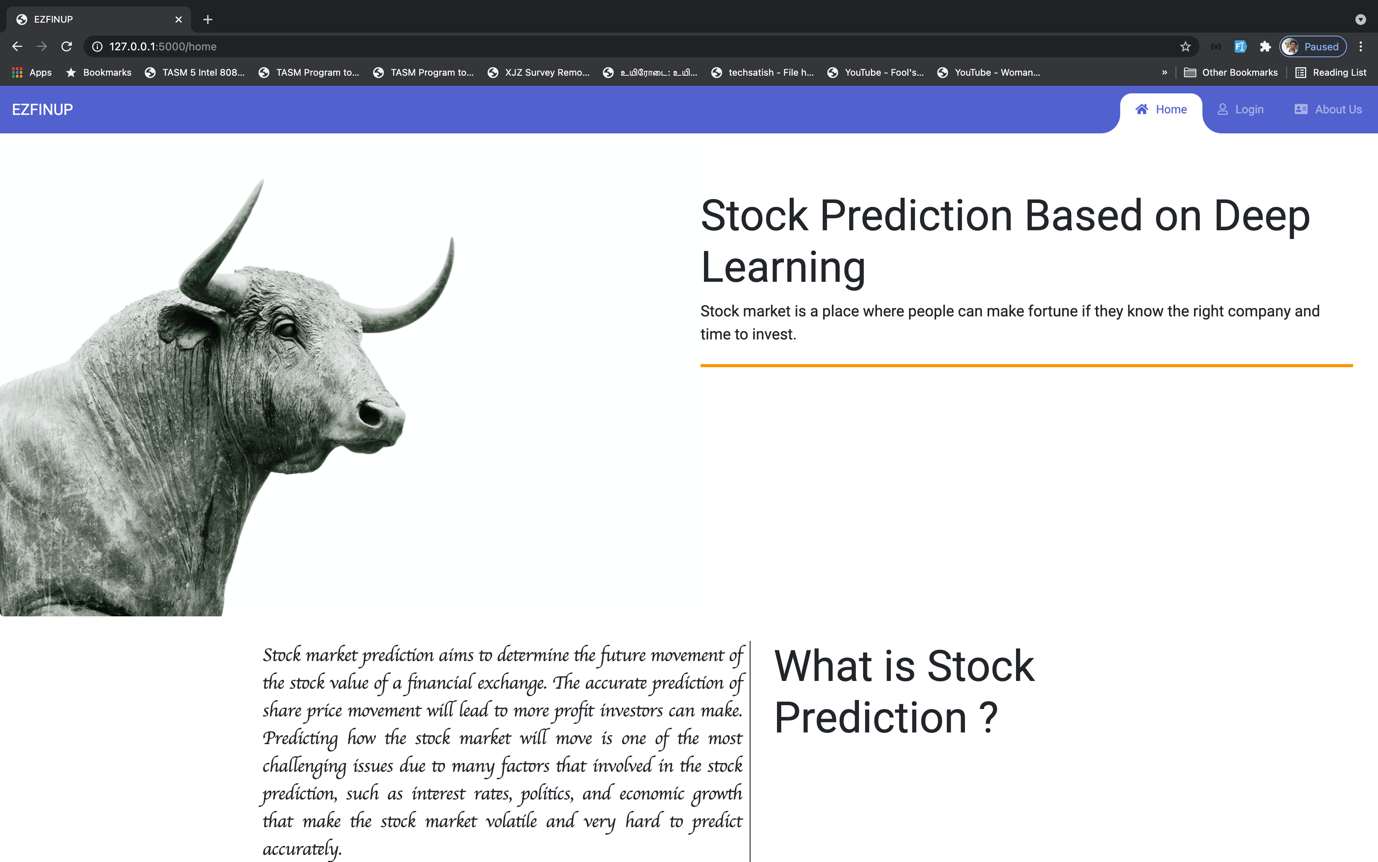
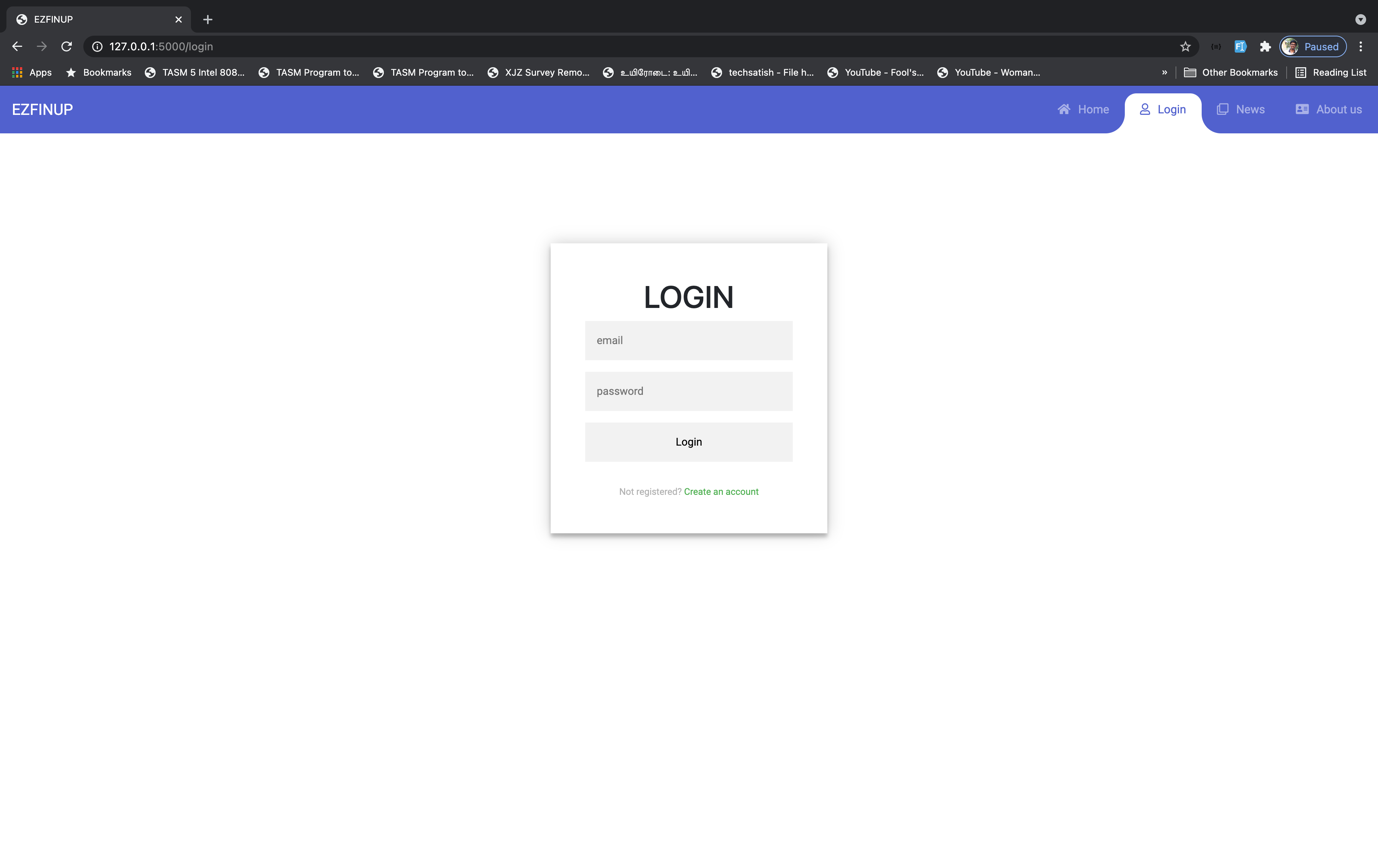
**REFERENCES**

1. Penglei Gao ,Rui Zhang and Xi Yang, “The Application of Stock Index Price Prediction with Neural Network”, Multidisciplinary Digital Publishing Institute, 18 August 2020
2. Padmaja Dhenuvakonda, R. Anandan, N. Kumar, “Stock Price Prediction Using Artificial Neural Networks”, Journal of Critical Reviews, 10 June 2020
3. Raghav Nandakumar, Uttamraj K R, Vishal R, Y V Lokeswari, “Stock Price Prediction Using Long Short Term Memory”, International Research Journal of Engineering and Technology (IRJET), vol. 05 Issue: 03, March 2018
4. R. Vanaga and B. Sloka, “Financial and capital market commission financing: aspects and challenges,” Journal of Logistics, Informatics and Service Science, vol. 7, no. 1, pp. 17–30, 2020.
5. Wenjie Lu, Jiazheng Li, Yifan Li, Aijun Sun, Jingyang Wang, "A CNN-LSTM-Based Model to Forecast Stock Prices", Complexity, vol. 2020, Article ID 6622927, 10 pages, 2020.
6. Hiransha M, Gopalakrishnan E.A, Vijay Krishna Menon, Soman K.P, “NSE Stock Market Prediction Using Deep-Learning Models”, International Conference on Computational Intelligence and Data Science, 2018

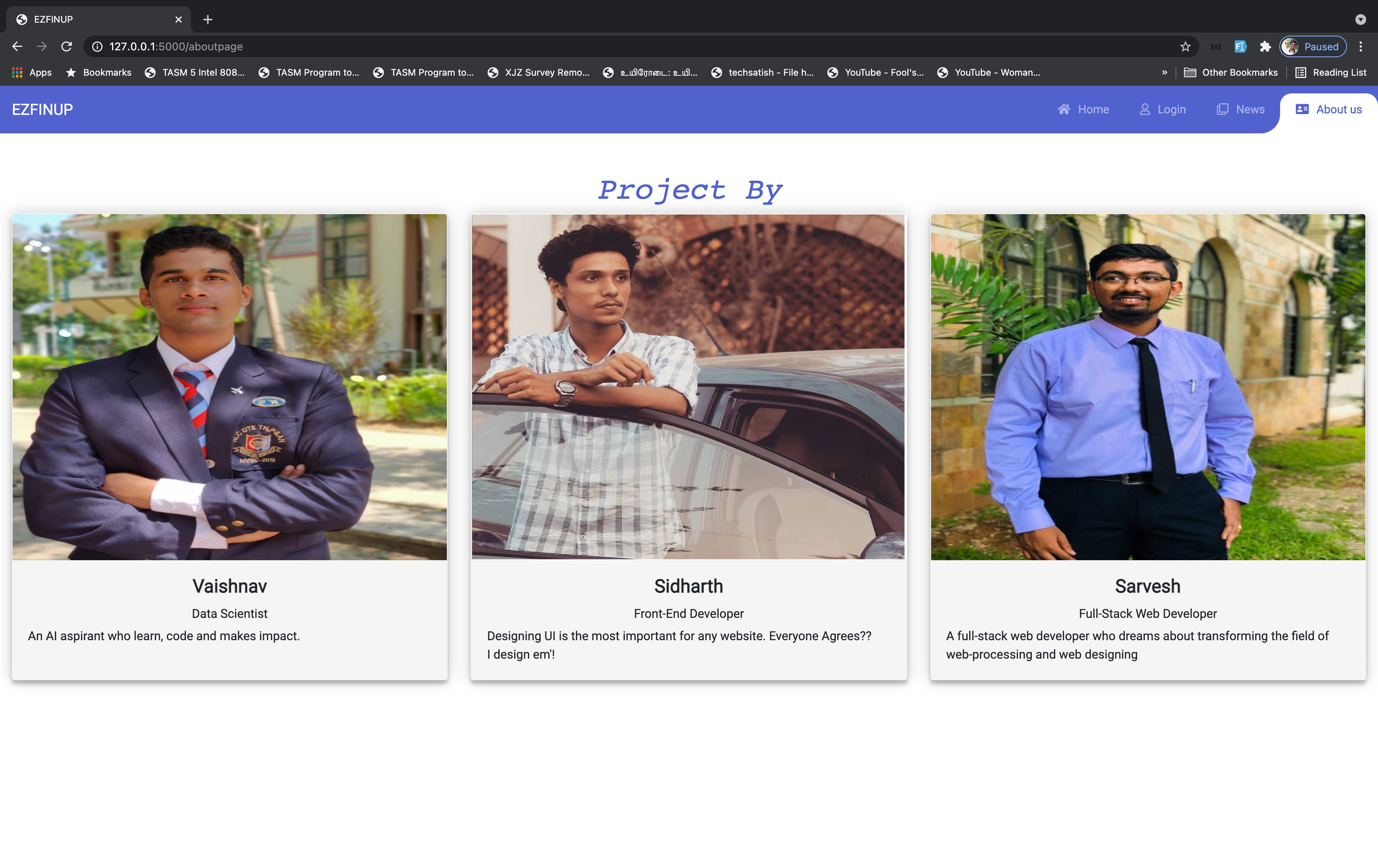
**APPENDIX A: SAMPLE SCREENS**

****

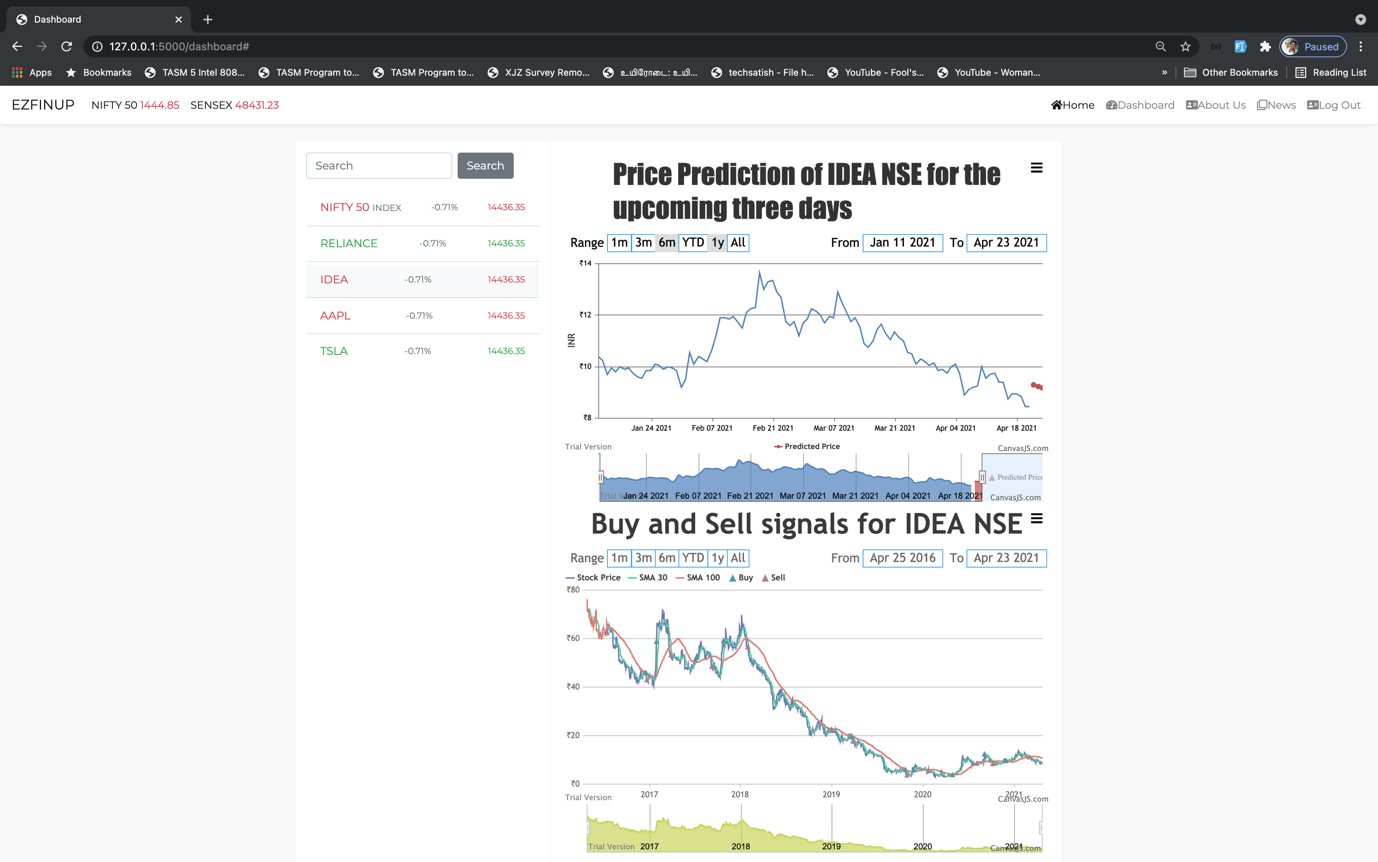
Homepage

****

Login Page

****

About Page



Dashboard

**APPENDIX B: SAMPLE CODE**

**app1.py – Website Backend Code**

from flask import Flask, render\_template,request, redirect, session, jsonify

import os, csv

from newsapi import NewsApiClient

import talib

import yfinance as yf

from patterns import candlestick\_patterns

import firebase\_admin

from firebase\_admin import credentials, auth

import pyrebase

import json

import pandas as pd

# Firebase Admin Init

cred = credentials.Certificate('fbAdminConfig.json')

firebase\_admin = firebase\_admin.initialize\_app(cred)

# Firebase Init

firebase = pyrebase.initialize\_app(json.load(open('fbConfig.json')))

# Init

newsapi = NewsApiClient(api\_key='0144a0f2461949b7b8896463a90399f3')

key="483bd89d87mshce534f1bd4ba6b0p182fcdjsnf08d8fc19c04"

# App Begins

app = Flask(\_\_name\_\_)

app.secret\_key = 'the random string'

# @app.before\_request

# def checkAuth():

# auth\_not\_required = ['home', 'homepage', 'login', 'signup', 'aboutpage']

# print("Request Endpoint ")

# print(request.path.strip('/'))

# print("Session User")

# print(session['user\_id'])

# if (request.path.strip('/') not in auth\_not\_required) and (session['user\_id'] == None):

# return render\_template('404.html')

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/news')

def news():

newsapi = NewsApiClient(api\_key="0144a0f2461949b7b8896463a90399f3")

topheadlines = newsapi.get\_top\_headlines(sources="bbc-news")

articles = topheadlines['articles']

desc = []

news = []

img = []

for i in range(len(articles)):

myarticles = articles[i]

news.append(myarticles['title'])

desc.append(myarticles['description'])

img.append(myarticles['urlToImage'])

mylist = zip(news, desc, img)

return render\_template('bbc.html', context=mylist)

@app.route('/dashboard')

def dashboard():

return render\_template('madhu\_dashboard.html')

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == "POST":

email = request.form.get('email')

password = request.form.get('password')

# try:

user = firebase.auth().sign\_in\_with\_email\_and\_password(email, password)

jwt = user['idToken']

session['user\_id'] = jwt

return redirect('/dashboard')

# except:

# return {'message': 'There was an error logging in'},400

else:

return render\_template('login.html')

@app.route('/logout')

def logout():

# firebase.auth().signOut()

session['user\_id'] = None

return redirect('/home')

@app.route('/signup', methods=['GET', 'POST'])

def signup():

if request.method == "POST":

name = request.form.get("name")

email = request.form.get("email")

password = request.form.get("password")

confirm\_password = request.form.get("cnf-password")

if password == confirm\_password:

# try:

user = auth.create\_user(

email=email,

password=password

)

# session['user\_id'] = user.idToken

# return redirect('/dashboard')

return jsonify(user)

# except:

# return {'message': 'Error creating user'},400

else:

return {'message': 'Error creating user. Password and Confirm Password don\'t match'},400

else:

return render\_template('create-acc.html')

# Candlestick routes

@app.route('/snapshot')

def snapshot():

with open('datasets/companies.csv') as f:

for line in f:

if "," not in line:

continue

symbol = line.split(",")[0] + '.NS'

print("Symbol")

print(symbol)

data = yf.download(symbol, start="2020-01-01", end="2021-03-01")

data.to\_csv('datasets/daily/{}.csv'.format(symbol))

return {

"code": "success"

}

@app.route('/screener')

def screener():

pattern = request.args.get('pattern', False)

stocks = {}

with open('datasets/companies.csv') as f:

for row in csv.reader(f):

stocks[row[0]] = {'company': row[1]}

if pattern:

for filename in os.listdir('datasets/daily'):

df = pd.read\_csv('datasets/daily/{}'.format(filename))

pattern\_function = getattr(talib, pattern)

symbol = filename.split('.')[0]

try:

results = pattern\_function(df['Open'], df['High'], df['Low'], df['Close'])

last = results.tail(1).values[0]

if last > 0:

stocks[symbol][pattern] = 'bullish'

elif last < 0:

stocks[symbol][pattern] = 'bearish'

else:

stocks[symbol][pattern] = None

except Exception as e:

print('failed on filename: ', filename)

return render\_template('screener.html', candlestick\_patterns=candlestick\_patterns, stocks=stocks, pattern=pattern)

return render\_template('create-acc.html')

# End of candlestick routes

@app.route('/aboutpage')

def aboutpage():

return render\_template('aboutpage.html')

@app.route('/home')

def homepage():

return render\_template('homepage.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**app.py – Stock Price Prediction Code**

from flask import Flask, render\_template, request

# from flask\_cors import CORS

import pandas\_datareader as pdr

from sklearn.preprocessing import MinMaxScaler

from newsapi import NewsApiClient

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

import numpy as np

df = pdr.get\_data\_yahoo(request.args.get('stockname'))

df1 = df.reset\_index()['Close']

scaler = MinMaxScaler(feature\_range=(0, 1))

df1 = scaler.fit\_transform(np.array(df1).reshape(-1, 1))

training\_size = int(len(df1) \* 0.65)

test\_size = len(df1) - training\_size

train\_data, test\_data = df1[0:training\_size, :], df1[training\_size:len(df1), :1]

# import numpy

# convert an array of values into a dataset matrix

def create\_dataset(dataset, time\_step=1):

dataX, dataY = [], []

for i in range(len(dataset) - time\_step - 1):

a = dataset[i:(i + time\_step), 0] ###i=0, 0,1,2,3-----99 100

dataX.append(a)

dataY.append(dataset[i + time\_step, 0])

return np.array(dataX), np.array(dataY)

time\_step = 100

X\_train, y\_train = create\_dataset(train\_data, time\_step)

X\_test, ytest = create\_dataset(test\_data, time\_step)

# reshape input to be [samples, time steps, features] which is required for LSTM

X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1], 1)

X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], 1)

### Create the Stacked LSTM model

import tensorflow as tf

import numpy as np

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM

model = tf.keras.Sequential([keras.layers.Dense(units=1, input\_shape=[1])])

model = Sequential()

if (request.args.get('stockname') =='IDEA.NS') or (request.args.get('stockname') =='AAPL') or (request.args.get('stockname') =='TSLA'):

model.add(LSTM(50, return\_sequences=True, input\_shape=(100, 1)))

model.add(LSTM(50, return\_sequences=True))

else:

model.add(LSTM(50, return\_sequences=True, input\_shape=(100, 1)))

model.add(LSTM(50, return\_sequences=True))

model.add(LSTM(50, return\_sequences=True))

model.add(LSTM(50, return\_sequences=True))

model.add(LSTM(50))

model.add(Dense(1))

if (request.args.get('stockname') =='IDEA.NS') or (request.args.get('stockname') =='AAPL') or (request.args.get('stockname') =='TSLA'):

model.compile(loss='mean\_squared\_error', optimizer='adam')

else:

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['accuracy'])

if (request.args.get('stockname') =='IDEA.NS') or (request.args.get('stockname') =='AAPL') or (request.args.get('stockname') =='TSLA'):

model.fit(X\_train, y\_train, validation\_data=(X\_test, ytest), epochs=10, batch\_size=64, verbose=1)

else:

model.fit(X\_train, y\_train, validation\_data=(X\_test, ytest), epochs=10, batch\_size=10, verbose=2) # Actual Epoch 50. Changing to 10 for current demo

train\_predict = model.predict(X\_train)

test\_predict = model.predict(X\_test)

##Transformback to original form --- rescaling

train\_predict = scaler.inverse\_transform(train\_predict)

test\_predict = scaler.inverse\_transform(test\_predict)

### Plotting

# shift train predictions for plotting

look\_back = 100

trainPredictPlot = np.empty\_like(df1)

trainPredictPlot[:, :] = np.nan

trainPredictPlot[look\_back:len(train\_predict) + look\_back, :] = train\_predict

# shift test predictions for plotting

testPredictPlot = np.empty\_like(df1)

testPredictPlot[:, :] = np.nan

testPredictPlot[len(train\_predict) + (look\_back \* 2) + 1:len(df1) - 1, :] = test\_predict

if request.args.get('stockname') =='AAPL' or request.args.get('stockname') == 'TSLA':

x\_input = test\_data[341:].reshape(1,-1)

# elif request.args.get('stockname') == 'TSLA':

# x\_input = test\_data[341:].reshape(1,-1)

else:

x\_input = test\_data[331:].reshape(1, -1)

temp\_input = list(x\_input)

temp\_input = temp\_input[0].tolist()

# demonstrate prediction for next 10 days

from numpy import array

lst\_output = []

n\_steps = 100

i = 0

if (request.args.get('stockname') =='IDEA.NS') or (request.args.get('stockname') =='AAPL') or (request.args.get('stockname') =='TSLA'):

while (i < 3):

if (len(temp\_input) > 100):

x\_input = np.array(temp\_input[1:])

x\_input = x\_input.reshape(1, -1)

x\_input = x\_input.reshape((1, n\_steps, 1))

yhat = model.predict(x\_input, verbose=0)

temp\_input.extend(yhat[0].tolist())

temp\_input = temp\_input[1:]

lst\_output.extend(yhat.tolist())

i = i + 1

else:

x\_input = x\_input.reshape((1, n\_steps, 1))

yhat = model.predict(x\_input, verbose=0)

temp\_input.extend(yhat[0].tolist())

lst\_output.extend(yhat.tolist())

i = i + 1

else:

while (i < 10):

if (len(temp\_input) > 100):

# print(temp\_input)

x\_input = np.array(temp\_input[1:])

x\_input = x\_input.reshape(1, -1)

x\_input = x\_input.reshape((1, n\_steps, 1))

# print(x\_input)

yhat = model.predict(x\_input, verbose=0)

temp\_input.extend(yhat[0].tolist())

temp\_input = temp\_input[1:]

# print(temp\_input)

lst\_output.extend(yhat.tolist())

i = i + 1

else:

x\_input = x\_input.reshape((1, n\_steps, 1))

yhat = model.predict(x\_input, verbose=0)

temp\_input.extend(yhat[0].tolist())

lst\_output.extend(yhat.tolist())

i = i + 1

if (request.args.get('stockname') =='IDEA.NS') or (request.args.get('stockname') =='AAPL') or (request.args.get('stockname') =='TSLA'):

day\_new = np.arange(1, 101) # testdata 100indexes

day\_pred = np.arange(101, 104) # 101-131-predicted

# FOR RELIANCE

else:

day\_new = np.arange(1, 101) # testdata 100indexes

day\_pred = np.arange(101, 111) # 101-131-predicted

if (request.args.get('stockname') =='AAPL') or (request.args.get('stockname') =='TSLA'):

numpyData3 = scaler.inverse\_transform(df1[1158:])

# FOR TSLA

# elif (request.args.get('stockname') =='TSLA'):

# numpyData3 = scaler.inverse\_transform(df1[1158:])

# FOR RELIANCE

else:

numpyData3 = scaler.inverse\_transform(df1[1131:])

numpyData4 = scaler.inverse\_transform(lst\_output)

daynew = {"x": day\_new.tolist(), "y": day\_pred.tolist()}

daypred = {"x": numpyData3.tolist(), "y": numpyData4.tolist()}

data = []

data.append(daynew)

data.append(daypred)

final\_data = {"data": data}

print(final\_data)

return final\_data

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5001, debug=False)

**app.py – Algo-trading Code**

from flask import Flask, request

import pandas as pd

import numpy as np

import pandas\_datareader as pdr

import math

from datetime import datetime

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

stock = pdr.get\_data\_yahoo(request.args.get('stockname'))

# print("DF")

# print(stock)

stock.reset\_index(inplace=True, drop=False)

# stock = df.reset\_index()['Close']

# print(stock)

Sma30 = pd.DataFrame()

Sma30['Prev Close Price'] = stock['Close'].rolling(window=10).mean()

# Creating Simple Moving Average with 100-day Window

Sma100 = pd.DataFrame()

Sma100['Prev Close Price'] = stock['Close'].rolling(window=60).mean()

data = pd.DataFrame()

data['stock'] = stock['Close']

data['Sma30'] = Sma30['Prev Close Price']

data['Sma100'] = Sma100['Prev Close Price']

# print(Sma30['Prev Close Price'].fillna("NONE"))

def buy\_sell(data):

sigPriceBuy = []

sigPriceSell = []

flag = -1

for i in range(len(data)):

if data['Sma30'][i] > data['Sma100'][i]:

if flag != 1:

sigPriceBuy.append(data['stock'][i])

sigPriceSell.append(np.nan)

flag = 1

else:

sigPriceBuy.append(np.nan)

sigPriceSell.append(np.nan)

elif data['Sma30'][i] < data['Sma100'][i]:

if flag != 0:

sigPriceBuy.append(np.nan)

sigPriceSell.append(data['stock'][i])

flag = 0

else:

sigPriceBuy.append(np.nan)

sigPriceSell.append(np.nan)

else:

sigPriceBuy.append(np.nan)

sigPriceSell.append(np.nan)

return (sigPriceBuy, sigPriceSell)

buy\_sell = buy\_sell(data)

data['Buy\_Signal\_Price'] = buy\_sell[0]

data['Sell\_Signal\_Price'] = buy\_sell[1]

# print("Stock")

# print(stock['Date'])

# End Result in JSON

result = {"dates": stock['Date'].fillna("NONE").dt.date.values.tolist(), "price": stock['Close'].values.tolist(), "SMA30": Sma30['Prev Close Price'].fillna("NONE").values.tolist(), "SMA100": Sma100['Prev Close Price'].fillna("NONE").values.tolist(), "buy\_point": data['Buy\_Signal\_Price'].fillna("NONE").values.tolist(), "sell\_point": data['Sell\_Signal\_Price'].fillna("NONE").values.tolist()}

# print("Result\n")

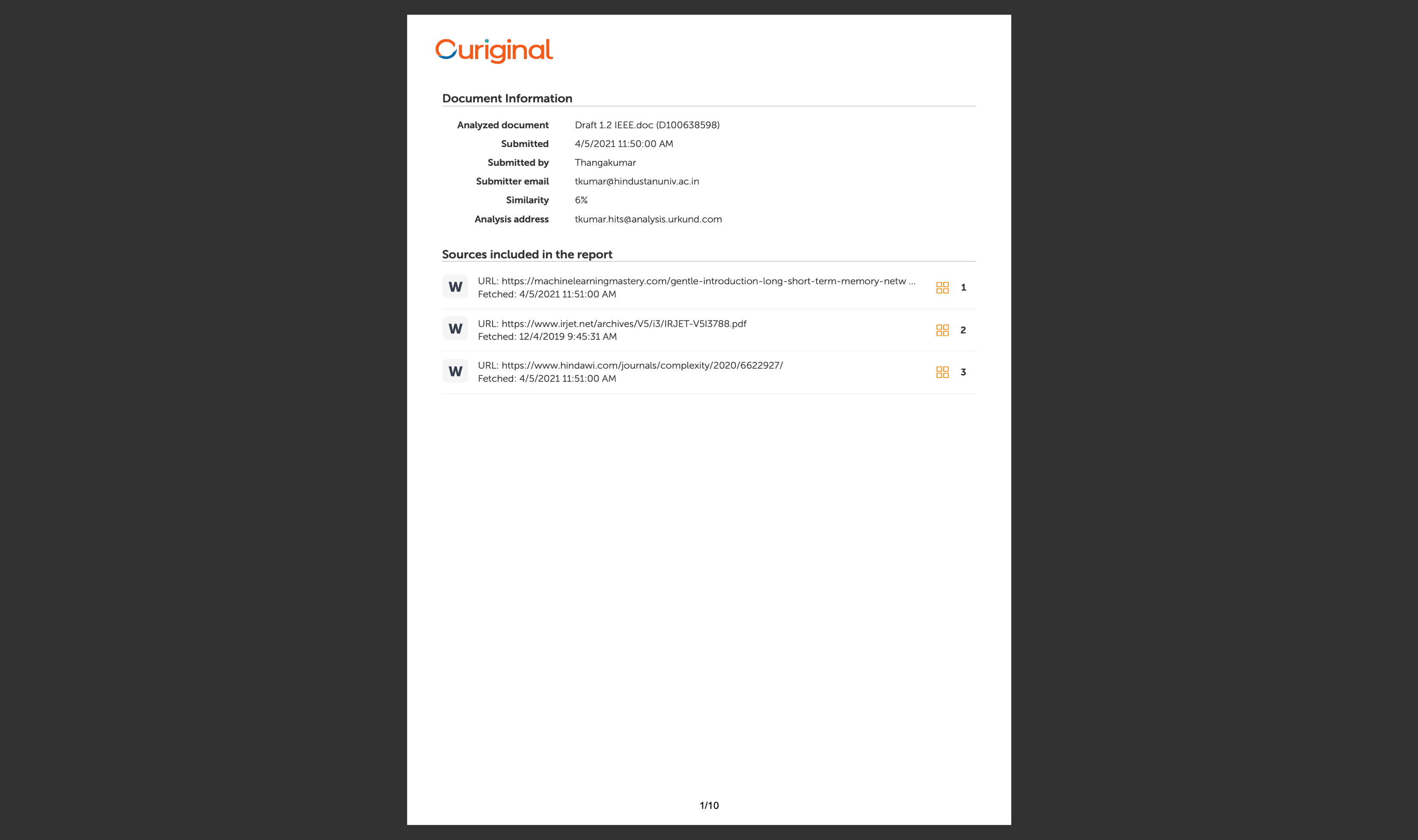
# print(result)

return result

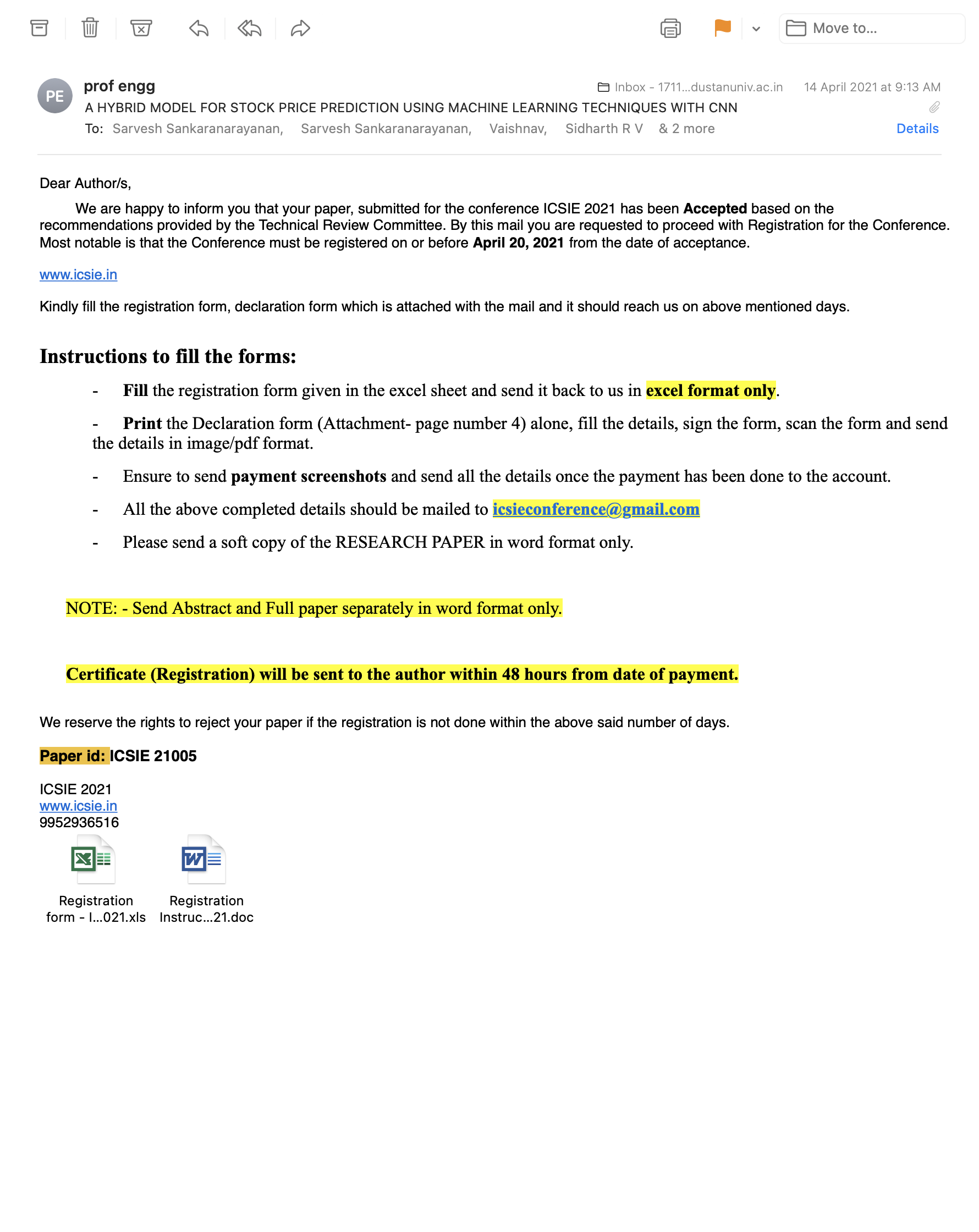
if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='127.0.0.1',port=5002 , debug=True)

**APPENDIX C: PLAGIARISM REPORT**

****

**APPENDIX D: PUBLICATION DETAILS**

****

**APPENDIX E: TEAM DETAILS**

**TEAM NUMBER:** T\_12

1. **NAME:** SARVESH S.

**CLASS:** CSE – 8A

**ROLL NUMBER:** 17113061

1. **NAME:** V. VAISHNAV

**CLASS:** CSE – 8A

**ROLL NUMBER:** 17113061

1. **NAME:** SIDHARTH R.V.

**CLASS:** CSE-8A

**ROLL NUMBER:** 17113035

1. **MENTOR NAME:** DR. THANGAKUMAR J

**DESIGNATION:** ASSOCIATE PROFESSOR