

# **PROJECT TITLE**

## **A MINI PROJECT REPORT**

**18CSC305J - ARTIFICIAL INTELLIGENCE**

*Submitted by*

**DIVYA UDAY SINGH [RA2011003010829]**

**ANIMESH SINGH [RA2011003010846]**

*Under the guidance of*

**ARULALAN V.**

Assistant Professor, Department of Computer Science and Engineering

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE & ENGINEERING**

of

**FACULTY OF ENGINEERING AND TECHNOLOGY**



S.R.M. Nagar, Kattankulathur, Chengalpattu District

**MAY 2023**

# **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

## **BONAFIDE CERTIFICATE**

Certified that Mini project report titled “**VISUALISING STOCKS** ” is the bona fide work of **DIVYA UDAY SINGH (RA2011003010829)** and **ANIMESH SINGH (RA2011003010846)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

### **SIGNATURE**

GUIDE NAME

**ARULALAN V**

Assistant Professor

Department of Computing  
Technologies

### **SIGNATURE**

Dr. M. Pushpalatha

**HEAD OF THE DEPARTMENT**

Professor & Head

Department of Computing  
Technologies

## **ABSTRACT**

The project "Visualizing Projects" aims to create a comprehensive and user-friendly platform for visualizing project management data. The platform will allow project managers to track and monitor project progress, identify bottlenecks, and make data-driven decisions. The system will leverage modern data visualization techniques to represent data in intuitive and insightful ways, allowing project managers to quickly identify trends, patterns, and anomalies. The project will be developed using modern web technologies and will be accessible from multiple devices. Ultimately, the project aims to help organizations optimize their project management processes and achieve greater success in their projects.

Creating a single-page web application using Dash (a python framework) and some machine learning models which will show company information (logo, registered name and description) and stock plots based on the stock code given by the user. Also the ML model will enable the user to get predicted stock prices for the date inputted by the user .

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>iii</b>
<b>TABLE OF CONTENTS</b>	<b>iv</b>
<b>LIST OF FIGURES</b>	<b>v</b>
<b>ABBREVIATIONS</b>	<b>vi</b>
<b>1 INTRODUCTION</b>	<b>7</b>
<b>2 LITERATURE SURVEY</b>	<b>9</b>
<b>3 SYSTEM ARCHITECTURE AND DESIGN</b>	<b>11</b>
<b>4 METHODOLOGY</b>	<b>14</b>
<b>5 CODING AND TESTING</b>	<b>19</b>
<b>6 SREENSHOTS AND RESULTS</b>	<b>29</b>
<b>7 CONCLUSION AND FUTURE ENHANCEMENT</b>	<b>31</b>
<b>REFERENCES</b>	<b>32</b>

# LIST OF FIGURES

Fig 1.1 Project Stages

Fig 3.1 System Architecture

Fig 3.2 Working Architecture

Fig 6.1 Web Layout output

Fig 6.2 Graph of stock

Fig 6.3 Stock Price

## ABBREVIATIONS

<b>AI</b>	Artificial Intelligence
<b>ML</b>	Machine Learning
<b>SVR</b>	Support Vector Regression
<b>EMA</b>	estimatedmoving average
<b>IR</b>	Infra red
<b>UART</b>	Universal Asynchronous Receiver/Transmitter
<b>IDE</b>	Integrated Development Environment

# CHAPTER 1

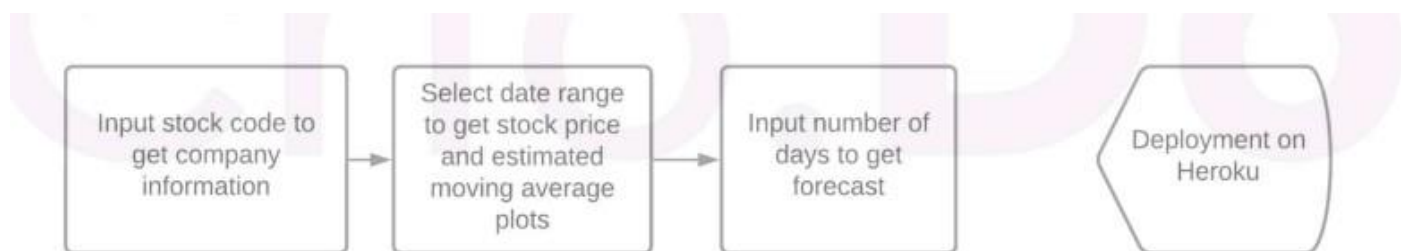
## INTRODUCTION

Stock investments provide one of the highest returns in the market. Even though they are volatile in nature, one can visualize share prices and other statistical factors which helps the keen investors carefully decide on which company they want to spend their earnings on.

Developing this simple project idea using the Dash library (of Python), we can make dynamic plots of the financial data of a specific company by using the tabular data provided by yfinance python library. On top of it, we can use a machine learning algorithm to predict the upcoming stock prices.

This project is a good start for beginners in python/data science and a good refresher for professionals who have dabbled in python/ML before. This web application can be applied to any company (whose stock code is available) of one's choosing, so feel free to explore!

### PROJECT STAGES



The project "Visualizing Stocks" aims to provide investors and traders with a comprehensive and intuitive platform for visualizing stock market data.

The platform will leverage modern data visualization techniques to represent stock market data in a way that is easy to understand and analyze.

The system will allow users to track stock prices, identify trends, and make data-driven investment decisions.

The project will be developed using modern web technologies and will be accessible from multiple devices, making it easy for users to access and analyze stock market data on the go. The goal of this project is to help investors and traders make informed decisions about their investments and achieve greater success in the stock market.

By providing a comprehensive and user-friendly platform for visualizing stock market data, this project will empower users to become more confident and effective in their investment strategies.

The stock market is a complex and constantly evolving environment, and making informed investment decisions can be a daunting task. Investors and traders need access to reliable and up-to-date information about stock prices, market trends, and other factors that can impact the value of their investments. This is where the project "Visualizing Stocks" comes in - by providing a powerful and user-friendly platform for visualizing stock market data, this project aims to help investors and traders navigate the complexities of the stock market and make more informed investment decisions.

The platform will leverage modern data visualization techniques to represent stock market data in a way that is easy to understand and analyze. Users will be able to track the performance of individual stocks, compare the performance of different stocks, and identify trends and patterns that can help them make more effective investment decisions. The platform will also include tools for analyzing market data and generating reports, allowing users to quickly and easily assess the health of the market and identify potential opportunities for investment.

Ultimately, the goal of this project is to help investors and traders become more confident and successful in the stock market. By providing a powerful and intuitive platform for visualizing stock market data, this project will empower users to make better investment decisions, achieve their financial goals, and build long-term wealth.



## **CHAPTER 2**

### **LITERATURE SURVEY**

Here is a brief literature survey for the project "Visualizing Stocks":

1. "Visualizing stock market data with heat maps and tree maps" by Andreas Kerren et al. This paper explores different data visualization techniques for representing stock market data, including heat maps and tree maps. The authors demonstrate how these techniques can be used to identify patterns and trends in stock market data.
2. "Interactive visual analysis of financial markets using self-organizing maps" by Fabian Beck et al. This paper presents a novel approach to visualizing stock market data using self-organizing maps (SOMs). The authors demonstrate how SOMs can be used to cluster and visualize large amounts of financial data, making it easier for investors and traders to identify patterns and trends.
3. "Interactive data visualization for financial market analysis" by Ronghua Liang et al. This paper presents a web-based platform for visualizing financial market data. The platform includes a range of interactive visualization tools, including candlestick charts, line charts, and scatter plots, as well as tools for data exploration and analysis.
4. "Stock price prediction using support vector regression on daily and upsampled time series data" by Saeid Asadi Bagloee et al. This paper explores the use of support vector regression (SVR) for predicting stock prices. The authors demonstrate how SVR can be used to model stock price trends and predict future stock prices, and how these predictions can be visualized using line charts and other data visualization techniques.

5. "Visualizing high-dimensional financial data using t-SNE" by Xiaofeng Zhang et al. This paper presents a technique for visualizing high-dimensional financial data using t-SNE, a popular dimensionality reduction algorithm. The authors demonstrate how t-SNE can be used to visualize large amounts of financial data in a way that is easy to interpret and analyze.

These papers provide valuable insights into different data visualization techniques and analytical tools that can be used for visualizing stock market data. By leveraging the insights gained from these studies, the project "Visualizing Stocks" can develop a powerful and effective platform for visualizing stock market data and helping investors and traders make more informed investment decisions.

## **CHAPTER 3**

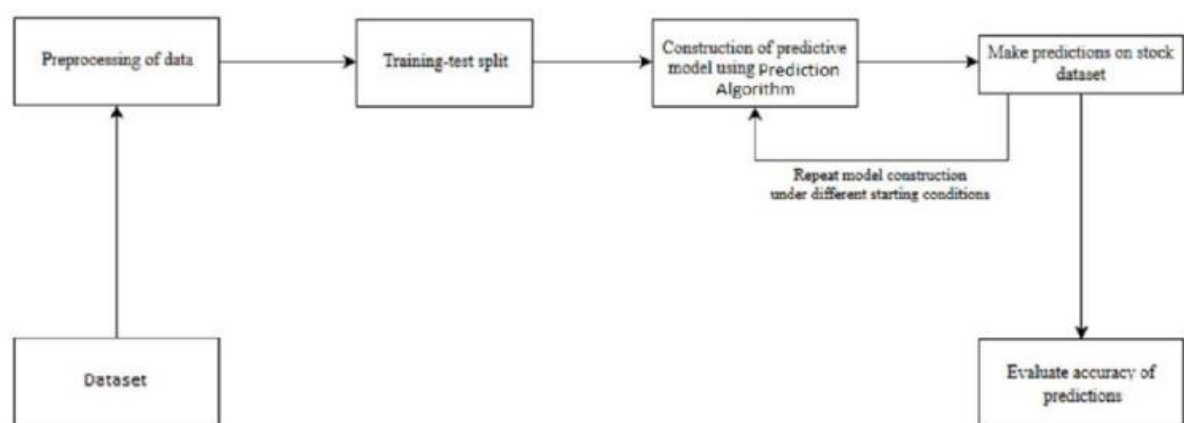
### **SYSTEM ARCHITECTURE AND DESIGN**

The system architecture and design for the project "Visualizing Stocks" can be broken down into the following components:

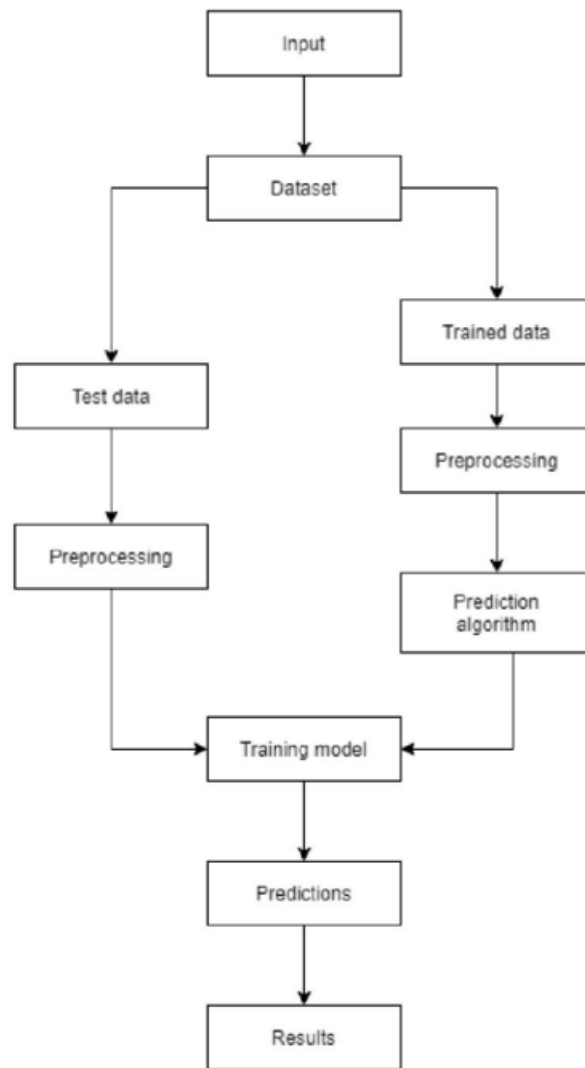
1. **Data Ingestion:** The first component of the system is responsible for ingesting stock market data from various sources, including stock exchanges, financial news outlets, and social media platforms. The data is processed and stored in a database for later use.
2. **Data Processing:** The second component of the system is responsible for processing the raw stock market data and transforming it into a format that can be easily visualized. This includes tasks such as data cleaning, data normalization, and data aggregation.
3. **Data Visualization:** The third component of the system is responsible for visualizing the processed stock market data in a way that is easy to understand and analyze. This includes a range of data visualization techniques such as candlestick charts, line charts, scatter plots, and heat maps.
4. **User Interface:** The fourth component of the system is responsible for providing a user-friendly interface for users to interact with the visualized stock market data. The user interface should be designed to be intuitive and easy to use, allowing users to quickly access and analyze the data they need.
5. **Analytics and Reporting:** The fifth component of the system is responsible for providing advanced analytics and reporting capabilities to users. This includes tools for identifying trends and patterns in the data, generating reports, and providing alerts when certain thresholds are met.

6. Integration: The final component of the system is responsible for integrating with other tools and systems used by investors and traders. This includes integration with trading platforms, investment management systems, and other financial applications.

The overall system architecture and design should be designed to be scalable, robust, and secure, with a focus on providing reliable and accurate stock market data to users. It should also be designed to be easily customizable, allowing users to customize the system to meet their specific needs and preferences. By leveraging modern web technologies and data visualization techniques, the project "Visualizing Stocks" can create a powerful and effective platform for helping investors and traders make more informed investment decisions.



**Fig. System Architecture**



## **WORKING ARCHITECTURE**

## **CHAPTER 4**

### **METHODOLOGY**

#### **Task 1**

##### **Getting started with required files and dependencies**

First validate the idea by doing a low level implementation (Proof of Concept) of the components involved in this project.

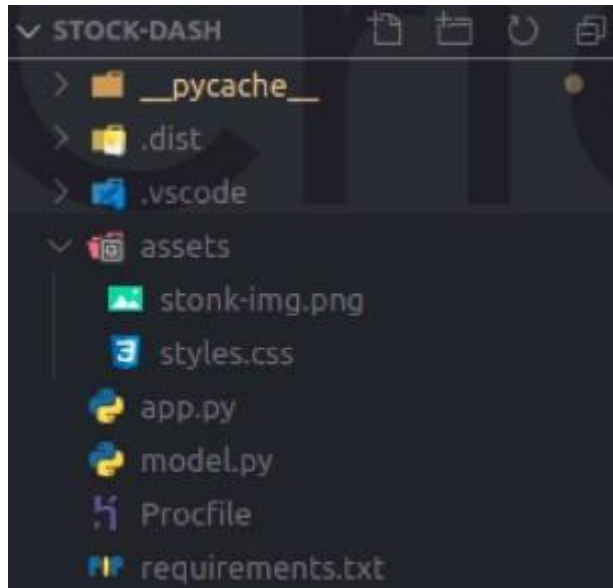
This helps you to:

1.  
Get more clarity around the unknowns.
2.  
Get a better understanding of the stages involved in the project.

We are going to set up the project's environment by setting up the application's starter files/folders. Also the dependencies to be installed will be covered here.

#### **Requirements**

In your working directory, you can create the following file structure.



You may follow the convention mentioned below -

app.py contains web layout and server function. We will be referring to it as our main file. model.py is where we will implement a machine learning model for forecasting the stock price.

The assets folder is where we keep our CSS files for styling and any other miscellaneous files like images (if u wish to include it in your site) requirements.txt is created so that other developers can install the correct versions of the required Python packages to run your Python code.

The Procfile is created for deployment using Heroku. It is not needed to run the app locally.

Install necessary libraries using pip package installer. It is recommended to use the following packages/libraries for this project.

```
1  dash==1.16.2
2  dash-core-components==1.12.1
3  dash-html-components==1.1.1
4  dash-renderer==1.8.2
5  dash-table==4.10.1
6  Flask==1.1.2
7  Flask-Compress==1.5.0
8  gunicorn==20.0.4
9  lxml==4.5.2
10 numpy==1.18.5
11 pandas==1.1.2
12 plotly==4.10.0
13 scikit-learn==0.23.2
14 scipy==1.5.2
15 sklearn==0.0
16 yfinance==0.1.54
```

## Tip

While installing the packages (preferably while using the terminal) you can install multiple packages/libraries in a single line of command (by separating them by spaces) like this: `$ pip install lib1 lib2 lib3`

numpy library is used for multi-dimensional array operations.

pandas is used for creating DataFrames to efficiently manage the data.

yfinance is a library that allows us to fetch financial data of a company (since its listing in the stock market) from its stock code directly.

gunicorn and lxml libraries will be used for the application's deployment i.e. to host the app on a target server.

sklearn and scikit-learn are tools used in the development of Machine Learning (ML) models.



## **Task 2**

### **Create basic website layout**

The basic layout of the application will be built using Dash in this task.

## **Task 3**

### **Styling the application's web page**

Using CSS we will style our webpage to make it look more neat and user friendly.

## **Task 4**

### **Generating a company's information and graphs**

We are going to use the yfinance python library to get company information (name, logo and description) and stock price history. Dash's callback functions will be used to trigger updates based on change in inputs.

## **Task 5**

### **Creating the machine learning model**

We are now going to build a machine learning model - Support Vector Regression (SVR) for predicting the stock prices.

## Task 6

### Deploying the project on Heroku

Now that your project is complete, feel free to deploy it on Heroku for free

.

The URL should be generated after following the aforementioned steps. It should look like `https://unique-name-here.herokuapp.com`

Heroku has some limitations on free plans. However, you can upgrade by paying a fee.

In case of deployment errors, Heroku maintains a log of it for you to check out.

You may use any other hosting services too if you are uncomfortable with using Heroku.

## CHAPTER 5

### CODING AND TESTING

#### 1. BASIC WEBSITE LAYOUT :-

Import relevant libraries as shown below:

```
import dash
import dash_core_components as dcc
import dash_html_components as html
from datetime import datetime as dt
```

Create a Dash instance and store it in an app variable. Also, store the application's server property in a server variable as it will be frequently used.

```
app = dash.Dash(__name__)  
server = app.server
```

Make the web layout using Dash HTML Components and Dash Core Components, then store it in the app's layout component i.e. the `app.layout`. Your code should look something like this:

```
app.layout = html.Div([item1, item2])
```

We mainly need two divisions (`.Div`) for the entire layout.

The first one is for our inputs like stock code, date range selector, number of days of forecast and buttons. These components are the ones which the user will be interacting with. They should be given appropriate IDs and class names for the upcoming tasks. You can follow the code given below:

```
html.Div([
    [
        html.P("Welcome to the Stock Dash App!", className="start"),
        html.Div([
            # stock code input
        ]),
        html.Div([
            # Date range picker input
        ]),
        html.Div([
            # Stock price button
            # Indicators button
            # Number of days of forecast input
            # Forecast button
        ]),
    ],
    className="nav")
```

The second division will be for the data plots and company's basic information (name, logo, brief intro) only. Leave the divisions inside them as blank and give unique IDs because these will get updated as you will see in the next task. You can refer to the code below:

```
html.Div(  
    [  
        html.Div(  
            [ # Logo  
              # Company Name  
            ],  
            className="header"),  
        html.Div( #Description  
            id="description", className="decription_ticker"),  
        html.Div([  
            # Stock price plot  
        ], id="graphs-content"),
```

```
        html.Div([  
            # Indicator plot  
        ], id="main-content"),  
        html.Div([  
            # Forecast plot  
        ], id="forecast-content")  
    ],  
    className="content")
```

Write the following at the end of the main file for running the app in development mode (mandate this step for now)

```
if __name__ == '__main__':  
    app.run_server(debug=True)
```

- 

Dash HTML Components are similar to HTML syntaxes, so try relating the two syntaxes together.

Dash Core Components are useful for building certain things like input methods (text, number, range, slider etc).

The equivalent of HTML's class is className in Dash layout.

DatePicker Range component allows the user to pick a start and end date. It will be

used to update the stock plots accordingly.

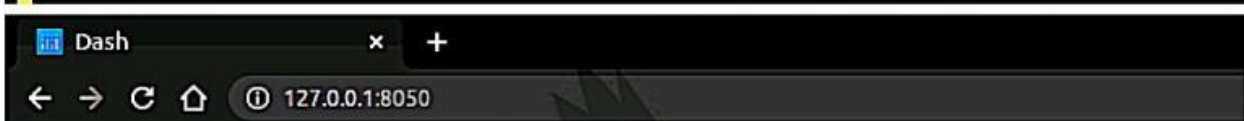
Use the following command in the terminal of the working directory to run your Dash app's server locally - `$python3 app.py`

## Expected Outcome

By now you should have the basic web page setup (as shown below in the second image) which can be seen by starting the server locally (as shown below in the first image).

```
sujoy@sujoy-VivoBook-ASUSLaptop-X412FAC-X412FA ~/D/M/stock-dash (master)>
/usr/bin/python3 "/home/sujoy/Documents/My Projects/stock-dash/app.py"
Dash is running on http://127.0.0.1:8050/

* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
```



Welcome to the Stock Dash App!

Input stock code:

<input type="text"/>	Submit		
Start Date	→ 01/04/2021		
Stock Price	Indicators	number of days	Forecast

## 2. Styling the application's web page :-

Give a class name to the parent division (the one which contains both our main divisions) you created in Task 1. Name it something appropriate like container.

Style the container division by giving the display property flex value. This will ensure that both the divisions are laid out in the same horizon.

```
.container {  
  display: flex;
```

Similarly, style the first division (the one containing our inputs). Again, give the

display property flex value and the flex-direction property column value. Give it a suitable width too, ideally less than 40%. You can keep the items aligned at the center.

```
.inputs {  
  width: 25vw;  
  align-items: center;  
  display: flex;  
  flex-direction: column;  
  justify-content: flex-start;  
  background-color: rgb(5, 107, 107);  
}
```

Give class names to any of the components you want to further style.



### **3. Generating Company's information and Graphs :-**

In the main file, import relevant libraries as shown below

```
import yfinance as yf  
  
import pandas as pd  
  
import plotly.graph_objs as go  
  
import plotly.express as px
```

Make callback functions to update the empty divisions (.Div) we created in the basic weblayout previously (for company information, stock graph and indicator graph). Dash's callback functions are called whenever an input component's property changes. List out theOutput along with their respective IDs and property of the components you want to change. After that list theInput along with their IDs and property of the components which will be used as a trigger for the change in Output components.

You may also enlist State` to just use the component values without using them as a trigger. You may refer to the code example below

```
@app.callback([
    Output("component-id-1", "property"),
    Output("# Output of component-id-2"),
],
[Input("# Input of component-id-2")],
[State("component-id-4", "property")])
def update_data(arg1, arg2): # input parameter(s)
    # your function here
    return output1, output2
```

In the first callback function, use the input field (for stock code) and submit buttons as **State** and Input components respectively. For the Output component use the first empty Div you had created in Task 2. Use the yfinance library's Ticker function to fetch the company info and return it from the first callback function (you may refer to the code snippet below)

```
ticker = yf.Ticker(val)
inf = ticker.info
df = pd.DataFrame().from_dict(inf, orient="index").T
return # df's first element of 'longBusinessSummary', df's first element
value of 'logo_url', df's first element value of 'shortName'
```

For making the second callback function, use the date range picker's start date, end date and also the stock price button as Input components. For the Output component use the second empty Div we had created in Task 2. You can download the stock price history with the download function of the yfinance library. We get a DataFrame in return. You can pass that DataFrame to a user defined function (which we will make) that returns the required plot (using plotly library). Finally, that plot can be returned from our callback function as a Dash Core Component Graph. You may refer to the code examples below.

```
df = yf.download(# input parameter, start_date str, end_date str )
df.reset_index(inplace=True)
fig = get_stock_price_fig(df)
return # plot the graph of fig using DCC function
```

User defined function for stock price graph generation from a given DataFrame

```
def get_stock_price_fig(df):
    fig = px.line(df,
                  x= # Date str,
                  y= # list of 'Open' and 'Close',
                  title="Closing and Opening Price vs Date")
    return fig
```

- 

The third callback will be for generating our indicator graph. You may use estimated moving average (EMA) for it. The callback function for it will be similar to the second one. However, we need to make a new user defined function which will return an EMA plot over time. You can refer to the code given below

```
def get_more(df):
    df['EWA_20'] = df['Close'].ewm(span=20, adjust=False).mean()
    fig = px.scatter(df,
                    x= # Date str,
                    y= # EWA_20 str,
                    title="Exponential Moving Average vs Date")

    fig.update_traces(mode= # appropriate mode)

    return fig
```

Till here we have completed the app.py file, i.e. the web layout and server functions are in place and hence the app is partially functional (prediction feature is yet to be implemented).

#### 4. CREATE ML MODEL :-

Make the model in the model.py file.

Use the support vector regression (SVR) module from the sklearn library. Fetch the stock prices for the last 60 days. Split the dataset into 9:1 ratio for training and testing respectively.

Use the rbf kernel in GridSearchCV for tuning your hyperparameters.

Then, train the SVR model with the training dataset.

Test your model's performance by using metrics such as Mean Squared Error (MSE)

and Mean Absolute Error (MAE) on the testing dataset.

After the model is built, make sure another callback function (as seen in Task 2) for the same is made in the main file i.e. app.py (where the model is to be imported).

## 5. Create a Heroku account.

Run `$ pip install gunicorn` in the terminal.

Run `$ pip freeze > requirements.txt` in the terminal. This creates a text file containing all our dependencies.

Create a Procfile in the working directory. In that paste web: gunicorn app:server  
Run `$ sudo snap install --classic heroku` on terminal to install the Heroku CLI. In the terminal, run the commands below:

```
$ heroku create unique-name-here
```

```
$ git add .
```

```
$ git commit -m 'Initial app template'
```

```
$ git push heroku master
```

Finally, run the app on one dyno by doing `$ heroku ps: scale web=1`

Deploying Dash

AppsHeroku CLI

The URL should be generated after following the aforementioned steps. It should look like `https://unique-name-here.herokuapp.com`

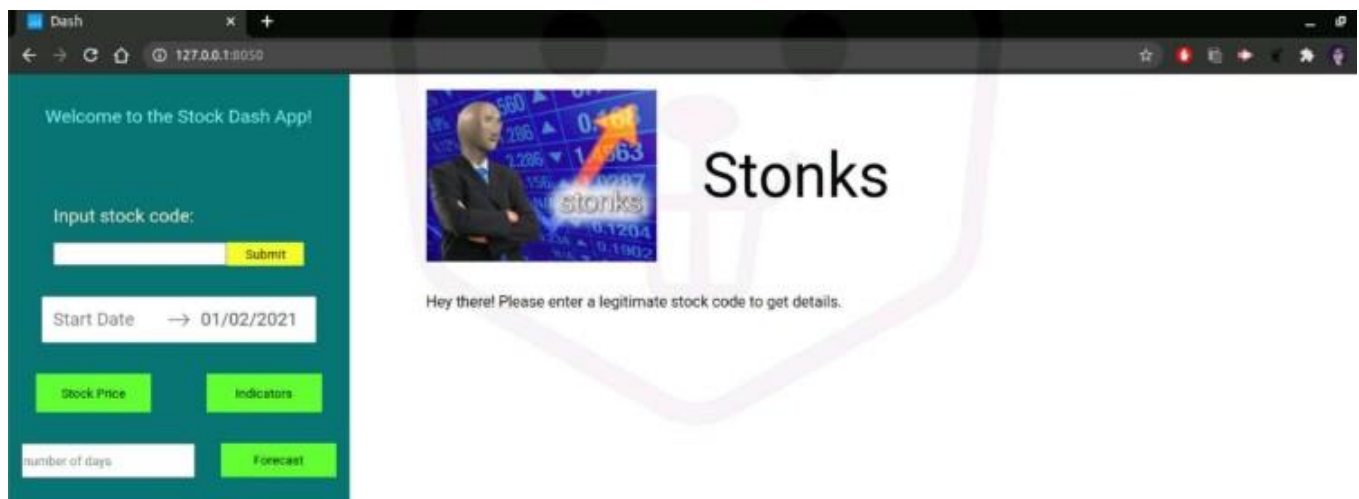
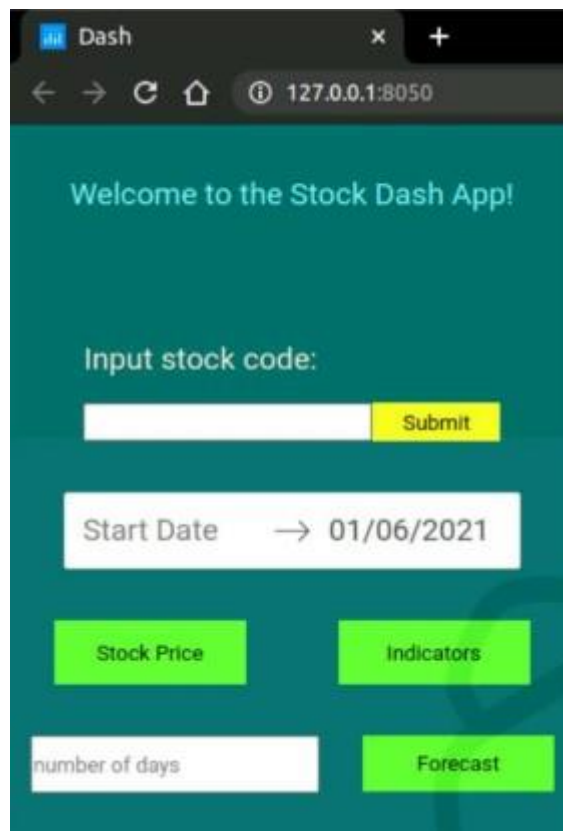
Heroku has some limitations on free plans. However, you can upgrade by paying a fee.

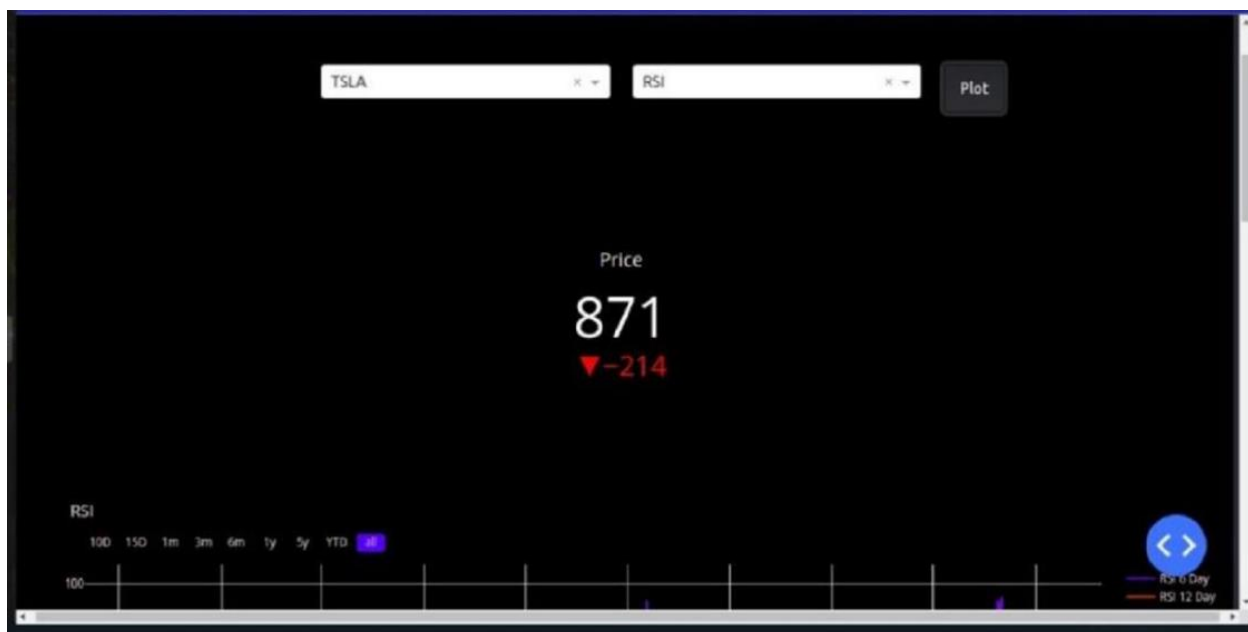
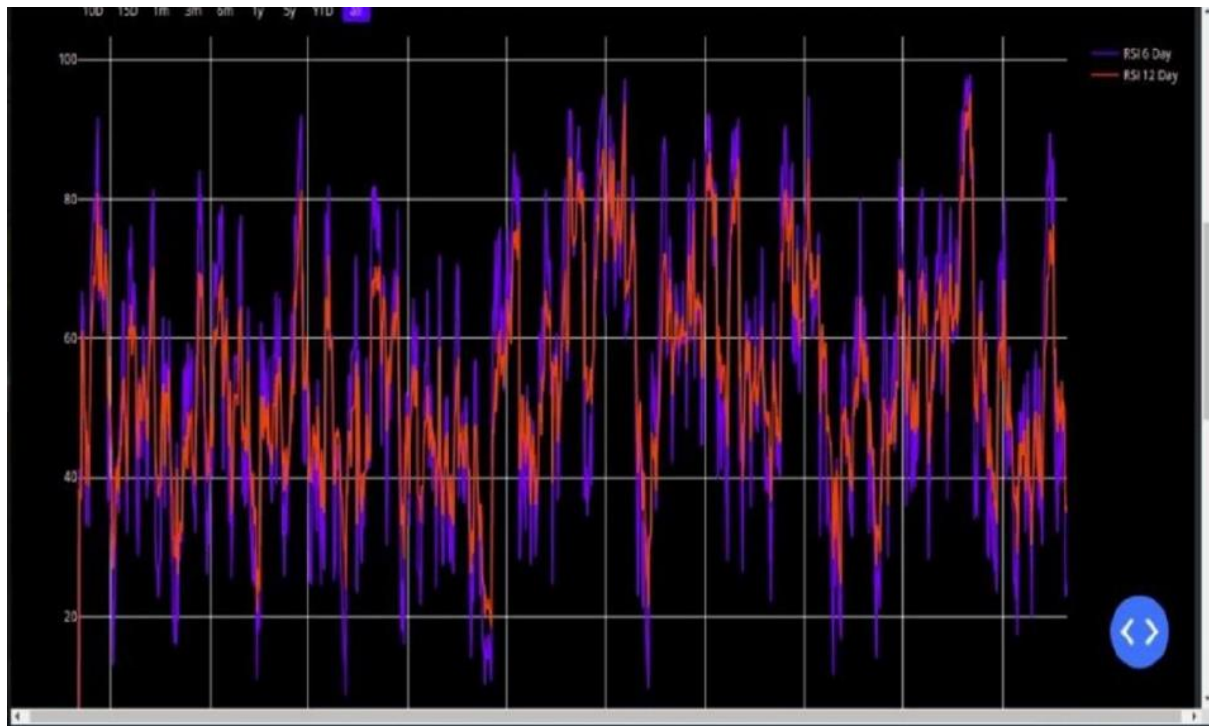
In case of deployment errors, Heroku maintains a log of it for you to checkout.

You may use any other hosting services too if you are uncomfortable with using Heroku.

## CHAPTER 6

### SCREENSHOTS AND RESULTS





## **CHAPTER 7**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

In conclusion, the project "Visualizing Stocks" aims to provide investors and traders with a powerful and user-friendly platform for visualizing stock market data, enabling them to make more informed investment decisions. By leveraging modern data visualization techniques and advanced analytics capabilities, the platform can help users identify trends and patterns in stock market data, compare the performance of different stocks, and generate reports and alerts to support their investment strategies.

However, there are several areas where the project can be further enhanced in the future. One potential enhancement is to incorporate machine learning algorithms to improve the accuracy of stock price predictions and other analytics. Another potential enhancement is to integrate social media and other external data sources into the platform, allowing users to track market sentiment and other factors that can impact the value of their investments.

In addition, the platform can be further customized to meet the specific needs and preferences of different types of users, such as individual investors, institutional investors, and day traders. This can include features such as personalized dashboards, custom reports, and integration with different trading platforms and financial applications.

Overall, the project "Visualizing Stocks" has the potential to revolutionize the way investors and traders access and analyze stock market data, providing them with a powerful and intuitive platform for making informed investment decisions. With ongoing development and enhancement, the platform can continue to evolve and improve, staying ahead of the curve in an ever-changing and competitive financial landscape.



## REFERENCES

- [1]. WWW.Google.Com W3 School.com
  
- [2]. Kimoto T, Asakawa K, Yoda M and Takeoka M (1990), Stock market prediction system with modular neural networks, Proc. International Joint Conference on Neural Networks, San Diego, Vol. 1, pp. 1-6
  
- [3].<https://www.quantinsti.com/blog/predictive-modeling-algorithmic-trading/>
  
- [4]. <https://www.quantinsti.com/blog/free-resources-learn-machine-learning-trading/>
  
- [5].<https://www.quantinsti.com/blog/use-decision-trees-machine-learning-predict-stock-movements/>
  
- [6]. Abraham A, Nath B and Mahanti P K (2001), Hybrid Intelligent Systems for Stock Market Analysis, Proceedings of the International Conference on Computational Science. Springer, pp. 337-345
  
- [7]. S. Hannon, "5 Rules For Predicting Stock Market Trends - StockTrader.com", StockTrader.com, 2016
  
- [8]. <https://arxiv.org/pdf/1301.4944.pdf>