

Progress Report of Mini Project

Subject Code: 3IT31

Academic Year 2020-21

Name of Students : Smit Ghelani, Parth Panchani, Jigar Shekhat
ID No. : 18IT401, 18IT410, 18IT427
Batch : A
Topic : Stock Price Prediction
Guide Name : Prof. Kanu Patel

BACHELOR OF ENGINEERING

In

INFORMATION TECHNOLOGY



**Birla Vishwakarma Mahavidhyalaya Engineering College, Vallabh
Vidhyanagar-388120**

Index

	Topics	Page No.
1.	Definition	3
2.	How it works?	3
3.	Project modules implementation work	4
4.	Test Cases of Implemented modules	8

1. Definition:

Stock price prediction using machine learning algorithm based on past data of stocks.

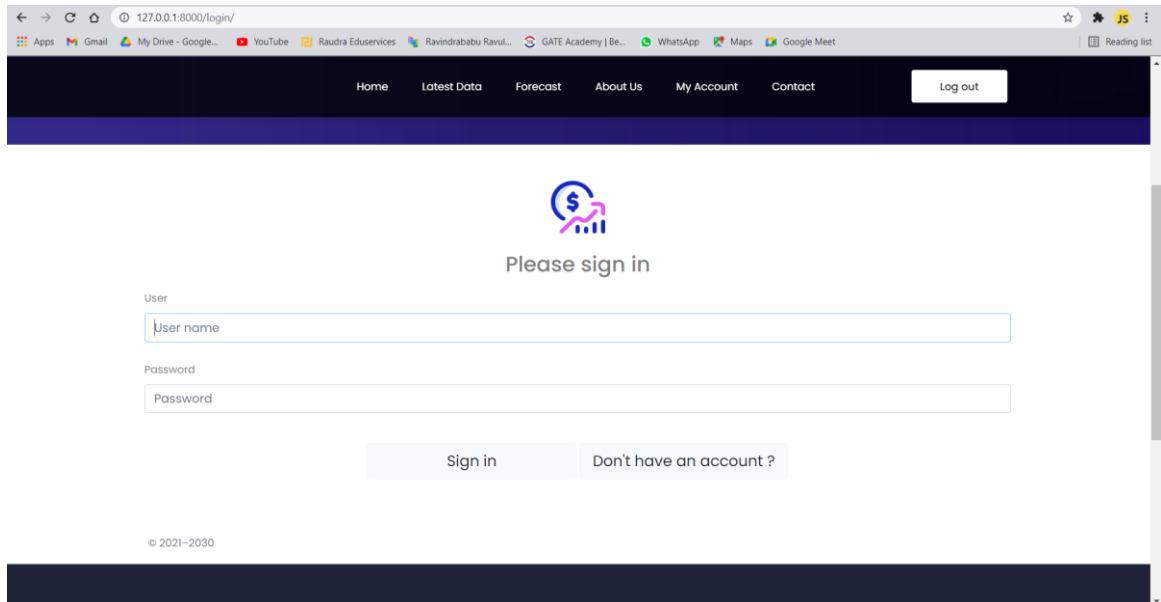
2. How it works?:

- We collect past data of stocks which are currently registered inside NSE using `get_history()` function of `nsepy` library when user tries to find prediction of particular stock.
- Collected data are in form of pandas data frame.
- Then we separate the Close and Volume columns from data-frame.
- Then we create Prediction column inside data-frame and allocate data to that frame by shifting 120 day closed prices and create 120 day empty slot for our predicted data.
- After this we normalize our data to scale it down less than 1.
- Then we split our dataset into train and test set as 80/20.
- We pass our training set to algorithm to train our model data and we are using gradient descent method to set slope and intercept for linear regression.
- Then we use closed prices of past 2 years to predict the stock values for 120 days.
- We plot the graph of closed prices against the date.

3. Project modules implementation work :

1) Login :

- login forms are one of the most important elements that a web page contains and hence designing these online forms is one of the most significant features when it comes to designing the website.
- When user didn't login then login page restrict their activity and don't give permission to use other tabs.



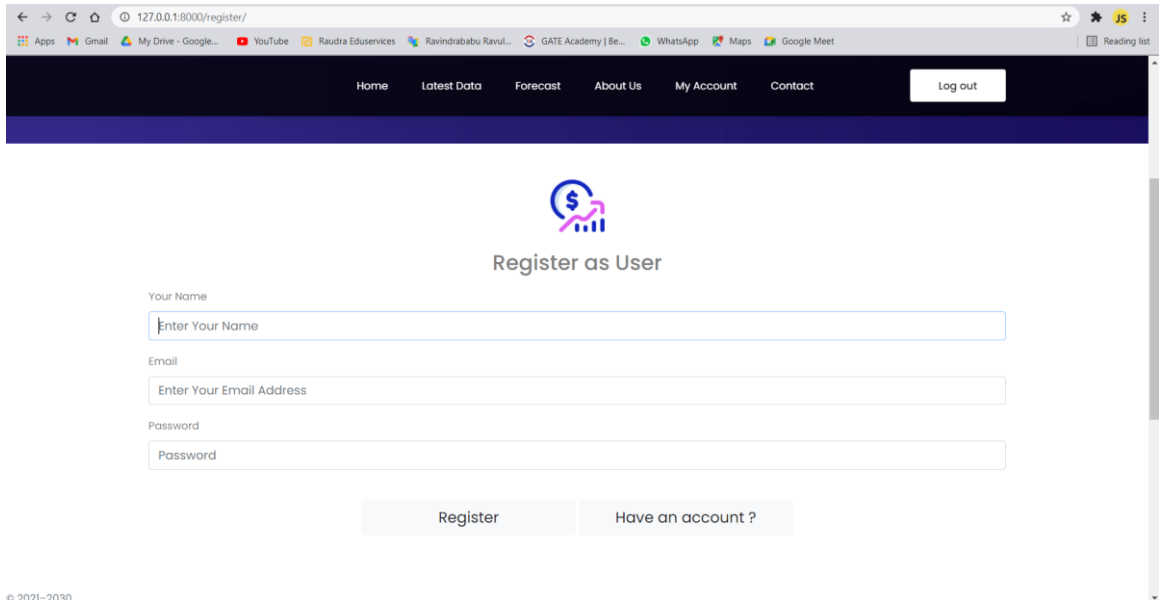
Img. 1 – Login Page

➤ Main logics blocks :

```
def login2(request):
    print(request)
    if request.method == "POST":
        login_user = request.POST.get('login_username')
        login_password = request.POST.get('login_password')
        user = authenticate(username=login_user, password=login_password)
        if user is not None:
            print("login")
            login(request, user)
            return redirect('/index')
        else:
            messages.success(request, 'Invalid Username or Password')
            return render(request, 'login.html')
    return render(request, 'login.html')
```

2) Sign Up/Register :

- If user don't have an account then they must have to create an account with valid name, email id and password.
- When user successfully created an account, they redirect to login page.



Img. 2 – Sign up page

➤ main logics blocks:

def register(request):

 if request.user.is_authenticated:

 return redirect('/')

 else:

 pass

 if request.method == "POST":

 reg_user = request.POST.get('reg_username')

 reg_email = request.POST.get('reg_email')

 reg_password = request.POST.get('reg_password')

 try:

 user = User.objects.get(username=reg_user)

 messages.success(request, 'You have already registered !')

 except User.DoesNotExist:

 special_characters = ""!@#\$%^&*()-+?_=<>/'"

 regex = '^([a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3})\$'

 error = 0

 if any(i.isdigit() for i in reg_user) or any(c in special_characters

for c in reg_user):

 messages.success(request, 'Username must contain alphabet

only !')

 error += 1

3IT31: Mini Project

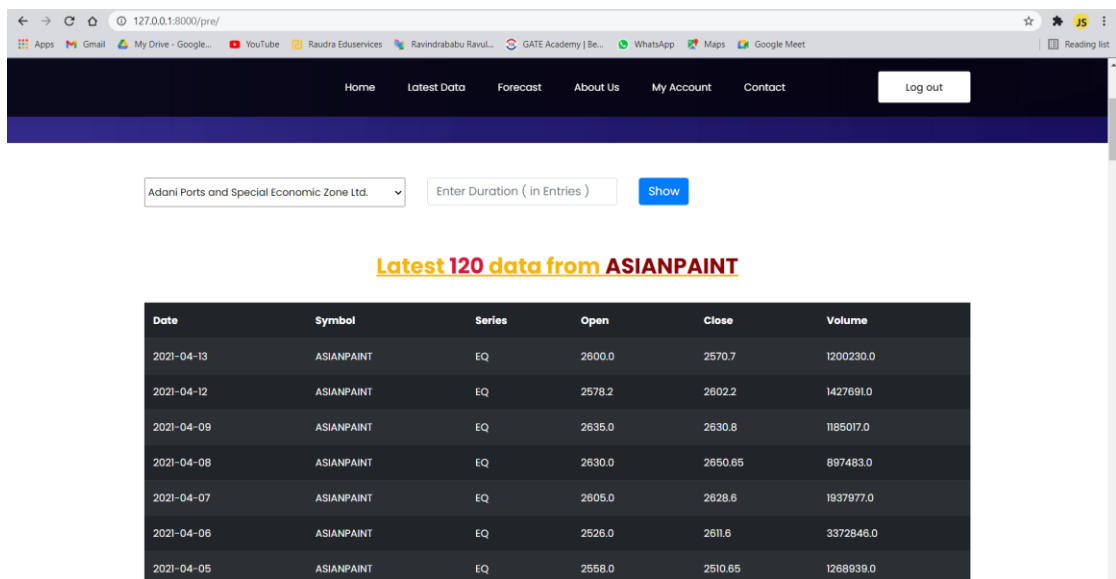
```
if (re.search(regex, reg_email)):
    pass
else:
    messages.success(request, 'Email must be in valid format !')
    error += 1
if len(reg_password) < 5 :
    messages.success(request, 'Password length must be 5 or more
than 5 !')
    error += 1
if error > 0:
    return redirect('/register')

user = User.objects.create_user(reg_user, reg_email,
reg_password)
user.save()
#messages.success(request, 'You have already registered !')
return redirect('/login')

return render(request, 'register.html')
```

3) Stock search :

- Whenever User choose a particular stock-name then this module gives the latest data about Open, Close and Volume of stocks.
- User have to enter number of entries according to their requirements.



Date	Symbol	Series	Open	Close	Volume
2021-04-13	ASIANPAINT	EQ	2600.0	2570.7	1200230.0
2021-04-12	ASIANPAINT	EQ	2578.2	2602.2	1427691.0
2021-04-09	ASIANPAINT	EQ	2635.0	2630.8	1185017.0
2021-04-08	ASIANPAINT	EQ	2630.0	2650.65	897483.0
2021-04-07	ASIANPAINT	EQ	2605.0	2628.6	1937977.0
2021-04-06	ASIANPAINT	EQ	2526.0	2611.6	3372646.0
2021-04-05	ASIANPAINT	EQ	2558.0	2510.65	1268939.0

Img. 3 – Stock Search

➤ **main logics blocks :**

```
def pre(request):
    if request.user.is_authenticated:
        pass
    else:
        return redirect('/login')

    if request.method == "POST":
        symbol_name = request.POST.get('symbol_name')
        duration = request.POST.get('dura')
        if all(i.isdigit() for i in duration):
            today = date.today()
            start = today + timedelta(-int(duration))
            stockData = nsepy.get_history(symbol=symbol_name,
start=start, end=today)
            stockData = stockData.reset_index()

            stockData['Date'] = stockData['Date'].astype(str)
            stockData['Open'] = stockData['Open'].astype(float)
            stockData['Volume'] = stockData['Volume'].astype(float)

            #chart = get_plot(stockData['Open'].tolist(),
stockData['Volume'].tolist())
            stockData = stockData.reindex(index=stockData.index[::-1])

            json_records =
stockData.reset_index().to_json(orient='records')
            data = json.loads(json_records)

            context = {'d': data,
                        'duration': duration,
                        'sym_name': symbol_name,
                        'visible': "visible",
                        }
            return render(request, 'pre.html', context)
        else:
            messages.success(request, 'Please enter duration in integer !')
            context = {'visible': "invisible"}
            return render(request, 'pre.html', context)
```

4) Price Prediction & Visualization:**• Implementation of linear regression:**

- Gradient descent is an optimization algorithm that works by efficiently searching the parameter space, intercept(c) and slope(m) for linear regression, according to the following rule:

$$\theta = \theta - \alpha (\delta / \delta \theta) J(\theta)$$

- Note that we used '=' to denote an assign or an update.
- The J(θ) is known as the cost function and α is the learning rate, a free parameter. In this tutorial, we're going to use a least squares cost function defined as following:

$$J(\theta) = 1/2n \sum_{(i=1 \text{ to } n)} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

- where n is the total number of training examples and $h_{\theta}(x^{(i)})$ is the hypothesis function defined like this:

$$h_{\theta}(x^{(i)}) = m * x^{(i)} + c$$

- where the super script (i) is used to denote the ith sample .
- We need derivatives for both c and m:

$$(\partial / \partial c) J(c, m) = 1/n \sum_{(i=1 \text{ to } m)} (h_{\theta}(x^{(i)}) - y^{(i)})$$

$$= 1/n \sum_{(i=1 \text{ to } m)} (c + m * x^{(i)} - y^{(i)})$$

$$(\partial / \partial m) J(c, m) = 1/n \sum_{(i=1 \text{ to } m)} (h_{\theta}(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$= 1/n \sum_{(i=1 \text{ to } m)} (c + m * x^{(i)} - y^{(i)}) x^{(i)}$$

- We need to be a little bit careful when we updates `c` and `m`. First, we calculate the temporary `c` and `m` with old `c` and `m`, and then we get new `c` and `m` from temp0 and temp1:

$$\text{temp0} := c - \alpha (\partial / \partial c) J(c, m)$$

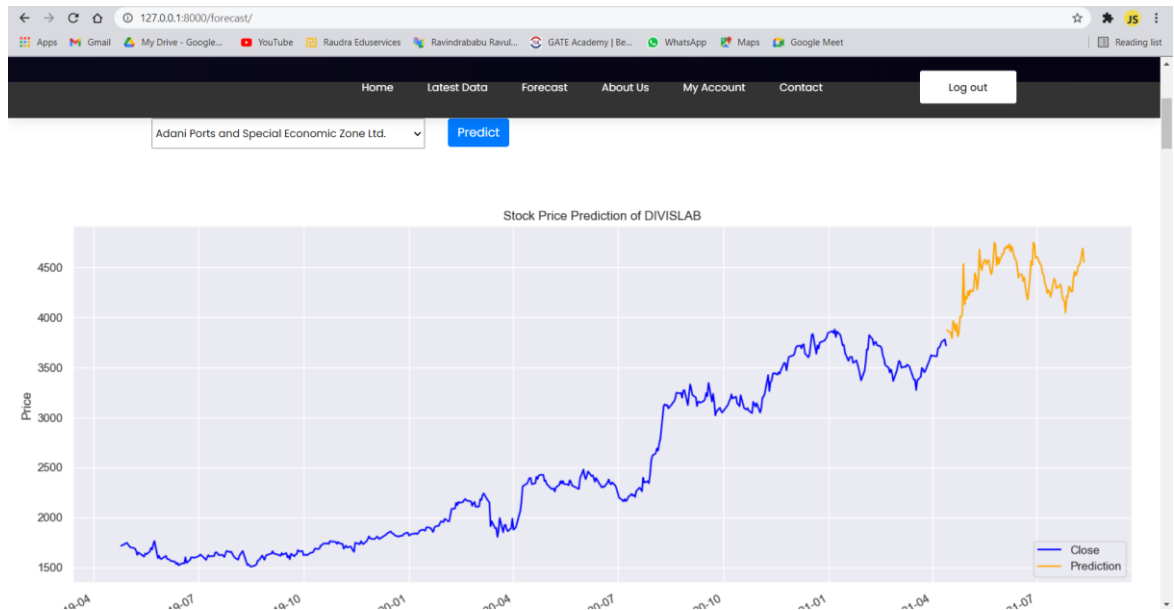
$$\text{temp1} := m - \alpha (\partial / \partial m) J(c, m)$$

$$c := \text{temp0}$$

$$m := \text{temp1}$$

3IT31: Mini Project

- User have to select a particular stock-name then this module gives prediction values of 120 days and gives the graphical representation.



Img. 4 – Prediction Algorithm Output

Past Data		Predicted Data	
Date	Price	Date	Prediction
2021-04-13	3847.0	2021-04-14	3870.94
2021-04-12	3760.0	2021-04-15	3857.60
2021-04-09	3718.5	2021-04-16	3857.71
2021-04-08	3698.55	2021-04-17	3845.16
2021-04-07	3701.0	2021-04-18	3793.15
2021-04-06	3605.0	2021-04-19	3964.18
2021-04-05	3616.1	2021-04-20	3925.97
2021-04-01	3618.0	2021-04-21	3865.38
2021-03-31	3600.0	2021-04-22	3934.00
2021-03-30	3500.0	2021-04-23	3810.57
2021-03-26	3450.0	2021-04-24	3845.00

- main logics blocks :

class LinearRegression:

```
def __init__(self, learning_rate=0.01, n_iters=1000):  
    self.lr = learning_rate  
    self.n_iters = n_iters  
    self.slope = None  
    self.c = None
```

```
def train(self, x, y):
    n_samples, n_features = x.shape
    self.slope = np.zeros(n_features)
    self.c = 0

    for _ in range(self.n_iters):
        y_predicted = np.dot(x, self.slope) + self.c
        dw = (1 / n_samples) * np.dot(x.T, (y_predicted - y))
        db = (1 / n_samples) * np.sum(y_predicted - y)
        self.slope -= self.lr * dw
        self.c -= self.lr * db

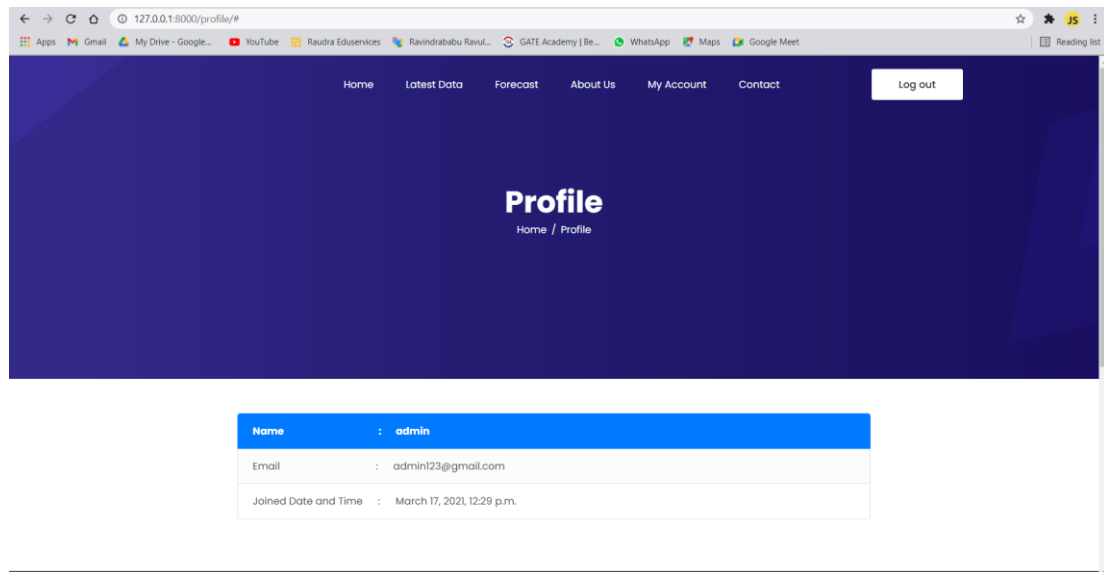
    def predict(self, x):
        y_approx = np.dot(x, self.slope) + self.c
        return y_approx
```

```
def plot1(org_df, pred_df, ttl='', x_label='x', y_label='y', color='blue'):
    global graph
    plt.clf()
    sns.set_theme(style="darkgrid")
    org_df['Close'].plot(figsize=(15, 6), color=color)
    pred_df['Prediction'].plot(figsize=(15, 6), color='orange')
    plt.legend(loc=4)
    set_labels(ttl, x_label, y_label)
    graph = get_graph()
```

```
def get_graph():
    buffer = BytesIO()
    buffer.flush()
    plt.savefig(buffer, format='png') #,dpi=700
    buffer.seek(0)
    image_png = buffer.getvalue()
    graph = base64.b64encode(image_png)
    graph = graph.decode('utf-8')
    print("print",buffer)
    buffer.close()
    return graph
```

5) My Profile :

- My Profile page have user name, user email id and joined date and time.



main logics blocks :

def profile(request):

if request.user.is_authenticated:

pass

else:

return redirect('/login')

context = {

'uname' : request.user.username,

'uemail' : request.user.email,

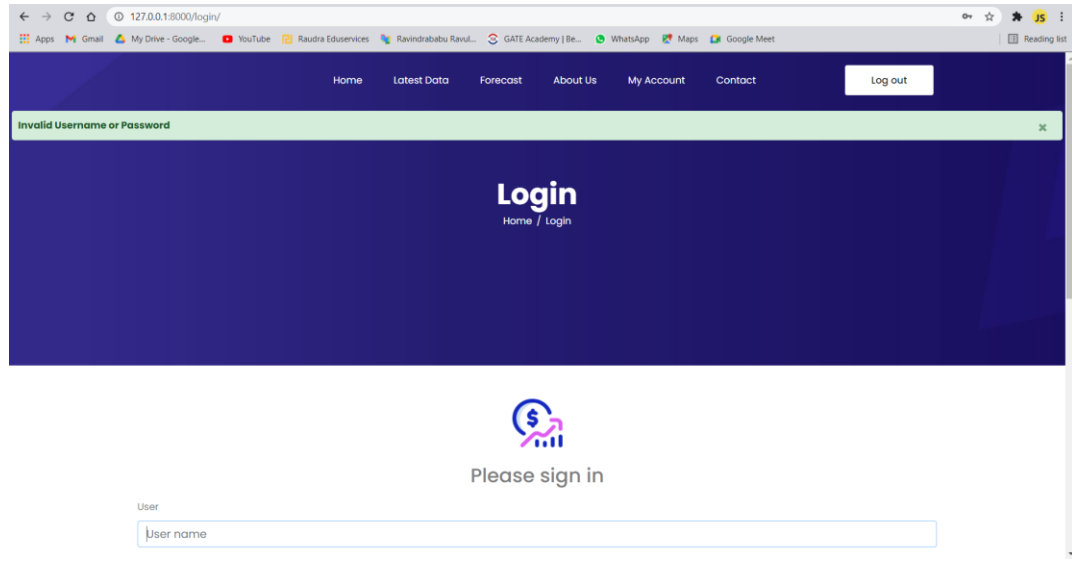
'udate' : request.user.date_joined}

return render(request, 'profile.html', context)

4. Test Cases of Implemented modules:

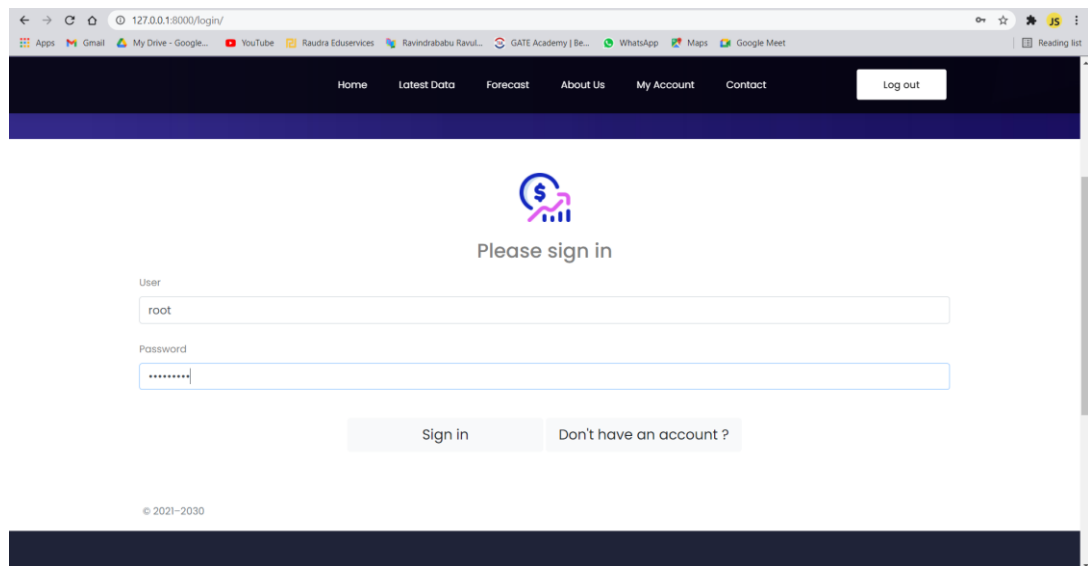
1) Login :

Case – 1 : When we enter invalid username or password, it denies the access.



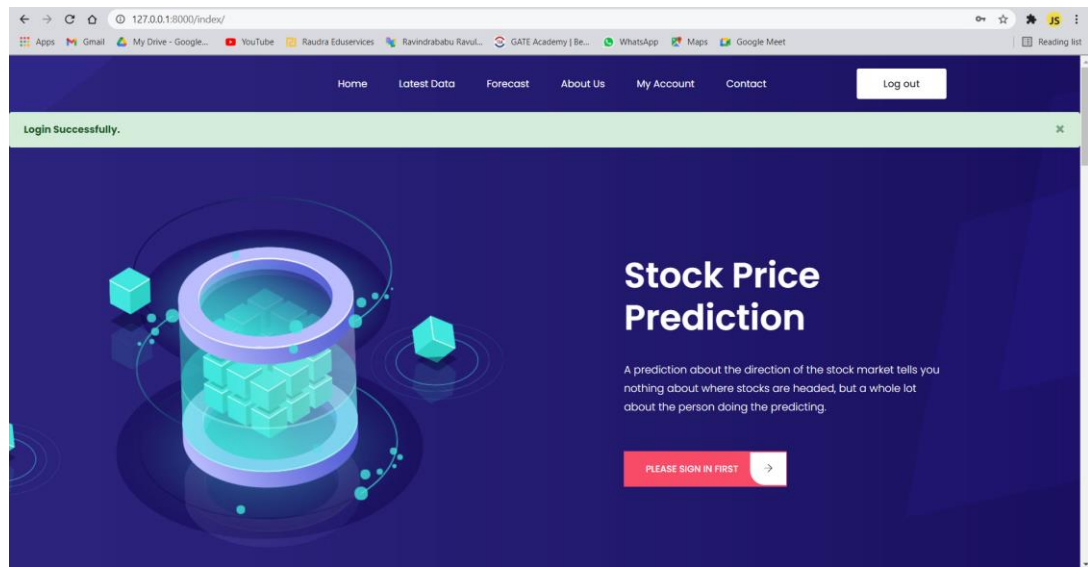
Img. 5 – Invalid Username and Password

Case – 2 : When we enter valid username and password, it allow access of homepage.



Img. 6 – Valid Input

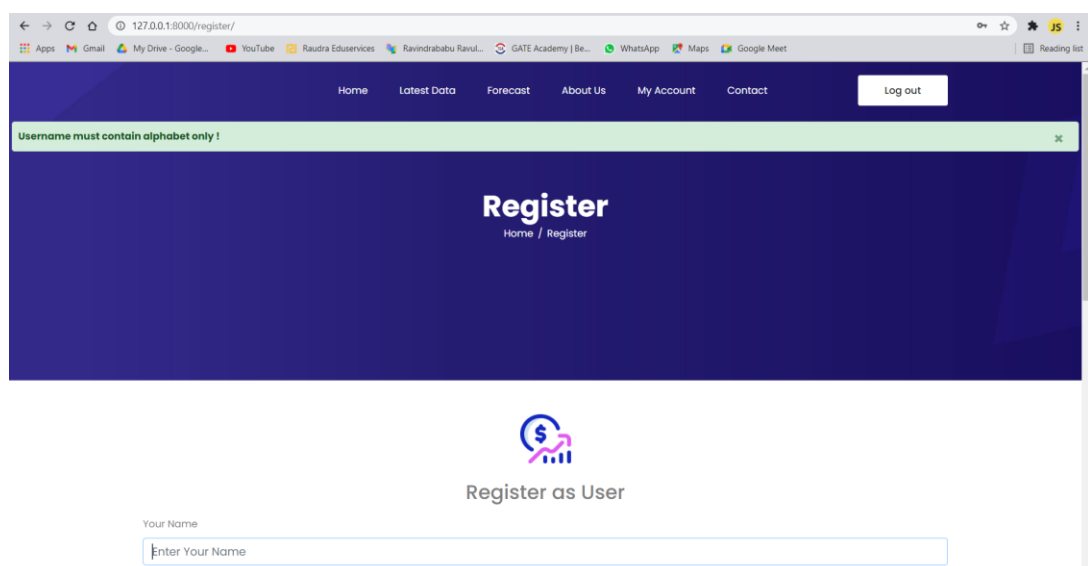
3IT31: Mini Project



Img. 7 – Successful login

2) Sign Up / Register :

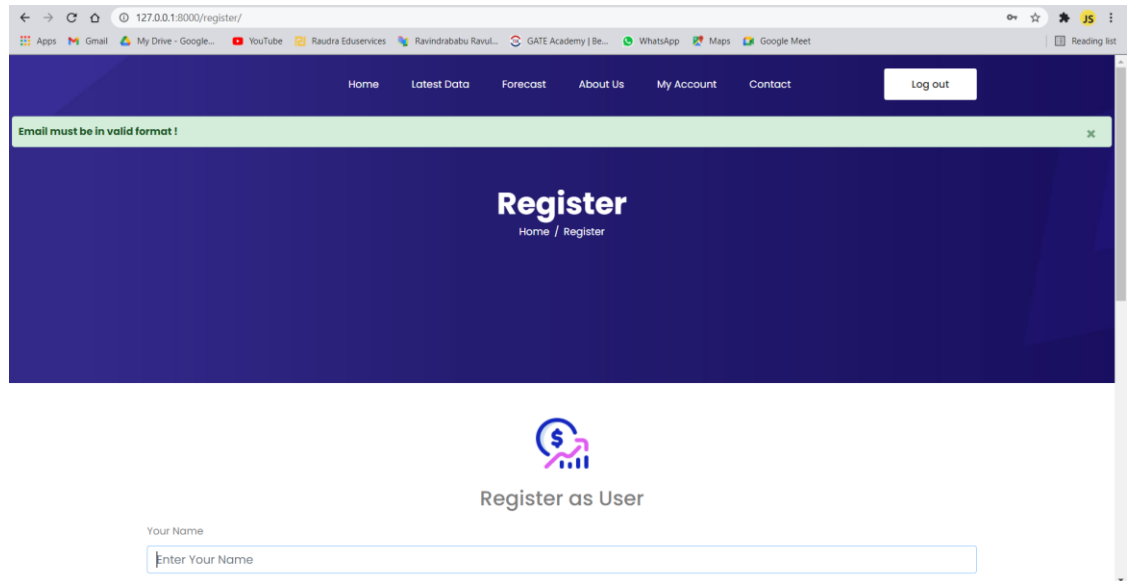
Case – 1 : When we enter non-alphabetic in username then it will denied and throws alert message



Img. 8 – Invalid Username input

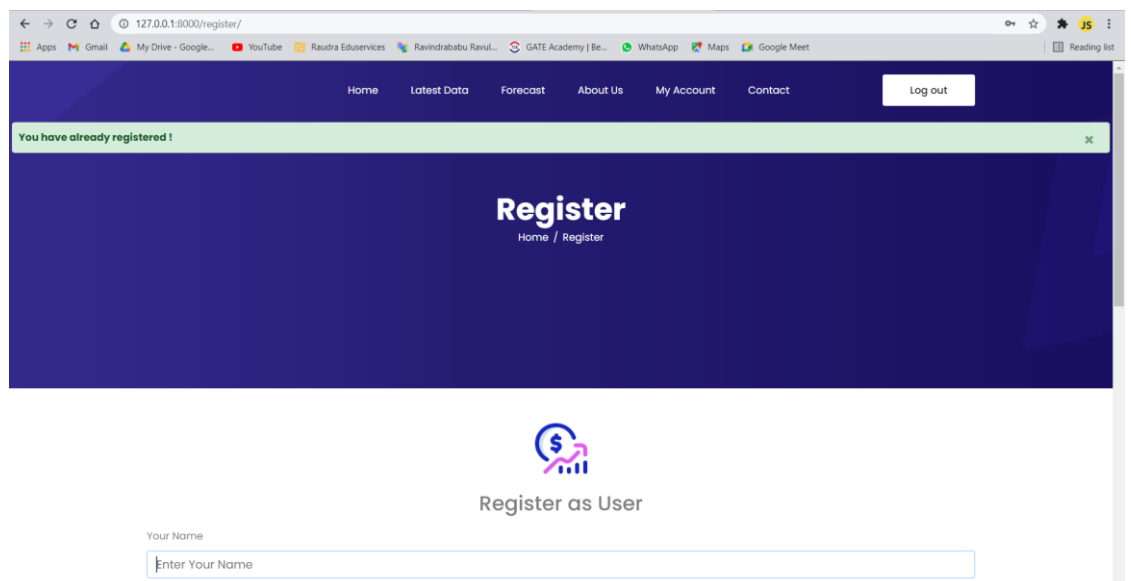
3IT31: Mini Project

Case – 2 : When we enter invalid email format then it will denied and throws alert message



Img. 9 – Invalid Email Format

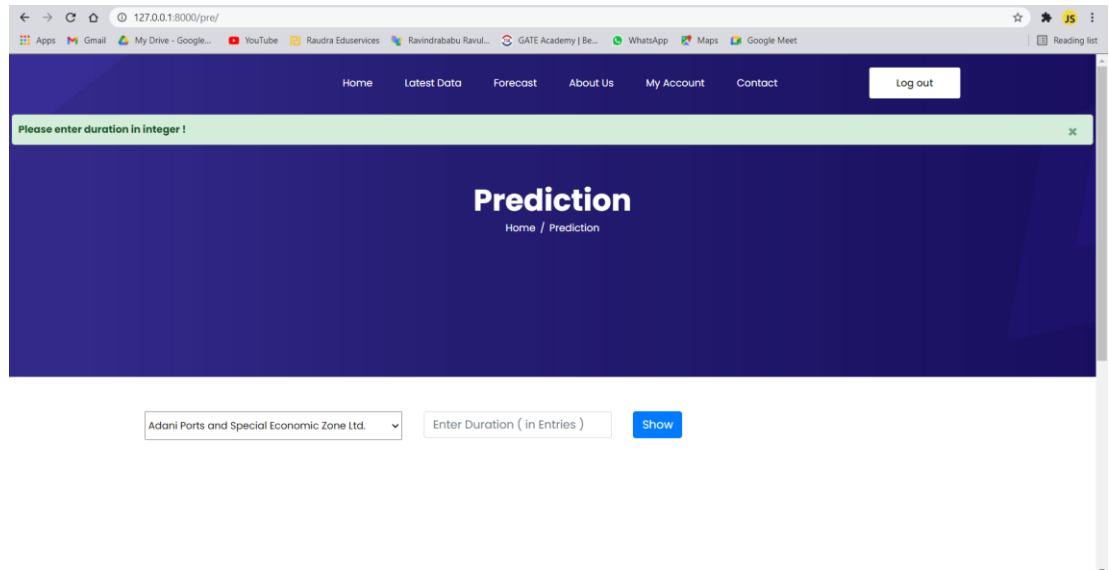
Case – 3 : When we have already registered, it throws Exception.



Img. 10 – Registered User

3) Stock-Search :

Case – 1 : When we enter alphabet or decimal value in duration field then it shows alert message.



Img. 11 – Invalid Duration

4) Price prediction :

Case - 1: In this case we compared the past 120 days of RELIANCE stock close prices with the prediction algorithm outputs. It is not pretty accurate but we can say that it gives idea about stock price increment or decrement.

Comparison table:

Date	Symbol	Close	Prediction
28-09-2020	RELIANCE	2216.25	1591.792
29-09-2020	RELIANCE	2245.05	1457.004
30-09-2020	RELIANCE	2234.35	1573.177
01-10-2020	RELIANCE	2225.25	1611.334
05-10-2020	RELIANCE	2212.2	3026.628
06-10-2020	RELIANCE	2210.35	2088.228
07-10-2020	RELIANCE	2257.5	2968.066
08-10-2020	RELIANCE	2239.25	2042.042
09-10-2020	RELIANCE	2233.45	2284.92
12-10-2020	RELIANCE	2237.05	1984.282
13-10-2020	RELIANCE	2280.7	1896.704
14-10-2020	RELIANCE	2287.5	1842.84
15-10-2020	RELIANCE	2206.5	1990.061

3IT31: Mini Project

16-10-2020	RELIANCE	2175.8	2444.675
19-10-2020	RELIANCE	2176.2	2191.133
20-10-2020	RELIANCE	2155.9	1939.29
21-10-2020	RELIANCE	2124.6	2113.759
22-10-2020	RELIANCE	2106.95	2109.337
23-10-2020	RELIANCE	2113.05	1835.619
26-10-2020	RELIANCE	2029.1	1795.338
27-10-2020	RELIANCE	2034.5	1733.506
28-10-2020	RELIANCE	2011.45	1777.981
29-10-2020	RELIANCE	2026.9	1850.719
30-10-2020	RELIANCE	2054.5	1844.236
02-11-2020	RELIANCE	1877.45	1714.986
03-11-2020	RELIANCE	1850.4	1849.103
04-11-2020	RELIANCE	1913.2	1836.627
05-11-2020	RELIANCE	1955	1807.884
06-11-2020	RELIANCE	2029.15	1717.886
09-11-2020	RELIANCE	2050.7	1908.679
10-11-2020	RELIANCE	2084.55	2105.3
11-11-2020	RELIANCE	1997.2	2083.723
12-11-2020	RELIANCE	1980	1966.181
13-11-2020	RELIANCE	1996.4	1976.959
14-11-2020	RELIANCE	2002.3	2851.328
17-11-2020	RELIANCE	1993.25	2161.85
18-11-2020	RELIANCE	1987.2	2008.396
19-11-2020	RELIANCE	1973.15	1916.176
20-11-2020	RELIANCE	1899.5	1838.543
23-11-2020	RELIANCE	1950.7	1902.309
24-11-2020	RELIANCE	1964.05	1928.554
25-11-2020	RELIANCE	1947.8	2187.808
26-11-2020	RELIANCE	1952.6	2023.14
27-11-2020	RELIANCE	1929.8	1895.44
01-12-2020	RELIANCE	1954.9	2481.625
02-12-2020	RELIANCE	1958.15	3325.143
03-12-2020	RELIANCE	1964.05	2354.187
04-12-2020	RELIANCE	1946.75	2261.038
07-12-2020	RELIANCE	1958.2	2446.205
08-12-2020	RELIANCE	1993.75	2464.213
09-12-2020	RELIANCE	2026.95	3290.699
10-12-2020	RELIANCE	2007	2800.512
11-12-2020	RELIANCE	2005.8	2504.178
14-12-2020	RELIANCE	1991.3	2675.388
15-12-2020	RELIANCE	1974.35	2288.555

3IT31: Mini Project

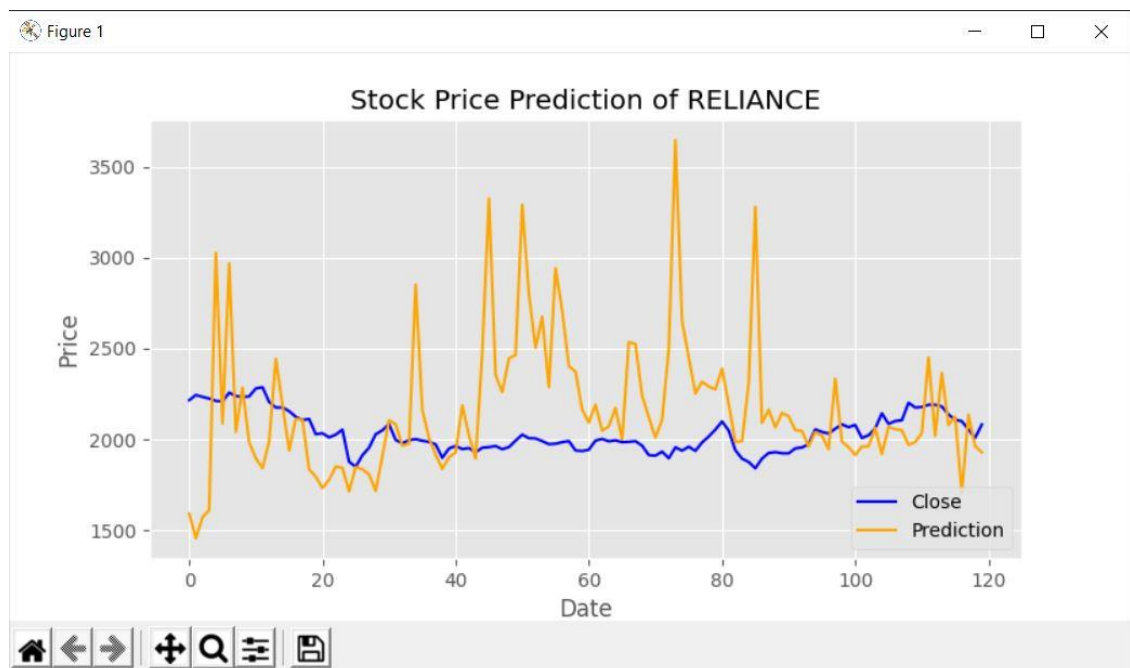
16-12-2020	RELIANCE	1976.55	2941.51
17-12-2020	RELIANCE	1985.6	2709.808
18-12-2020	RELIANCE	1991.55	2403.796
21-12-2020	RELIANCE	1939.7	2372.833
22-12-2020	RELIANCE	1936.7	2164.479
23-12-2020	RELIANCE	1943.85	2093.773
24-12-2020	RELIANCE	1994.15	2191.815
28-12-2020	RELIANCE	2003.3	2048.924
29-12-2020	RELIANCE	1990.05	2072.625
30-12-2020	RELIANCE	1995.5	2173.552
31-12-2020	RELIANCE	1985.3	2000.127
01-01-2021	RELIANCE	1987.5	2535.998
04-01-2021	RELIANCE	1990.85	2525.599
05-01-2021	RELIANCE	1966.1	2240.647
06-01-2021	RELIANCE	1914.25	2123.989
07-01-2021	RELIANCE	1911.15	2010.172
08-01-2021	RELIANCE	1933.7	2108.964
11-01-2021	RELIANCE	1897.25	2495.733
12-01-2021	RELIANCE	1957.05	3645.217
13-01-2021	RELIANCE	1938.8	2652.261
14-01-2021	RELIANCE	1960.6	2450.851
15-01-2021	RELIANCE	1937.45	2253.477
18-01-2021	RELIANCE	1983.95	2317.441
19-01-2021	RELIANCE	2016.4	2291.146
20-01-2021	RELIANCE	2054.7	2275.803
21-01-2021	RELIANCE	2099.4	2389.941
22-01-2021	RELIANCE	2049.6	2194.685
25-01-2021	RELIANCE	1941	1987.261
27-01-2021	RELIANCE	1895	1990.373
28-01-2021	RELIANCE	1876.55	2310.346
29-01-2021	RELIANCE	1841.95	3278.566
01-02-2021	RELIANCE	1895.3	2090.682
02-02-2021	RELIANCE	1925.8	2164.709
03-02-2021	RELIANCE	1930.65	2066.736
04-02-2021	RELIANCE	1924.3	2146.341
05-02-2021	RELIANCE	1923.75	2128.008
08-02-2021	RELIANCE	1951.45	2051.847
09-02-2021	RELIANCE	1956.15	2046.52
10-02-2021	RELIANCE	1974.3	1961.942
11-02-2021	RELIANCE	2055.7	2040.886
12-02-2021	RELIANCE	2041.6	2021.198
15-02-2021	RELIANCE	2032.6	1945.874

3IT31: Mini Project

16-02-2021	RELIANCE	2059.5	2334.424
17-02-2021	RELIANCE	2083.25	1987.922
18-02-2021	RELIANCE	2067.7	1958.467
19-02-2021	RELIANCE	2080.3	1914.77
22-02-2021	RELIANCE	2008.1	1961.096
23-02-2021	RELIANCE	2023.45	1962.502
24-02-2021	RELIANCE	2061	2063.872
25-02-2021	RELIANCE	2144.35	1920.226
26-02-2021	RELIANCE	2085.8	2069.211
01-03-2021	RELIANCE	2101.7	2057.028
02-03-2021	RELIANCE	2106	2050.746
03-03-2021	RELIANCE	2202.1	1970.615
04-03-2021	RELIANCE	2175.85	1985.798
05-03-2021	RELIANCE	2178.7	2034.587
08-03-2021	RELIANCE	2191.1	2451.756
09-03-2021	RELIANCE	2191.05	2019.868
10-03-2021	RELIANCE	2181.95	2365.84
12-03-2021	RELIANCE	2137.6	2079.788
15-03-2021	RELIANCE	2108.9	2126.025
16-03-2021	RELIANCE	2100.6	1706.33
17-03-2021	RELIANCE	2055.35	2136.639
18-03-2021	RELIANCE	2009.1	1963.894
19-03-2021	RELIANCE	2082	1929.669

Table 2 - Closed price and Predicted price comparison

Comparison Graph:



Img. 12 – Comparison Graph