1.3 BACKGROUND

As time is proceeding ahead, technology is improving and evolving every single moment. No one can claim something to be “latest” because the moment someone does so we can see the presence of something newer and better in front of our very own eyes. People endeavor to cope up with the dynamic changes but it is pretty difficult due to lack of availability of adequate and sufficient resources and technology. Two of the basic fundamental intentions of technology are to make things that are not complicated to be understood by the user and makes working of the user more convenient.

Financial analysts investing in stock market usually are not aware of the stock market behavior. They are facing the problem of trading as they do not properly understand which stocks to buy or which stocks to sell in order to get more profits.

In today’s world, all the information pertaining to stock market is available. Analyzing all this information individually or manually is tremendously difficult. As such, automation of the process is required. This is where Data mining techniques help. Understanding that analysis of numerical time series gives close results, intelligent investors use machine learning techniques in predicting the stock market behavior. This will allow financial analysts to foresee the behavior of the stock that they are interested in and thus act accordingly.

1.4 PROBLEM STATEMENT

Problem statement was to predict increase or decrease in stock price for next day. We addressed this as classification problem.. The most interesting task is to predict the market. So many methods are used for completing this task. Methods vary from very informal ways to many formal ways a lot. This tech is categorized as Prediction Methods, Traditional Time Series, Tech Analysis Methods, Mach Learning Methods and Fundamental Analysis Methods. The criteria to this category are the kind of tool and the kind of data that these methods are consuming in order to predict the market. What is mutual to the technique is that they are predicting and hence helping from the market's future behavior.

We are going to implement the following:

- In this project, we-are trying to review the possibility to apply two-known techniques which is neural-network and data-mining in stock market prediction. Extract useful information from a huge amount of data set and data mining is also able to predict future trends and behaviors through neural network. Therefore, combining both these methods could make the prediction much suitable and reliable.
- The most important for predicting stock market prices are neural networks because they are able to learn nonlinear-mappings between inputs and outputs.
- It may be possible to perform better than traditional analysis and other computerbased methods with the neural-networks ability to learn-nonlinear, chaotic-systems.

1.5 OBJECTIVE AND PURPOSE

In the past decades, there is an increasing interest in predicting markets among economists, policymakers, academics and market makers. The objective of the proposed work is to study and improve the supervised learning algorithms to predict the stock price.

The input to our system will be historical data from any finance. Appropriate data would be applied to find the stock price trends. Hence the prediction model will notify the up or down of the stock price movement for the next trading day and investors can act upon it so as to maximize their chances of gaining a profit.

The purpose of this project is to comparatively analyze the effectiveness of prediction algorithms on stock market data.

1.7 SCOPE OF THE PROJECT

- Basically the main objective of this project is to collect the stock information for some previous years and then accordingly predict the results for the predicting what would happen next. So for we are going to use of two well-known techniques neural network and data mining for stock market prediction. Extract useful information from a huge amount of data set and data mining is also able to predict future trends and behaviors through neural network. Therefore, combining both these techniques could make the prediction more suitable and much more reliable.

- As far as the solutions for the above problems, the answer depends on which way the forecast is used for. So the procedures that we will be using have proven to be very applicable to the task of forecasting product demand in a logistics system. Many techniques, which can prove useful for forecasting-problems, have shown to be inadequate to the task of demand forecasting in logistics systems.

1.8 PROJECT SPECIFICATIONS

SOFTWARE SPECIFICATIONS:

- 64-bit Operating System: Windows 8 or Higher.
- Python 3.5
- Flask Framework
- MySQL Database

2. LITERATURE REVIEW

2.1 LIST OF BOOKS AND WEBSITES REFERRED

https://www.quantinsti.com/blog/working-neural-networks-stock-price-prediction/

https://en.m.wikipedia.org/wiki/Time_series

https://enlight.nyc/projects/stock-market-prediction/

https://www.datacamp.com/community/tutorials/lstm-python-stock-market

https://towardsdatascience.com/stock-prediction-in-python-b66555171a2

2.2 EXISTING SYSTEMS AND DRAWBACKS

Number of developers have researched and studied on the stock price prediction. The have got success to achieve their goal, but not fully. Because there is a limitation of market. Any good or bad news in the market will change the direction of the stock price. There this prediction of stock is limited toward study and research only.

Every other developer have tried to predict the stock price to its best. They use graphs and many more other techniques to check the accuracy. Therefore, common user is unable to read the graph and predict the value.

3. METHODOLOGY

The components used in this project can't be specific, since this project is a prototype for all computers. As such, certain prerequisites are as follows:

3.1 TOOLS AND TECHNOLOGIES

PYTHON 3.5

Python is having large number of fameworks which makes coding easier and helps in saving development time.

FLASK FRAMEWORK

Flask is considered more Pythonic than the Django web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

MYSQL

HTML & CSS

RECURRENT NEURAL NETWORK

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and belong to the most promising algorithms out there at the moment because they are the only ones with an internal memory.

RNN's are relatively old, like many other deep learning algorithms. They were initially created in the 1980's, but can only show their real potential since a few years, because of the increase in available computational power, the massive amounts of data that we have nowadays and the invention of LSTM in the 1990's.

Because of their internal memory, RNN's are able to remember important things about the input they received, which enables them to be very precise in predicting what's coming next.

This is the reason why they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more because they can form a much deeper understanding of a sequence and its context, compared to other algorithms.

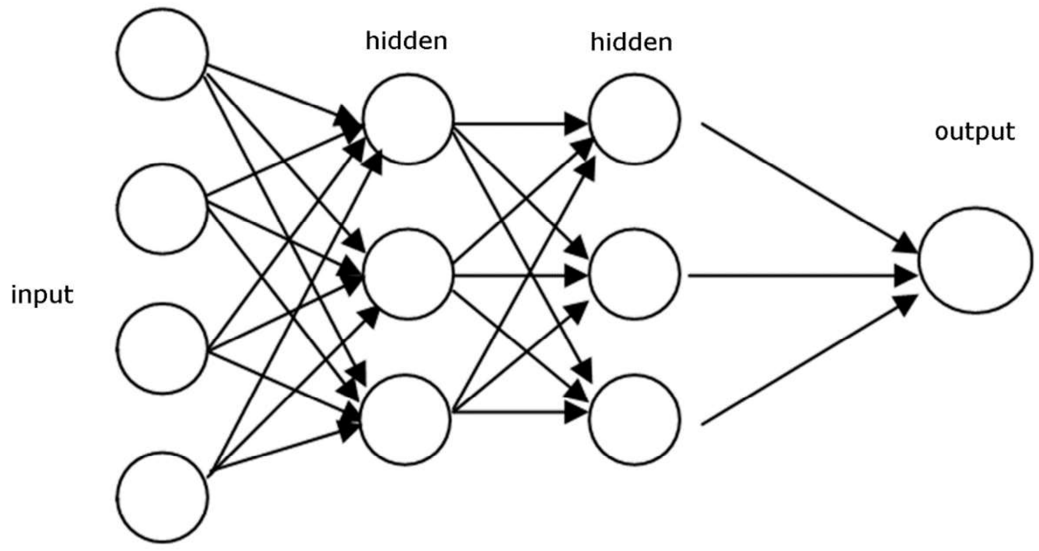


Fig 1

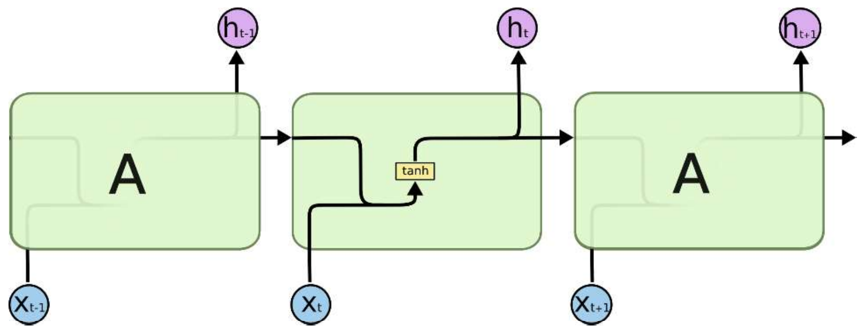
MODEL USED

LSTM (LONG-SHORT TERM MEMORY)

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.[1] They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



The repeating module in a standard RNN contains a single layer.

Fig 2

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It’s very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

3.2 DFD (DATA FLOW DIAGRAM)

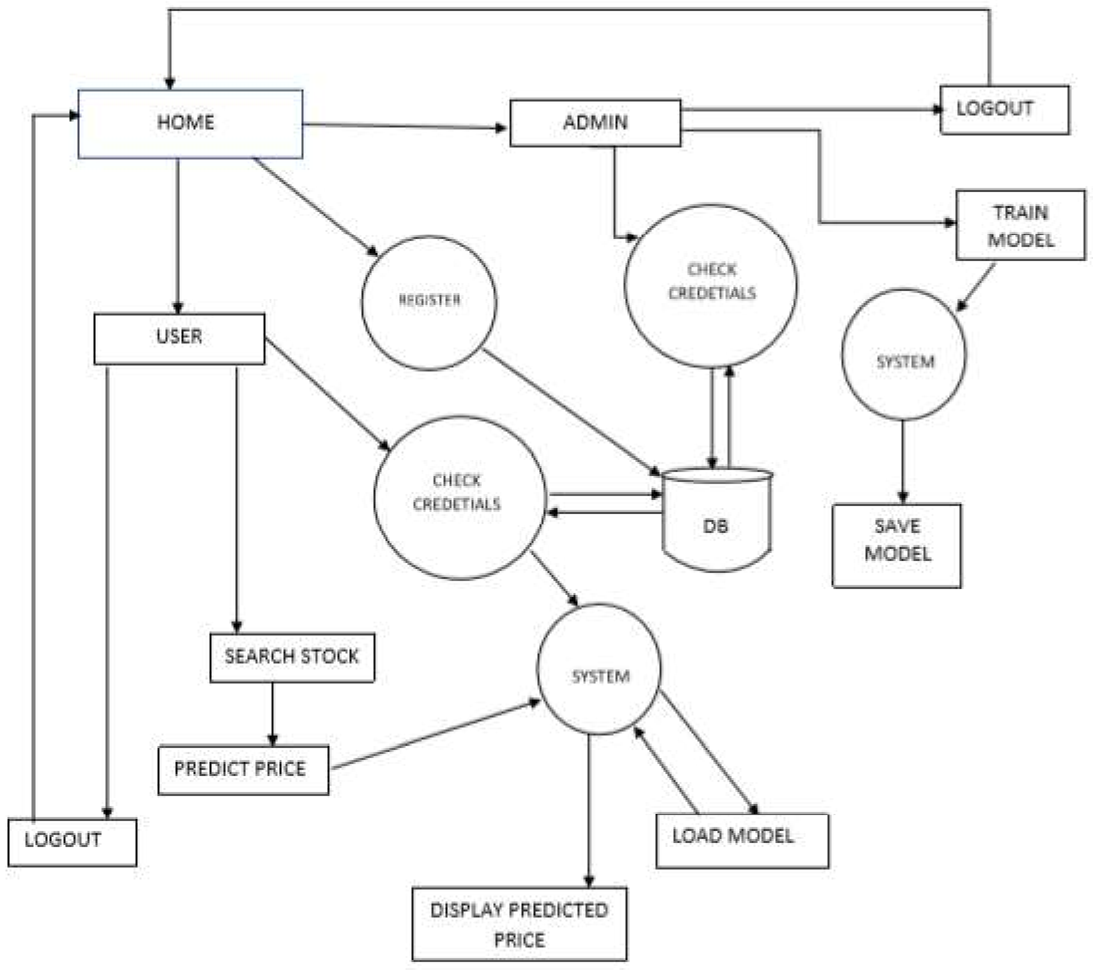
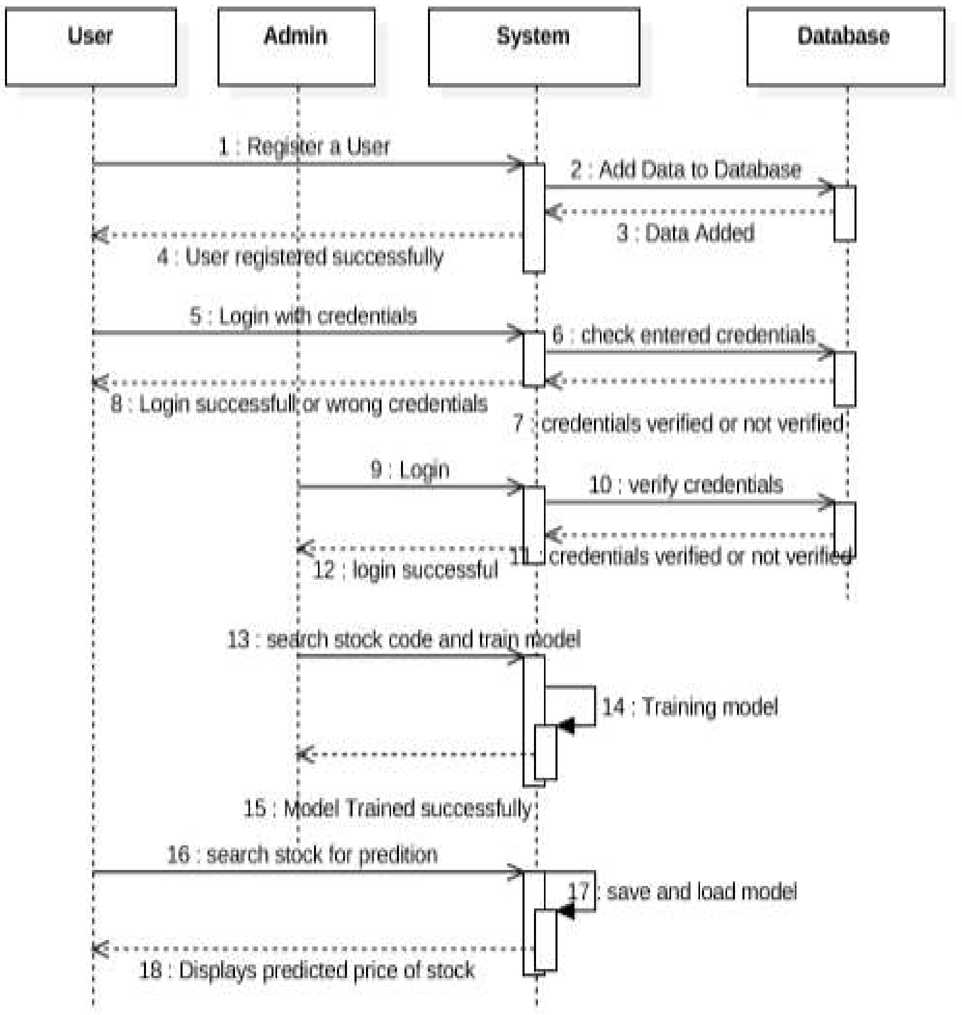


Fig 3

3.3 SEQUENCE DIAGRAM



User
Admin
System
Database
1 : Register a User
2 : Add Data to Database
3 : Data Added
4 : User registered successfully
5 : Login with credentials
6 : check entered credentials
7 : credentials verified or not verified
8 : Login successful or wrong credentials
9 : Login
10 : verify credentials
11 : credentials verified or not verified
12 : login successful
13 : search stock code and train model
14 : Training model
15 : Model Trained successfully
16 : search stock for predition
17 : save and load model
18 : Displays predicted price of stock

Fig 4

3.4 USE-CASE DIAGRAM

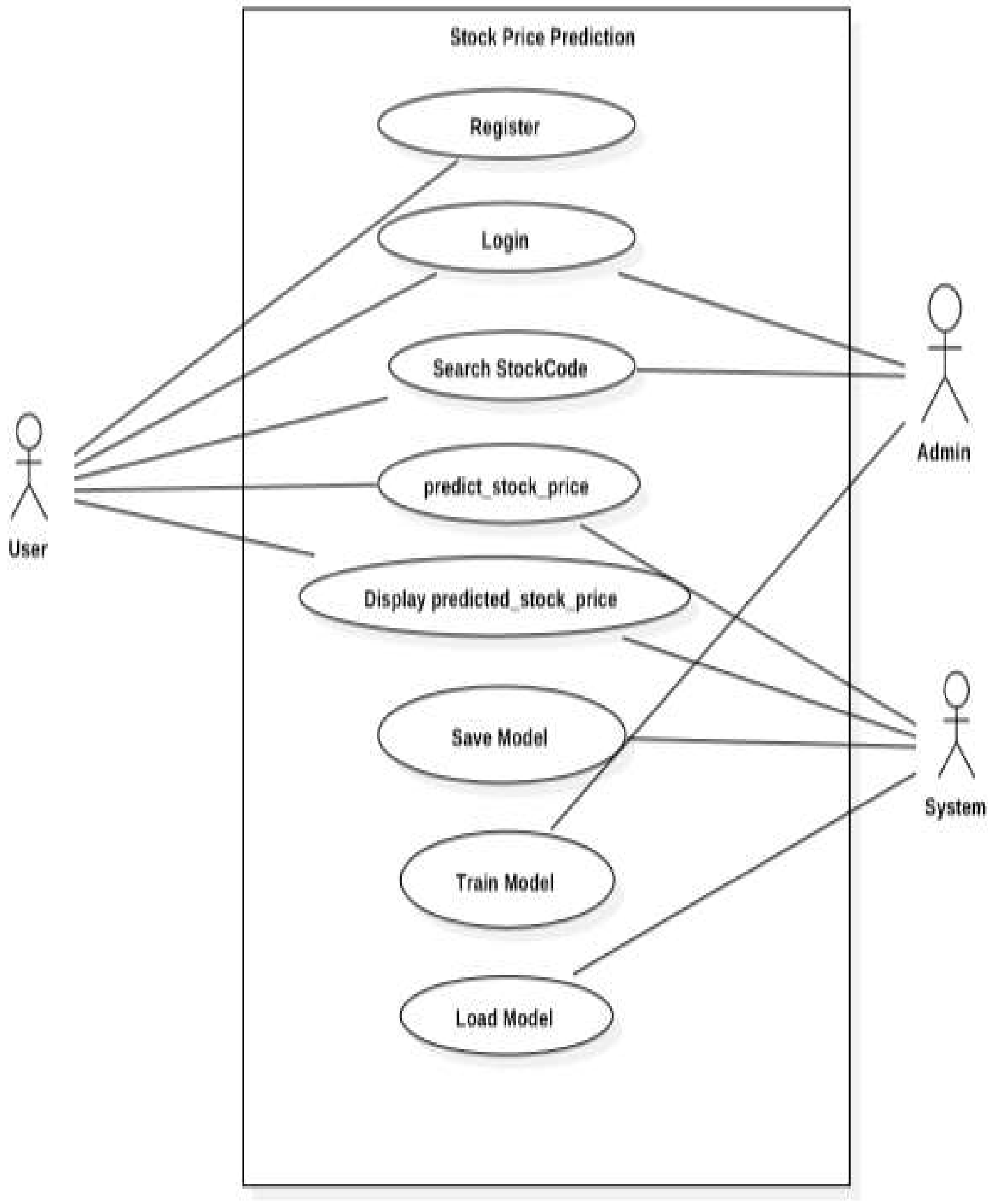


Fig 5

4. IMPLEMENTATION

4.1 CODING

4.1.1. LSTM_Predictor.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
from sklearn.preprocessing import MinMaxScaler
# from nsepy import get_history
from datetime import date
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM
from datetime import datetime, timedelta
from keras.models import load_model
import os
import quandl
from keras import backend as K


##############################################
##
##  Fetching and formatting Data
##
##############################################

scaler = ''

def getTrainingFilePath(stockCode):
	dir = os.path.dirname(__file__)
	filename = os.path.join(dir, "\\StockApp\\LSTM_Models\\"+stockCode+".h5")
	return filename

def getData(stockCode):
	print("Fetching Data")
	end = datetime.today() - timedelta(days=1)
	start = datetime.today() - timedelta(days=765)
	#Get Value of a stock from given range (765 days as per given inputs)
	df=quandl.get("NSE/"+stockCode,start_date = start, end_date =  end, api_key =
"D8VV4PzqvQaWTR3yZtDV").reset_index()
```

```
	data = df.sort_index(ascending=True, axis=0)
	#Copy all the import parameters into new_data
	new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Last', 'Close', 'Open',
'Total Trade Quantity'])

	for i in range(0,len(data)):
		new_data['Date'][i] = data['Date'][i]
		new_data['Close'][i] = data['Close'][i]
		new_data['Last'][i] = data['Last'][i]
		new_data['Open'][i] = data['Open'][i]
		new_data['Total Trade Quantity'][i] = data['Total Trade Quantity'][i]

	#setting index
	new_data.index = new_data.Date
	new_data.drop('Date', axis=1, inplace=True)

	return new_data

def scaleData(new_data):
	print("Scaling Data")
	#creating train and test sets
	dataset = new_data.values
	train = dataset[0:len(dataset),:]
	#valid = dataset[700:,:]
	valid=[]
	global scaler
	#converting dataset into x_train and y_train
	scaler = MinMaxScaler(feature_range=(0, 1))
	scaled_data = scaler.fit_transform(dataset)

	x_train, y_train = [], []
	for i in range(60,len(train)):
		sdx, sdy = [], []
		for j in range(0,4):
			sdx.append(scaled_data[i-60:i,j])
			sdy.append(scaled_data[i,j])
		x_train.append(sdx)
		y_train.append(sdy)
	x_train, y_train = np.array(x_train), np.array(y_train)
	#x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1]*x_train.shape[2],1))
	return (x_train, y_train)


def trainning(stockCode,x_train, y_train):
    ##############################################################
    ##
    ##  Training
    ##
    ##############################################################
    # create and fit the LSTM network
    print("Training Data")
    global model
    model = Sequential()
    model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],60)))
    model.add(LSTM(units=50))
    model.add(Dense(4))
    model.compile(loss='mean_squared_error', optimizer='adam')
    model.fit(x_train, y_train, epochs=2, batch_size=1, verbose=2)
    print("Saving at "+getTrainingFilePath(stockCode))
    model.save(getTrainingFilePath(stockCode))

def predictClose(stockCode, new_data):
    ##############################################################
    ##
    ##  Prediction
    ##
    ##############################################################
    global scaler
    #global model
    model = load_model(getTrainingFilePath(stockCode))
    inputs = new_data[len(new_data) - 60:].values
    #inputs = inputs.reshape(-1,1)
    inputs  = scaler.fit_transform(inputs)

    X_test = []
    for i in range(0,inputs.shape[0]):
        tx=[]
        for j in range(0,4):
            tx.append(inputs[0:60,j])
        X_test.append(tx)
    X_test = np.array(X_test)

    #X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    print("Making a Prediction")
    closing_price = model.predict(X_test)
    closing_price = scaler.inverse_transform(closing_price)

    TR_closing_price=[]
```

```
        for i in range(closing_price.shape[0]):
                tr=[]
                for j in range(closing_price.shape[1]):
                        tr.append(closing_price[i,j])

                TR_closing_price.append(tr)

        #print("Actual : "+str(valid[-1:].item(1))+" | Predicted : "+str(TR_closing_price[-1][1]))
        #print("Actual : "+str(train[-1:].item(1))+" | Predicted : "+str(TR_closing_price[-1][1]))
        return TR_closing_price[-1][1]

def getEstimate(stockCode):
        print("Getting Estimates")

        new_data = getData(stockCode)
        x_train, y_train = scaleData(new_data)
        #ftrainning(stockCode,x_train, y_train)
        closing_price = predictClose(stockCode, new_data)
        K.clear_session()
        return closing_price

def prepareModel(stockCode):
        print("Preparing model")
        new_data = getData(stockCode)
        x_train, y_train = scaleData(new_data)
        trainning(stockCode,x_train, y_train)
        closing_price = predictClose(stockCode, new_data)
        K.clear_session()
        return closing_price

if __name__ == "__main__":
        print("Stock Price Predictor")
        stock = input("Enter Stock Code: ")
        prepareModel(stock)
        print(getEstimate(stock))
```

4.1.2 App.py

```
from flask import Flask,request,render_template,session, redirect
from flaskext.mysql import MySQL
import os
import LSTM_Predictor_InProgress
```

```
from flask import flash


mysql = MySQL()

app=Flask(__name__)
app.config['MYSQL_DATABASE_USER']='root'
app.config['MYSQL_DATABASE_PASSWORD']='root'
app.config['MYSQL_DATABASE_DB']='db_stockapp'
app.config['MYSQL_DATABASE_HOST']='localhost'
mysql.init_app(app)



@app.route('/')
def main():
        return render_template('layout.html')


@app.route('/home')
def home():
        if not session.get('logged_in'):
                return render_template('login.html')
        else:
                return render_template('home.html',msg="You're logged
in",user=session.get('inputEmail'))

@app.route('/admin')
def admin():
        if not session.get('logged_in'):
                return render_template('login.html')
        else:
                return render_template('admin.html',msg="You're logged
in",user=session.get('inputEmail'))

POST_EMAILID = ""
@app.route('/login', methods=['GET','POST'])
def login():
        global POST_EMAILID
        if request.method == 'POST':

                POST_EMAILID = str(request.form['inputEmail'])
                POST_PASSWORD = str(request.form['inputPassword'])
```

```
                    cursor = mysql.connect().cursor()
                    cursor.execute("SELECT * FROM UserDetails WHERE email = 
'"+POST_EMAILID+"' AND password = '"+POST_PASSWORD+"'")
                    result = cursor.fetchone()

                    if result:
                              print(POST_EMAILID)
                              session['logged_in'] = True
                              session['inputEmail'] = request.form['inputEmail'].split('@')[0]
                              if POST_EMAILID == "admin@gmail.com":
                                        return redirect('/admin')
                              else:
                                        return redirect('/home')
                    else:
                              flash('wrong password!')
          elif POST_EMAILID == "admin@gmail.com":
                    return admin()
          else:
                    return home()




@app.route('/register')
def register():
          return render_template('register.html')


@app.route('/register', methods=['GET','POST'])
def registerUser():
          try:
                    _name = request.form['inputName']
                    _email = request.form['inputEmail']
                    _password = request.form['inputPassword']
                    # validate the received values
                    if _email and _password and request.method == 'POST':
                              #do not save password as a plain text
                              # save edits
                              sql = "INSERT INTO userdetails(name, email, password) VALUES(%s, %s, 
%s)"
                              data = (_name, _email,_password,)
                              conn = mysql.connect()
                              cursor = conn.cursor()
                              cursor.execute(sql, data)
```

```
                        conn.commit()
                        # flash('User added successfully!')
                        return redirect('/login')
                else:
                        return 'Error while adding user'
        except Exception as e:
                print(e)
        finally:
                cursor.close()
                conn.close()




@app.route('/contact')
def contact():
        return render_template('contact.html')


@app.route('/about')
def about():
        return render_template('about.html')


@app.route("/logout")
def logout():
        session['logged_in'] = False
        return home()

@app.route('/Predict', methods=['POST'])
def Predict():
        pred=LSTM_Predictor_InProgress.getEstimate(str(request.form['stockcode']))
        pred = str(pred)
        session['prediction'] = pred
        return render_template('home.html',msg="Login
Successful!!",user=session.get('inputEmail'),pred=pred)
        # POST_EMAILID ="admin@gmail.com"
        # if POST_EMAILID == "admin@gmail.com":
        #       pred=LSTM_Predictor_InProgress.prepareModel(str(request.form['stockcode']))
        # else:
        #       pred=LSTM_Predictor_InProgress.getEstimate(str(request.form['stockcode']))
```

```
@app.route('/Training', methods=['POST'])
def Training():
        print("Admin Checkpoint")
        # return render_template('home.html',msg="Login
Successful!!",user=session.get('inputEmail'),pred=pred)
        pred=LSTM_Predictor_InProgress.prepareModel(str(request.form['stockcode']))
        pred = str(pred)
        session['prediction'] = pred
        print("Data Trained Success")
        return render_template('admin.html',pred="Training Successful")

# @app.route('/Train', methods=['POST'])
# def Train():
#       pred=LSTM_Predictor_InProgress.prepareModel(str(request.form['stockcode']))
#       pred = str(pred)
#       session['prediction'] = pred
#       return render_template('home.html',msg="Login
Successful!!",user=session.get('inputEmail'),pred=pred)


if __name__ == '__main__':
        app.secret_key = os.urandom(12)
        app.run(debug=True)
```

4.1.3 home.html

```
{% extends "layout.html" %}

{% block content %}

  <div class="jumbotron">
    {% if session['logged_in'] %}
    <p>{{ msg }} , {{ user }}</p>

                {% if session['logged_in'] %}
                <div>
                        <form class="form-signin" action="{{ url_for('Predict')}}" method="post">
                                <label for="stockcode">Enter Stock Code</label>
                                <input type="text" name="stockcode" id="stockcode"
placeholder="Stock Code" required autofocus>
                                <button id="btnPredict" type="submit">Predict</button>
                                <button id="btnPredict" type="submit">Predict</button>
```

```
                                        {% if session['prediction'] %}
                                                </br>
                                                </br>
                                                <p>Projected Close Price for Today is {{ pred }}</p>
                                        {% endif %}
                                </form>
                        </div>

                        {% endif %}

    {% else %}
    <p>This is home page html</p>
    {% endif %}

  </div>


{% endblock %}
```

4.1.4 layout.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Stock Predictor</title>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
rel="stylesheet">

    <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
    <link href="http://getbootstrap.com/assets/css/ie10-viewport-bug-workaround.css"
rel="stylesheet">

    <!-- Custom styles for this template -->
    <link href="http://getbootstrap.com/examples/jumbotron-narrow/jumbotron-narrow.css"
rel="stylesheet">

    <link rel="stylesheet" href="/static/css/style.css" type="text/css">

  </head>

  <body>

    <div class="container">
      <div class="header clearfix">
```

```
        <nav>
          <ul class="nav nav-pills pull-right">
            {% if session['logged_in'] %}
              <li role="presentation"><a>Welcome, {{ session['inputEmail'].capitalize() }}</a> </li>
              <li role="presentation"><a href="{{ url_for('home') }}">Home</a></li>
              <li role="presentation"><a href="{{ url_for('logout') }}">Logout</a></li>
            {% else %}
            <li role="presentation"><a href="{{ url_for('login') }}">Login</a></li>
                            <li role="presentation"><a href="{{ url_for('register') }}">Sign Up</a></li>
            {% endif %}
            <li role="presentation"><a href="{{ url_for('about') }}">About</a></li>
            <li role="presentation"><a href="{{ url_for('contact') }}">Contact</a></li>
          </ul>
        </nav>
        <h3 class="text-muted">Stock Predictor App</h3>
      </div>

      {% block content %}
      {% endblock %}

      <footer class="footer">
        <p>© 2018 Datta Meghe College Project</p>
      </footer>

    </div>

    <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
    <script src="http://getbootstrap.com/assets/js/ie10-viewport-bug-workaround.js"></script>
  </body>
</html>
```

4.1.5 login.html

```
{% extends "layout.html" %}

{% block content %}

{% if session['logged_in'] %}
<p>You are logged in </p>

{% else %}

<div class="jumbotron">
```

```
	<div class="login">
		<div class="login-form">
			<div class="app-title">
				<form class="form-signin" action="{{ url_for('login')}}"
method="post">
					<label for="inputemail" class="sr-only">Email
Address</label>
					<input type="email" name="inputEmail" id="inputEmail"
class="form-control" placeholder="Email Address" required autofocus>
					<label for="inputPassword" class="sr-only">Password</label>
					<input type="password" name="inputPassword"
id="inputPassword" class="form-control" placeholder="Password" required>
					<button id="btnSignIn" class="btn btn-lg btn-primary btn-
block" type="submit">Sign in</button>
				</form>
			</div>
		</div>
	</div>
</div>

{% endif %}

{% endblock %}
```

4.1.6 register.html

```
{% extends "layout.html" %}

{% block content %}

{% if session['logged_in'] %}
<p>You are logged in </p>

{% else %}

<div class="jumbotron">
	<div class="login">
		<div class="login-form">
			<div class="app-title">
				<form class="form-signin" action="{{ url_for('registerUser')}}"
method="post">
					<label for="inputname" class="sr-only">Name</label>
					<input type="text" name="inputName" id="inputEmail"
class="form-control" placeholder="Name" required>
```

```
                                                  <label for="inputemail" class="sr-only">Email
Address</label>
                                                  <input type="email" name="inputEmail" id="inputEmail"
class="form-control" placeholder="Email Address" required autofocus>
                                                  <label for="inputPassword" class="sr-only">Password</label>
                                                  <input type="password" name="inputPassword"
id="inputPassword" class="form-control" placeholder="Password" required>
                                                  <button id="btnSignIn" class="btn btn-lg btn-primary btn-
block" type="submit">Register</button>
                                        </form>
                              </div>
                    </div>
          </div>
</div>

{% endif %}

{% endblock %}
```

4.1.7 admin.html

```
{% extends "layout.html" %}

{% block content %}

  <div class="jumbotron">
    {% if session['logged_in'] %}
    <p>{{ msg }}  {{ user }}</p>

                    {% if session['logged_in'] %}
                    <div>
                              <form class="form-signin" action="{{ url_for('Training')}}" method="post">
                                        <label for="stockcode">Enter Stock Code</label>
                                        <input type="text" name="stockcode" id="stockcode"
placeholder="Stock Code" required autofocus>
                                        <button id="btnPredict" type="submit">Train Data ( Admin
)</button>
                                        {% if session['prediction'] %}
                                                  </br>
                                                  </br>
                                                  <p>Training Status : {{ pred }}</p>
                                        {% endif %}
                              </form>
                    </div>

                    {% endif %}

    {% else %}
   <p>This is home page html</p>
    {% endif %}

  </div>


{% endblock %}
```

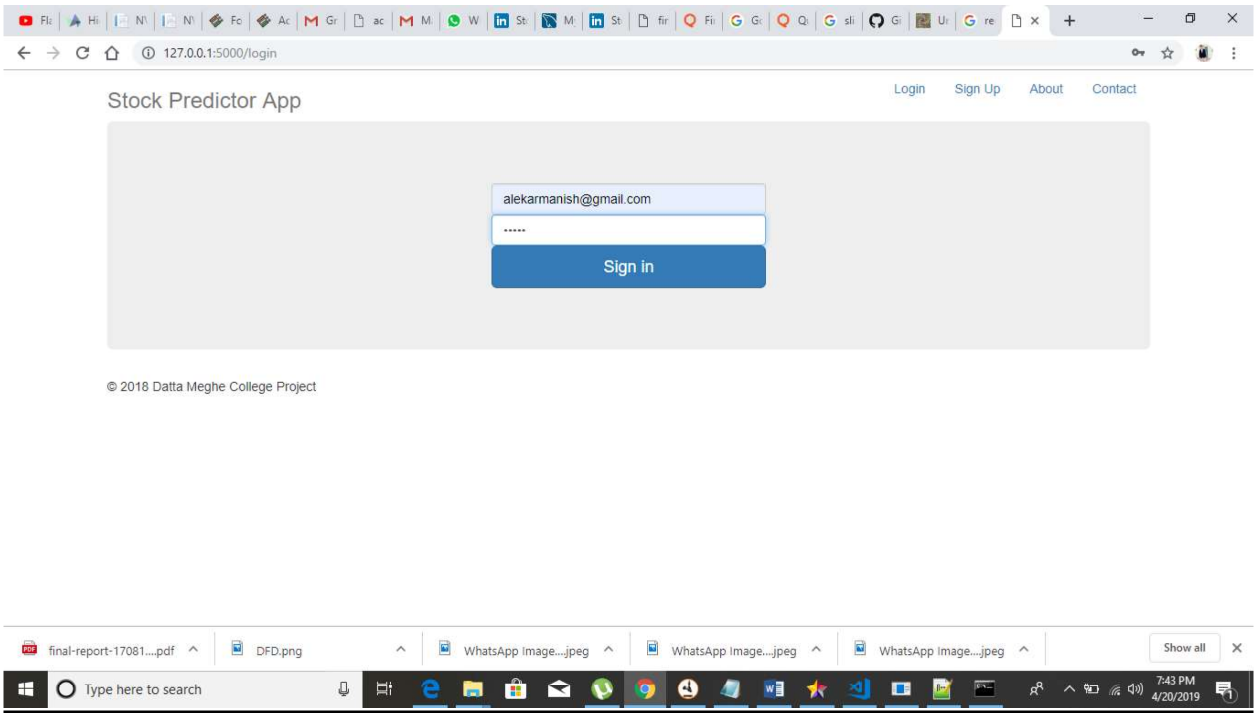
5. RESULTS AND DISCUSSION

5.1 DATA COLLECTION

QUANDL

This project attempts to predict the stock value with respect to the stock's previous value and trends. It requires historic data of stock market as the project also emphasizes on data mining techniques. So, it is necessary to have a trusted source having relevant and necessary data required for the prediction. We are using Quandl library to fetch National Stock Exchange data.

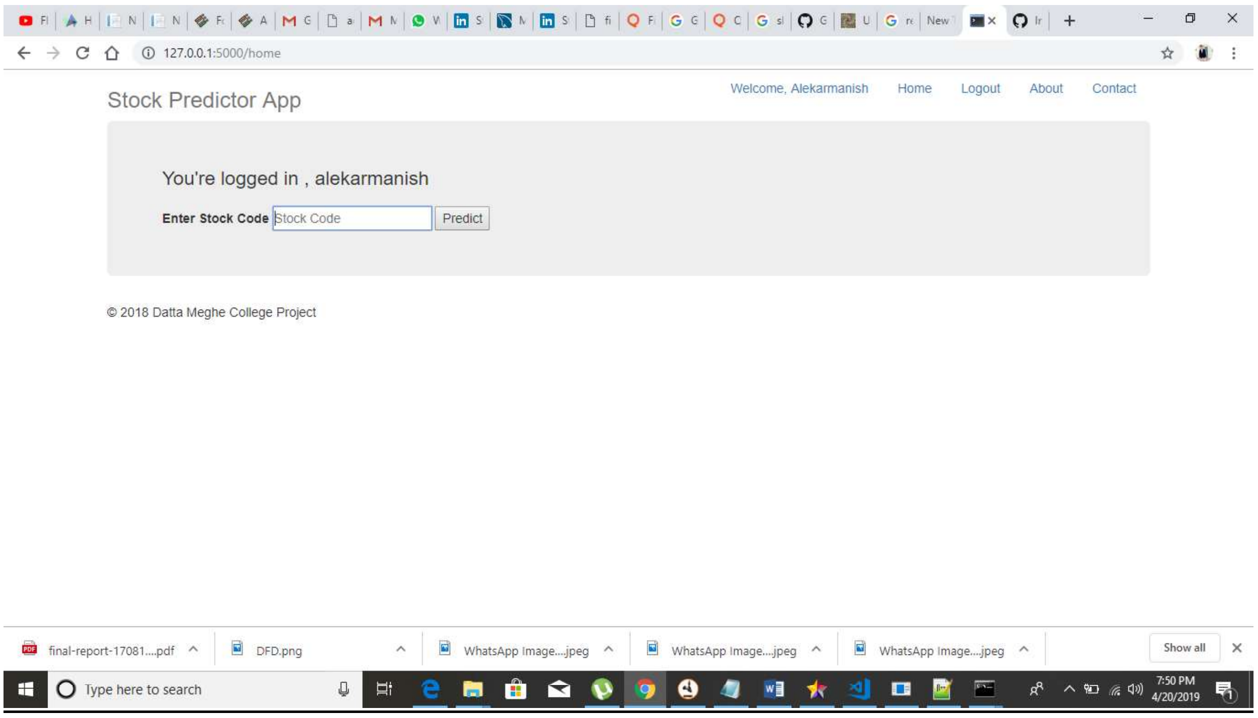
5.2 SCREENSHOTS



alekarmanish@gmail.com

© 2018 Datta Meghe College Project

Fig 6

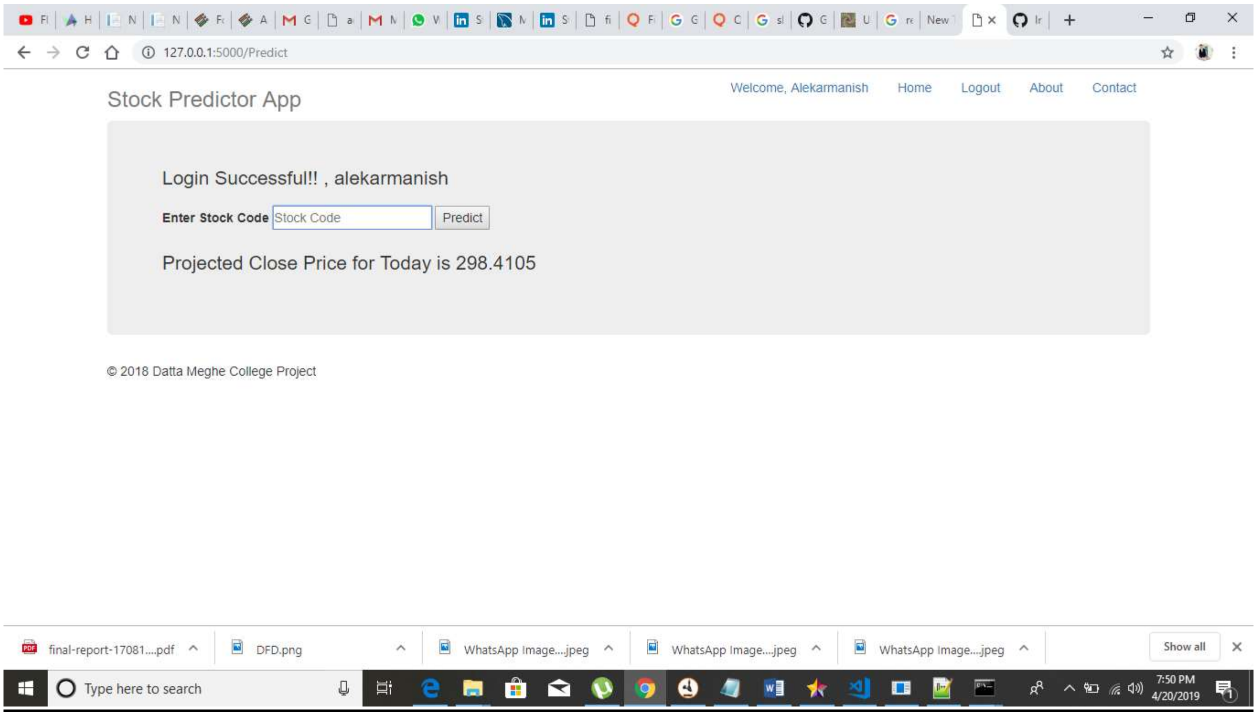

127.0.0.1:5000/home

Stock Predictor App

You're logged in , alekarmanish

© 2018 Datta Meghe College Project

Fig 7


127.0.0.1:5000/Predict

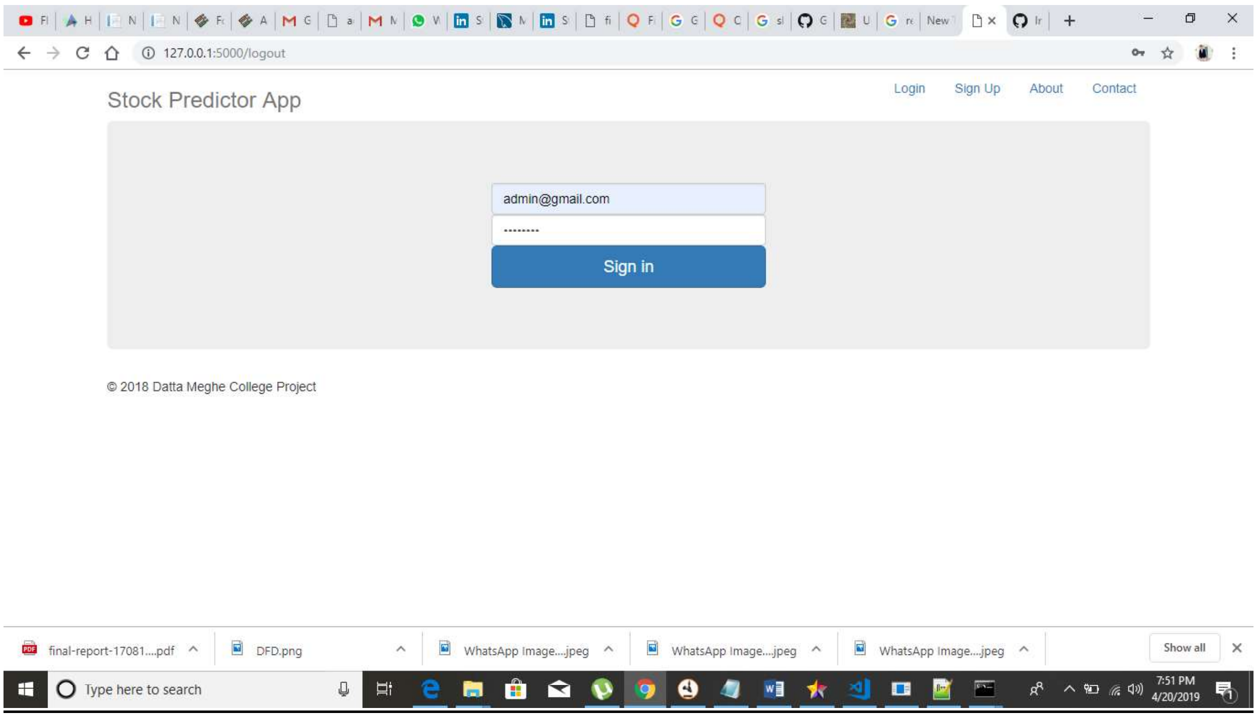
Stock Predictor App

Login Successful!! , alekarmanish

Projected Close Price for Today is 298.4105

© 2018 Datta Meghe College Project

Fig 8



127.0.0.1:5000/logout

Stock Predictor App

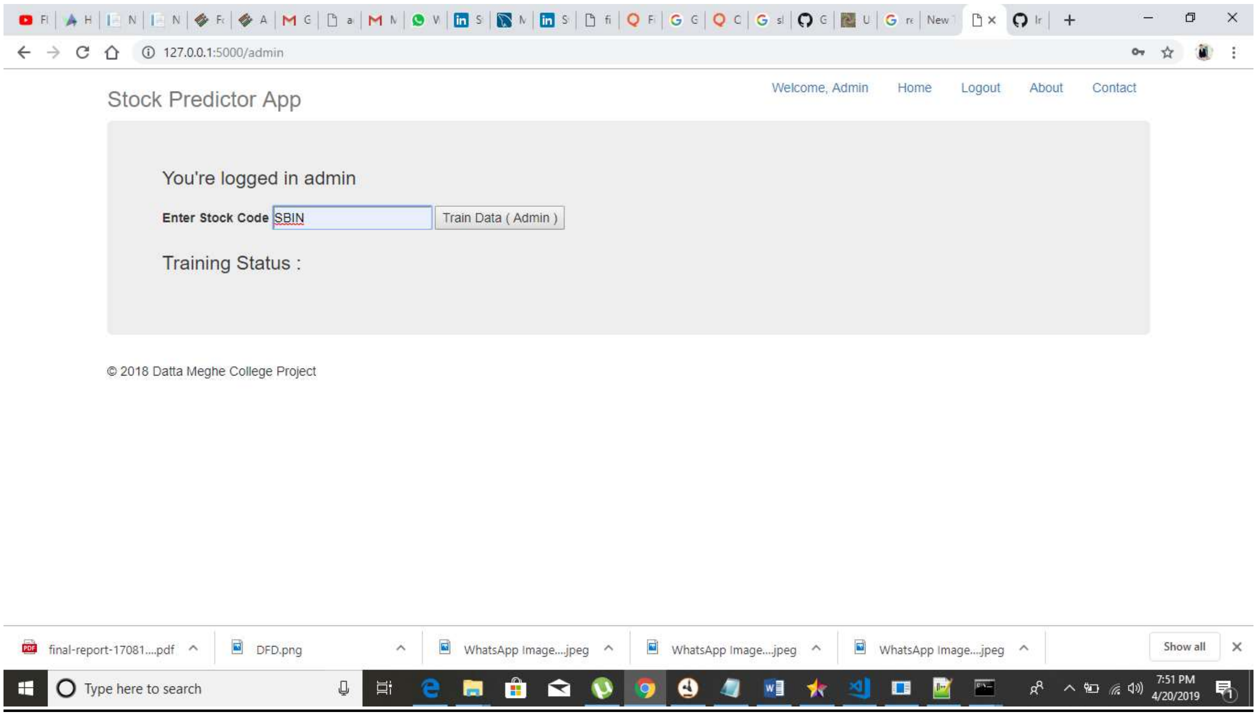
Login Sign Up About Contact

admin@gmail.com

Sign in

© 2018 Datta Meghe College Project

Fig 9



127.0.0.1:5000/admin

Stock Predictor App

Welcome, Admin Home Logout About Contact

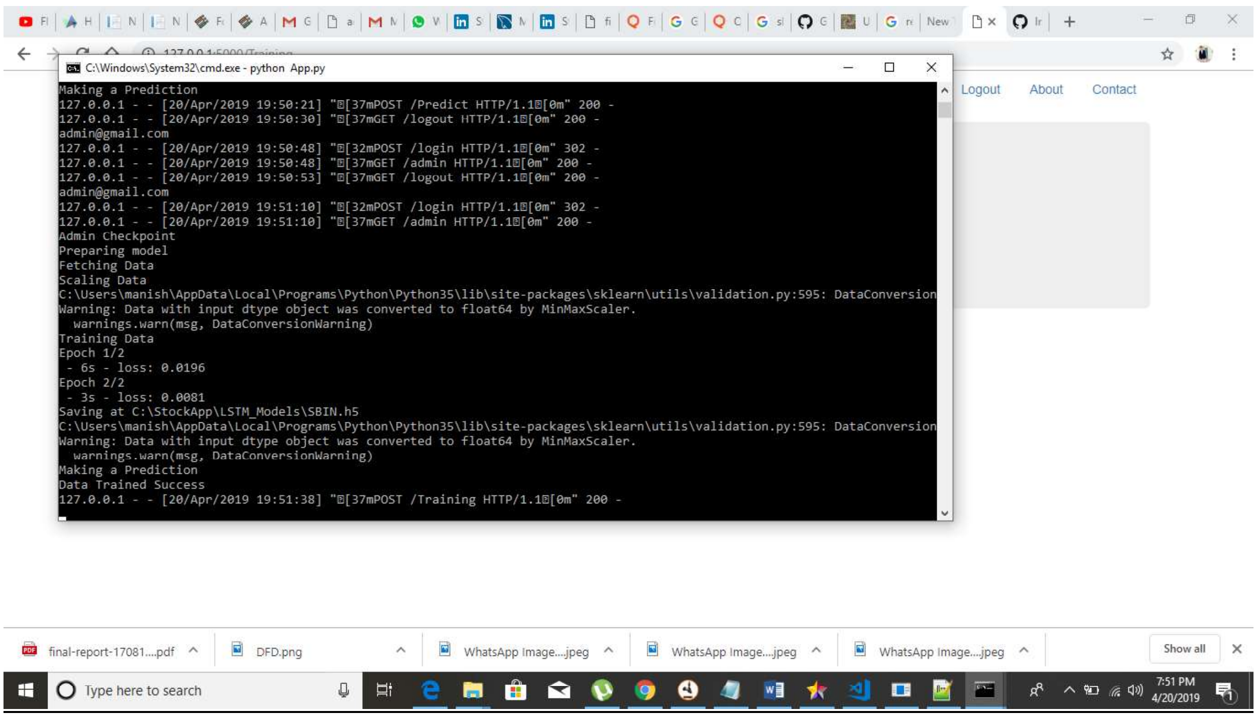
You're logged in admin

Enter Stock Code SBIN Train Data (Admin)

Training Status :

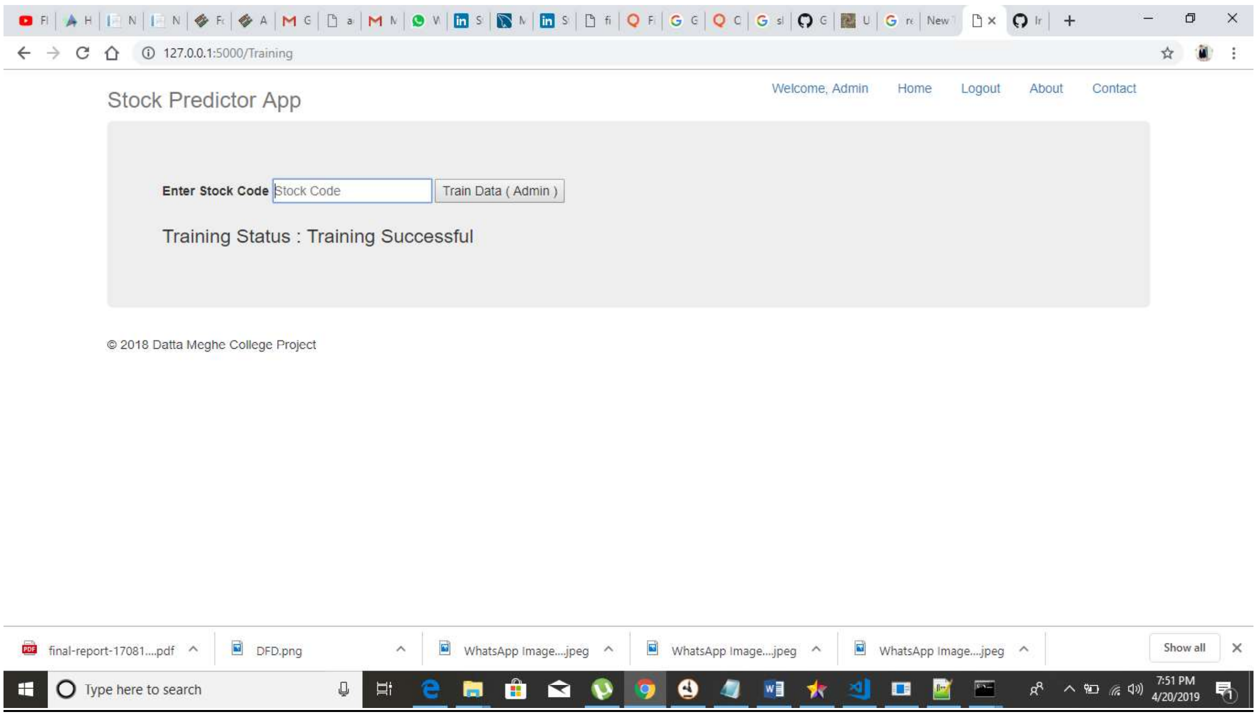
© 2018 Datta Meghe College Project

Fig 10



```
Making a Prediction
127.0.0.1 - - [20/Apr/2019 19:50:21] "□[37mPOST /Predict HTTP/1.1□[0m" 200 -
127.0.0.1 - - [20/Apr/2019 19:50:30] "□[37mGET /logout HTTP/1.1□[0m" 200 -
admin@gmail.com
127.0.0.1 - - [20/Apr/2019 19:50:48] "□[32mPOST /login HTTP/1.1□[0m" 302 -
127.0.0.1 - - [20/Apr/2019 19:50:48] "□[37mGET /admin HTTP/1.1□[0m" 200 -
127.0.0.1 - - [20/Apr/2019 19:50:53] "□[37mGET /logout HTTP/1.1□[0m" 200 -
admin@gmail.com
127.0.0.1 - - [20/Apr/2019 19:51:10] "□[32mPOST /login HTTP/1.1□[0m" 302 -
127.0.0.1 - - [20/Apr/2019 19:51:10] "□[37mGET /admin HTTP/1.1□[0m" 200 -
Admin Checkpoint
Preparing model
Fetching Data
Scaling Data
C:\Users\manish\AppData\Local\Programs\Python\Python35\lib\site-packages\sklearn\utils\validation.py:595: DataConversion
Warning: Data with input dtype object was converted to float64 by MinMaxScaler.
  warnings.warn(msg, DataConversionWarning)
Training Data
Epoch 1/2
 - 6s - loss: 0.0196
Epoch 2/2
 - 3s - loss: 0.0081
Saving at C:\StockApp\LSTM_Models\SBIN.h5
C:\Users\manish\AppData\Local\Programs\Python\Python35\lib\site-packages\sklearn\utils\validation.py:595: DataConversion
Warning: Data with input dtype object was converted to float64 by MinMaxScaler.
  warnings.warn(msg, DataConversionWarning)
Making a Prediction
Data Trained Success
127.0.0.1 - - [20/Apr/2019 19:51:38] "□[37mPOST /Training HTTP/1.1□[0m" 200 -
```

Fig 11



Stock Predictor App

Enter Stock Code Stock Code

Training Status : Training Successful

© 2018 Datta Meghe College Project

Fig 12

6. CONCLUSION

6.1 CONCLUSION

In this project, We have demonstrated a machine learning approach to predict stock market trend using different Pneural networks. Result shows how we can use history data to predict stock movement with reasonable accuracy. Also, we can conclude that LSTM performs better compare to other models.
For this implementation, We would like to conclude that if we incorporate all the factors that affect stock performance and feed them to neural network with proper data preprocessing and filtering, after training the network we will be able to have a model which can predict stock momentum very accurately and this can result into better stock forecasting and profit for financial firms. It helps the users in detecting the market trend pattern and other conditions. Insight on this data through visualization to predict future stock behavior and value at risk for each stock.
To predicting the stock market is it helps you to invest wisely to make good profits.

6.2 LIMITATIONS

The main aim of this system is to provide a general idea of where the stock market is headed. It is only limited to a very basic prediction model. Thus, it cannot be used as a critical decision making tool. By incorporating only limited number of parameters, there is certain degree of accuracy. Since, there are many indeterminate parameters that directly affect stock market, each and every one of them cannot be taken into account. So, our model only depends on the relationship of our selected parameters with the share price.
This system is limited to only certain users that have knowledge of stock market.

6.3 FEASIBILITY STUDY

- The stock market is a place where buyers and sellers converge. When there are more buyers than sellers, the price increases.

When there are more sellers than buyers, the price decrease.

- It has more to do with emotion than logic. Because emotion is unpredictable, stock market movements unpredictable. It's futile to try to predict where markets are going. They are designed to be unpredictable.
- There are some fundamental financial indicators by which a company's Stock value can be estimated. Some of the indicators and factors are: Price-to-Earning(P/E) Ratio, Price-to-Earning Growth(PEG) Ratio, Price-to-sales(P/S) Ratio, price-to-cash Flow(P/CF) Ratio, Price-to-Book Value(P/BV) Ratio and Debt-to-Equity Ratio.

Requirements:

FUNCTIONAL REQUIREMENTS:

1. **User Interface**: The user is required to select which company is he interested in

   amongst various companies that have been provided.

2. **Admin Login**: Admin can manually manage the external factors that can affect the stock price

   Ex.New Company Policy that is been criticized, Drop In Company's profit,

   Unexpected change in senior leadership of the company

3. **User Login**

4. **Libraries**: ***Pandas*** is a software library written for the Python programming. It offers

data structures, operations for manipulating numerical tables and time-series.

Matplotlib is a plotting library for python &amp; its numerical mathematics extension

NumPy. ***SciPy*** makes use of matplotlib. Pandas data reader (up to date remote data

access for pandas, works for multiple versions of pandas***). Datetime*** module supplies

classes for manipulating dates and times in both simple and complex way.

NON FUNCTIONAL REQUIREMENTS:

1. **Reliability**: The reliability of the product will be dependent on the accuracy of the data- date of purchase, how much stock was purchased, high and low value range as well as opening and closing figures. Also the stock data used in the training would determine the reliability of the software.

2**. Security**: The user will only be able to access the website using his login details and will not be able to access the computations happening at the back end.

3**. Maintainability**: The maintenance of the product would require training of the software by recent data so that there commendations are up to date. The database has to be updated with recent values.

4. **Portability**: The website is completely portable and the recommendations completely.

trustworthy as the data are dynamically updated.

7. REFERENCES

https://www.quantinsti.com/blog/working-neural-networks-stock-price-prediction/

https://en.m.wikipedia.org/wiki/Time_series

https://enlight.nyc/projects/stock-market-prediction/

https://www.datacamp.com/community/tutorials/lstm-python-stock-market

https://towardsdatascience.com/stock-prediction-in-python-b66555171a2