SuperStonks

Shounak Rangwala
RUID:191004996

Table of Contents

Abstract:

Stock market prediction is probably the most sought-after application in predictive analysis worldwide. Millions have tried and millions more will keep trying to somehow make sense of the trends in the stock market and game the system so as to make a boatload of money. In this web application, I have tried to come up with 2 predictive models that can help traders gain a peek behind the curtains. Most trading platforms/applications give a detailed stat sheet of the stock and leave the decision making on the trader. Sometimes this can be too overwhelming, and the naïve trader can make costly mistakes. This web application has been designed to help these traders make good financial decisions with respect to buying or selling a stock by making accurate stock price predictions over different periods of time. At its current version, SuperStonks boasts an accuracy of 54% for long term predictions and 47% for short term predictions.

Introduction:

SuperStonks is a web application that provides the user a deeper insight into the stock market. It focusses on predicting the stock prices of 10 stocks both in the long term time interval as well as in the short term time interval. It also displays the trends of the stock prices across an adjustable time frame so the user has the option of eyeballing the stock trend and thus making a sound financial decision. This web-app is designed to be used by both experienced and novice traders and also provides resources to better understand market terminologies.

As a newcomer to the stock market trading scene, I believed that the learning curve to navigating this is very steep. I had no notion which were the best stocks to invest in, when to buy the stocks and when to sell the stocks, how long to keep holding the stocks, how to options and futures work and how did they differ from traditional stocks. The platforms that I mostly used to trade (like Robinhood) did not present me with a deep enough understanding of the stock market to answer my above questions. All I had in front of me were columns of numbers and data and no way of making sense of these. There was no way to educate yourself on these topics except to go over to the internet and browse through Investopedia. An application that can predict stock price trends, give stock recommendations and also provide a curated set of resources so as to be a one stop shop for any trader was something that could address all the pain points of a new stock trader.

Why this solution is unique?

- SuperStonks is a web application that runs seamlessly on your browser, both pc and mobile. It uses the Django web framework, so the scalability and the performance of the web application are better than most other web applications that are using predictive models.

- The backend database is MySQL powered so the scalability in terms of data gathered and faster CRUD operations.
- Neural network model used for predicting stock prices for a long-time interval (more than 1 day). The service stores historical data of stocks in the database and the model trains on this data every time the service is used and makes a prediction for the stock price in the desired time interval.
- Polynomial curve fitting algorithm used to predict stock prices on a short-term time interval. It collects the real time data which are the stock prices at different time intervals of 5 seconds and then plots them out on a graph. This is useful for the day traders and the frequent traders.
- A list of resources that I found most useful in understanding the stock market terminology all in one place so the user can refer to those if he wants to learn/understand a specific topic better.

Related works:

Stock market predictions are generally technical or fundamental. The technical predictions use the trends in the stock prices – how high and how low they go and for how long to make predictions. Fundamental methods generally focus on market sentiment analysis and taking into notice the headlines for the day which are most likely to effect the stock market. The fundamental approach is more useful when you are predicting for a long time period (over months or years). While coding algorithms the technical analysis way provides a more intuitive appeal to be used because it can be used for both short term and long-term predictions while a more fundamental analysis technique would employ complicated methods of NLP and deep learning. Although the accuracy of such a model is better than a normal technical analysis model (which uses just stock prices), the designing and implementation of such a heavy model slows down the performance and the computation needs cannot be satisfied by most PC's and mobile devices. A tradeoff between accuracy and fast light-weight calculation exists in determining the optimal model to use in the web application. The research done in this field is extensive with artificial Neural networks coming out as the most versatile and preferred models used for prediction. Some recent research has been done using NLP and time series but the computers used in those papers have specifications much better than commonly found.

Background of services:

Predictive models:

- Neural network: A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Since the problem presented is more along the lines of regression, neural networks with a good backpropagation algorithm seems to be the

perfect model to use. Most models show an accuracy of 50 to 60 % with some even going to 70%. This is used mainly when the data is abundant, and the model can be trained effectively in all base cases.

- Polynomial curve fitting: A variant of regression analysis, this is a popular method in the field of statistics where a curve (the degree of the polynomial can be decided beforehand) is made to pass through the data points and the best fitting coefficients are then evaluated and then the curve is used for future analysis.

Resources:

- A collection of internet resources that can equip the new stock trader with the basics as well as the intricacies of stock trading so as to help them make better decisions and also use the predictions made by SuperStonks to their benefit.

Detailed description of the service:

Technology used:

The tech stack of SuperStonks includes the following:

- Data Source – Yahoo Finance: Yahoo finance has been one of the mainstays in the stock market world, all the information regarding different stocks, options and futures can be found comprehensively on Yahoo Finance. Thus it makes sense to use this as the data source for our service. The predictive model cannot work if the data is in any way defective or erroneous and using a reputable source like Yahoo Finance eliminates that worry for us. The one issue I faced was that Yahoo Finance goes not have their own API for developers to use so I had to use an open source API called “yahoo_fin”. More will be discussed about this later.
- Web framework – Django: Django is the ideal choice for this application because of the scalable and flexible nature of it. As far as standard MVC architecture goes, Django based applications have a far better performance than other frameworks including Ru by-On-Rails. The reusability of code makes the application look sleek and neat. Also Django REST framework is an additional toolkit than is often paired with Django to maintain user authentication. This makes the application secure and prevents data leaks. Lastly, Django admin enables the developer to maintain a track of both the database models (Django comes with a dbSQLite database) as well as the users who are going to use the application.
- Database – MySQL: As I mentioned before Django comes with a SQLite database as a default. This is useful when you are making pet Django projects because the data needed to be stored is not massive and the entire database can be stored as part of the project. However this is not viable in real-world scenarios especially in applications that are deployed to the users/clients. The amount of data to be stored increases each day and so the database has to adapt and become scalable. MySQL databases have long been the industry standard of databases. They are a relational database and the option of having more tables that can be linked together provides for smoother and faster querying. The queries are written in SQL. Django as a framework has good documentation regarding switching over to a MySQL database and considering all the

benefits of MySQL and of simulating the real world scenario, MySQL databases were used in this application.

- Scripts/Models – Python: Since Django is a Python based web framework, it makes perfect sense to have all the scripts and the code all in Python. An interpreted, high-level programming language, Python is very intuitive to the coder and is highly maintainable. Also, Python is the preferred language for all Data science and machine learning applications because of the vast resource of different libraries that can be used in Python. Some of these libraries include Pandas, Sci-kit, PyTorch and Tensorflow. Since SuperStonks is going to be dealing with large amounts of data and will be using predictive models, it made sense to use pandas as a library to do the data manipulation part and Tensorflow to create the Neural Networks using the in-built Sequential module. Also there exist serializer libraries in Python that can be used to collect data from the source (Yahoo Finance) on a sequential basis.
- Web – HTML/CSS: The templates that make for the web pages that will be displayed I the web application are coded in HTML. Like all web pages, HTML is paired with styling sheets as well as JavaScript to read the data being sent over from both the scripts as well as the databases. I am using Bootstrap as my CSS library to create a better-looking UI with the navigation bar and the buttons that are being used in the application. Also, I am using Highcharts.js which is a JavaScript library that makes graphs and visual representation from JSON data (which is the format in which the data will be communicated between the frontend (HTML) and the backend (MYSQL and python scripts).

MVC Architecture in SuperStonks:

Models: - The MySQL server running on my system can be interfaced with using the MySQL workbench. In order to sync that with the Django project we need to decide on the models – the layout/schema of the data that is going to be stored in the database as well as the way of connecting the MySQL database with the project. Lets me explain these aspects separately:

- Creating the models in Django: The models.py will be reference for the application to see what kind of data is being used, the different attributes and features of it etc. To define these in models.py is important because if defined incorrectly, the entire database will have to be recreated later in the project and the data that is stored will be erased because of its incorrect format (which renders it useless for our predictions). For this application, I came up with 3 different types of data models that will be needed. They are Stocks, RealTime and Historical data models. In all these models I set the value of managed to “True”, because that way Django is given the permission to create and maintain (update,delete etc) the database tables.
    - Stocks: This data will include just 2 values, a stock ID which will be unique for each stock and a stock ticker which is the label for the stock in the stock market.

```
class Stocks(models.Model):
    sid = models.IntegerField(primary_key=True)
    ticker = models.CharField(max_length=10, blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'stocks'
```

- RealTime: This is the real time stock data that is collected from Yahoo Finance. The entries are correct to the last second and include the stock price, the volume of shares traded and other metrics that can be used in the predictive model training. The unique identifier of each entry will include the stock ID, the date of the entry and the time of the entry.

```
class RealTime(models.Model):
    sid = models.OneToOneField('Stocks', models.DO_NOTHING, db_column='sid', primary_key=True)
    dat = models.DateField()
    tim = models.TimeField()
    open_value = models.FloatField(blank=True, null=True)
    low = models.FloatField(blank=True, null=True)
    high = models.FloatField(blank=True, null=True)
    close_value = models.FloatField(blank=True, null=True)
    volume = models.IntegerField(blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'real_time'
        unique_together = (('sid', 'dat', 'tim'),)
```

- Historical: This includes the summary of the stock price from the open to the close of business on the stock market. The highest price, the lowest price and the volume traded for that day are parts of the attributes. This entries to this table are made once a day and so the unique identifier for the model will be the stock ID and the date of the entry. We use this data for the long-term prediction after collecting the data for multiple days.

```
class Historical(models.Model):
    sid = models.OneToOneField('Stocks', models.DO_NOTHING, db_column='sid', primary_key=True)
    dat = models.DateField()
    open_value = models.FloatField(blank=True, null=True)
    low = models.FloatField(blank=True, null=True)
    high = models.FloatField(blank=True, null=True)
    close_value = models.FloatField(blank=True, null=True)
    volume = models.IntegerField(blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'historical'
        unique_together = (('sid', 'dat'),)
```

- Connecting the MySQL server to Django: We need to set up connections in 2 ways. The first is through the settings.py file in Django where the default database is set so that the models can be used by Django to create the Tables there as well as when the data is

being called in the templates. Usually this is configured to the SQLLite database so after referring to the Django docs the modified connection looks like this:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'SuperStonks',
        'USER': 'root',
        'PASSWORD': [redacted],
        'HOST': '127.0.0.1',
        'PORT':     '3306',
    }
}
```

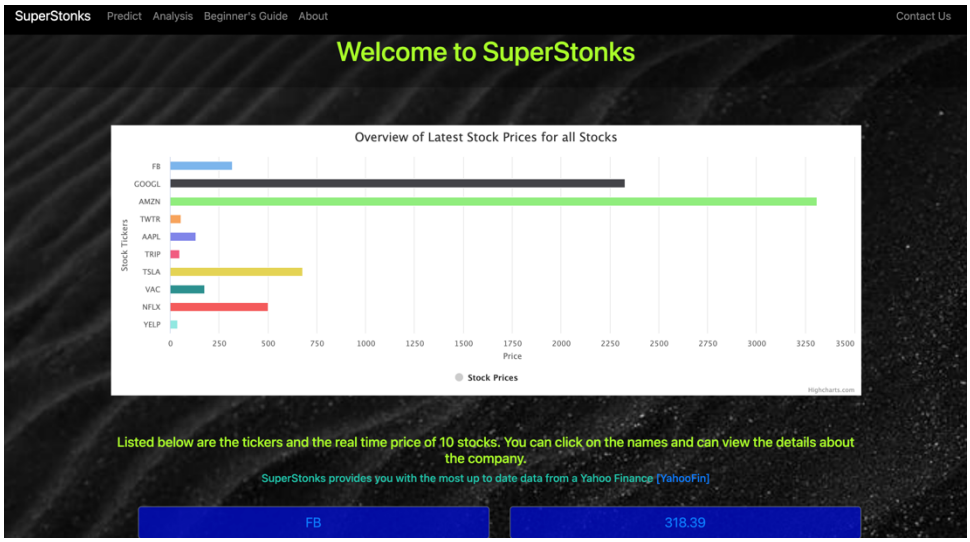
While the above connection makes the data available to be used by the views and the templates of Django, we need to set up another way of connecting with the database for our python scripts which will be reading, updating and inserting data into the database. The scripts concerned with the predictive models will be reading data while the scripts concerned with getting new data from Yahoo will be updating the database regularly. For this I used the mysql_connector library. This is mainly to be used to connect python programs to the database. There is extensive documentation of this available on the internet and I have only used a couple of methods in my implementations. The connection using this gets to be done like so:

```
# connect to mysql database
mydb = mysql.connector.connect(
    host="127.0.0.1",
    user="root",
    passwd=[redacted],
    database="SuperStonks",
    buffered=True,

)

mycursor = mydb.cursor()
mycursor.execute("SET SQL_SAFE_UPDATES = 0;")
```

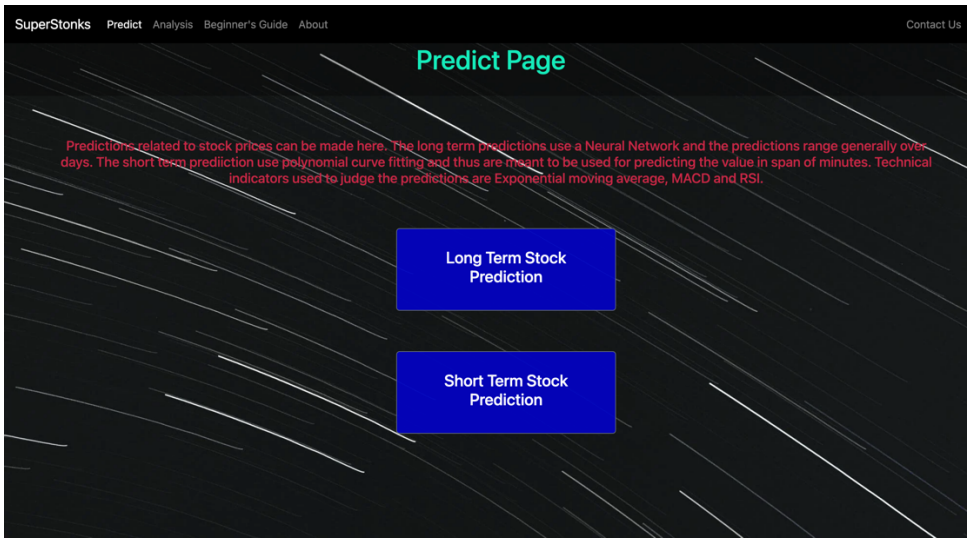
Views: In the traditional MVC architecture the views are the webpages/html files that are going to be displayed on the web browser. In Django these are called templates. We are going to be using a number of different templates for our web applications. The fun part about using Django is the reusability of code and the similar parts of the html code which are being

repeated across different templates can be extracted. Below is the list of all the templates that are used (their description and appearance)

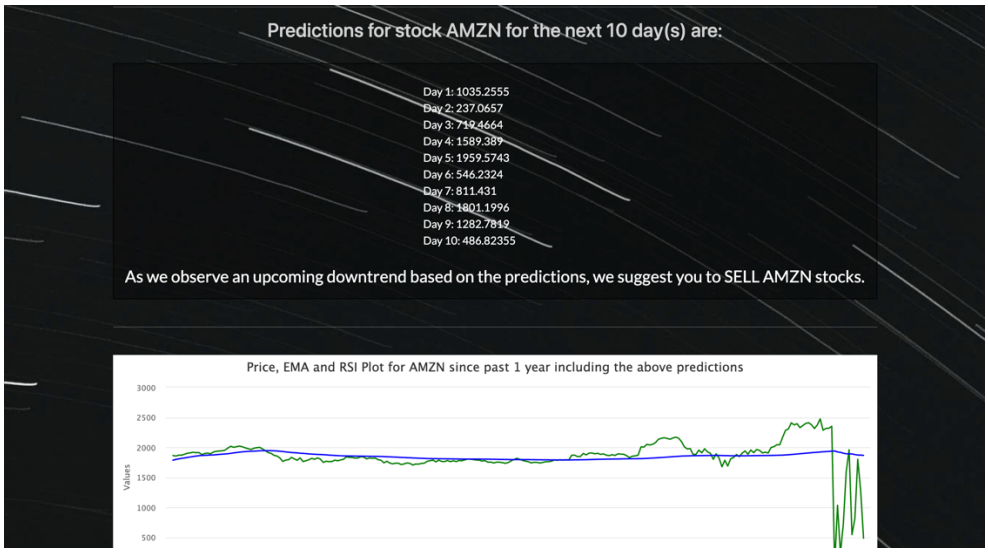
- Home.html : the landing page of the application. This is your starting point and it will show a navigation bar at the top, a graph showing the real time prices of the 10 stocks that we will be focusing on as well as a table of links depicting the name of the stock and the price of the stock. By clicking on the name yo go to another template which describes the company, while clicking on the price takes you to the analysis template.



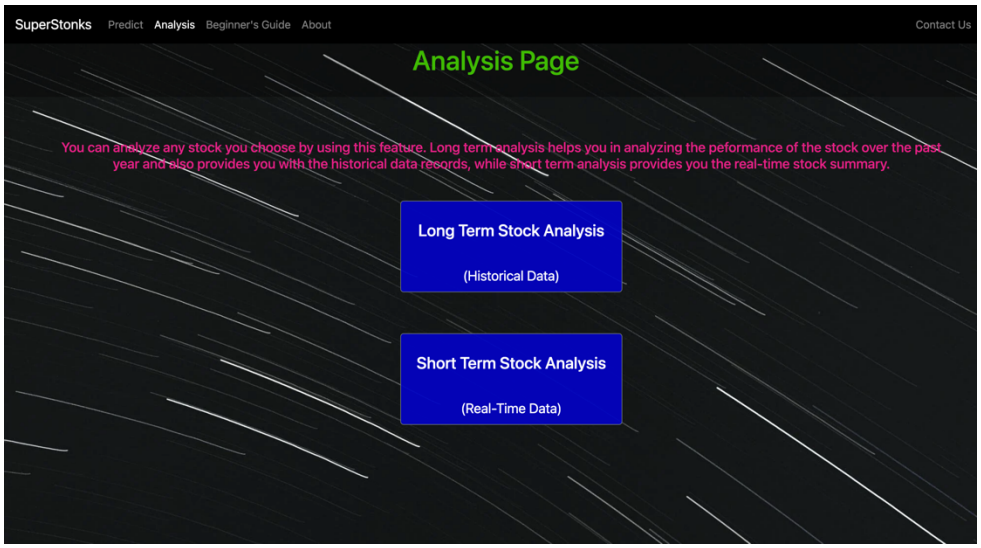
- Predict.html: The home page for the predictive models for the web applications. It will show a brief message describing the 2 models and their uses (long term and short term). Lastly it will display 2 buttons on clicking you go into that particular models template.



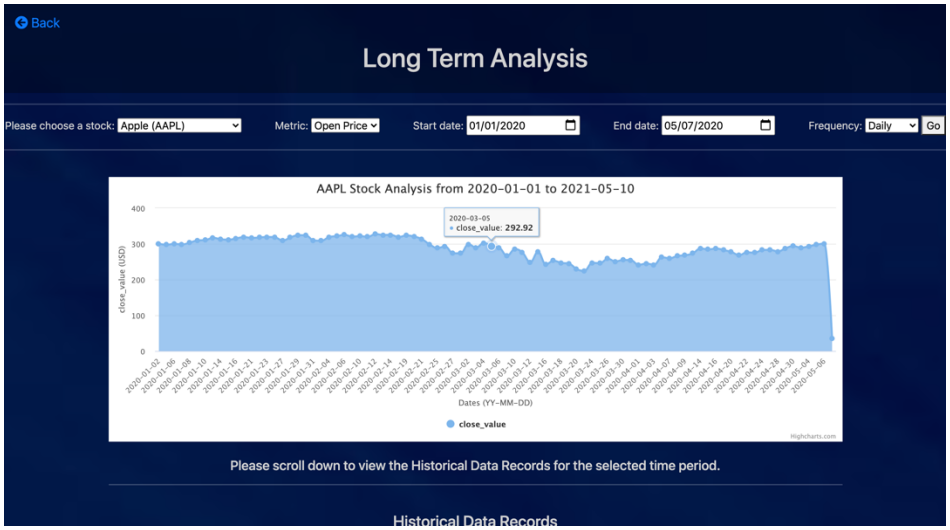
- Predict_form.html: This template is similar for both long term and short term prediction. It displays a form which asks you to select a stock and the time duration you want your prediction for. After clicking submit, the page updates itself to show the result below the form and clears the form for you to enter some other input.



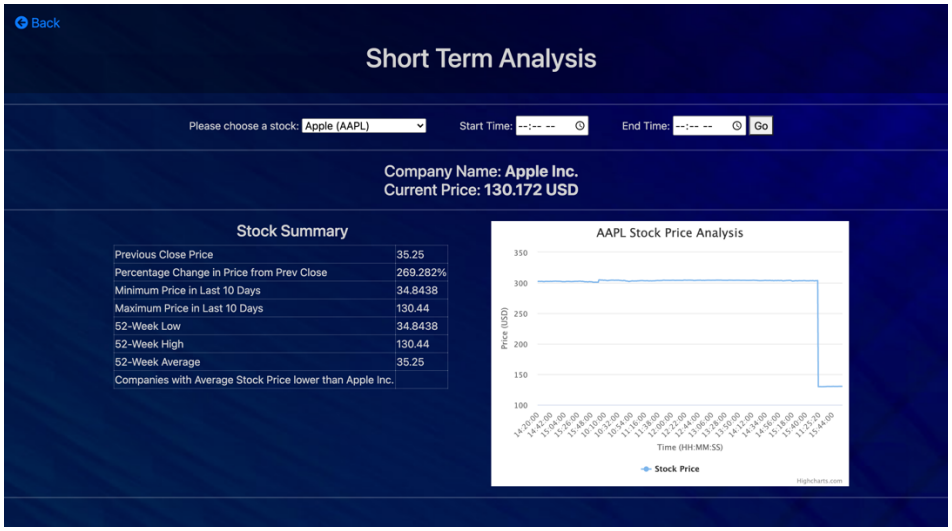
- Analysis.html: this template is the home template for the analysis part of the application. This is basically a statistical summary for the long term data (historical data) and the short term data (real-time data). The page displays a brief message about the analysis being done and then shows 2 buttons one for each type of data.



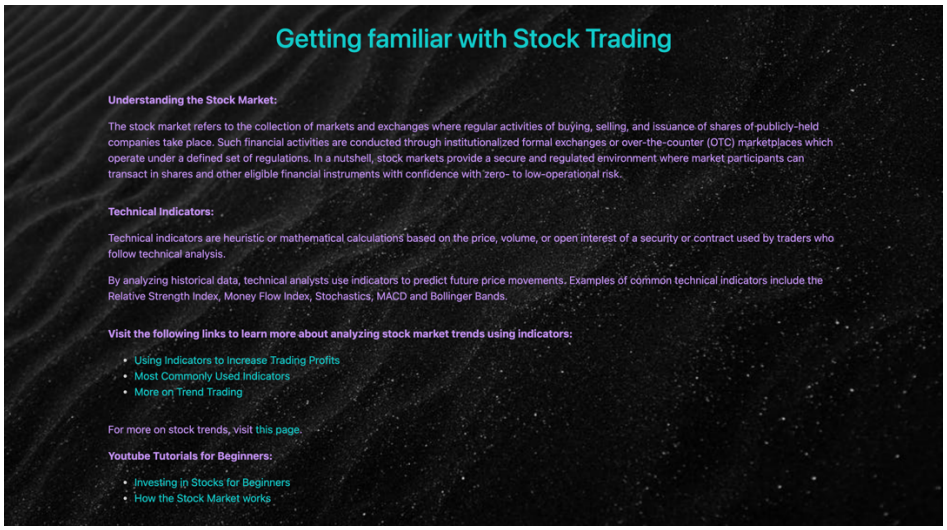
- Historical_data.html: This is the long term analysis page. The page shows a form on the top where the user can choose the stock and the time period where he wants the analysis to be done. After you click submit, the graph will be displayed showing the trend of the stock prices history.



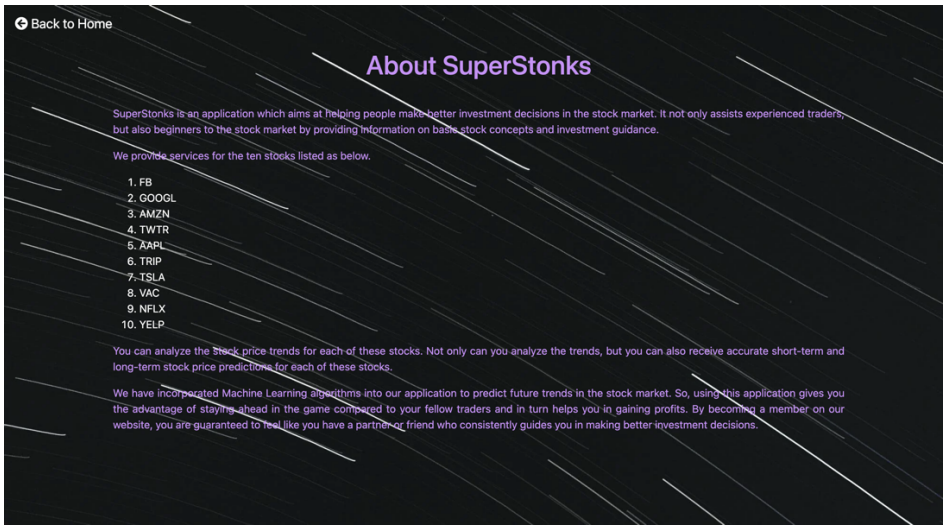
- Shortterm.html: This is the short term analysis page. The page shows a form on the top where the user can choose the stock and the time period where he wants the analysis to be done. After you click submit, the graph will be displayed showing the trend of the stock prices history.



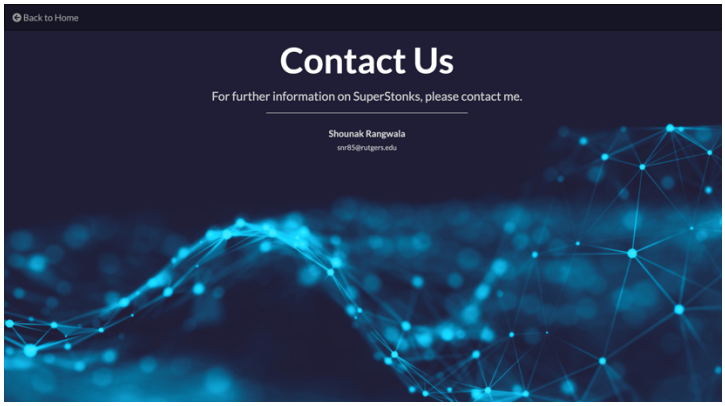
- Learning_page.html: This template shows the resources and the links to other websites from which a new user can learn about the stock market before he starts using the application or the predictive models.



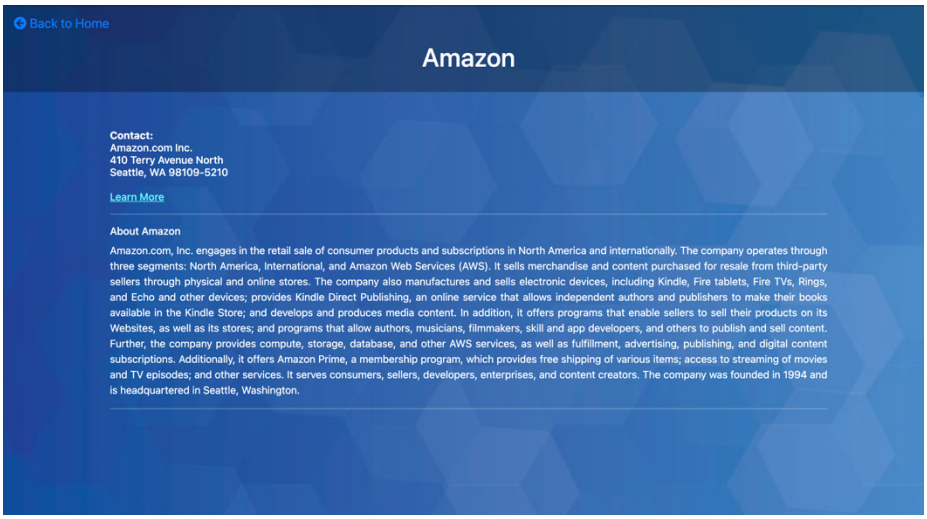
- About.html: describes the SuperStonks application, the motivation behind making this application and how it works



- Contact.html: A basic contact us template



- Company templates: These are the 10 templates which show the descriptions of the 10 companies whose stocks are being monitored by SuperStonks.



Controller: The controller part of the service is basically divided up into the views.py file and the urls.py file. The urls.py file calls upon the routers which route the different urls the application will use to the different views we make in the views.py. The views.py is the most important file in the entire Django project and acts like the spine, connecting the templates to the database and the scripts. For example, the myhome function written in views.py works as follows. When the request to get the home.html page on the browser is sent to the server, the views.py file sends this request to the myhome function. The function then queries the database and gets the names of the stocks from the Stocks table and the real time prices of the stock from the RealTime table. Once the data is stored in lists in the function, a chart is made using that data and the function returns a render method which specifies the template to be rendered (in this case home.html) and the different arguments that will be sent to the home.html to display in itself, namely the names of the stocks and the graph of their values. The following is the list of the most important functions the views.py file contains:

- Myhome: calls upon the default landing page of the app and sends data to be displayed on the web page.
- Long_analysis: calls upon the long_analysis.html page which both takes in a POST request from the form, gets the historical data from the database and displays a chart of it below.
- Short_analysis: calls upon the short_analysis.html page which both takes in a POST request from the form, gets the real-time data from the database and displays a chart of it below.
- Long_predict: gets the name of the stock and the number of days needed for the prediction from the request and then uses them as arguments and calls upon the neural network. The output (prediction) from the neural network is then taken and used in variables to make a chart/graph. Finally, the function renders the predict_form.html page with the results of the prediction sent as json graph data.

- Short_predict: gets the name of the stock and the number of minutes needed for the prediction from the request and then uses them as arguments and calls upon the neural network. The output (prediction) from the curve fitting algorithm is then taken and used in variables to make a chart/graph. Finally, the function renders the predict_form.html page with the results of the prediction sent as json graph data.

Data collection:

The feature that makes this application useful and unique is the dynamic data collection. The data used while predicting reflects the latest trend in the market and so the user is not given results on historical data collected over years, but over days. The real time data gets collected every working day from the Yahoo Finance website. The frequency of this is every minute. Historical data gets collected every weekday day at the close of the day. A simple print statement makes the new entries visible in the console and you can see new entries coming in.

```
('365', '2021-05-10', '13:08:00', '130.85', '129.48', '131.26', '130.2100067138672', '746676
44.0')
('267', '2021-05-10', '13:08:00', '3319.09', '3289.07', '3330.78', '3291.610107421875', '450
6685.0')
('146', '2021-05-10', '13:08:00', '322.38', '318.75', '322.93', '319.0799865722656', '152128
77.0')
('178', '2021-05-10', '13:08:00', '2363.89', '2346.16', '2371.25', '2351.929931640625', '144
4862.0')
('711', '2021-05-10', '13:08:00', '504.62', '501.16', '508.55', '503.8399963378906', '313278
6.0')
('579', '2021-05-10', '13:08:00', '665.8', '660.22', '690.00', '672.3699951171875', '2339614
1.0')
('292', '2021-05-10', '13:08:00', '54.77', '53.37', '54.92', '53.7900009155273 44', '21748560
.0')
('971', '2021-05-10', '13:08:00', '38.86', '36.86', '40.12', '39.459999084472656', '1051601.
0')
('632', '2021-05-10', '13:08:00', '172.38', '171.53', '178.98', '177.36000061035156', '38494
9.0')
('413', '2021-05-10', '13:08:00', '42.33', '42.22', '47.25', '44.16999816894531', '5358728.0
')
```

The way this is done is by using a scheduler library. I am using the apscheduler library which has the Backgroundscheduler module. This is act as a scheduler and regularly perform a job that we give it chronologically from a start time to an end time at set intervals. SO the 2 jobs given to the scheduler in SuperStonks are to get historical data and real-time data. The function is called in the apps.py file of the main web app so each time you start the service, the schedulers are put into action and newer data gets added to the database.

Prediction models and algorithms:

We are using two types of algorithms, short-term and long-term algorithms.

1. Short-term predictions: 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. 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.

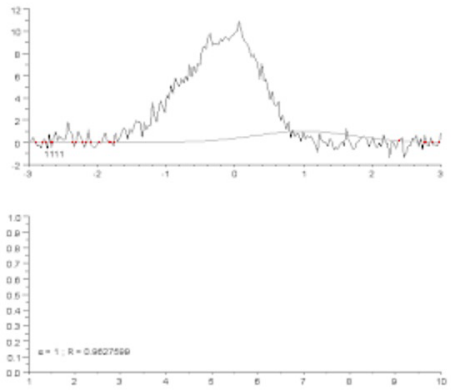
The input and output for this prediction:
Input:

- Stock ticker symbol
- Number of minutes you want to be predicted

Output:

- List of the values predicted
- Suggestion to buy/sell depending on the market trend
- Graph showing the trend of the indicators used to evaluate the results

Algorithm used:
The algorithm used is Polynomial curve fitting.
Given a set of data points (xi, yi), i = 1, 2, 3....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:

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

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 updating of the parameters w and M based on the training data which is used in the fitting process.

TO find the value of M needed for the model, I had to try out different values and settled on M=2. Also the weights of the polynomial were decided by evaluating the least square error and finding out which values gave the least error. The process involved using matrices to represent these weights and then changing them over iterations.

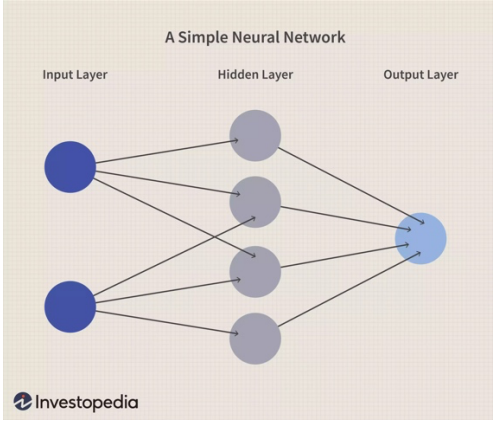
The model then would take in N different time points and the values of the stock at those time points and then predict the value of the (N+1)st time point.

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. 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.

   The input and output for this prediction:
   Input:
   - Stock ticker symbol
   - Number of days you want to be predicted

   Output:
   - List of the values predicted
   - Suggestion to buy/sell depending on the market trend
   - Graph showing the trend of the indicators used to evaluate the results

   Algorithm used:
   The algorithm used is Neural Network.
   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, σ.



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

- Calculating the predicted output ŷ, 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. 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.

SuperStonks NN Architecture:

- Number of layers: 5 (1 Input layer, 2 hidden layers with 10 and 8 neurons respectively and final output layer)
- Activation function: ReLU for hidden layers and linear activation for output layer
- Training: Split the data into standard 80% training data and 20% testing data. Training was done for 50 epochs.

A moving window approach was taken where all the values inside the window will be passed through as training data and the value predicted will be the first value outside the window. The window then moves one entry forward and does the same for all the entries in the training data. The window approach is better for our case because this also accounts for the changing trends in the stock prices over time. For my neural network I have set the window size as 5 because the stock trends rarely stay the same after any 5 days.

Indicators:

So after the predictions are made by the models, the experienced traders will have their way of deciding if they want to buy/sell or hold their stocks. However the new traders will not know the intricacies of judging trends and may end up making blunders. For the interest of such users, SuperStonks provides a suggestion attached with each prediction done which may be to buy, to hold or to sell the stock. This is a suggestion made by studying the past, present and predicted stock price and the indicators used for this suggestion are as follows:

1. Exponential moving average (EMA):

(Source: Investopedia)

The exponential moving average (EMA) is a technical chart indicator that tracks the price of an investment (like a stock or commodity) over time. The EMA is a type of weighted moving average (WMA) that gives more weighting or importance to recent price data. Like the simple moving average, the exponential moving average is used to see price trends over time and watching several EMAs at the same time is easy to do with moving average ribbons.

The formula for calculating the EMA is a matter of using a multiplier and starting with the SMA. There are three steps in the calculation (although chart applications do the math for you):

1. Compute the SMA
2. Calculate the multiplier for weighting the EMA
3. Calculate the current EMA

The calculation for the simple moving average is the same as computing an average or mean. That is, the SMA for any given number of time periods is simply the sum of closing prices for that number of time periods, divided by that same number.

So, the final formula of the EMA would be something like this:

$$EMA = Price(t) * k + EMA(y) * (1 - k)$$

Where,

t= today,

y=yesterday,

N= number of days in the EMA

k= 2/(N+1)

Defined by their characteristic three-dimensional shape that seems to flow and twist across a price chart, moving average ribbons are easy to interpret. The indicators trigger buy and sell signals whenever the moving average lines all converge at one point. Traders look to buy on occasions when shorter-term moving averages cross above the longer-term moving averages from below and look to sell when shorter moving averages cross below from above.

2. Relative strength index (RSI):

The relative strength index (RSI) is a momentum indicator used in technical analysis 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. Generally, when the RSI surpasses the horizontal 30 reference level, it is a bullish sign and when it slides below the horizontal 70 reference level, it is a bearish sign. Put another way, one can intepret RSI values of 70 or above indicate a security is becoming overbought or overvalued and may be primed for a trend reversal or corrective price pullback. An RSI reading of 30 or below indicates an oversold or undervalued condition.

During trends, the RSI readings may fall into a band or range. During an uptrend, the RSI tends to stay above 30 and should frequently hit 70. During a downtrend, it is rare to see

the RSI exceed 70, and the indicator frequently hits 30 or under. These guidelines can help determine trend strength and spot potential reversals. For example, if the RSI isn't able to reach 70 on a number of consecutive price swings during an uptrend, but then drops below 30, the trend has weakened and could be reversing lower.
The opposite is true for a downtrend. If the downtrend is unable to reach 30 or below and then rallies above 70, that downtrend has weakened and could be reversing to the upside. Trendlines and moving averages are helpful tools to include when using the RSI in this way.

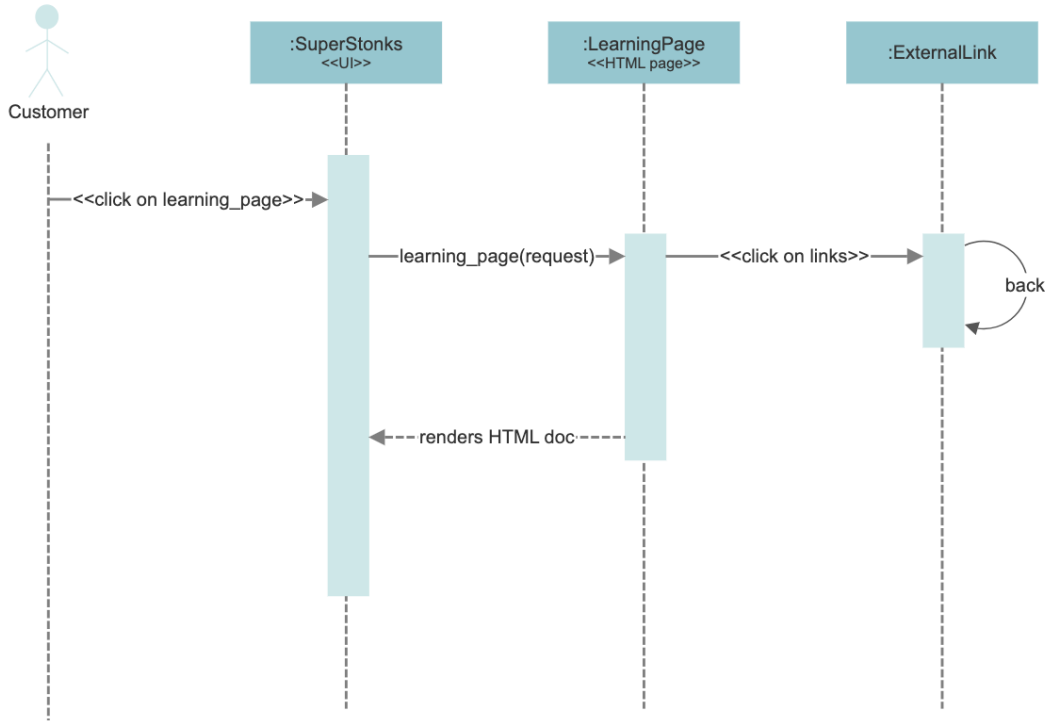
3. Moving Average convergence/divergence (MACD) :
Moving average convergence divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. The MACD is calculated by subtracting the 26 period exponential moving average (EMA) from the 12-period EMA.

How are the suggestions being made?

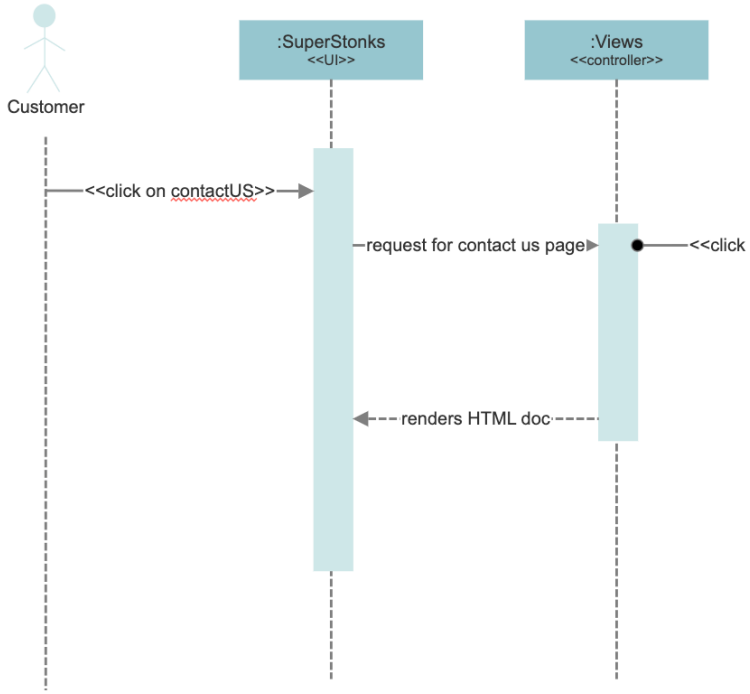
- Calculate the EMA for 26 data points (EMA26) and 12 data points (EMA12) for short term or calculate the EMA for 50 data points (EMA50) and 200 data points (EMA200) for long term.
- Calculate the MACD from these values by subtracting EMA26 from EMA12 (for short term) or subtracting EMA200 from EMA50 (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.
- 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

Interaction diagrams:

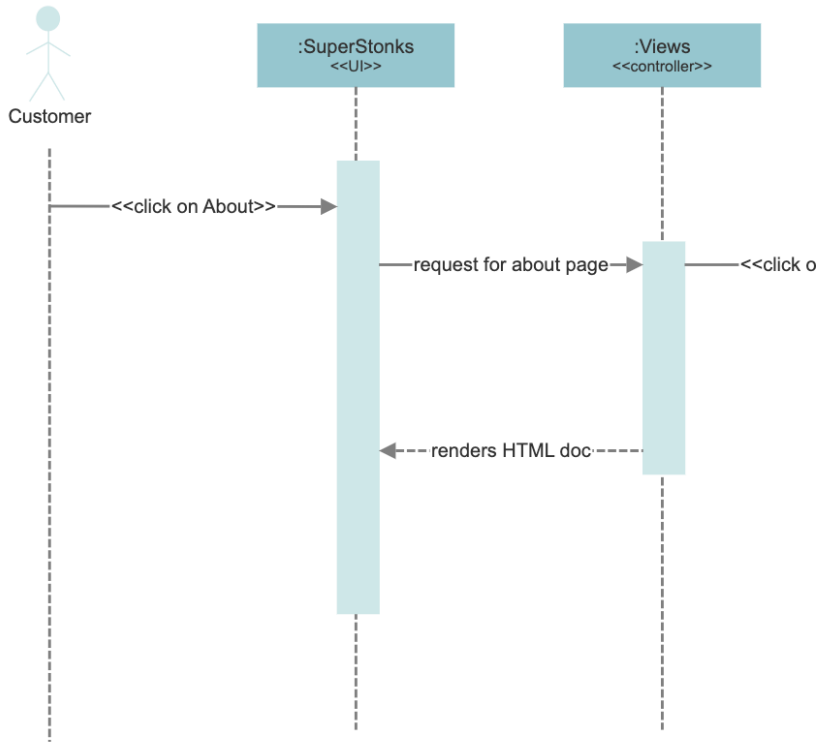
1. Navigating to the learning page: how to get from the home page to the learning page which has different links that lead to outside websites



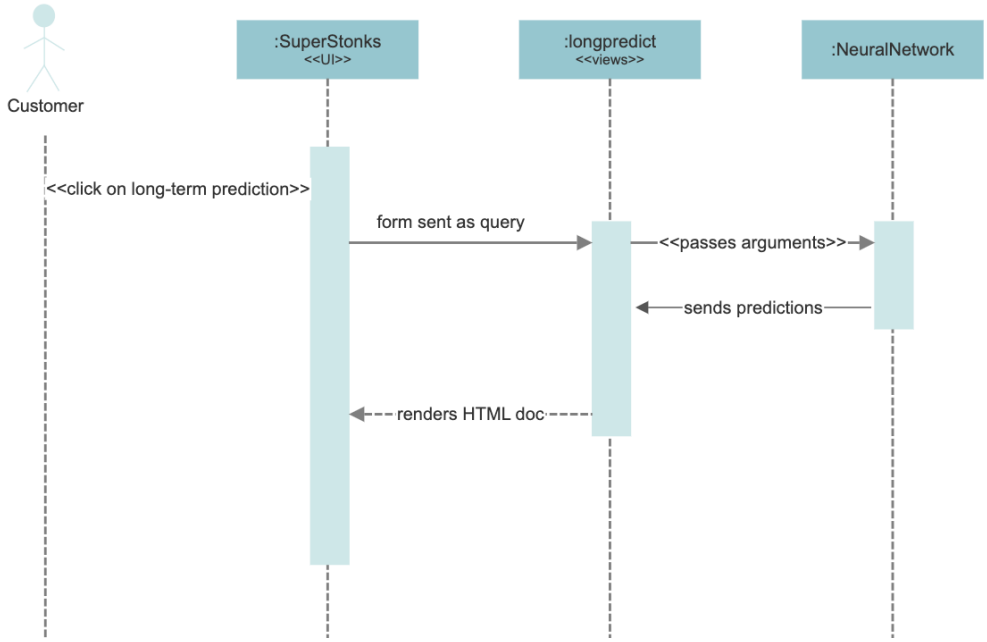
2. Navigating to the contact us page: how to go from the home page to the contact us page



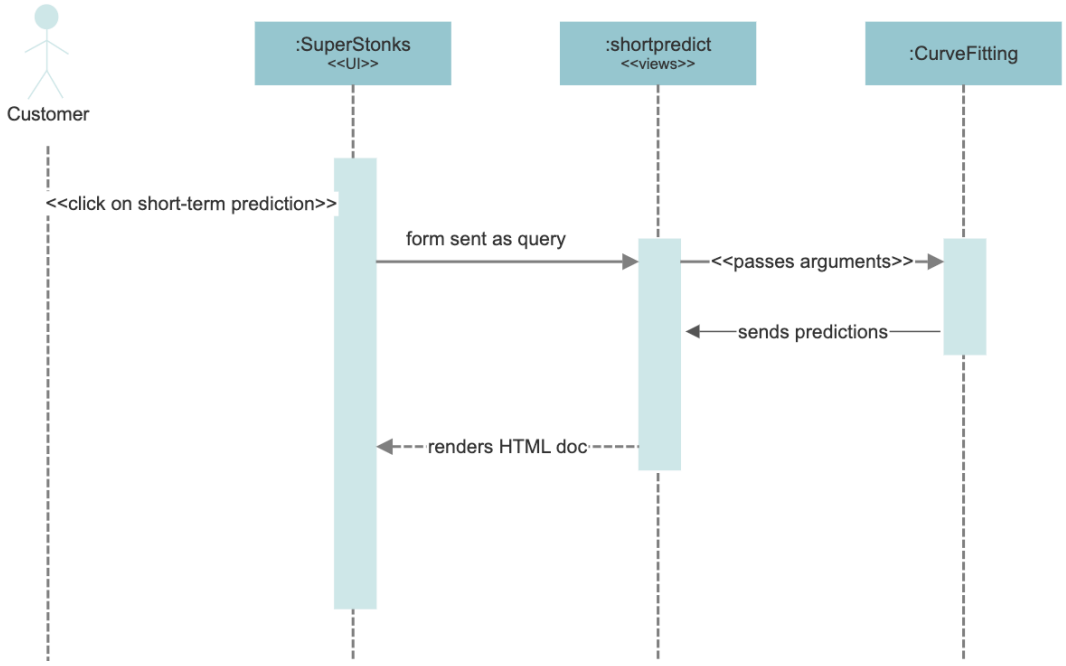
3. Navigating to the about us page: How to go from the home page to the about page which describes what SuperStonks is



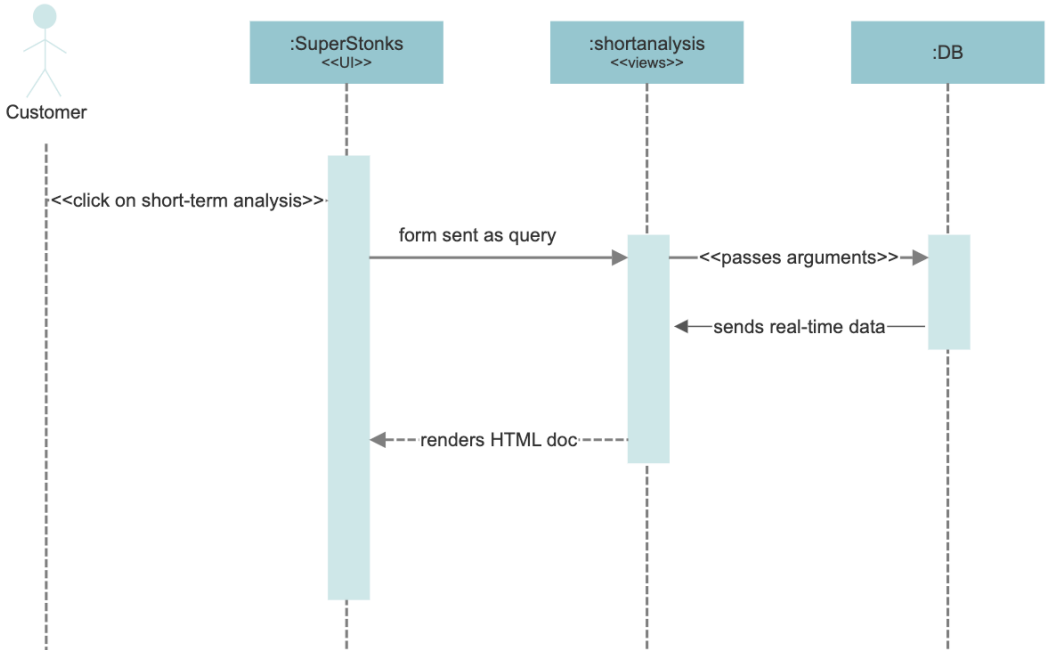
4. Predictions: long-term : Shows how the input for the predictions is being taken from the user and then the results are being displayed on the browser.



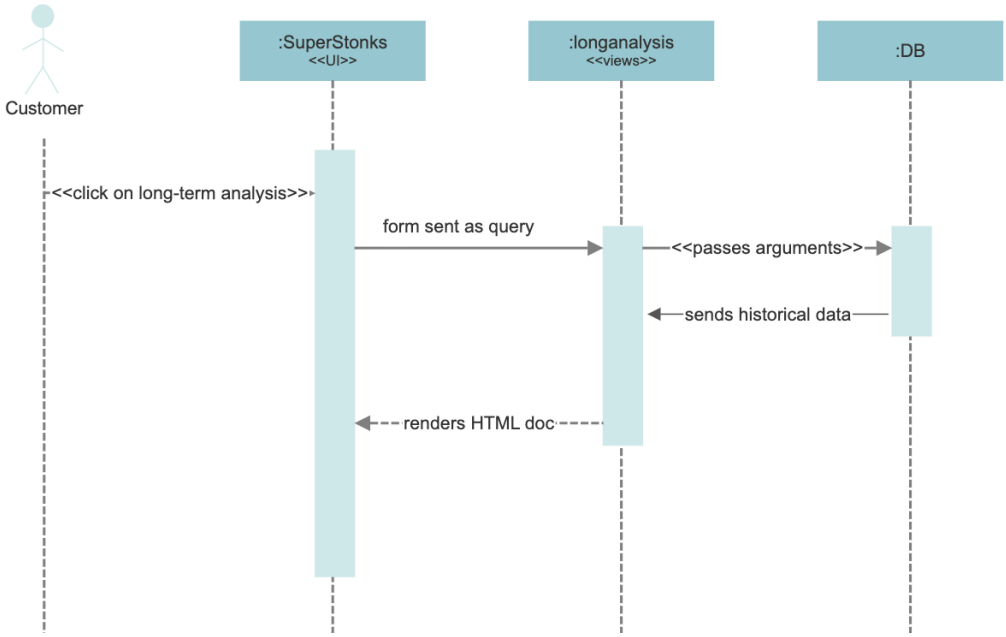
5. Predictions: short-term: shows how the input is being taken from the user and the results are being displayed on the browser



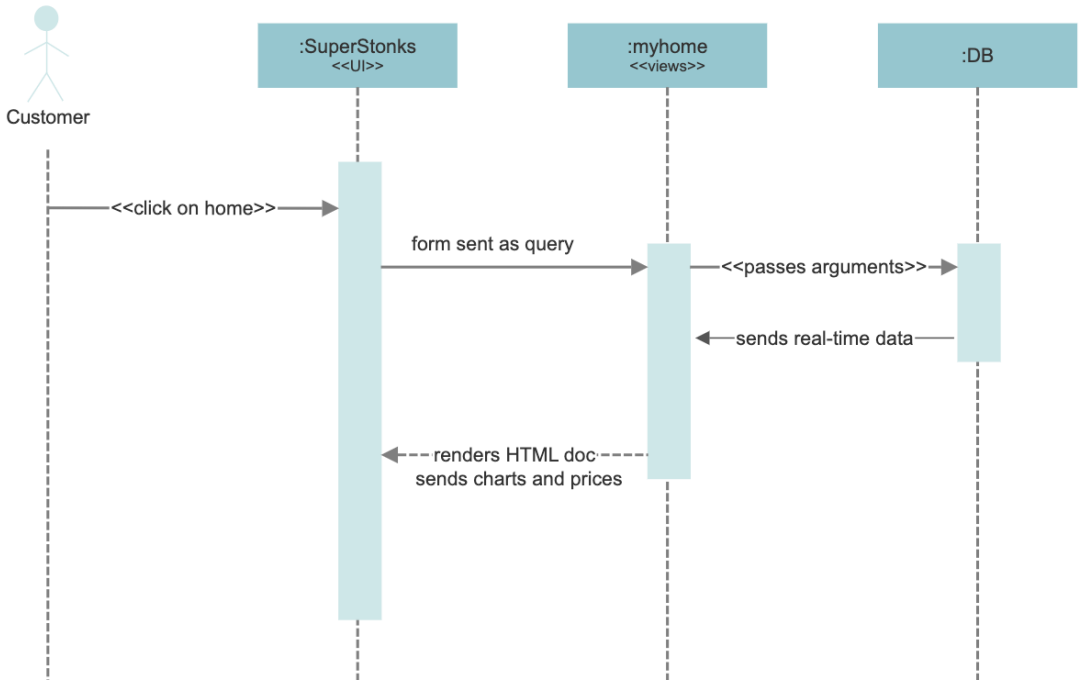
6. Analysis: short-term: collects the real-time data from the database and then makes a graph about it and displays it on the browser



7. Analysis: long-term: collects the historical data from the database and then makes a graph about it and displays it on the browser



8. Get real time stock prices: the home page shows the latest prices of the stocks and displays it in a chart so this is how the entire data is brought from the databse to the html page and rendered.



Conclusion:

SuperStonks is a Django based web application which is both easy to use as well as unique. Imagine a web application that is easily accessible in your browsers, that can give you the stock predictions on the top 10 stocks in the market. This application gives you access to the prediction models free of cost and the accuracy of the models are 54% and 47% respectively. The product itself was not without its difficulties. The connection between the mySQL database and the Django project was difficult to be set up when the data was being accessed by the templates and when it is being accessed by the python scripts. The updating of the data every day/minute was difficult to set up and sending the data to the database was difficult too.

Future Work:

1. Add a more personable experience by adding a login and a signup feature so that more users can login and save their personalized predictions and also access their portfolios so that the predictions are specified for them rather than just general market trends
2. Add more stocks to the existing 10 ones so that the limitation there is alleviated. This gives us more flexibility in terms of services provided. We can also include options and futures into the mix.
3. We can try to maintain a cache of data in the browser so that for successive calls to make predictions, the database is not queried repeatedly and this will improve the speed of the entire application.

References:

1. Zekic, Marijana. "Neural network applications in stock market predictions-a methodology analysis." *proceedings of the 9th International Conference on Information and Intelligent Systems*. Vol. 98. No. 1. Citeseer, 1998.
2. Nti, Isaac Kofi, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. "A systematic review of fundamental and technical analysis of stock market predictions." Artificial Intelligence Review (2019): 1-51.
3. Nardo, Michela, Marco Petracco-Giudici, and Minás Naltsidis. "Walking down wall street with a tablet: A survey of stock market predictions using the web." Journal of Economic Surveys 30.2 (2016): 356-369.
4. Dase, R. K., and D. D. Pawar. "Application of Artificial Neural Network for stock market predictions: A review of literature." International Journal of Machine Intelligence 2.2 (2010): 14-17.