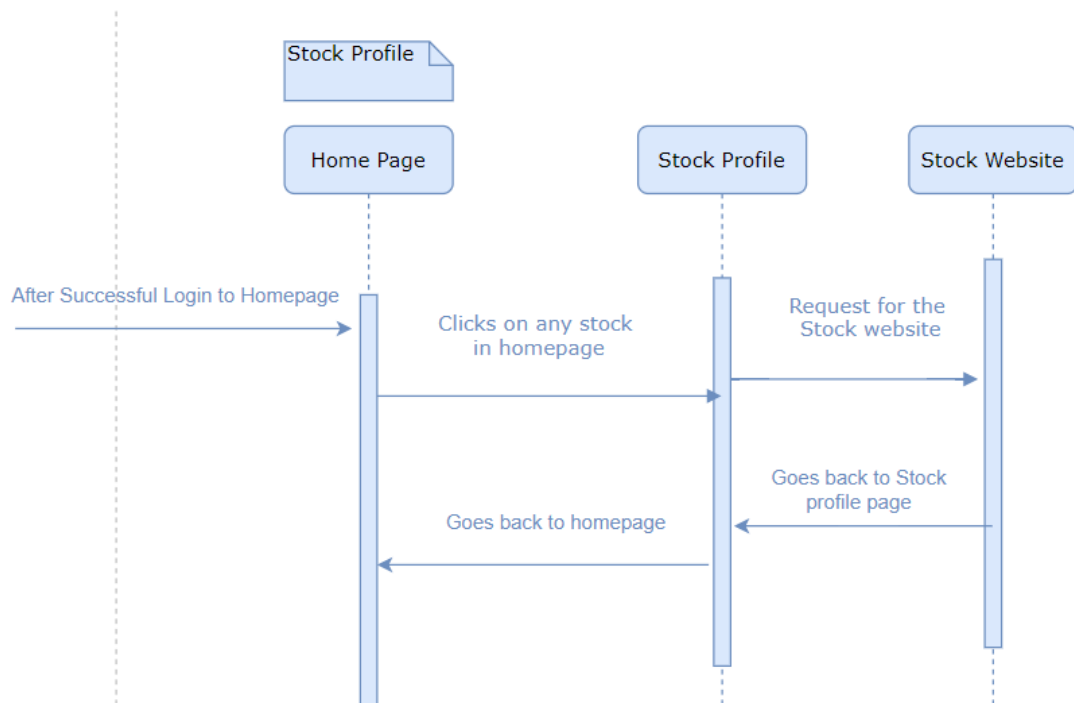
The interaction diagram for the above use cases is shown below:





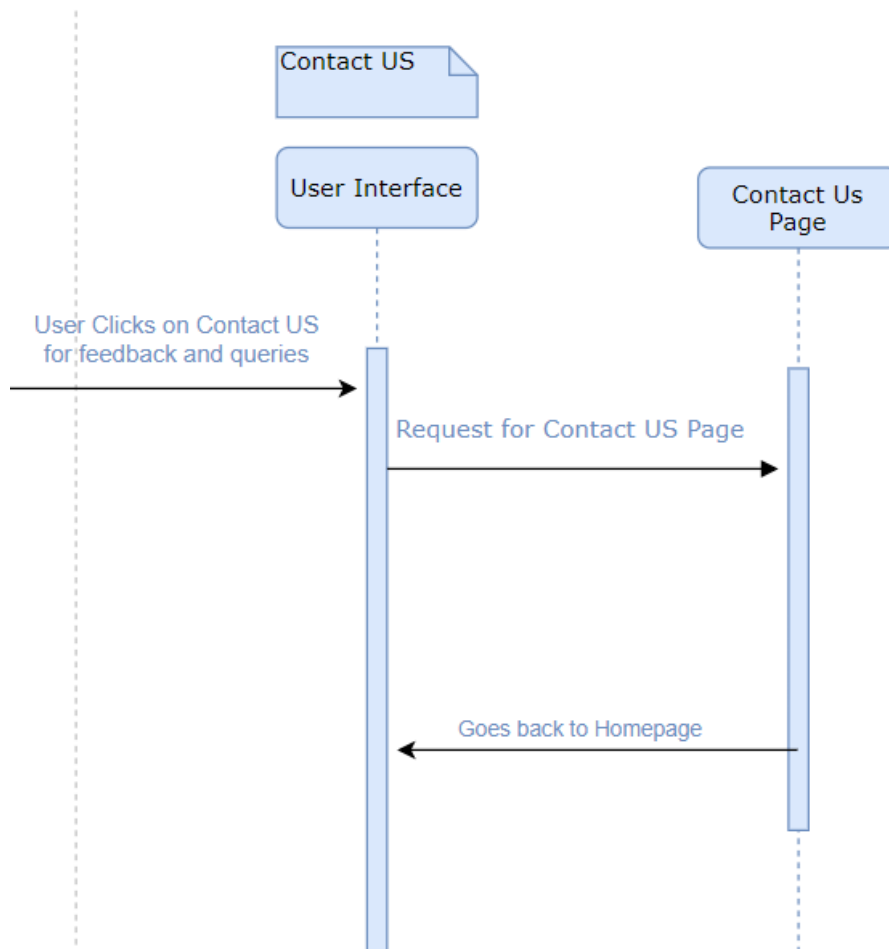
## 18. Stock Profiles

To provide some background about the company and what the company does, we provide a Profile page for each stock where the users can learn more about the company by visiting the official company website from this page.



## 19. Contact Us Page

Contact Us page has the contact details of the application owners for the users to contact us in case of any queries or feedback about the application.



## 5. Implementation

The implementation of this application involved the development of four major modules necessary for its functioning. They are:

- Data Collection and Updation
- Prediction Algorithms and Indicators Used
- Web Service
- User Interface

The implementation of these modules is described in the sub-sections below.

### 5.1. Data Collection and Updation

#### 5.1.1. Stocks Considered

The basic idea of the data collection module is to collect the relevant data necessary for StockUp to predict the stock prices for both the long term and short term periods. We collected the stock data for the following 10 stocks. Their corresponding stock symbols are as listed below:

1. **Apple (AAPL)**
2. **Amazon (AMZN)**
3. **Facebook (FB)**
4. **Google (GOOGL)**
5. **Netflix (NFLX)**
6. **TripAdvisor (TRIP)**
7. **Tesla (TSLA)**
8. **Twitter (TWTR)**
9. **Marriott Vacations Worldwide Corporation (VAC)**
10. **Yelp (YELP)**

#### 5.1.2. Data and Database Tables

There are two types of data that StockUp is supposed to handle, the historical data and the real-time data.

##### a. Historical Data

The historical data requires the following attributes of each stock.

- **Opening Price:** The opening price is the price at which a security first trades upon the opening of an exchange on a trading day.
- **Closing Price:** Closing price generally refers to the last price at which a stock trades during a regular trading session.

- **High:** It means the highest price in a given period of time (one day in the case of historical data)
- **Low:** It means the lowest price in a given period of time (one day in the case of historical data)
- **Volume:** Volume is the number of shares of a security traded during a given period of time.
- **Date:** The dates corresponding to the stock data records.

#### b. Real Time Data

The real time data requires the following attributes of each stock.

- **Price:** The price of the stock at that point of time.
- **Volume:** It is similar in definition to the volume in historical data.
- **Date:** The dates corresponding to the stock data records.
- **Time:** The time corresponding to the stock data records.

Besides the above required fields, we have also stored the Open, High, Low and Volume attributes even for real-time data to make use of these fields for analysis purposes in the user interface. The price attribute in the real-time table is stored in a field called Close.

The time interval between two data records in the historical table is one day while the time interval between two data records in the real-time table is one minute. We have also added an additional column called sid which represents the Stock ID. The sid is an integer attribute which is used to uniquely identify a stock and its corresponding data in the historical and real-time tables.

As the historical and real-time tables house enormous amounts of data, we have used the sid field as an integer ID instead of the ticker symbols as it is more efficient to store integers than strings.

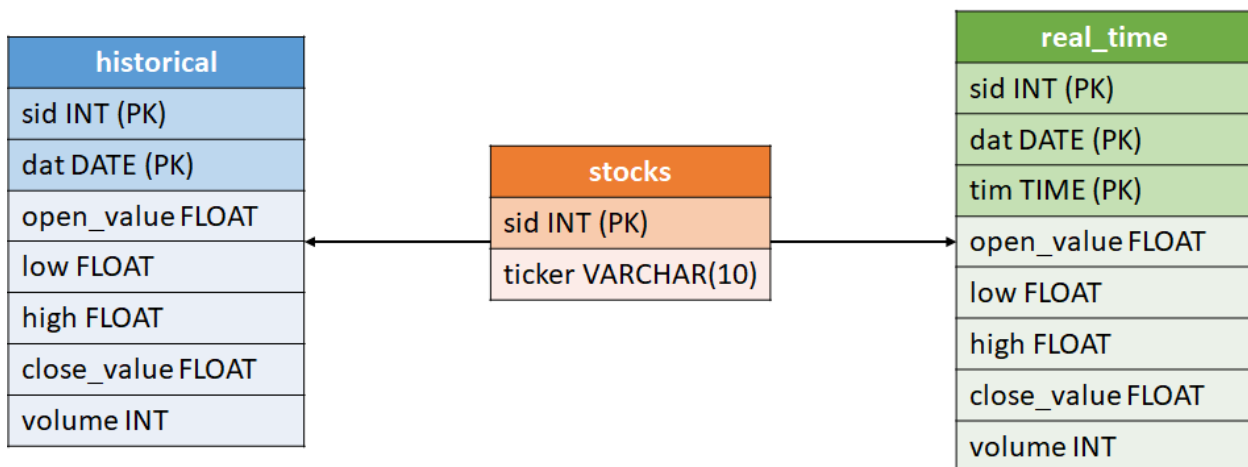
We created another table called Stocks to map the sid values to the stocks. In this table, we have two fields, one for the sid and one for the stock ticker. The data in this column is as below.

sid	ticker
365	AAPL
267	AMZN
146	FB

178	GOOGL
711	NFLX
413	TRIP
579	TSLA
292	TWTR
632	VAC
971	YELP

We had tried to obtain the stock details of Yahoo as well, but the details of the stock were private. We then considered a similar stock, Yahoo Japan (YOJ.SG) and tried to access its details. We were able to download the stock data and relevant details; however we were denied access while fetching data periodically from the web using the APIs. Due to this, we used a different stock, Yelp (YELP).

### 5.1.3. Database Schema for Storing Stock Data



### 5.1.4. MySQL Commands

#### 1. Creation of Database

```
CREATE DATABASE appdata;
```

#### 2. Stocks Table

```
CREATE TABLE stocks (
```

```
sid INT PRIMARY KEY,  
ticker VARCHAR(10)  
);
```

### 3. Insert into Stocks

```
INSERT INTO TABLE stocks (sid, ticker)  
VALUES (711, 'NFLX');
```

### 4. Historical Table

```
CREATE TABLE historical (  
sid INT,  
dat DATE,  
open_value FLOAT,  
low FLOAT,  
high FLOAT,  
close_value FLOAT,  
volume INT,  
PRIMARY KEY (sid, dat),  
FOREIGN KEY (sid) REFERENCES Stocks (sid)  
);
```

As hist\_stocks table hosts the data of all stocks, sid alone cannot be a primary key, but sid along with date can form the primary key as there is only one record for one day per stock. However, sid is a foreign key in this table which references the primary key sid in the stock\_syms table.

### 5. RealTime Table

```
CREATE TABLE real_time (  
sid INT,  
dat DATE,  
tim TIME,  
open_value FLOAT,  
low FLOAT,  
high FLOAT,  
close_value FLOAT,  
volume INT,  
PRIMARY KEY (sid, dat, tim),  
FOREIGN KEY (sid) REFERENCES Stocks (sid)  
);
```

In the real time data table, we will have several records for each stock for each day. Hence, we make the collection of sid, dat and tim columns to be the primary key collectively.

Note: We named the columns as dat and tim to store the date and time values as we wanted to avoid the confusion with the predefined keywords 'date' and 'time' in the MYSQL server.

## **6. Initial Load of Historical Data**

We wanted to store the historical data of the past one year into the historical table to lay the data foundation for our prediction algorithms. For this purpose, we downloaded the historical data of all the ten stocks for the past one year and loaded the csv data directly into the database using the Load command of MySQL.

```
LOAD DATA INFILE 'path_to_csvfile/filename.csv'
INTO TABLE historical
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

### ***5.1.5. Update Database with Data from Web Source***

After setting up the database and the historical data, we then had to update the historical and real-time tables on a daily basis so that the data in the tables always stays up-to-date at any given point of time. To do this, we need to fetch the data on a daily basis from some web source hosting the stock data and update our database with this data. For this purpose of data fetch, we used an API called Yahoo Finance and coded this functionality using the Python language. We are using Python to build this module due to its ease of use and built-in functions for working with the MySQL database.

We initially used another API called as AlphaVantage but we switched to Yahoo Finance later as we observed two issues during the course of our project:

1. The AlphaVantage API had a daily and minute-wise call limit. It allowed only 5 API calls in a minute and as a result, we could only fetch data for 5 stocks within a minute using a single API key and then wait another minute to fetch data for the remaining 5 stocks. By doing this, we were losing a portion of the real-time data.
2. We observed that the API was not always fetching up-to-date data. Sometimes, it was only fetching data until the previous day or previous hour and this issue seemed also resulted in some data loss.

Hence, we researched all the other APIs available in the market and finally decided on using the Yahoo Finance API.

### **5.1.6. Connection to MySQL Database from Python**

Python has a mysql-connector package which helps us in connecting the MySQL database from our Python program so that we can access the database and make any necessary retrievals or updates to the data. In order to be able to use it, we need to install the mysql-connector-python package using pip command. This command for Windows is:

```
python -m pip install mysql-connector-python
```

The code to access the database is:

```
mydb = mysql.connector.connect(  
    host="<hostname>",  
    user="<username>",  
    passwd="<password>"  
)
```

We then create a cursor object on this database which is used in accessing the database and to execute the SQL queries on the data in the database.

```
mycursor = mydb.cursor()
```

### **5.1.7. Yahoo Finance API**

Yahoo Finance is an API that is useful to retrieve both historical and real-time stock data from the Yahoo Finance website. The yahoo\_fin is the package written for Python 3 which scrapes historical stock price data, in addition to providing current information on market caps, dividend yields, and which stocks comprise the major exchanges. Additional functionalities include scraping income statements, balance sheets, cash flows, holder information, and analyst data. The package includes the ability to get live stock prices, capture cryptocurrency data, and get the most actively traded stocks on a current trading day.

The yahoo\_fin package has two modules, which are:

1. stock\_info
2. options

stock\_info has numerous methods to obtain the stock information, while the options module can be used to import other methods. For our project, we have used two methods from the stock\_info module.

To use this API in python, we need to install the package using pip command.

```
python -m pip install yahoo_fin
```



Then, we need to import this package in our code.

```
import yahoo_fin as yf
```

### **a. Fetching Historical Data:**

We used the `get_data()` method from the API to fetch the historical data everyday. This method downloads historical price data of a stock into a python pandas data frame. It offers the functionality to pull daily, weekly, or monthly data. In our code, we pass two input parameters to this method. They are:

- **ticker:** Ticker symbol of the stock for which we want to fetch historical data
- **start\_date:** Passing the current date in this parameter returns one historical data record of the current day.

The code snippet for using this method is:

```
data = yf.get_data(ticker=stock, start_date=date.today()),
```

Where stock can be any stock ticker symbol like 'NFLX', 'AAPL' and so on.

As it returns a dataframe object, we access the required values using the following code:

```
open_value = data.iloc[0, :]['open']
close_value = data.iloc[0, :]['close']
low = data.iloc[0, :]['low']
high = data.iloc[0, :]['high']
volume = data.iloc[0, :]['volume']
```

We then write this data to the historical table in our database using the following code:

```
dat = date.today()
sql = "INSERT IGNORE INTO historical
(sid,dat,open_value,low,high,close_value,volume) VALUES
(%s,%s,%s,%s,%s,%s,%s)"
val = (str(sid), str(dat), str(open_value), str(low), str(high),
str(close_value), str(volume))
mycursor.execute(sql, val)
mydb.commit()
```

### **b. Fetching Real-Time data:**

We used the `get_quote_table()` method of the Yahoo Finance API to fetch real-time data. This function basically scrapes the primary table found on the quote page of an input stock symbol. The possible parameters are:

- **ticker:** The stock symbol for which the data is being scraped by the API.

- **dict\_result:** This is a boolean parameter and is set to True by default. This implies that the function returns the results in a dictionary format; otherwise the results are returned in a data frame.

Method invocation code:

```
details = yf.get_quote_table(stock)
```

We access the attributes similar to the historical data fetch code and store them in the real\_time table.

### 5.1.8. Scheduling Data Collection Module

As the data needs to be up-to-date at any given point of time, we scheduled our data collection program in the web-service using the APScheduler package of Python. The stock market timings are 9:30 AM to 4:00 PM on weekdays.

#### Scheduling Historical Data Fetch

Hence, the historical data fetch happens at 5:00 PM as the stock market has to close for the day in order for the historical record to be generated by Yahoo Finance. This way, our historical data table gets updated with one record of the current day on every working day.

#### Scheduling Real-Time Data Fetch

For real time data fetch, we scheduled it to run only on weekdays and in between the above timings as there won't be any new data in the other timings. However, real-time data fetch occurs every minute, that is, the method gets invoked every minute between 9:30 AM to 4:00 PM on every working day and the real\_time data table gets updated every minute.

By doing this, we are making sure that our data is guaranteed to stay up-to-date at any given point of time.

### 5.1.9. Screenshots for Stock Data Tables

a. **SELECT \* FROM stocks;**

	sid	ticker
▶	365	AAPL
	267	AMZN
	146	FB
	178	GOOGL
	711	NFLX
	413	TRIP
	579	TSLA
	292	TWTR
	632	VAC
	971	YELP

b. `SELECT * FROM historical;`

	sid	dat	open_value	low	high	close_value	volume
	146	2020-04-28	188.66	182.57	189.2	182.91	19309256
	292	2020-04-28	30.4	28.5734	30.45	28.79	22925081
	365	2020-04-28	285.08	278.2	285.83	278.58	26219110
	632	2020-04-28	83.01	76.425	83.24	78.96	738018
	178	2020-04-28	1283.2	1230.65	1284.76	1232.59	2905828
	711	2020-04-28	419.99	402.91	421	403.83	9862533
	267	2020-04-28	2372.1	2306	2373	2314.08	4916361
	971	2020-04-28	21.6	20.17	21.79	20.4	1164013
	413	2020-04-28	19.07	18.66	19.48	19.1	2382171
	579	2020-04-28	795.64	756.9	805	769.12	15035721
	971	2020-03-27	20.5	18.02	20.67	18.59	1222961
	711	2020-03-27	359.09	353	366.54	361.769	3840298
	413	2020-03-27	18.94	17.76	19.19	18.015	1217918
	146	2020-03-27	158.2	154.75	158.56	155.825	13098769
	365	2020-03-27	252.75	248.38	252.92	250.67	22525784
	579	2020-03-27	505	494.04	510.53	505.38	7375075
	178	2020-03-27	1127.47	1108.44	1133.77	1115.89	1412932
	292	2020-03-27	25.56	24.76	25.82	24.98	10074145
	632	2020-03-27	58.21	56.415	66.07	61.355	315779

c. `SELECT * FROM real_time;`

	sid	dat	tim	open_value	low	high	close_value	volume
▶	632	2020-05-01	14:53:00	79.77	75.78	80.2	77.1	266061
	146	2020-05-01	14:53:00	201.6	199.05	207.28	201.43	23306532
	178	2020-05-01	14:53:00	1324.09	1309.66	1351.43	1312.28	1671077
	267	2020-05-01	14:53:00	2336.8	2258.25	2362.44	2302.61	8168675
	579	2020-05-01	14:53:00	755	683.04	772.77	704.201	28567769
	292	2020-05-01	14:53:00	28	27.19	28.44	27.8894	22563041
	365	2020-05-01	14:53:00	286.25	285.85	299	291.15	49196132
	971	2020-05-01	14:53:00	21.49	20.56	21.8	21.8	879865
	711	2020-05-01	14:53:00	415.1	411.73	427.97	416.82	6651638
	413	2020-05-01	14:53:00	19.5	18.39	19.52	18.66	1129571
	971	2020-05-01	14:52:00	21.49	20.56	21.79	21.77	876070
	146	2020-05-01	14:52:00	201.6	199.05	207.28	201.3	23269334
	711	2020-05-01	14:52:00	415.1	411.73	427.97	416.82	6643677
	178	2020-05-01	14:52:00	1324.09	1309.66	1351.43	1312.5	1659047
	267	2020-05-01	14:52:00	2336.8	2258.25	2362.44	2301.52	8157657
	292	2020-05-01	14:52:00	28	27.19	28.44	27.86	22503212
	365	2020-05-01	14:52:00	286.25	285.85	299	290.89	49128373
	413	2020-05-01	14:52:00	19.5	18.39	19.52	18.68	1127835
	579	2020-05-01	14:52:00	755	683.04	772.77	703	28529970

## **5.2. Prediction Algorithms and Indicators Used**

We have written algorithms for two types of predictions, one is Short-Term Prediction and the other is Long-Term Prediction.

### ***5.2.1. Short Term Prediction***

This type of prediction is done by using real-time data. As the real-time data is minute-wise data, short term prediction also provides minute-wise predictions for the future as it makes use of the real-time data and hence observes patterns in the data in a minute-wise fashion. Therefore, using short term prediction, we can provide stock price predictions for the next few minutes on the same day. This type of prediction is particularly useful for day traders. Day traders are the stock market participants who participate in day trading. Day trading is a strategy in which stock traders buy and sell throughout the day with a goal of making small profits with each trade. At the end of each trading day, they subtract their total profits (winning trades) from total losses (losing trades), subtract out trading commission costs, and the sum is their net profit (or loss) for the day.

#### ***5.2.1.1. Inputs and Outputs for Short Term Prediction***

The inputs to the prediction function are:

- Stock Ticker Symbol (Eg: 'NFLX')
- Number of Predictions (Eg: 5)

The outputs from the function are:

- List of Stock Price predictions whose length will be the number of predictions given as input to the function.
- Suggestion (either BUY or SELL)

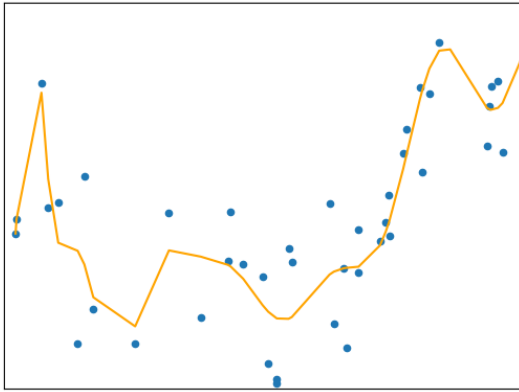
This prediction function uses the Polynomial Curve Fitting algorithm to make future stock price predictions. Besides the algorithm, it uses technical indicators such as EMA, RSI and MACD to return buy or sell suggestions as well. These indicators will be discussed in section 4.2.3.

#### ***5.2.1.2. Polynomial Curve Fitting for Short Term Predictions***

We have used the polynomial curve fitting algorithm for short-term prediction of stock prices. The algorithm uses real-time stock data to predict the future stock prices for the near future, say a few minutes or hours.

## What is Polynomial Curve Fitting?

Given a set of data points  $(x_i, y_i)$ ,  $i = 1, 2, 3, \dots, N$  where  $x$  is the input value to which  $y$  is the corresponding target value. Polynomial curve fitting finds out the underlying regularity in this set of data points.



This regularity can be modeled in the following way:

$$\hat{y}(x, w) = \sum_{k=0}^M w_k x^k, \text{ where}$$

$M$  – order of the polynomial

$w$  – vector of polynomial coefficients

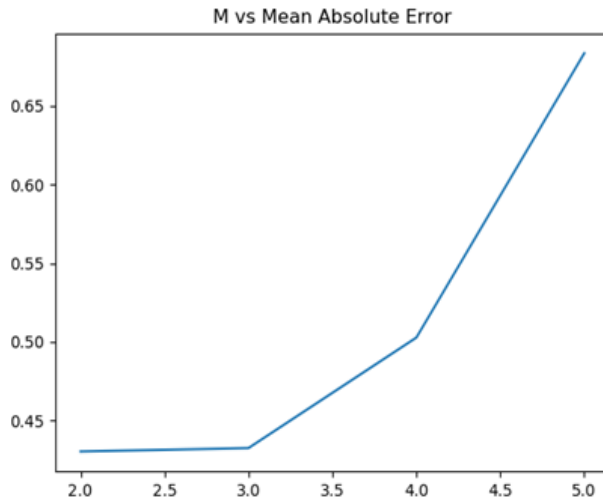
It then uses this regularity to determine the target value  $y'$  when we are provided with a new input value  $x'$ . Initially, we set random values of  $w$  and  $M$ . We then need to fit the model using the training data and use this fit model to make predictions. Fitting is the process of updation of the parameters  $w$  and  $M$  based on the training data which is used in the fitting process.

### Determining and Updating $M$ :

Picking an appropriate order of the polynomial is important because otherwise, it might result in the overfit or underfit of data. As a general rule, we should pick  $M$  at least several orders lesser than the number of data points.

We analyzed the performance of the algorithm for various sizes of the dataset at hand with different values of  $M$  and accordingly set its value to maximize the performance of the algorithm.

We tried updating the value of  $M$  using the rule:  $N \geq 5(M+1)$  to  $N \geq 10(M+1)$ . Therefore, according to this rule, we tried different values of  $M$ . However, when we analyzed the performance of the algorithm with these values, we observed that the algorithm performed better with least prediction errors at  $M = 2$  for any set of training data. Hence, we fixed the value of  $M$  to be 2.



### Determining and Updating w:

We also auto-adjusted the weight vector of polynomial coefficients based on the datasets so that it will minimize the least squares error and make accurate predictions.

We wish to minimize the error/cost function. This can be given by:

$$E(w) = \frac{1}{2} \sum_{i=1}^N (t_i - y(x_i, w))^2$$

where,

$t_i$  is the observed/actual value at time  $i$

$y(x_i, w)$  is the predicted value at time  $i$ .

$$E(w) = \sum_{i=1}^N (t_i - (\sum_{k=0}^M w_k x^k))^2 = \sum_{i=1}^N (t_i - (w_0 + \sum_{k=1}^M w_k x^k))^2$$

To find  $w^*$  which minimizes  $E(w)$ , we need to find the partial derivative of  $E(w)$  w.r.t.  $w$ :

$$\frac{dE}{dw_0} = - \sum_{i=1}^N (t_i - (w_0 + \sum_{k=1}^M w_k x^k)) x^0 = 0,$$

$$\frac{dE}{dw_1} = - \sum_{i=1}^N (t_i - (w_0 + \sum_{k=1}^M w_k x^k)) x^1 = 0,$$

$$\frac{dE}{dw_2} = - \sum_{i=1}^N (t_i - (w_0 + \sum_{k=1}^M w_k x^k)) x^2 = 0,$$

.

$$\frac{dE}{dw_M} = - \sum_{i=1}^N (t_i - (w_0 + \sum_{k=1}^M w_k x^k)) x^M = 0.$$

Putting these equations in the matrix form, we obtain

$$\begin{bmatrix} 1 & \dots & x_1^M \\ \vdots & \ddots & \vdots \\ 1 & \dots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_N \end{bmatrix},$$

which can be written as  $X.w = T$

In matrix notation, the polynomial fit is given by the equation  $y = X.a$ , where  $y = T$  and  $a = w$  in this case.

Multiplying both sides with  $X^T$ ,

$$X^T . X . w = X^T . T,$$

$$w^* = (X^T X)^{-1} X^T . T$$

Therefore, this is how we determined and kept updating the weight vector based upon the data such that it minimizes the least squares errors in the predictions.

#### **5.2.1.3. Input and Output to the Curve Fitting algorithm:**

The input in the input-output pairs for the curve fitting algorithm will be the time points (eg: 1, 2, 3..so on) and the output will be the price at that point of time. Therefore, the first data point in the training set will have input value  $x = 1$  and if we have  $N$  data points, then we will find out the price for the input  $N+1$ , that is at the  $(N+1)$ th point of time.

### **5.2.2. Long Term Prediction**

This type of prediction is done by using historical data. As the historical data is day-wise data, long term prediction also provides day-wise predictions for the future as it makes use of the historical data and hence observes patterns in the data in a day-wise fashion. Therefore, using long term prediction, we can provide stock price predictions for the next few days. This type of prediction is particularly useful for long term traders in the stock market. Long term traders are the stock market participants who participate in long term trading. Long term trading, otherwise known as position trading, refers to a trading style in which the trader will hold on to a position for an extended period of time. A position trade can last anywhere from a few weeks to a couple of years.

#### **5.2.2.1. Inputs and Outputs for Long Term Prediction**

The inputs to the prediction function are:

- Stock Ticker Symbol (Eg: 'NFLX')
- Number of Predictions (Eg: 5)

The outputs from the function are:

- List of Stock Price predictions whose length will be the number of predictions given as input to the function.
- Suggestion (either BUY or SELL)

This prediction function uses Artificial Neural Networks to make future stock price predictions. Besides the algorithm, it uses technical indicators such as EMA, RSI and MACD to return buy or sell suggestions as well. These indicators will be discussed in the next sub-section.

#### 5.2.2.2. Artificial Neural Networks for Long Term Predictions

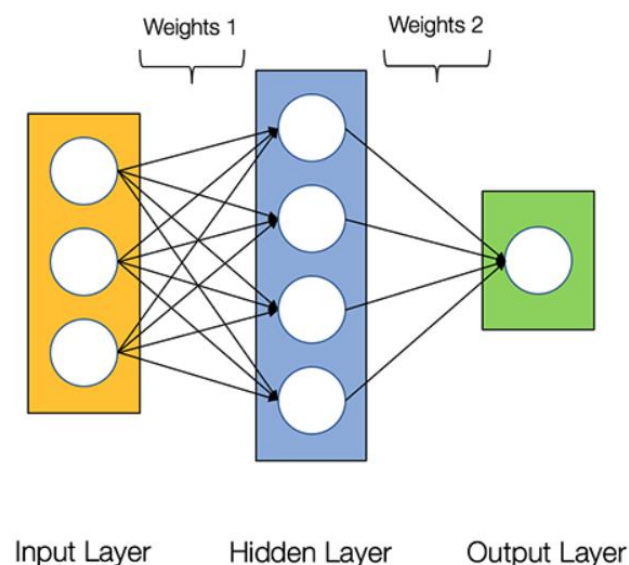
We have used the Artificial Neural Networks for long term prediction of stock prices. The algorithm uses historical stock data to predict the future stock prices for the long term, that is the upcoming days.

#### Concept behind Artificial Neural Networks

Neural Networks are basically a mathematical function that map a given input to a desired output. They consist of the following components:

- An input layer,  $x$
- An arbitrary amount of hidden layers
- An output layer,  $\hat{y}$
- A set of weights and biases between each layer,  $W$  and  $b$
- A choice of activation function for each hidden layer,  $\sigma$ .

The diagram below represents the architecture of a two-layer neural network.





The output  $\hat{y}$  of a simple 2-layer Neural Network is:

$$\hat{y} = \sigma(W_2 \sigma(W_1 x + b_1) + b_2)$$

The right values for the weights and biases determines the strength of the predictions. The process of fine-tuning the weights and biases from the input data is known as training the Neural Network.

Each iteration of the training process consists of the following steps:

- Calculating the predicted output  $\hat{y}$ , known as feedforward
- Updating the weights and biases, known as backpropagation

The equation above represents the feedforward step. The Loss Function allows us to evaluate the goodness of the model's predictions and our goal in training is to find the best set of weights and biases that minimizes the loss function. We can use any kind of loss functions such as RMSE, MAE and so on. We have used the MSE (Mean Squared Error) as our Loss function, which is the mean of the squared errors in all predictions. It can be given by the following equation:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

In backpropagation, we update the weights and biases depending on the amount of loss resulting from the loss function above. In order to do this, we need to know the derivative of the loss function with respect to the weights and biases. If we have the derivative, we can simply update the weights and biases by increasing/reducing with it (refer to the diagram above). This is known as gradient descent.

We use the Chain rule as below to obtain the derivatives needed for backpropagation.

$$Loss(y, \hat{y}) = \sum_{i=1}^n (y - \hat{y})^2$$

$$\frac{\partial Loss(y, \hat{y})}{\partial W} = \frac{\partial Loss(y, \hat{y})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial W} \quad \text{where } z = Wx + b$$

$$= \frac{2(y - \hat{y})}{n} * \text{derivative of actvtn function} * x$$

$$= \frac{2(y - \hat{y})}{n} * z(1-z) * x$$

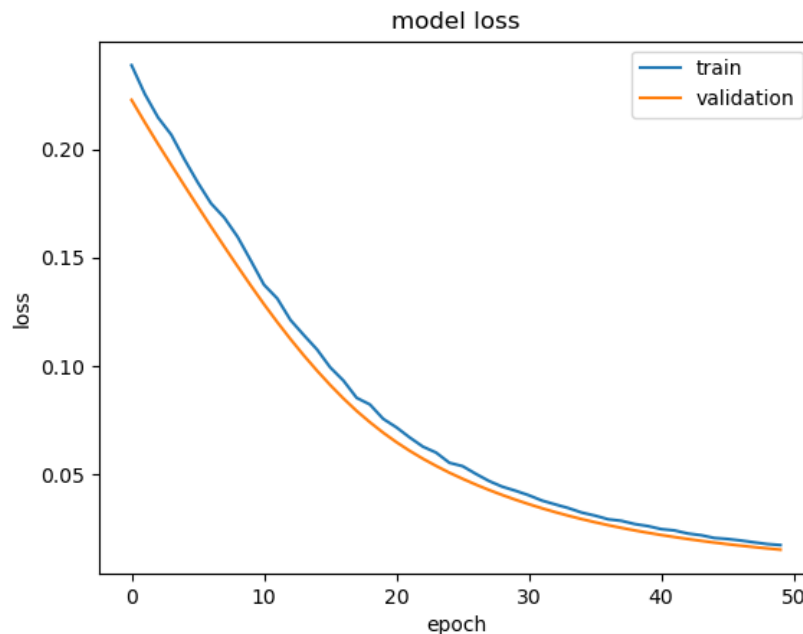
### Our Implementation of the Neural Network:

In our case, we are performing Supervised Machine Learning using regression in neural networks. We used a neural network with three layers, two hidden layers and one output layer. The first hidden layer has 10 neurons while the second hidden layer comprises 8

neurons. We have used the relu activation function on the hidden layers and linear activation function on the output layer as these are the most suitable for regression purposes. Regression and classification are categorized under the same umbrella of supervised machine learning. The main difference between them is that the output variable in regression is numerical (or continuous) while that for classification is categorical (or discrete).

### **Training our Model:**

We train the model by splitting the dataset as 80 percent training and 20 percent testing data and iterate this process for 50 epochs. As we can see from the plot below, the loss decreases for both training and testing set with each increasing epoch.



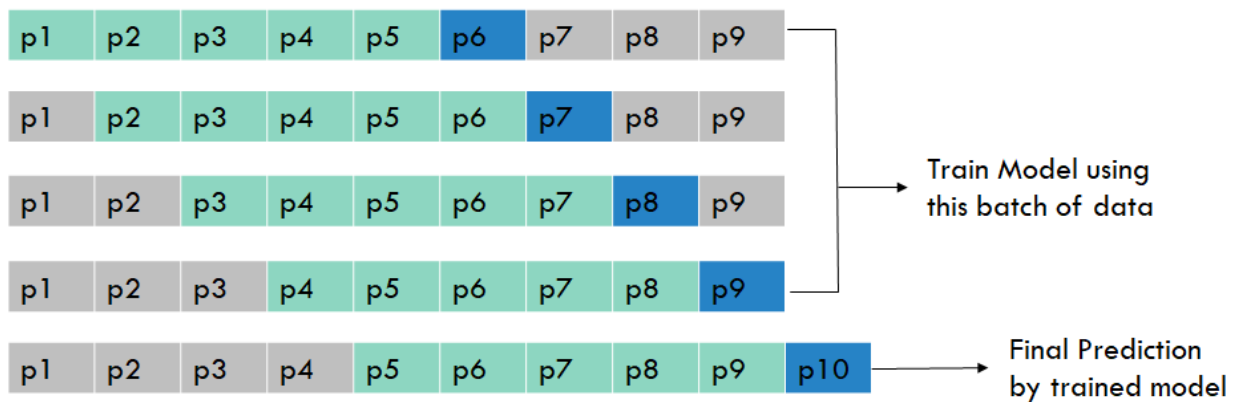
Once the model is fit using the training data, we then use this model to make the stock price predictions for future.

#### ***5.2.2.3. Input and Output to the Neural Network:***

Both the input and output to the network will be the stock prices. We used the concept of a sliding window to provide the inputs and outputs to the neural network. This can be used to recognize the patterns in the price movements. For example, if there are a set of 9 prices which we consider as the historical data for training the network and the size of the sliding window is 5, then we train the network using the following set of prices as input and outputs.

Input	Output
Price1, Price2, Price3, Price4, Price5	Price6
Price2, Price3, Price4, Price5, Price6	Price7
Price3, Price4, Price5, Price6, Price7	Price8
Price4, Price5, Price6, Price7, Price8	Price9

We train the network with the above set of training data for 50 epochs. We then make the stock price prediction for the next day which is Price10 by passing the input [Price5, Price6, Price7, Price8, Price9] to the neural network. To summarize using the following diagram for a better understanding:



Where, p1, p2...p9 are the 1st, 2nd.....9th prices in the list of prices used as training data to train the model to predict p10th price. Example, if we want to predict the price for tomorrow which is p10, we then consider p1 to p9 as the stock prices for the past 9 days including the current day.

### 5.2.3. Technical Indicators Used

We made use of three technical indicators to observe the trends for the stocks based on the past, present and predicted data and used this observation to provide a buy or sell suggestion accordingly. The data which we send to the strategy using the technical indicators to make a suggestion will include the data of the stock which we have in our database and also the predicted prices of the stock which we predicted using our short term and long term algorithms. For example, if we predicted five prices for the stock AAPL using 50 prices of that stock from our database, then we will send the 50+5 = 55 prices for AAPL to the Indicator Model strategy to come up with a buy/sell/hold suggestion.

The three indicators we used are:

### **1. Exponential Moving Average (EMA)**

Moving averages act as a technical indicator to show you how a security's price has moved, on average, over a certain period of time. Moving averages are often used to help highlight trends, spot trend reversals, and provide trade signals.

An exponential moving average (EMA) is a type of moving average (MA) that places a greater weight and significance on the most recent data points. The exponential moving average is also referred to as the exponentially weighted moving average. An exponentially weighted moving average reacts more significantly to recent price changes than a simple moving average (SMA), which applies an equal weight to all observations in the period. Traders often use several different EMA lengths, such as 10-day, 50-day, and 200-day moving averages.

The 12- and 26-day exponential moving averages (EMAs) are often the most quoted and analyzed short-term averages. The 12- and 26-day are used to create indicators like the moving average convergence divergence (MACD) and the percentage price oscillator (PPO). In general, the 50- and 200-day EMAs are used as indicators for long-term trends. When a stock price crosses its 200-day moving average, it is a technical signal that a reversal has occurred.

EMA is calculated in the following way:

$$\text{EMA} = \text{Price}(t) \times k + \text{EMA}(y) \times (1 - k)$$

where:

t=today,

y=yesterday,

N=number of days in EMA,

$$k = 2 \div (N + 1)$$

### **2. Relative Strength Index (RSI)**

The relative strength index (RSI) is a momentum indicator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset. The RSI is displayed as an oscillator (a line graph that moves between two extremes) and can have a reading from 0 to 100. An asset is usually considered overbought when the RSI is above 70% and oversold when it is below 30%.

The RSI was designed to indicate whether a security is overbought or oversold in relation to recent price levels. The RSI is calculated using average price gains and losses over a given period of time. The default time period is 14 periods with values bounded from 0 to 100.

In an uptrend or bull market, the RSI tends to remain in the 40 to 90 range with the 40-50 zone acting as support. During a downtrend or bear market the RSI tends to stay between the 10 to 60 range with the 50-60 zone acting as resistance.

RSI is calculated using the following formula:

$$RSI = 100 - [100 / (1 + (\text{Average of Upward Price Change} / \text{Average of Downward Price Change}))]$$

### ***3. Moving Average Convergence/Divergence (MACD)***

Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. The MACD is calculated by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA.

The result of that calculation is the MACD line. A nine-day EMA of the MACD called the "signal line," is then plotted on top of the MACD line, which can function as a trigger for buy and sell signals. Traders may buy the security when the MACD crosses above its signal line and sell - or short - the security when the MACD crosses below the signal line.

The formula for MACD for short-term period is:

$$MACD = 12\text{-Period EMA} - 26\text{-Period EMA}$$

The formula for MACD for long-term period can be:

$$MACD = 50\text{-Period EMA} - 200\text{-Period EMA}$$

We came up with the following strategy wherein we collectively used all of the above three indicators to understand the market trends and momentum and then finally give a buy or sell suggestion.

#### **Strategy using Technical Indicators to provide Buy/Sell suggestion:**

- Calculate the EMA for 26 data points (EMA26) and 12 data points (EMA12) for short term or calculate the EMA for 50 data points (EMA50) and 200 data points (EMA200) for long term.
- Calculate the MACD from these values by subtracting EMA26 from EMA12 (for short term) or subtracting EMA200 from EMA50 (for long term).
- If the MACD is positive, then it means that the market is in an uptrend. Similarly, if MACD is negative, then it means that the market is in a downtrend.
- To verify if the market is going to stay in the observed trend for a significant amount of time, we use RSI.
- We then calculate RSI by considering 14 data points (RSI14) for short term and calculate RSI for 70 data points (RSI70) for long term.

- If RSI is below 30, then it means that the market is oversold, and the trend is likely to reverse. This in turn means that the price is going to increase and that the market is about to go into an uptrend. If RSI is above 70, then it means that the market is overbought, and the trend will reverse. This means that there will be a downtrend soon. If RSI is between 30-70, then whatever trend is observed, it will continue.
- Using the above points, we came up with the following scenarios to determine if it is profitable to buy or sell the stock.
  - MACD shows uptrend
    - ❑ If RSI < 70, then suggestion will be BUY
    - ❑ If RSI > 70, then suggestion will be HOLD
  - MACD shows downtrend
    - ❑ If RSI > 30, then suggestion will be SELL
    - ❑ If RSI < 30, then suggestion will be HOLD

By using the combination of the prediction algorithms and the technical indicators as described above, we predict the future stock prices and provide a buy, hold or sell suggestion accordingly.

## 5.3. Web Service

We have used the Django REST framework to implement the Web Service for our project. One of the main reasons why we picked Django to build the web service for our project was because we have coded all our backend in Python. As Django is built on Python, we figured it would be more efficient to use Django as the integration would be smoother. The Django framework is well documented which helps beginners to build a web service efficiently.

Also, Django has a lot to offer and has the ability to accommodate any modern web apps structures. Stacking Django and Javascript web frameworks to build modern web apps is one of the best ways to stack backend and frontend frameworks. The web server developed by our team runs on the local host as of now and it facilitates the execution of all the project requirements right from the data collection phase, invoking the prediction algorithms, maintaining the admin and user databases and up until managing the user interface.

### 5.3.1. Implementation

Firstly, we created a virtual environment called 'virtualstockenv' for our application. What is a Virtual Environment?

At its core, the main purpose of Python virtual environments is to create an isolated environment for Python projects. This means that each project can have its own dependencies, regardless of what dependencies every other project has.

The command to create the Python virtual environment in Windows is:

```
$ python -m venv <name_of_virtual_environment>
```

In our case it was,

```
$ python -m venv virtualstockenv
```

We then activated the virtual environment

```
$ cd virtualstockenv
```

```
$ cd Scripts
```

```
$ activate.bat
```

This command will activate the virtual environment and get it running.

We then installed the modules necessary for Django to function.

```
$ python -m pip install django djangoframework
```

We then created a project called 'virtualstockenv' inside the virtual environment and an application called 'stockapp' inside this project.

```
$ django-admin startproject virtualstockenv
```

```
$ cd virtualstockenv
```

```
python manage.py startapp stockapp
```

We then made changes in the virtualstockenv/settings.py file as below by adding additional lines.

```
INSTALLED_APPS = [  
    'virtualstockenv',  
    'stockapp.apps.StockAppConfig',  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'users.apps.UsersConfig',  
    'rest_framework',  
    'crispy_forms',  
]
```

The settings.py file is an important component of the server. It holds sensitive data used by the server. The information about the configurations necessary for the application is stored in this file. It also holds the information of the database like name of the database,

username, password, host name and the port number that can be used by the web service we created to access our local database. This file also holds the information of the time zones.

We need to be able to use the stock data we have stored in our database during the Data Collection phase. For this, we first need to connect to the MySQL database management system. To facilitate this, we need to update the same settings.py file as below.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': "<database_name>",
        'USER': "<username>",
        'PASSWORD': "<password>",
        'HOST': "localhost",
        'PORT': "3306",
    }
}
```

Next step we followed was generating the models for our data. A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you're storing. Django web applications access and manage data through Python objects referred to as models. Generally, each model maps to a single database table. Models define the structure of stored data, including the field types and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc.

We can either create new models or already use the existing database schema to let Django to automatically generate the models for us using the existing schema. As we already had the database schema and the data set up, we used the inspectdb command of Django to get automatically generated models and saved these models into a file called models.py.

```
$ python manage.py inspectdb > models.py
```

This command generated all the models into the models.py file. We then moved this file from /virtualstockenv to /virtualstockenv/stockapp to be able to use it for data access operations.

We then ran the migrate command to install any extra needed database records such as admin permissions and users and content types

```
$ python manage.py migrate
```



In this way, all the data we had in our database initially was then linked to the Django Framework such that we could access the data in all the database tables from the application through the Models generated for each table.

For example: The schema of our stocks table is as below.

	Field	Type	Null	Key
►	sid	int	NO	PRI
	ticker	varchar(10)	NO	

The model generated by Django for the above table is as follows:

```
class Stocks(models.Model):  
    sid = models.IntegerField(primary_key=True)  
    ticker = models.CharField(max_length=10)  
    class Meta:  
        managed = False  
        db_table = 'stocks'
```

The sid and ticker attributes of the Stocks Model correspond to the fields in the stocks table as the value of db\_table = 'stocks'. These fields are mapped to IntegerField and CharField pertaining to the data types of these fields in the stocks table in the database.

We then create an admin account for our application by executing the below command. The admin is responsible for maintaining the application and he has extra privileges to the application and the data compared to the other application users. He can also add and delete user accounts from his account.

```
python manage.py createsuperuser
```

This command asks us to enter a username and password for the admin account. Upon entering these details, the admin account is created.

We then make use of the urls.py file in virtualstockenv/stockapp/ folder to register the URL endpoints for our application.

A sample endpoint registered in our urls.py file is:

```
path('Contact/', views.contact, name='contact'),
```

The first argument represents the endpoint which is localhost:<port\_number>/stockapp/Contact. The second argument represents the function which will be invoked when this endpoint is hit. This functionality will be a part of the views.py file in the virtualstockenv/stockapp/ folder. The third argument represents the name for this endpoint so that we refer to it in the other files on the service.

All the functionalities which are invoked when the application endpoints are hit are stored in the views.py file. Some of these functionalities also invoke the backend algorithms for predictions.

We have scheduled the data collection code and the code for thresholds to run at the specified intervals using APScheduler. To use Django for scheduling, we created a new folder called stockupdate in virtualstockenv folder. We placed the data\_collection.py file in this folder. We then created a new python file called updater.py in the same folder to write the code for scheduling. The scheduling code is written in this file under the start() method.

By using this scheduling code, whenever we start the server, Django will automatically start the scheduler and invokes the data fetch code and thresholds code at the specified intervals.

- We invoke the get\_real() function of data\_collection.py file every minute between 10:00 AM and 4:00 PM on stock market working days.
- We invoke the get\_hist() function of data\_collection.py file once at 5:00 PM on all stock market working days.
- We invoke the thresholds (used for NotifyMe) similar to the get\_real() function, that is, every minute on weekdays between stock market timings.

To render the output from the functionalities in the views.py file to the UI HTML pages, we create a table called templates inside stockapp folder and place all our HTML files in that location. We return the output from a function in the views.py to a HTML page in the following way:

```
return render(request, 'stockapp/<file_name.html>', context)
```

The context is a python dictionary object which will contain the output which we want to display on the HTML file which we specify as the second parameter in the code above.

We also need to send data like user inputs from the HTML page to the functionalities in the views.py file in the form of POST requests so that we will be able to make use

of this data and return the desired output to the user. Example: To get the stock name input by the user labelled with the name 'stock' on the UI, we use the following code:

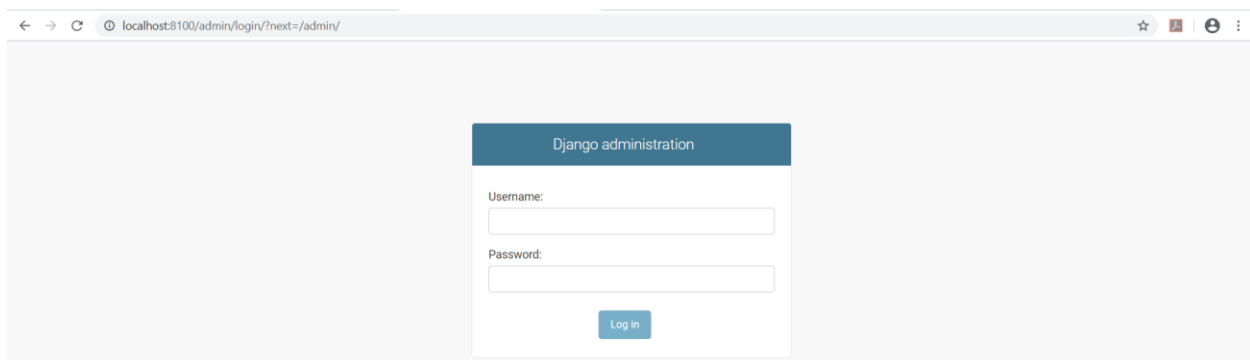
```
stock_name = request.POST.get('stock')
```

We then run the server by running the command from the virtualstockenv/ folder:

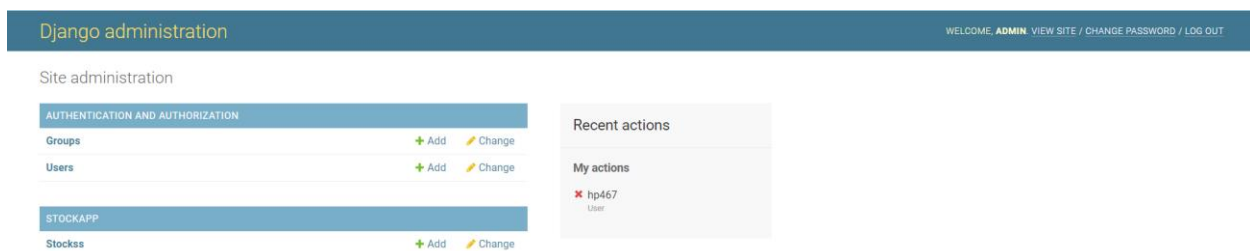
```
python manage.py runserver 8100
```

This will start the server on the localhost on the port 8100. We can then hit the URL <http://localhost:8100> and we will be redirected to the user landing page.

To access the admin account, we need to go to the URL <http://localhost:8100/admin>



The below page is seen once the admin logs in. The recent action shows that the admin deleted the user account with username 'hp467'. The admin has the options of Changing Password and Logging out of the application on the navbar on the right.



The admin can add or delete users from the application from the below page.

Django administration

WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · Authentication and Authorization · Users

Select user to change

ADD USER +

Q  Search

Action:  Go 0 of 4 selected

<input type="checkbox"/>	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	Admin	stockupalerts@gmail.com			
<input type="checkbox"/>	hp467	hp467@scarletmail.rutgers.edu	Hari Priya	Ponnakanti	
<input type="checkbox"/>	php16	haripriyaponnekanti@gmail.com	Priya	Ponnekanti	
<input type="checkbox"/>	ps1029	ps1029@rutgers.edu	Pranav	Shivkumar	

4 users

FILTER

By staff status

All  
Yes  
No

By superuser status

All  
Yes  
No

By active

All  
Yes  
No

The admin can also manage the stocks table by either adding, updating or deleting records from the table through the below page.

Django administration

WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · Stockapp · Stockss

Select stocks to change

ADD STOCKS +

Action:  Go 0 of 10 selected

<input type="checkbox"/>	STOCKS
<input type="checkbox"/>	Stocks object (971)
<input type="checkbox"/>	Stocks object (711)
<input type="checkbox"/>	Stocks object (632)
<input type="checkbox"/>	Stocks object (579)
<input type="checkbox"/>	Stocks object (413)
<input type="checkbox"/>	Stocks object (365)
<input type="checkbox"/>	Stocks object (292)
<input type="checkbox"/>	Stocks object (267)
<input type="checkbox"/>	Stocks object (178)
<input type="checkbox"/>	Stocks object (146)

10 stockss

## 5.4. User Interface

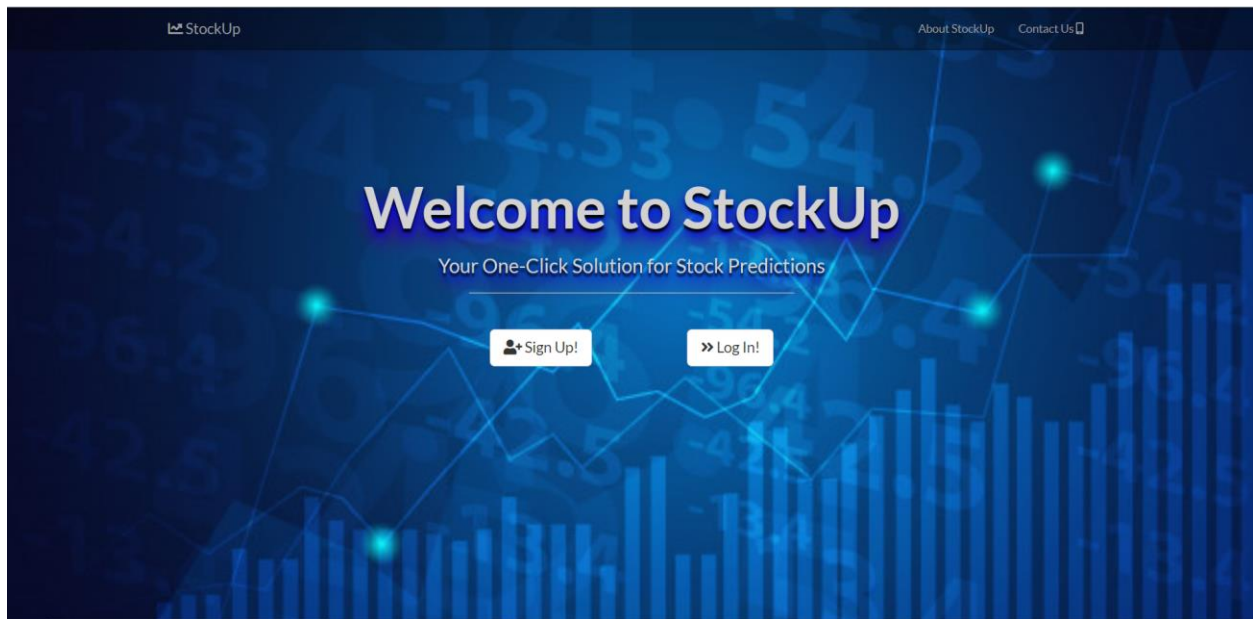
We used HTML, CSS and Bootstrap to design the User Interface. The UI covers all the use-cases mentioned in Section 4.

### 1. Landing Page

This is the first page which the users will see when they visit our application website. It displays the title of our application, which is 'StockUp' and the tagline. It mainly has two options, one for signing up for a new account on the website and another is to login to an already existing user account.

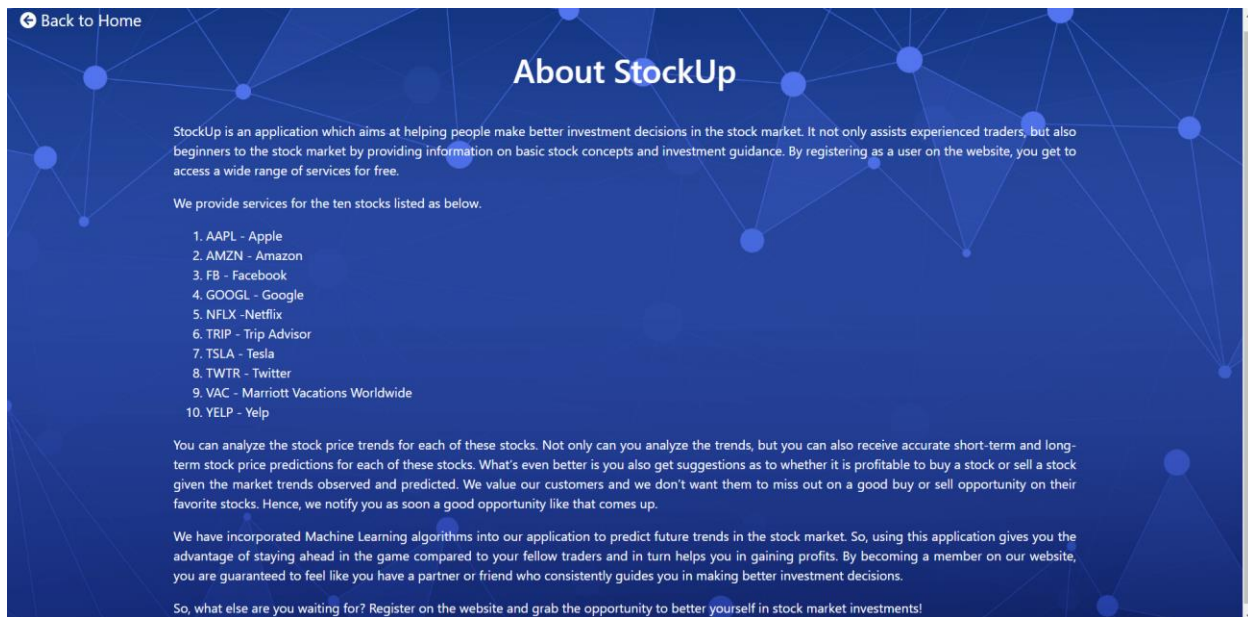
- New users can choose the Sign Up option
- Already existing users can choose the Login option

Besides these options, there are two more options for users even before signing up or logging in to the application. One option is the About StockUp page, another option is the Contact Us page which we will describe in detail in the next screenshots.



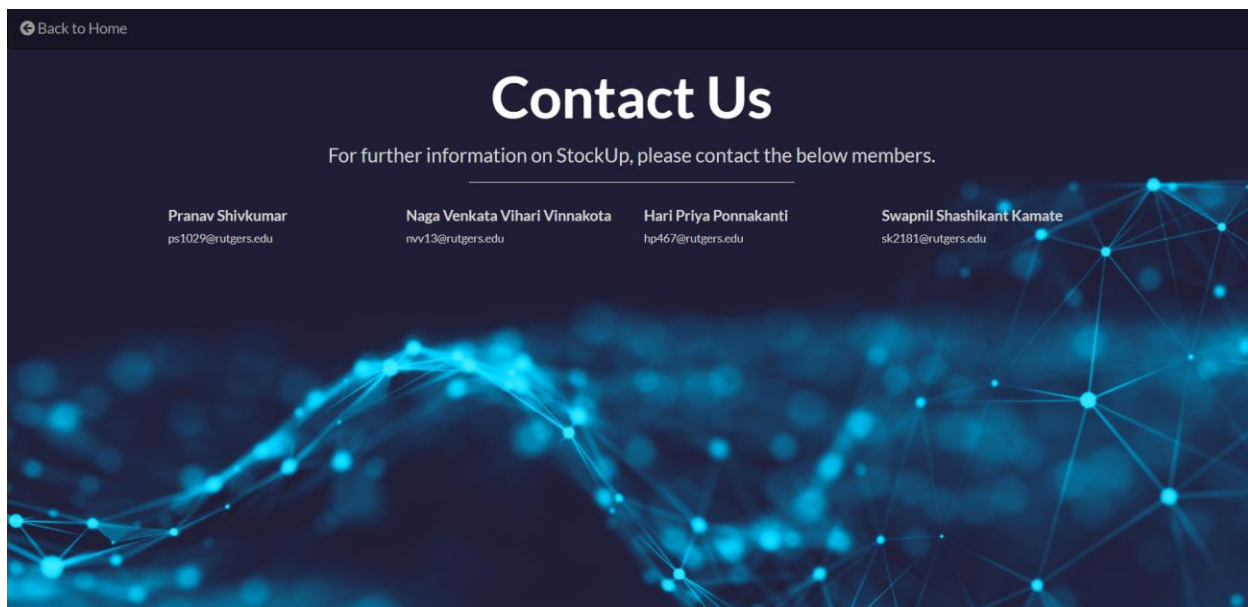
### 2. About StockUp Page (Use Case: 7)

This page helps in giving users a brief overview about our application and the benefits they will gain upon signing up on our website and becoming a user. They can see what all stocks they can get stock information and performance analysis for and what all services they can make use of once they become an application user.



### 3. Contact Us Page (Use Case: 19)

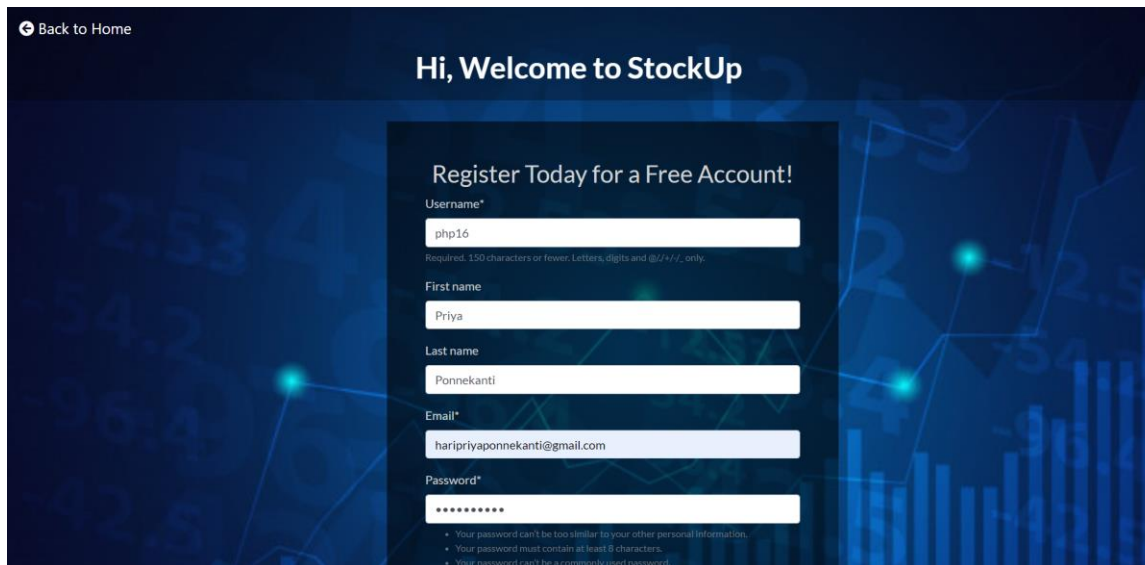
This page displays the names and contact details of all the application owners. Users can use these details to contact any of the owners to either get more information on StockUp or to provide any feedback about the application.



#### 4. Sign-Up/Registration Page (Use Case: 1)

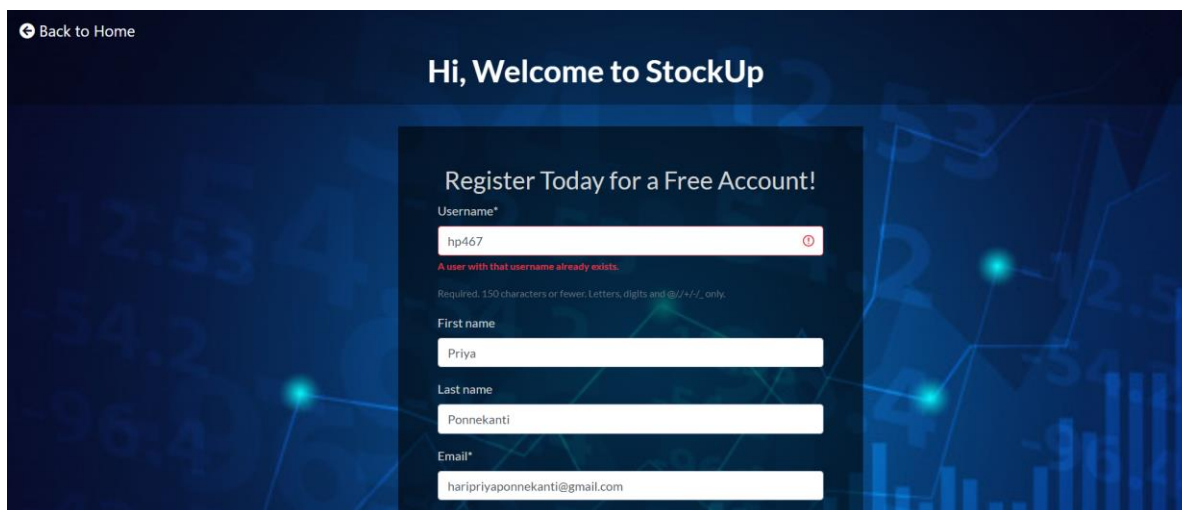
This page enables the user to create a user account and register as a user on our application. The user will have to provide the following details on the Sign-Up page:

- Username
- First name
- Last name
- Email Address
- Password



The screenshot shows a registration form titled "Hi, Welcome to StockUp" and "Register Today for a Free Account!". The form fields are filled with the following data: Username: php16, First name: Priya, Last name: Ponnekanti, Email: haripriyaponnekanti@gmail.com, and Password: a masked password. Below the password field, there are three bullet points: "Your password can't be too similar to your other personal information.", "Your password must contain at least 8 characters.", and "Your password can't be a commonly used password."

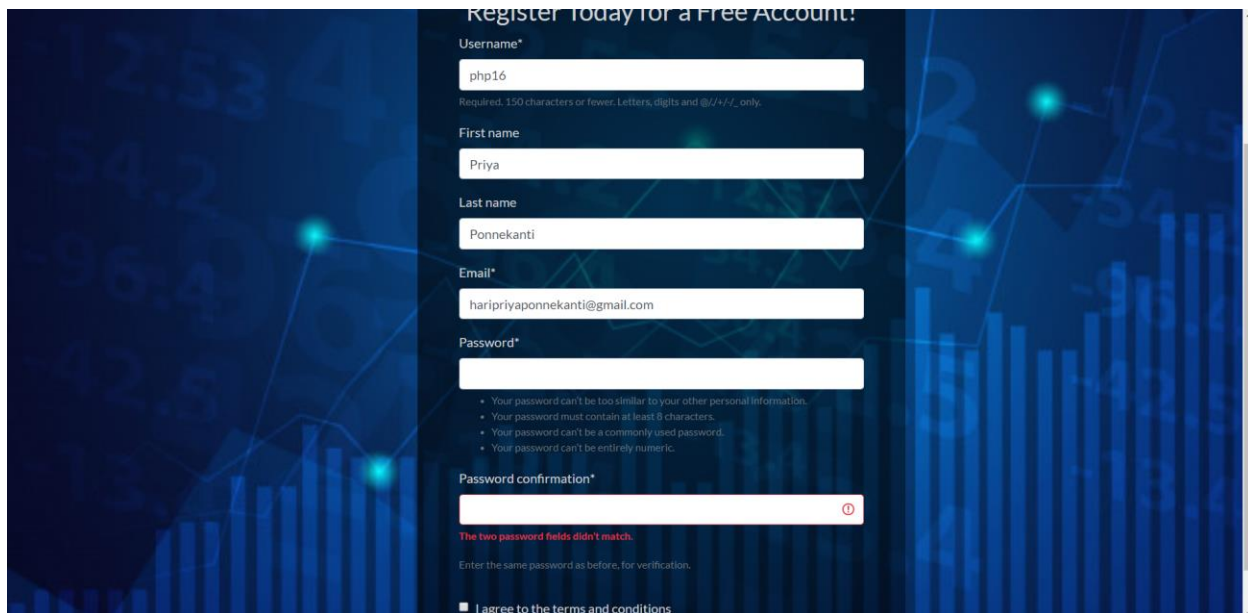
Each user needs to have a unique username. If a new user enters a username which is already used by another of our application users, then we display an error and request the user to enter a new username as that one is already in use.



The screenshot shows the same registration form as above, but with an error message displayed below the Username field: "A user with that username already exists." The Username field now contains "hp467" and has a red error icon. The other fields (First name, Last name, Email, Password) remain the same as in the previous screenshot.



The user is asked to re-enter the password for verification and security purposes. If the two passwords do not match, then we display an error message that the two passwords which the user enters need to match.



Register Today for a Free Account!

Username\*  
php16  
Required: 150 characters or fewer. Letters, digits and @/./\_/- only.

First name  
Priya

Last name  
Ponnekanti

Email\*  
haripriyaponnekanti@gmail.com

Password\*  
  

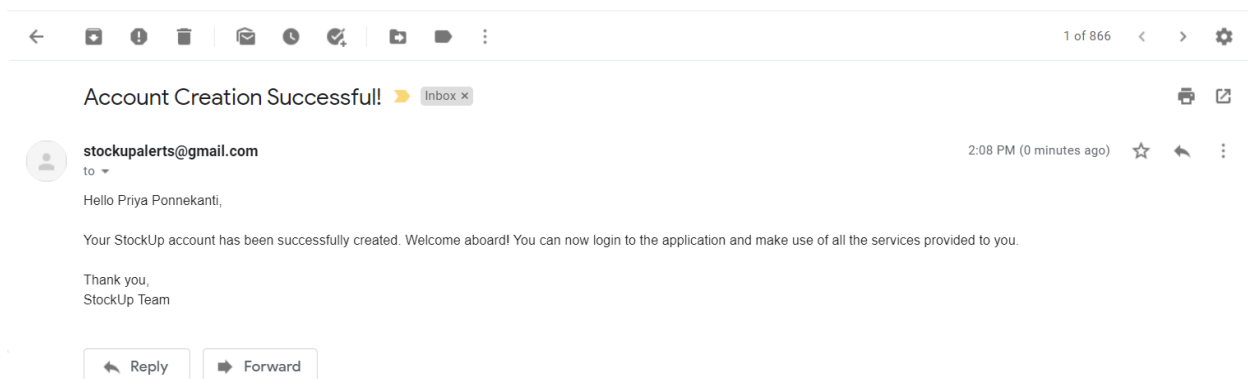
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation\*  
  
The two password fields didn't match.

Enter the same password as before, for verification.

☐ I agree to the terms and conditions

Once the user provides all the correct required details, an account is created for him in our database and we send him an email to the Email address he provided while signing up. This email informs him that his account has been successfully created.

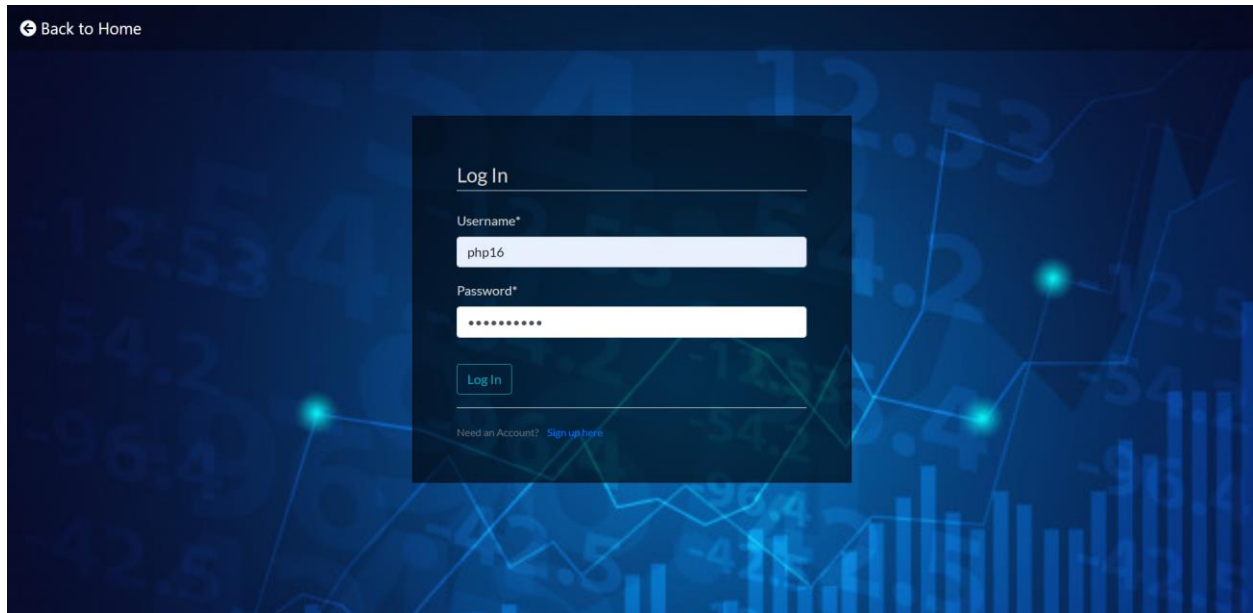


Once the account is created, the user is redirected to the login page.



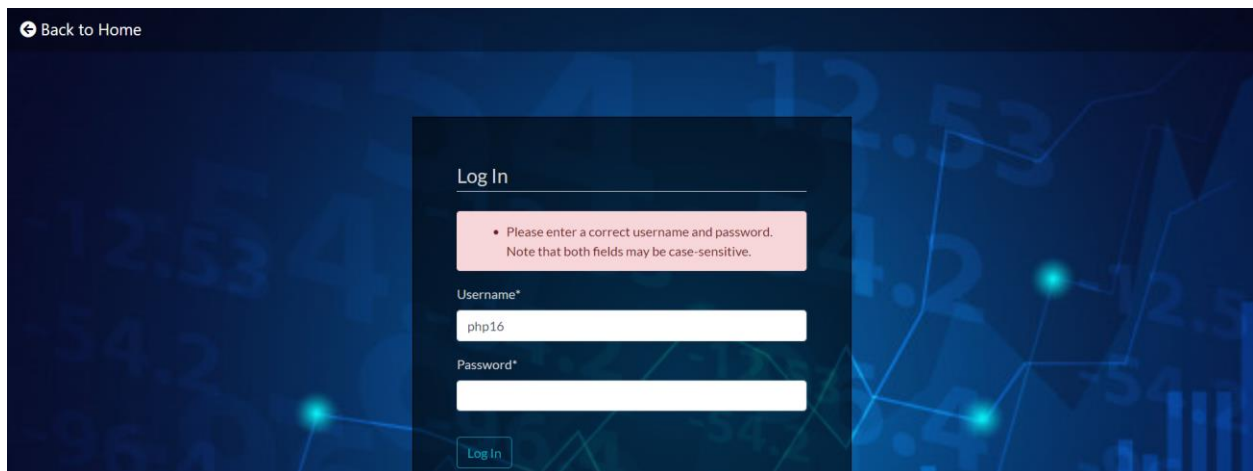
## 5. Login Page (Use Case: 2)

This is the page where users provide their credentials like username and password to login to the application. If a user is visiting our website for this first time, he/she has to sign up first through the Sign-Up page.



The screenshot shows a login page with a dark blue background featuring a faint line graph and numerical data. At the top left, there is a link "Back to Home" with a left-pointing arrow. The main content is a white "Log In" form. The form has a title "Log In" followed by a horizontal line. Below this, there are two input fields: "Username\*" containing the text "php16" and "Password\*" which is masked with dots. A "Log In" button is positioned below the password field. At the bottom of the form, there is a link "Need an Account? Sign up here".

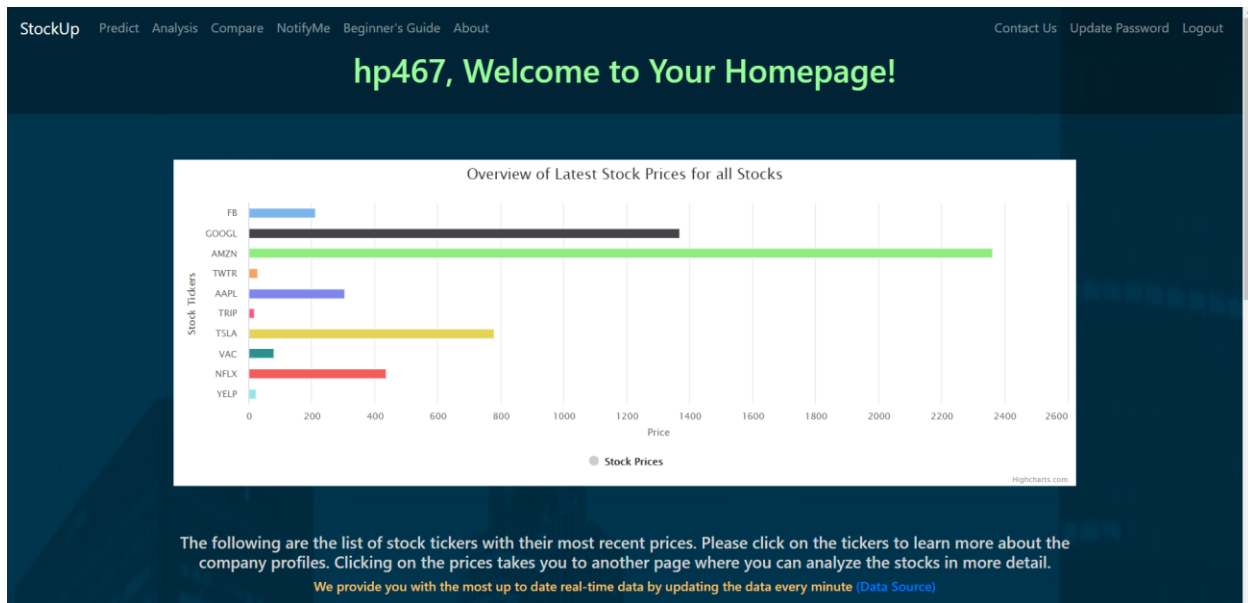
We perform a username and password check on the login page. If the user enters either an incorrect username or an incorrect password, then we display an error message requesting the user to enter the correct details in order to be able to login to the application.



This screenshot shows the same login page as before, but with an error message displayed. The error message is contained in a pink rectangular box and reads: "Please enter a correct username and password. Note that both fields may be case-sensitive." The "Username\*" field still contains "php16", and the "Password\*" field is empty. The "Log In" button remains visible below the password field.

## 6. Home Page (Use Case: 12)

This is the first page which the user will be redirected to once he logs in to the application. This page contains the following contents: The user will see a graph which displays the most recent stock prices of all the stocks used in our application.

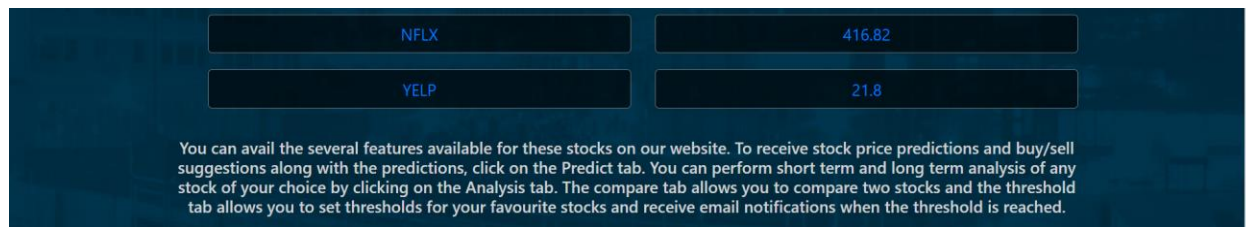


When he scrolls down, he will also see the exact values of the prices in the form of a table which contains the stock ticker symbols and their corresponding latest prices. At the bottom of the page, he will be informed about which pages will help him use what kind of services.

The following are the list of stock tickers with their most recent prices. Please click on the tickers to learn more about the company profiles. Clicking on the prices takes you to another page where you can analyze the stocks in more detail.

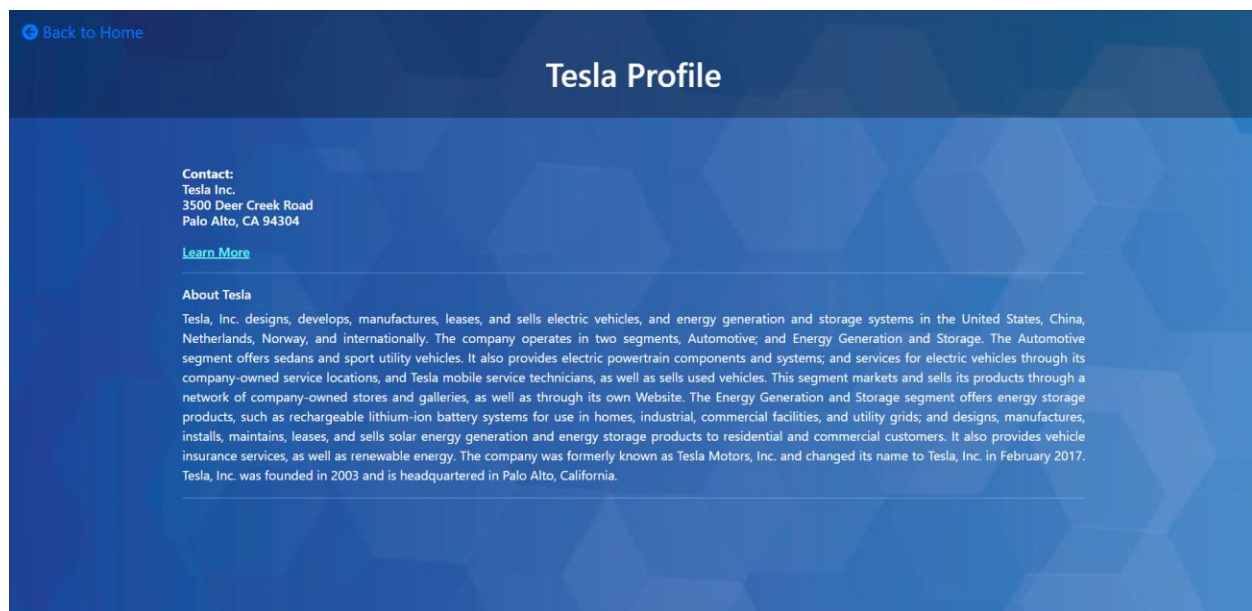
We provide you with the most up to date real-time data by updating the data every minute (Data Source)

FB	211.1
GOOGL	1367.9
AMZN	2362.81
TWTR	28.735
AAPL	303.58
TRIP	18.75
TSLA	780.0
VAC	79.39
NFLX	436.13



## 7. Stock Profile Pages (Use Case: 18)

On Clicking on any of the stock tickers on the Home Page, the user will be redirected to the profile page for that specific stock wherein he can learn more about that company or organization and he/she can also visit the company website from the profile page to know what exactly the company does. By clicking on TSLA ticker, the user will be redirected to the Tesla Profile page as below.

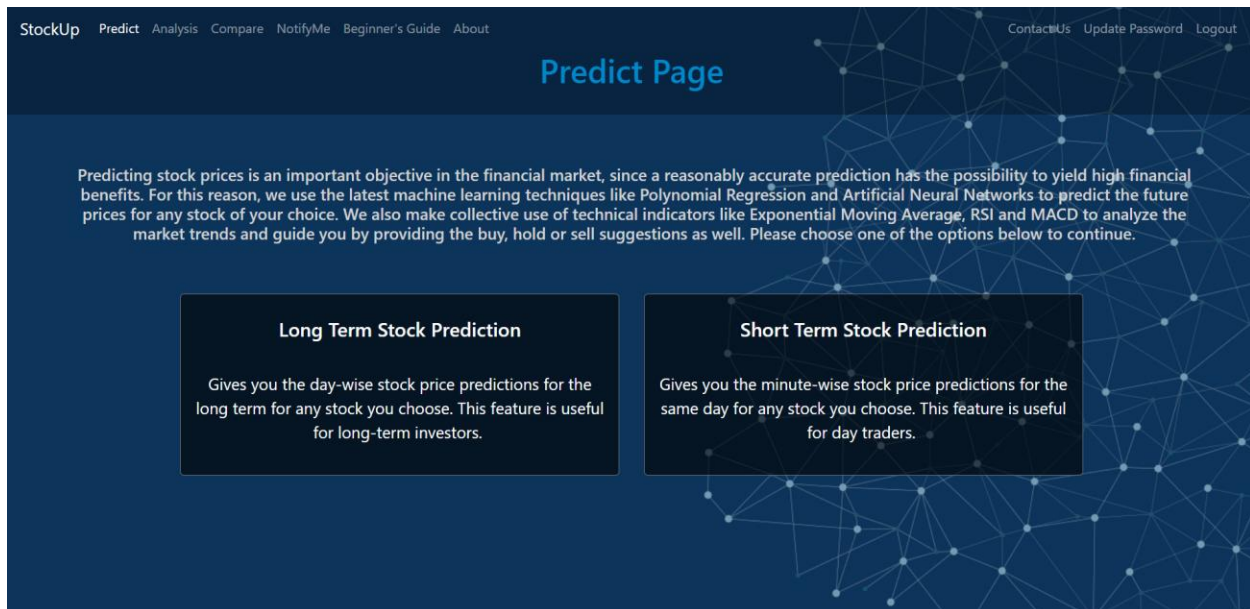


## 8. Predict Page (Use Case: 8)

This page is displayed when the user clicks on the Predict tab in the Navbar. It provides two options to the user:

- Long-Term Stock Prediction
- Short-Term Stock Prediction

The user can click on any of these two options to get stock price predictions and suggestions.



### 9. Long Term Prediction Page (Use Case: 8, 9, 10)

When the user clicks on the Long Term Stock prediction, he will be redirected to this page wherein he can select any stock from the drop-down and input the number of predictions he wants to see for this stock.

Back

## Long Term Prediction

Please input the required details below. We then provide you with the number of predictions you input for the stock you choose. We also give you a suggestion as to whether you can buy or sell the stock based on the predictions. Besides this, you also get to see a graph below the predictions where you can analyze the trends of that stock by observing the curves for the price, EMA and RSI.

Please choose a stock:

Google (GOOGL)

Enter the number of predictions you want:

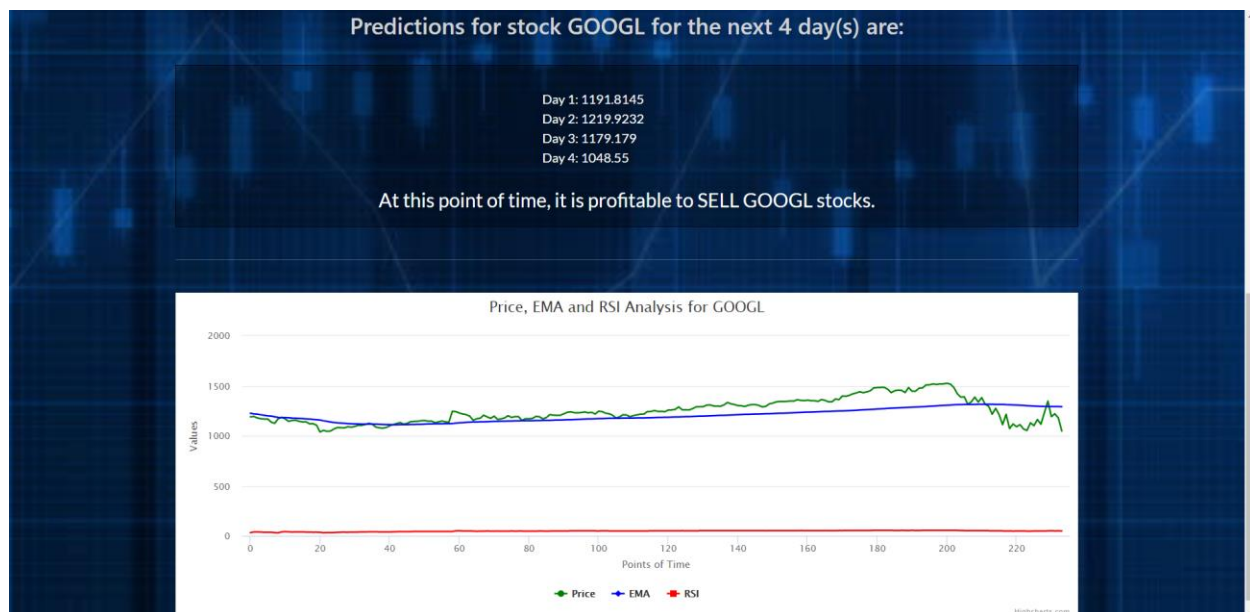
4

Submit!

Once he gives the input and clicks on Submit and scrolls down, he will be able to see the list of predictions obtained for that stock and also he will get a buy or sell suggestion for that stock based on the predictions made.



Besides these, he will also be able to view a graph which plots the price, EMA and RSI values for that graph over the long term period which is from the past one year.



## 10. Short Term Prediction Page (Use Case: 8, 9, 10)

On the predict page, if the user clicks on Short Term Stock Prediction, he/she will be redirected to this page which is similar to the Long Term Prediction Page except that the algorithm for short term prediction uses real time data. The user needs to input the same details in this form similar to the long term prediction form.

Back

### Short Term Prediction

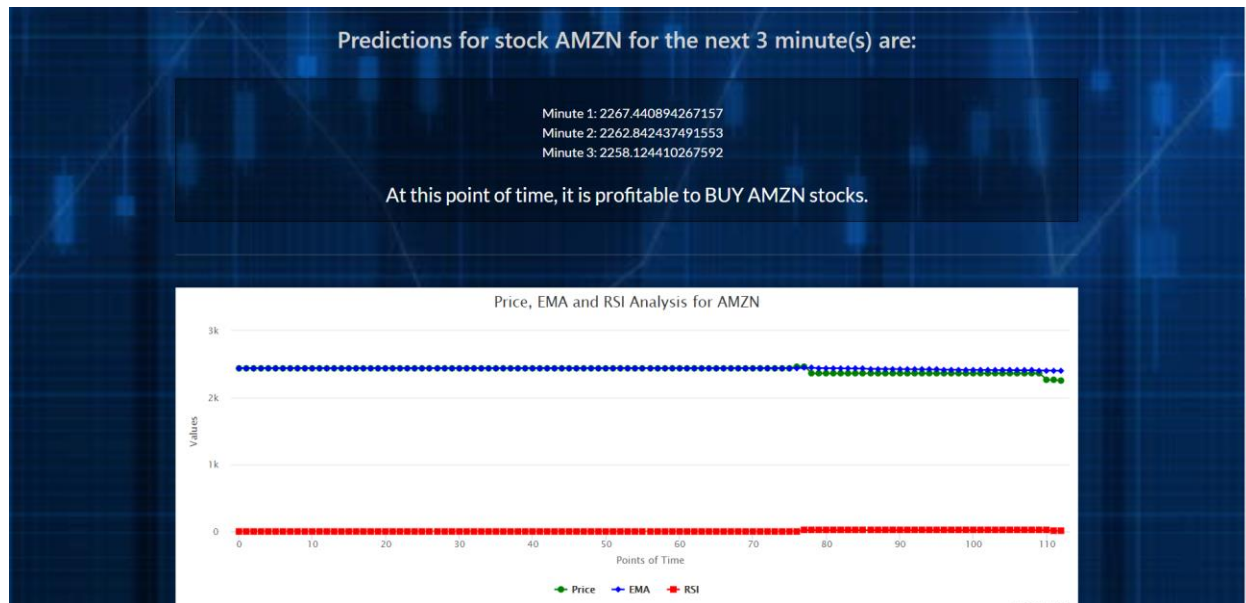
Please input the required details below. We then provide you with the number of predictions you input for the stock you choose. We also give you a suggestion as to whether you can buy or sell the stock based on the predictions. Besides this, you also get to see a graph below the predictions where you can analyze the trends of that stock by observing the curves for the price, EMA and RSI.

Please choose a stock:  
Amazon (AMZN)

Enter the number of predictions you want:  
8

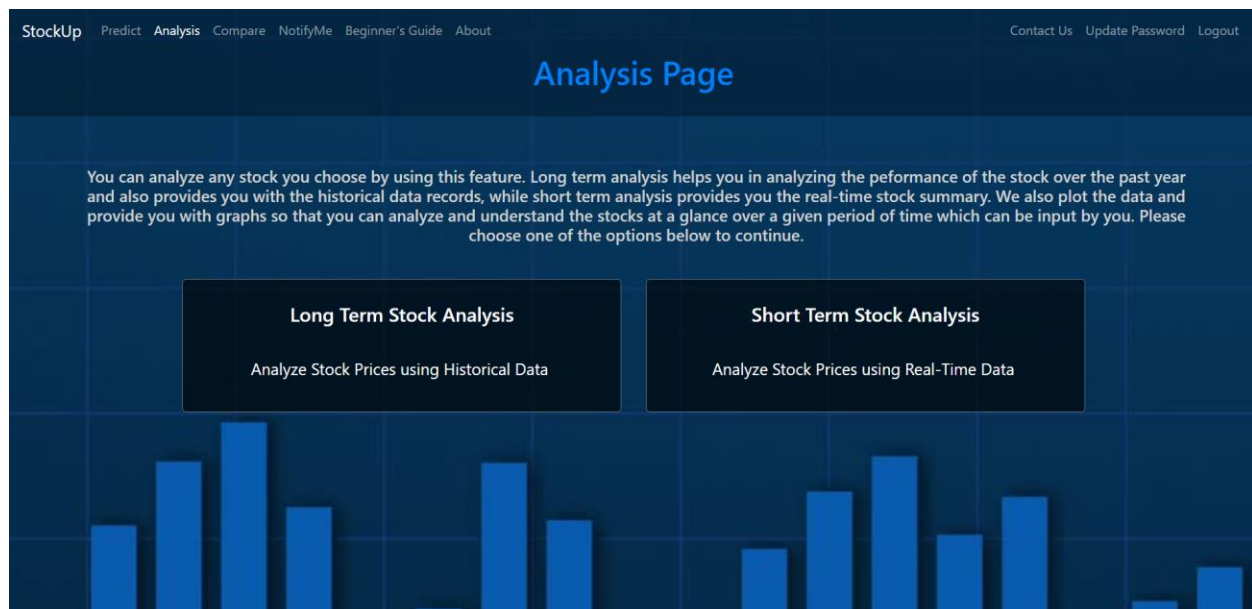
Submit!

Once he clicks on submit, the user gets the list of predictions obtained for the next few minutes (as input by the user), a buy/sell suggestion based on these predictions and a graph plotting the price, EMA and RSI values for the short-term period which is the past few hours. These plots also include the predictions made by the short term polynomial curve fitting algorithm.



## 11. Analysis Page (Use Case: 11)

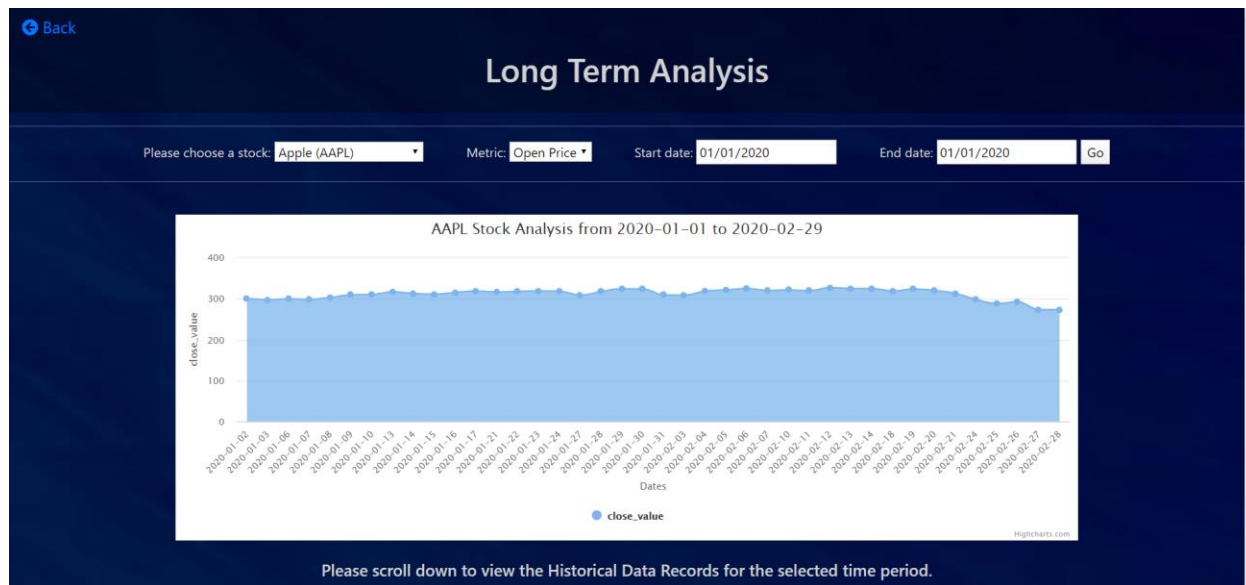
When the user clicks on the Analysis tab in the navbar, he will go to this page where he has two options: Long Term Stock Analysis and Short Term Stock Analysis.



## 12. Long Term Analysis Page (Use Case: 11, 13)

In this page, the user can choose any stock for which he wants to perform long term analysis for, using historical data. He can also choose a metric from any of the following: **{Open Price, Close Price, Low, High or Volume}**.

He can also input the start date and end date. After he provides all the required inputs as above, he will be able to view a graph will a plot of all the values of the selected metric between selected date range.



Below the plot, he will also be able to view the historical records for the selected stock within the selected range of dates.

Please scroll down to view the Historical Data Records for the selected time period.

Date	Open	Low	High	Close	Volume
Jan. 2, 2020	296.24	295.19	300.6	300.35	33870100
Jan. 3, 2020	297.15	296.5	300.58	297.43	36580700
Jan. 6, 2020	293.79	292.75	299.96	299.8	29596800
Jan. 7, 2020	299.84	297.48	300.9	298.39	27218000
Jan. 8, 2020	297.16	297.16	304.44	303.19	33019800
Jan. 9, 2020	307.24	306.2	310.43	309.63	42527100
Jan. 10, 2020	310.6	308.25	312.67	310.33	35161200
Jan. 13, 2020	311.64	311.15	317.07	316.96	30383000
Jan. 14, 2020	316.7	312.17	317.57	312.68	40488600
Jan. 15, 2020	311.85	309.55	315.5	311.34	30480900
Jan. 16, 2020	313.59	312.09	315.7	315.24	27207300
Jan. 17, 2020	316.27	315.0	318.74	318.73	34454100
Jan. 21, 2020	317.19	316.0	319.02	316.57	27710800
Jan. 22, 2020	318.58	317.31	319.99	317.7	25458100
Jan. 23, 2020	317.92	315.65	319.56	319.23	26118000
Jan. 24, 2020	320.25	317.52	323.33	318.31	36634400
Jan. 27, 2020	310.06	304.88	311.77	308.95	40485000
Jan. 28, 2020	312.6	312.19	318.4	317.69	40558500
Jan. 29, 2020	324.45	321.38	327.85	324.34	54057300
Jan. 30, 2020	320.54	318.75	324.09	323.87	31689800
Jan. 31, 2020	320.93	308.29	322.68	309.51	49897100
Feb. 3, 2020	304.3	302.22	313.49	308.66	43496400

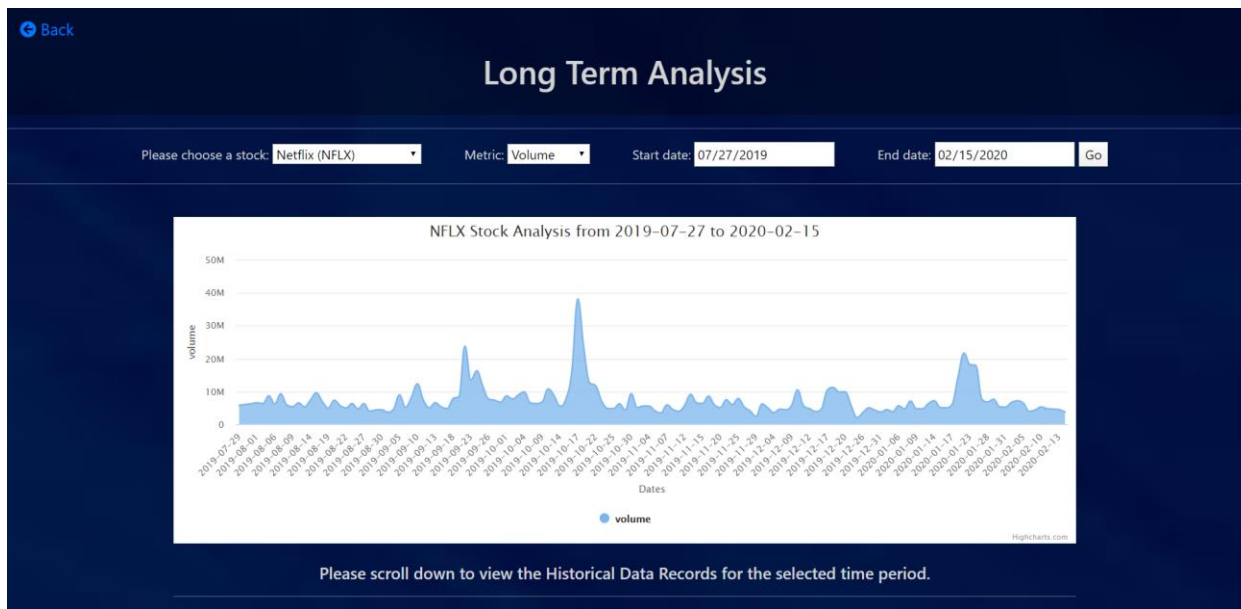
Jan. 23, 2020	317.92	315.65	319.56	319.23	26118000
Jan. 24, 2020	320.25	317.52	323.33	318.31	36634400
Jan. 27, 2020	310.06	304.88	311.77	308.95	40485000
Jan. 28, 2020	312.6	312.19	318.4	317.69	40558500
Jan. 29, 2020	324.45	321.38	327.85	324.34	54057300
Jan. 30, 2020	320.54	318.75	324.09	323.87	31685800
Jan. 31, 2020	320.93	308.29	322.68	309.51	49897100
Feb. 3, 2020	304.3	302.22	313.49	308.66	43496400
Feb. 4, 2020	315.31	313.63	319.64	318.85	34154100
Feb. 5, 2020	323.52	318.95	324.76	321.45	29706700
Feb. 6, 2020	322.57	320.26	325.22	325.21	26356400
Feb. 7, 2020	322.37	318.0	323.4	320.03	29421000
Feb. 10, 2020	314.18	313.85	321.55	321.55	27337200
Feb. 11, 2020	323.6	318.71	323.9	319.61	23580800
Feb. 12, 2020	321.47	321.47	327.22	327.2	28432600
Feb. 13, 2020	324.19	323.35	326.22	324.87	23686900
Feb. 14, 2020	324.74	322.85	325.98	324.95	20028400
Feb. 18, 2020	315.36	314.61	319.75	319.0	38132800
Feb. 19, 2020	320.0	320.0	324.57	323.62	23496000
Feb. 20, 2020	322.63	318.21	324.65	320.3	25141500
Feb. 21, 2020	318.62	310.5	320.45	313.05	32388500
Feb. 24, 2020	297.26	289.23	304.18	298.18	55548800
Feb. 25, 2020	300.95	286.13	302.53	288.08	57668400
Feb. 26, 2020	286.53	286.5	297.88	292.65	49513700
Feb. 27, 2020	281.1	272.96	286.0	273.52	80151400
Feb. 28, 2020	257.26	256.37	278.41	273.36	106721200

The below is the plot for TRIP stock and the metric chosen is high. The date range is from 12th June, 2019 to 1st January, 2020.



The below is the plot for NFLX stock and the metric chosen is volume. The date range is from 27th July, 2019 to 15th February, 2020.

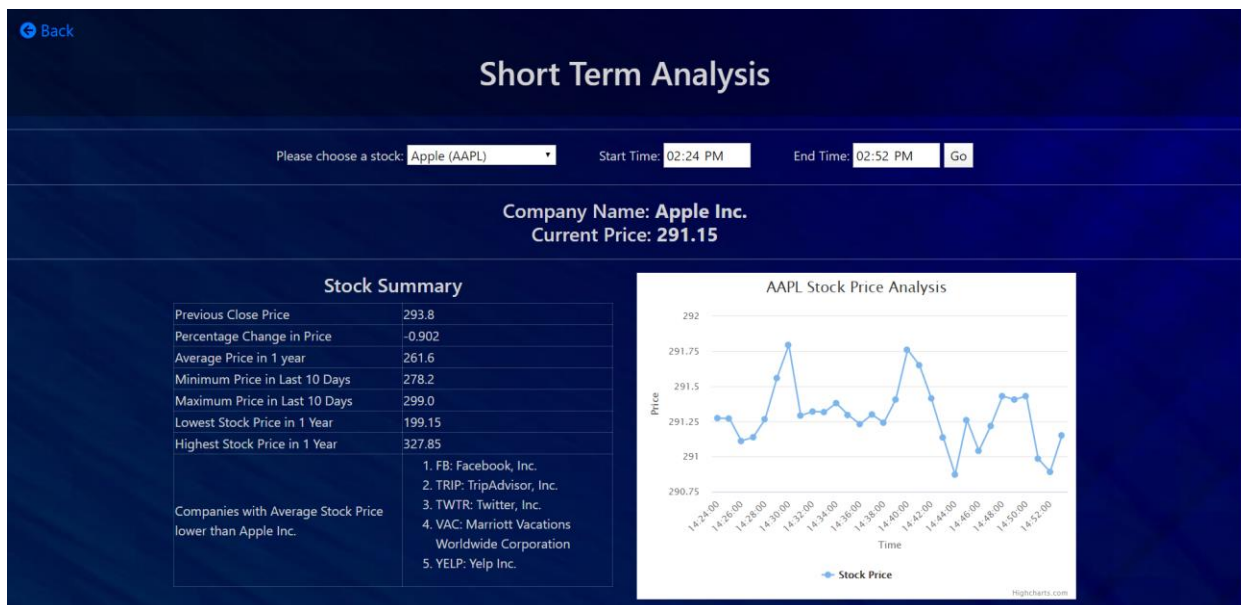




### 13. Short Term Analysis Page (Use Case: 11, 14)

When the user clicks on the Short Term analysis option on the Analysis page, he will be redirected to this page. On this page, he can analyze the stock in the short term period using real-time data. He needs to choose any stock of his choice from the drop down and also select the start time and end time. He can view the stock summary on the left side of the page which contains various details as shown below.

He sees a graph on the right side which plots the price of the stock between the selected range of time on that day.



#### 14. Compare Stocks Page (Use Case: 15)

This page is displayed when the user clicks on the Compare tab in the navbar. He can use this page to compare the performance of two or more stocks either in the short term or the long term.

He needs to choose multiple stocks and also the short term or long term option and click on Submit.

StockUp Predict Analysis Compare NotifyMe Beginner's Guide About Contact Us Update Password Logout

## Compare Stocks

Are you confused as to which stock to invest in? We make the decision process smoother for you by helping you to compare two or more stocks to analyze their performance both in the long term (past one year) as well as the short term (one day). You just need to input the details below and we will provide you with the graphs for the price and volumes traded for each stock. We also give you further details on the average values in the long and short time periods over the Open price, Close Price, Low, High and Volume traded. Please provide the necessary details below.

Select stocks by holding Ctrl key:

- Apple (AAPL)
- Amazon (AMZN)
- Facebook (FB)
- Google (GOOGL)

Select the term:

- Short Term
- Long Term

Submit

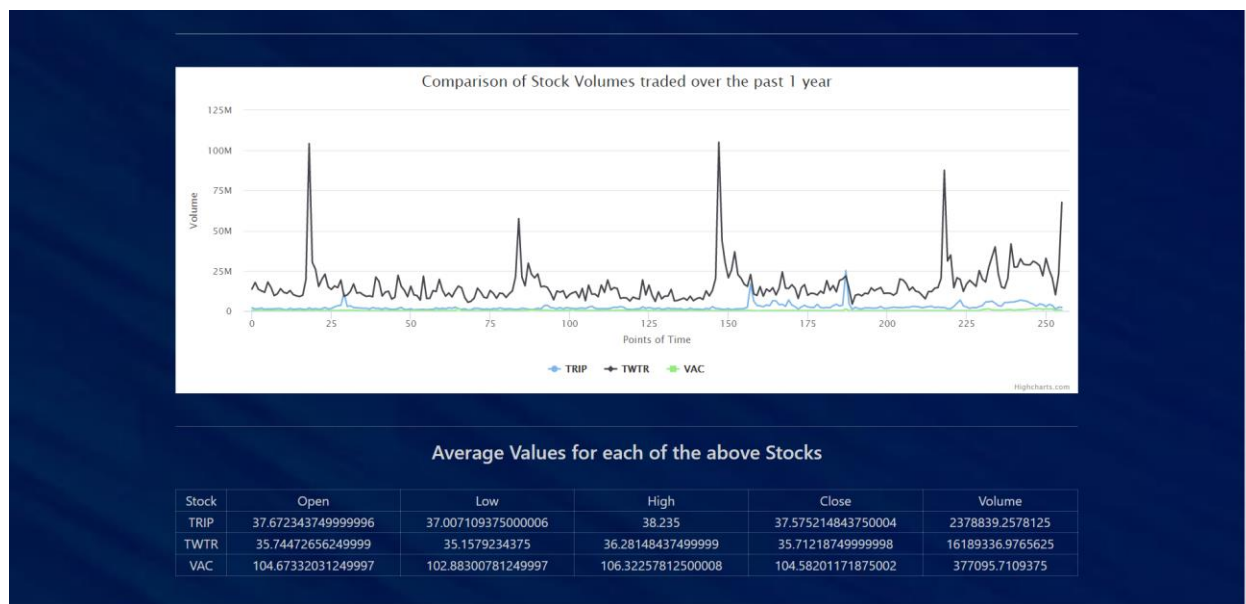
He will then be able to see two plots.

One plot shows the price movements or trends in each of the stocks.



Another plot shows the number of volumes traded for each of the selected stocks. Below this graph, the user can also see the average values of all the attributes of the stocks over

the short-term or long-term periods (short term is past one day while long term is past one year).



## 15. NotifyMe Page (Use Case: 16, 17)

This page is displayed when the user clicks on the NotifyMe tab in the Navbar.

StockUp Predict Analysis Compare **NotifyMe** Beginner's Guide About Contact Us Update Password Logout

## NotifyMe

We do not want you to miss out any good trading opportunity on the stocks which you are interested in. We therefore give you the option of specifying a percentage change in price for your favourite stocks so that we can notify you through email whenever the stock price reaches or crosses the percentage you set and bring it to your notice immediately. You can refer to the current price of all stocks on the left and accordingly specify the desired percentage change in price for any stock of your choice in the form on the right.

Stock	Current Price
FB	211.1
GOOGL	1367.9
AMZN	2362.81
TWTR	28.735
AAPL	303.58
TRIP	18.75
TSLA	780.0
VAC	79.39
NFLX	436.13
YELP	22.79

Please select a stock:

Please enter a percentage:

This functionality helps the user in setting a price percentage for any of the stocks he's interested in so that he gets notified as and when the stock price reaches or crosses that percentage. For this, he needs to choose a stock and input the percentage he wants.

Below the form, he will be able to view the list of stocks, their corresponding price percentages which have already been set by the user and which have not yet been satisfied and the price of that stock when that percentage was set.

The screenshot displays the StockUp application interface. At the top, there is a table listing various stocks and their current prices. To the right of this table is a form for setting a percentage change for a selected stock. Below the form is a section titled 'List of Percentages set by You' which contains a table showing the desired percentage change and the stock price when the percentage was set.

Stock	Current Price
FB	211.1
GOOGL	1367.9
AMZN	2362.81
TWTR	28.735
AAPL	303.58
TRIP	18.75
TSLA	780.0
VAC	79.39
NFLX	436.13
YELP	22.79

Please select a stock:  
Apple (AAPL)

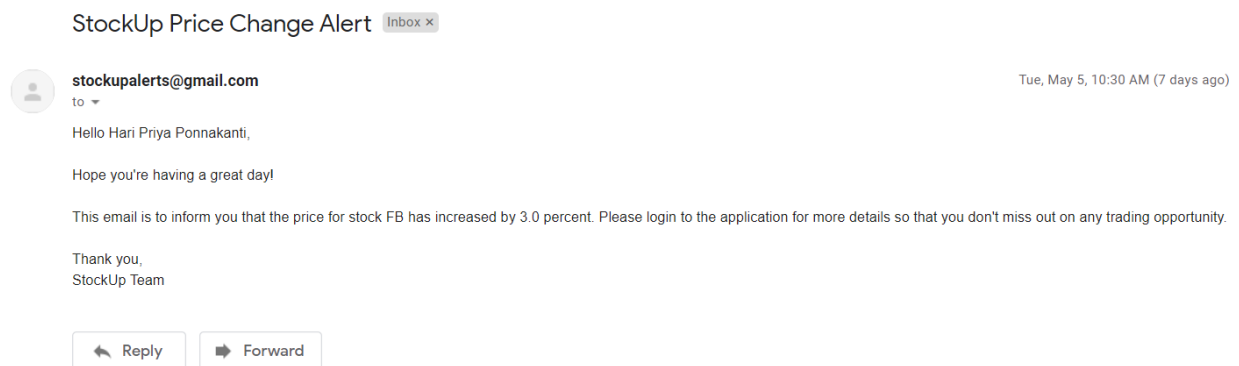
Please enter a percentage:  
Percentage Change

Submit!

### List of Percentages set by You

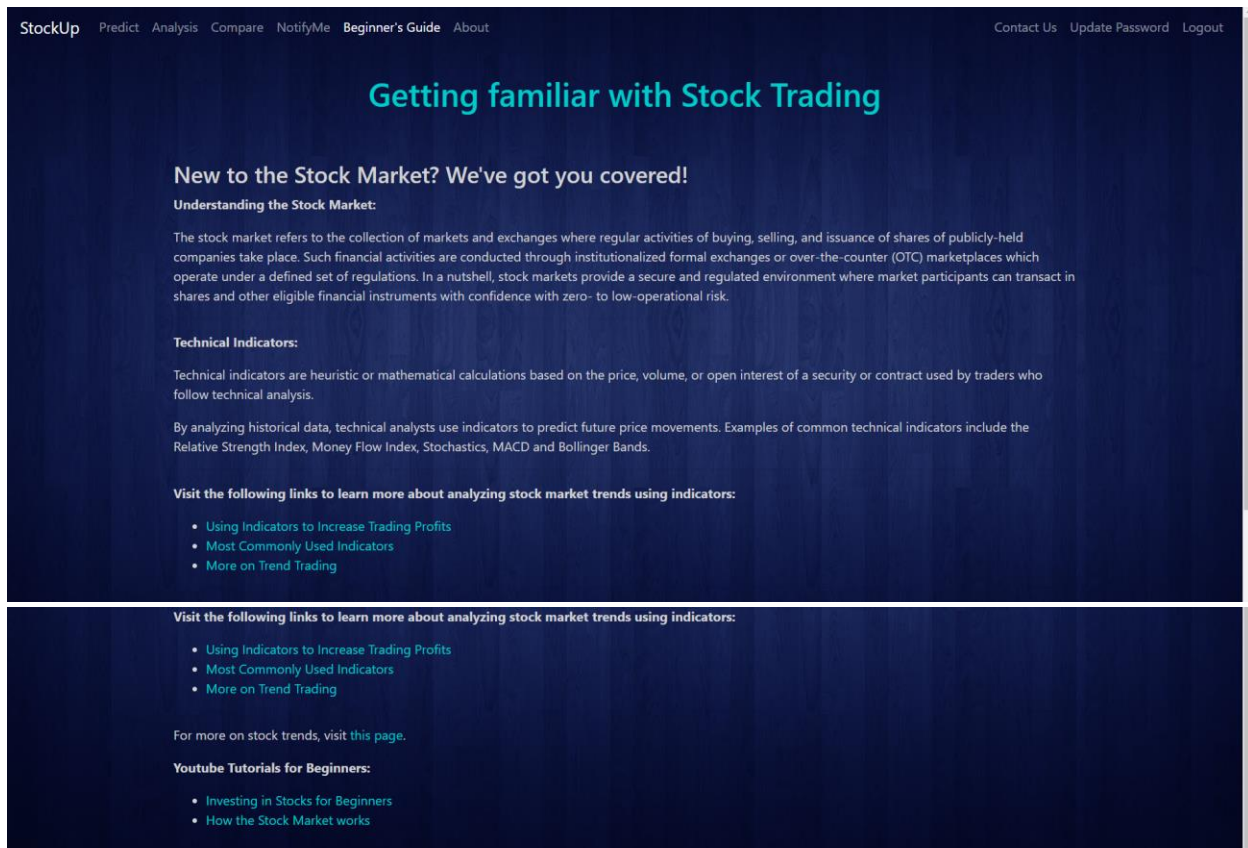
Stock Symbols	Desired Percentage Change for the Stock	Stock Price when Percentage was Set
AMZN	5.0	2280.15

When any of the percentage gets satisfied, then we notify the user immediately by sending an email so that he doesn't miss out on any good trading opportunity for that stock.



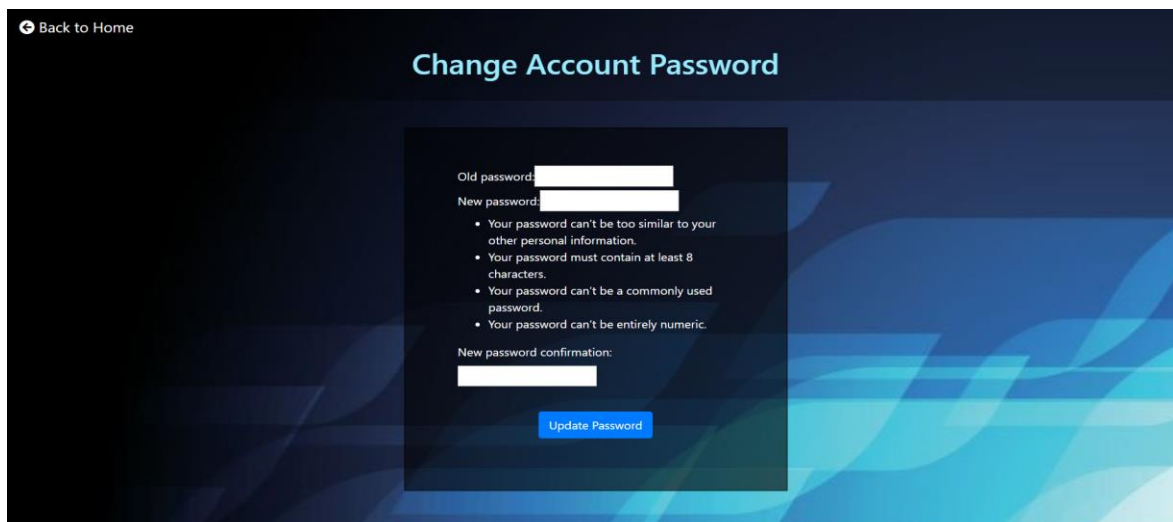
## 16. Beginner's Guide Page (Use Case: 6)

This page is displayed when the user clicks on the Beginner's Guide tab on the Navbar. This page is to help familiarize the users, especially beginners to the stock market, with stock market and stock trading. We explain the basic concepts in brief and also provide links to other websites and youtube tutorials to learn more about it.



## 17. Update Password Page (Use Case: 5)

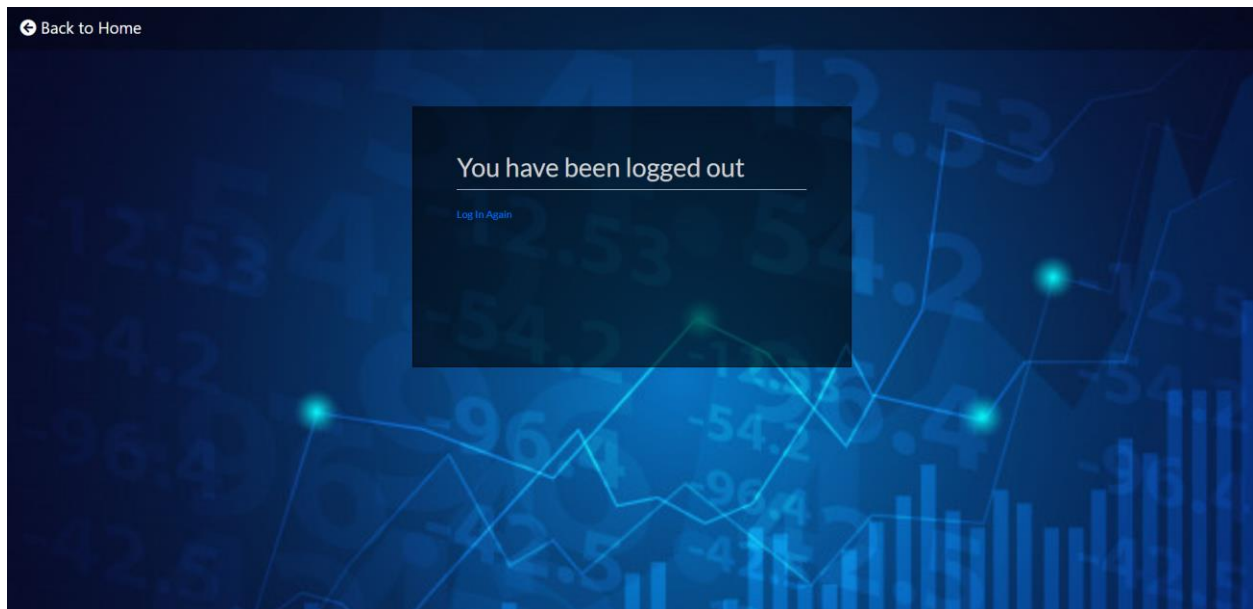
When the user clicks on Update Password on the Navbar, he will see this page. The user can update the password by providing the old password and the new password and this will update the account password for the user.





## 18. Logout Page (Use Case: 4)

When the user clicks on the Logout option on the Navbar, then he will be logged out of the application and he will see the below page. By clicking on Back to Home, he will be redirected to the Landing Page and if he clicks on login again, he will be redirected to the Login Page.



## 6. Conclusion

In this project we have developed a web application to predict prices for a stock for short term and long term. We used Polynomial curve fitting for short term and Neural Networks for long term stock prediction. We have also included additional features where users can contact us in case of any issues and the application also notifies the users through email for a few features. For example when there is a fall in any stock price which the user desired, then an email is sent to the user notifying him about the price decline so that he doesn't miss out on any good buy opportunity. We have created a user interface using HTML, CSS, Bootstrap and JavaScript. We have considered both historical and real time data for stock predictions. We implemented our web service using the Django REST framework. We have added several use cases to make it as user friendly as possible.

We faced several issues while developing this project in all the modules starting from data collection, backend, server and the User Interface. However, we overcame all the bottlenecks and finally came up with several use cases in addition to the basic

requirements. The step by step procedure involved in software engineering development style has helped us to build these use cases from a conceptual point to a realized module.

We as a team members put extra efforts to achieve all the requirements. The course helped us achieve this project successfully by working on a full-stack end to end web application.

## **7. Future Work**

1. The system can be developed into a price alert feature with small scale industries.
2. Adding more number of stocks into our list will give wider options to users.
3. Having a written blog of the latest trends of the stocks would be helpful to users.
4. Write an Ansible script to directly install and run the application just by running the script.

## **8. Contribution Breakdown**

We believe that each team member should have a complete idea about each module of the project. So, we divided each module into separate tasks so that each one of us can contribute our individual parts to each module and finally have a thorough understanding of each detail in our project. Therefore, we all contributed equally to the project.

## **9. References**

1. [https://miro.medium.com/max/1400/0\\*g9Q81AEpvLTH4gwq.jpg](https://miro.medium.com/max/1400/0*g9Q81AEpvLTH4gwq.jpg)
2. [https://miro.medium.com/max/640/1\\*ELey2wytIZvKYFLbLbhCoA.png](https://miro.medium.com/max/640/1*ELey2wytIZvKYFLbLbhCoA.png)
3. [https://www.thestreet.com/.image/t\\_share/MTY3NTQxNTA4OTgxOTkxMzEw/11-best-short-term-investments-in-2019.jpg](https://www.thestreet.com/.image/t_share/MTY3NTQxNTA4OTgxOTkxMzEw/11-best-short-term-investments-in-2019.jpg)
4. <https://cdn3.vectorstock.com/i/1000x1000/38/87/long-term-investment-vector-22633887.jpg>
5. <https://previews.123rf.com/images/foxaon/foxaon1603/foxaon160300554/53724839-stock-market-chart-background.jpg>
6. [24839-stock-market-chart-background.jpg](https://previews.123rf.com/images/foxaon/foxaon1603/foxaon160300554/53724839-stock-market-chart-background.jpg)