

Stock Market Tracking App



Dipali Singh - RA1911030010138

Project Description

In the present times when investors need to be two steps ahead at all time, we need a web-means that can be as fast and efficient. Investors hold back from dabbling in the stock market due to lack of information. Our aim while building this website is to facilitate and assist investors to keep track of the stock market. We want to make a user-friendly website that will allow investors to favourite the stocks they have invested in and track their stocks by analysing the percentage increase or percentage decrease alongside with graphs over a given timeline. The app will be complete with blogs and helpline that will give more insights into the current market status.



Roles and Models

The roles of the different stakeholders are as follows:

Manage and development of the project.

Manage the finances and development.

Manage the marketing and development.



Roles and Models

Stakeholder Name	Activity / Area / Phase	Interest	Influence	Priority (High / Medium/Low)
Regional Head of Sales & Marketing	Subscription using mobile App	High	High	1
Finance Account Receivable consultant	Multiple Currency Payment	High	Low	3
Developer	Development of the app	High	High	1
Stock broker	User	High	Low	3



Roles and Models

Stakeholder Name	Activity / Area / Phase	Interest	Influence	Priority (High / Medium/Low)
Customer	User	High	High	2
Supplier	Make resources available	High	High	1
Investors	Provide financial assistance	High	High	1
Parent companies	Monitor the functioning	Low	High	3
Stock broker	User	High	Low	3

An illustration of a man and a woman in a modern office setting. The man, with a beard and wearing a red t-shirt and dark trousers, stands with his hands in his pockets, looking towards the woman. The woman, with dark curly hair and wearing a black and white striped long-sleeved shirt and blue jeans, is seated on a white stool, working on a laptop on a wooden desk. A pink lamp hangs above the desk, and a pink vase with a plant is on the desk. The background is a light gray wall with a large white circle.

Methodology:

Agile software development

In software development, agile practices involve discovering requirements and developing solutions through the collaborative effort of self-organizing and cross-functional teams and their customer/end user.



Communication Plans for Stakeholders

The different stakeholders involved must have a proper channel of communication to ensure the proper flow of development. Communication is extremely crucial to maintain the quality standard.

A few ways to ensure proper communication are listed below:

- Frequent meetings over online platforms like g-meet.
- Providing a single-page dashboard and communicating via meetings.
- Weekly report for High Interest and Low Influence.



Project Management Plan

Focus Area	Details
Integration Management	<p>Stock market tracking app is a web app that will allow users to track stock market data. Our project team will hold little to no hierarchy with all out teammates working together in collaboration with outside parties. The different parts of the project will be integrated and the change will be managed using agile project methodology.</p>
Scope Management	<p>The scope of our project involves the tracking and transaction of stocks. The different stakeholders are:</p> <ul style="list-style-type: none">>Investors>Developers>Stock broker >Customers



Project Management Plan

Schedule Management	<p>Web app with detailed stock market analysis followed by a full stack app. Steps involved are:</p> <ul style="list-style-type: none">• Planning Road map.• Requirement Analysis• Backend Software• Backend Development• Frontend Development• Database Implementation• Testing
Quality Management	<p>The web app will include the necessary data encryption and will be continuously improved to maintain quality</p>



Project Management Plan

Resource Management

Estimate and Manage the need

People: Dipali Singh

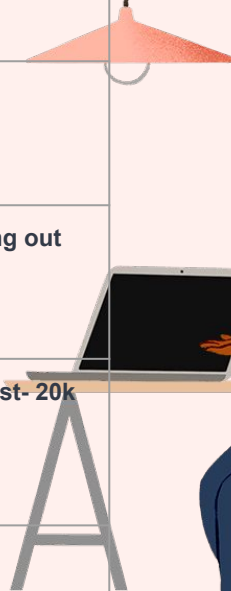
Finance: 65k

Physical: Facilities, IT Infrastructure



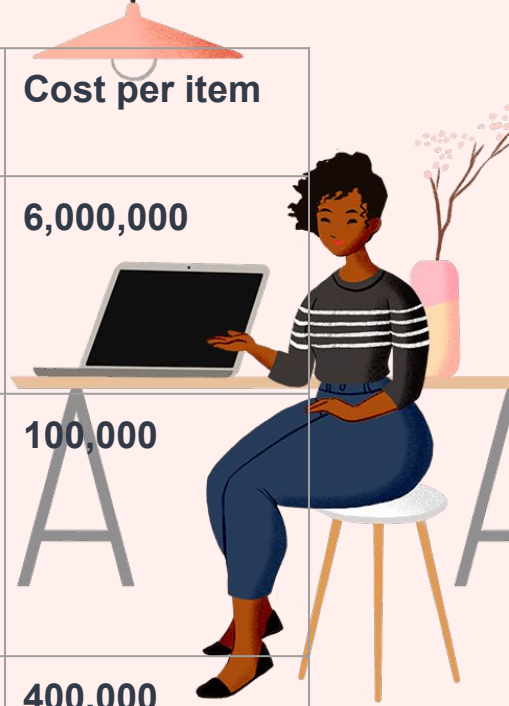
Estimation -

Activity Description	Sub - Task Desc.	Efforts in hours	Costs in INR
Design the web app	To create a webapp using python and django	6-12	Frontend- 30k Backend- 30k Integration- 5k
Research and development	Evaluating of software tech trends and incorporating them with updates and patches.	4-8	For Research - 15K For Rolling out patches/ updates - 5K
Data analytics	Administration Data, Database, Management, and evaluating stock market trends.	4-5	Data Analyst- 15k Data scientist- 20k
Identify Data Source for displaying units of Energy Consumption	Go through Interface contract (Application Data Exchange) documents	5	15K
Marketing	Advertisement	8-10	1L



Maintenance and Support Cost

Category	Details	Qty	Cost per annum	Cost per item
People	Network, System, Middleware and DB admin	5	2,000,000	6,000,000
License	Operating System, Database, Middleware, IDE	4	10000	100,000
Infrastructure	Server, Storage and Network	3	20000	400,000



Risk



Risk Identification

Risk identification and categorization:

- The predictions can be wrong due to inaccuracy in the app.
- Sometimes wrong stocks can be shown which may lead to misinformation.
- Out of date information can be shown.
- Sometimes personalised stocks may be incorrect due to inaccurate predictions.



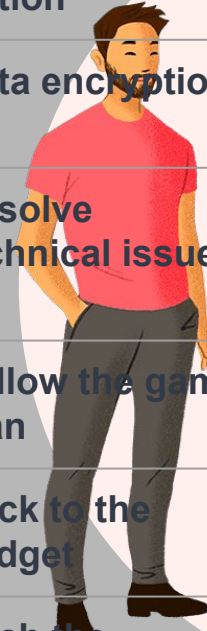
List (Describe) Register

Risk ID#	Risk Description	Impact Description
R01	Hacking of application	Has a high impact as it can leak information
R02	Technical difficulties like Data Security.	High impact technical risk
R03	Unable to meet deadlines.	Low impact risk
R04	Insufficiency of the budget	Medium impact risk as it slows down development
R05	Lack of stakeholder engagement	Medium impact risk



Managing Risk

Risk ID#	Status	Risk Appetite	Action
R01	Open	Avoid	Data encryption
R02	Open	Mitigate	Resolve technical issues
R03	Open	Mitigate	Follow the game plan
R04	Open	Avoid	Stick to the budget
R05	Open	Accept	Pitch the product

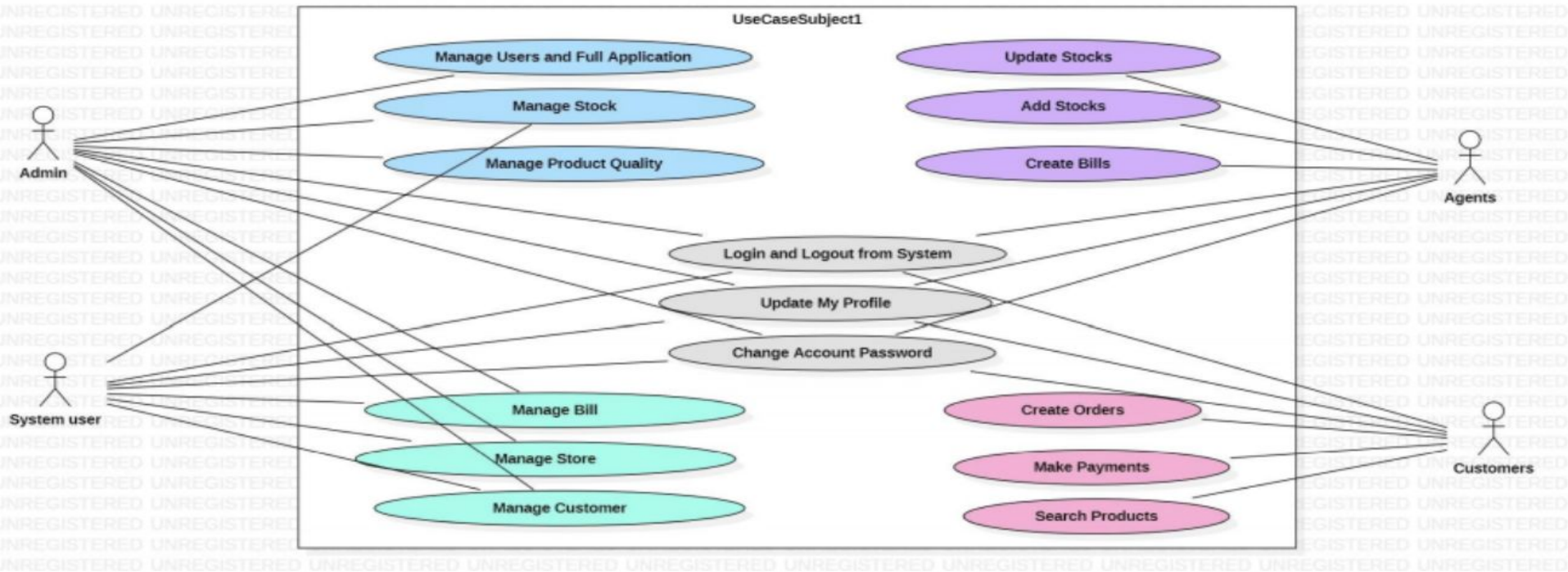
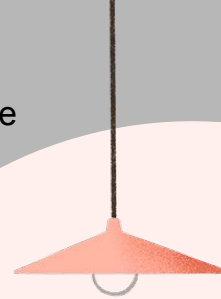


Architecture and Design of the System

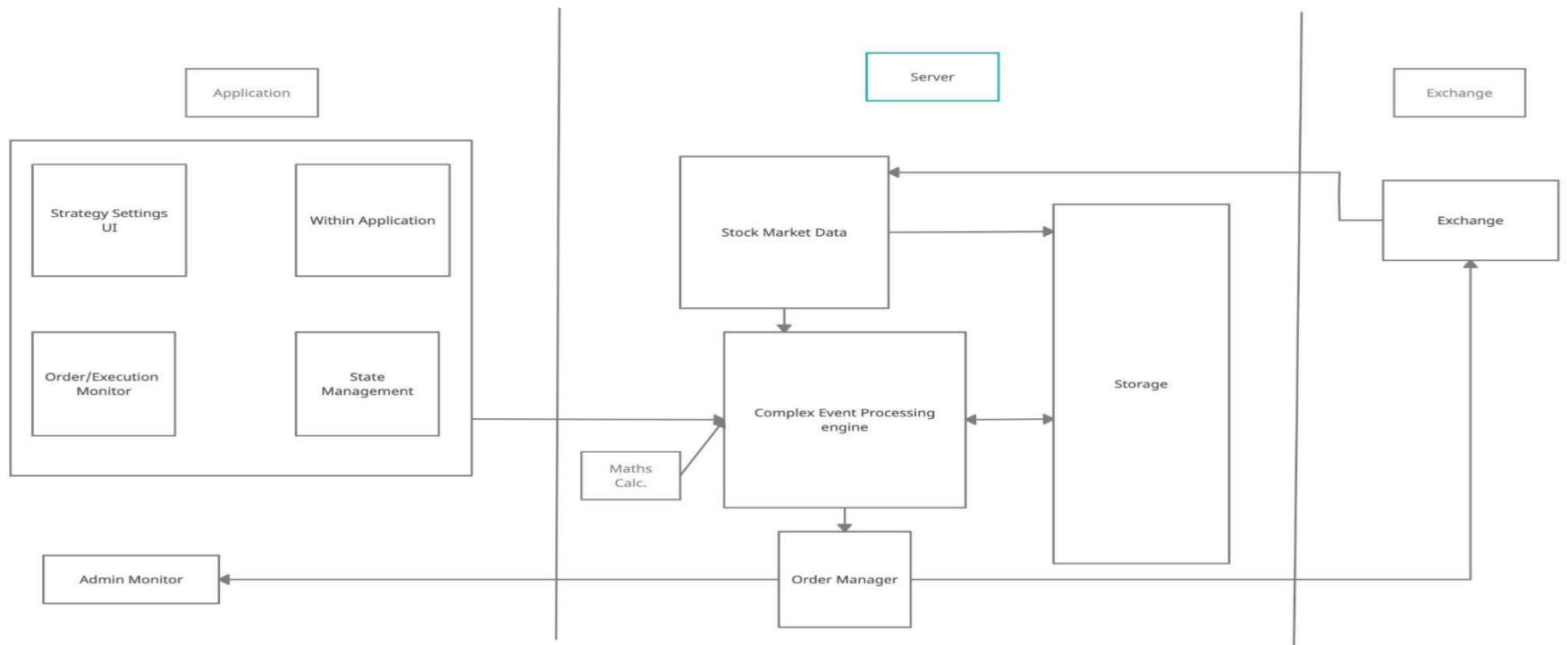


Use case Diagram

This use case diagram is a graphic depiction of the interactions among the elements of the Stock Management System. It represents the methodology used in system analysis to identify, clarify, and organize system requirements of Stock Management System.



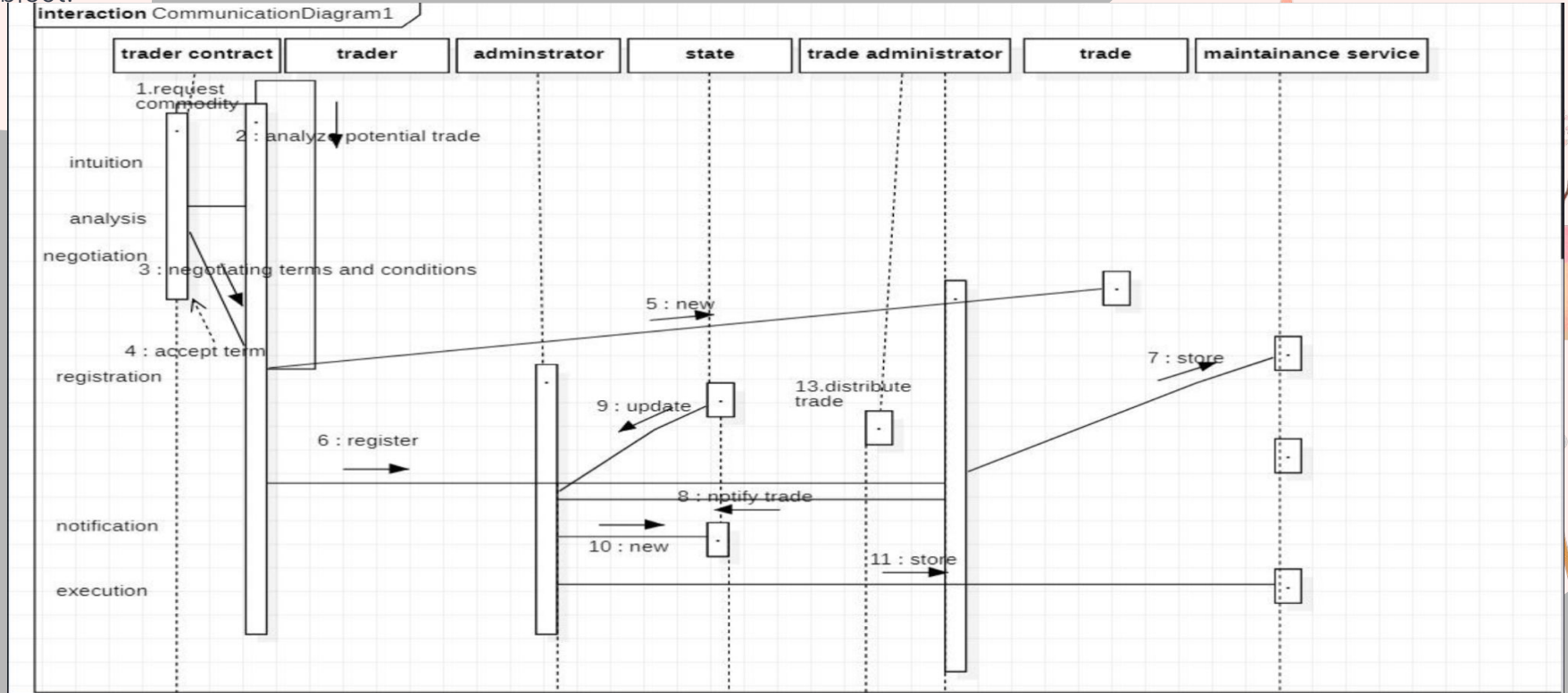
Architecture Diagram – An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap.



Architecture Diagram Of App

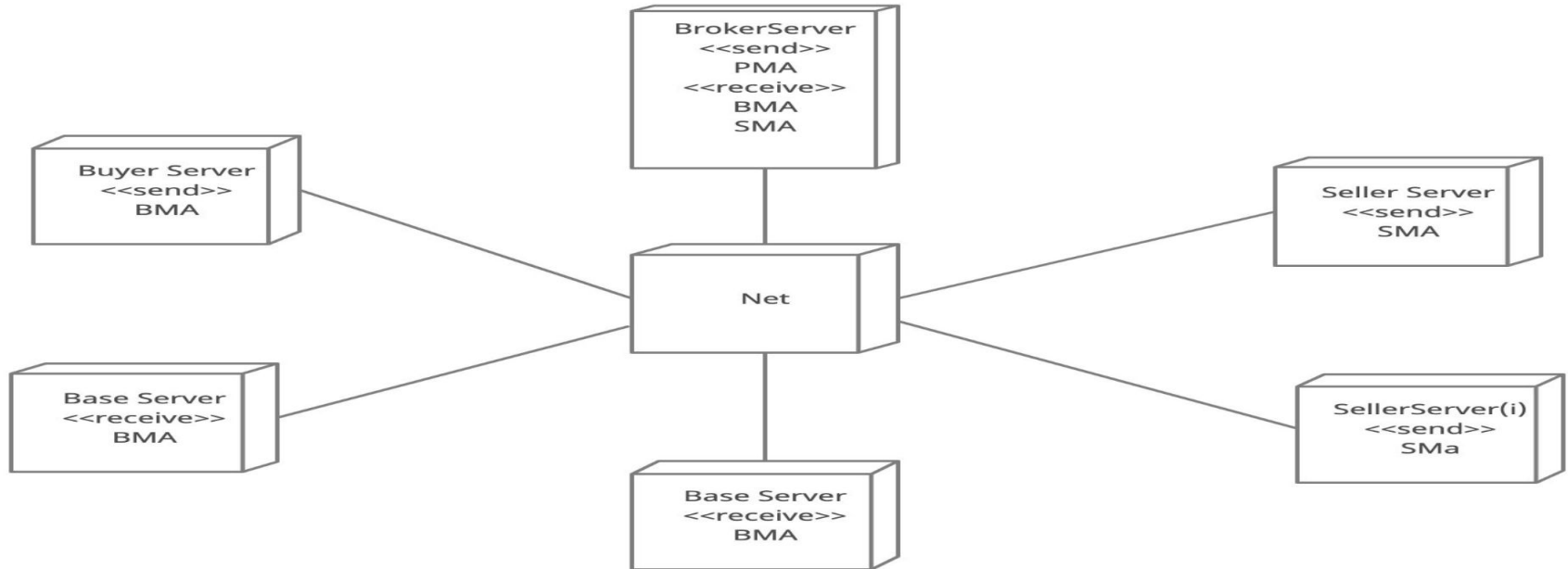
Collaboration Diagram

– A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.



Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams are a kind of structure diagram used in modeling the physical aspects of an object-oriented system. The deployment diagram of our application is shown below:



Deployment Diagram for Stock Marketing App

Sample Frontend Design -

Stock Portfolio

AboutAdd StockDelete Stock

Get Stock Quote

Stock Quote

Add Stock...

Add To Portfolio

Add Stock

Company Name	Stock Price	Previous Close	Market Cap	YTD Change	52Wk High	52Wk Low
Alphabet, Inc.	\$1227.49	\$1239.41	\$851167691897	0.175454%	\$1289.27	\$970.11
Facebook, Inc.	\$197.7	\$195.94	\$563982744000	0.453113%	\$208.66	\$123.02
Apple, Inc.	\$209.665	\$209.68	\$964685438200	0.327691%	\$233.47	\$142
Amazon.com, Inc.	\$1900	\$1912.45	\$939846400000	0.236043%	\$2050.5	\$1307

Delete goog

Delete fb

Delete aapl

Delete amzn

Implementation of Module 1

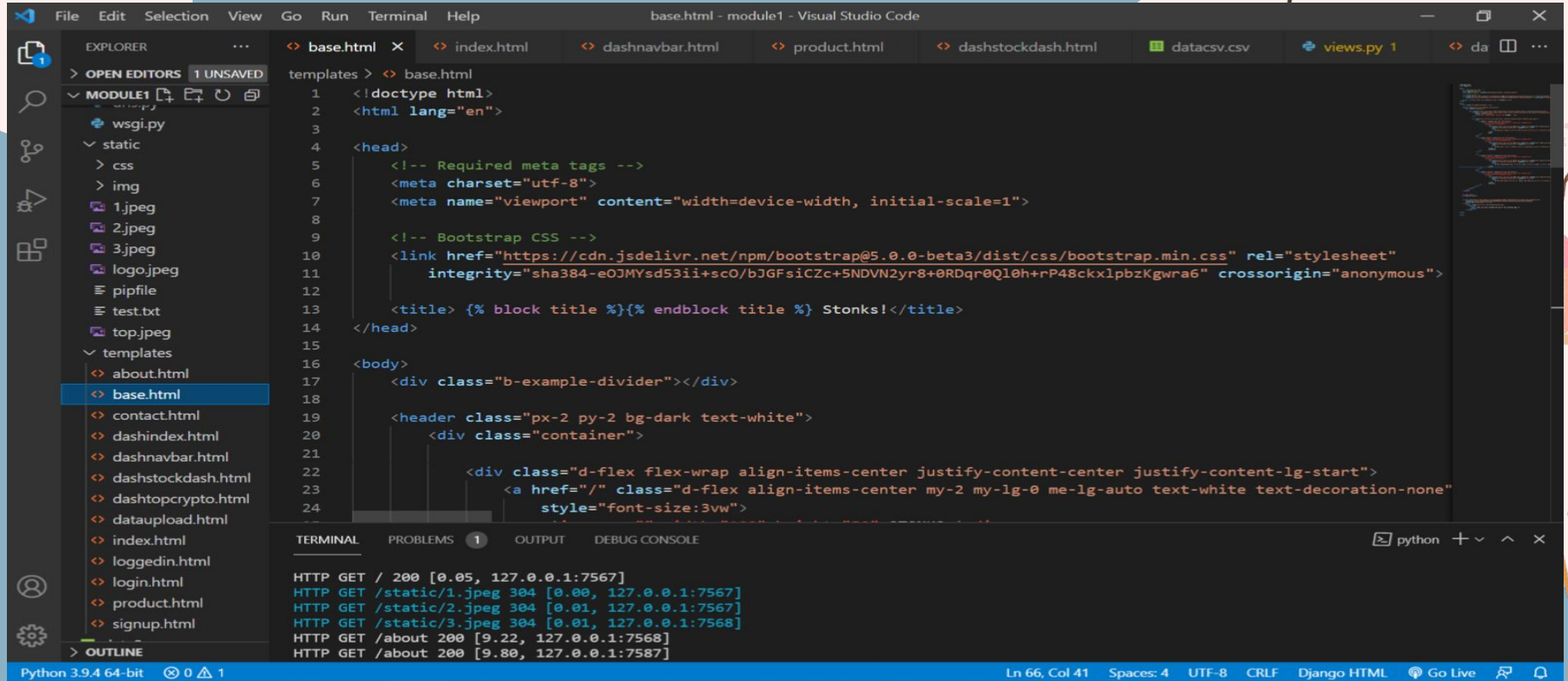
Code of Module 1

The code for the first module is can be found in the following link:

<https://drive.google.com/drive/u/0/folders/1d114nrIDAKcCiGxYD5I6gC7bBieDlIsB>



Code:



The screenshot shows the Visual Studio Code interface with a Django project. The Explorer sidebar on the left displays the project structure, including files like `wsgi.py`, `static` (with `css` and `img` subfolders), `templates`, and `base.html` (which is selected). The main editor area shows the content of `base.html`, which is a Django template using Bootstrap 5. The template includes a `<doctype html>` declaration, a `<html lang="en">` declaration, a `<head>` section with meta tags for charset and viewport, and a link to Bootstrap CSS. The body contains a `<div class="b-example-divider">` and a `<header class="px-2 py-2 bg-dark text-white">` with a `<div class="container">` containing a `<div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-start">` with a `` with `style="font-size:3vw">`.

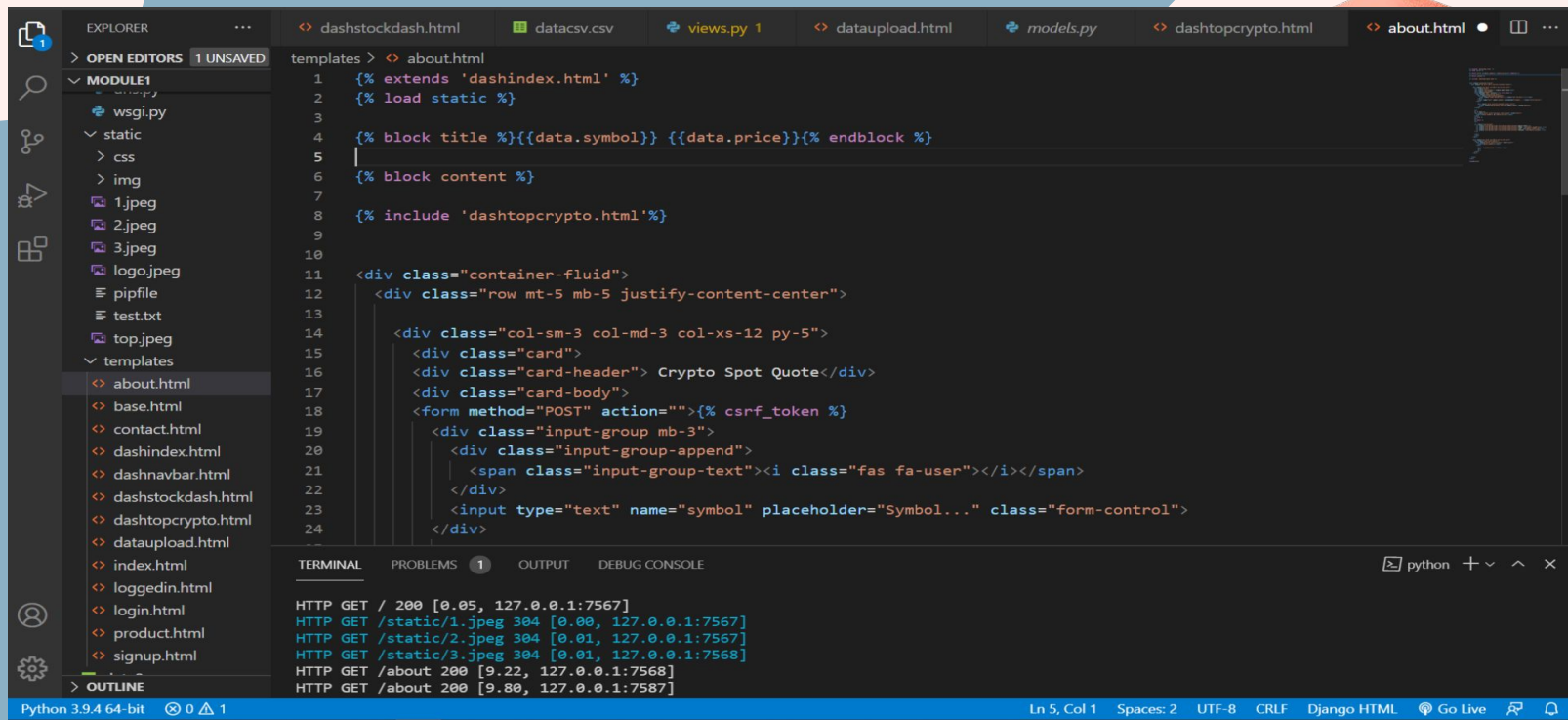
```
1 <!doctype html>
2 <html lang="en">
3
4 <head>
5     <!-- Required meta tags -->
6     <meta charset="utf-8">
7     <meta name="viewport" content="width=device-width, initial-scale=1">
8
9     <!-- Bootstrap CSS -->
10    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet"
11        integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+RP48ckxlpbzKgwra6" crossorigin="anonymous">
12
13    <title> {% block title %}{% endblock title %} Stonks!</title>
14 </head>
15
16 <body>
17     <div class="b-example-divider"></div>
18
19     <header class="px-2 py-2 bg-dark text-white">
20         <div class="container">
21
22             <div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-start">
23                 <a href="/" class="d-flex align-items-center my-2 my-lg-0 me-lg-auto text-white text-decoration-none">
24                     style="font-size:3vw">
```

The TERMINAL panel at the bottom shows the following output:

```
HTTP GET / 200 [0.05, 127.0.0.1:7567]
HTTP GET /static/1.jpeg 304 [0.00, 127.0.0.1:7567]
HTTP GET /static/2.jpeg 304 [0.01, 127.0.0.1:7567]
HTTP GET /static/3.jpeg 304 [0.01, 127.0.0.1:7568]
HTTP GET /about 200 [9.22, 127.0.0.1:7568]
HTTP GET /about 200 [9.80, 127.0.0.1:7587]
```

The status bar at the bottom indicates: Python 3.9.4 64-bit, 0 errors, 1 warning, Ln 66, Col 41, Spaces: 4, UTF-8, CRLF, Django HTML, Go Live, and icons for search, run, and notifications.

Code:



```
templates > < about.html
1  {% extends 'dashindex.html' %}
2  {% load static %}
3
4  {% block title %}{{data.symbol}} {{data.price}}{% endblock %}
5  |
6  {% block content %}
7
8  {% include 'dashtopcrypto.html'%}
9
10
11  <div class="container-fluid">
12    <div class="row mt-5 mb-5 justify-content-center">
13
14      <div class="col-sm-3 col-md-3 col-xs-12 py-5">
15        <div class="card">
16          <div class="card-header"> Crypto Spot Quote</div>
17          <div class="card-body">
18            <form method="POST" action="">{% csrf_token %}
19              <div class="input-group mb-3">
20                <div class="input-group-append">
21                  <span class="input-group-text"><i class="fas fa-user"></i></span>
22                </div>
23                <input type="text" name="symbol" placeholder="Symbol..." class="form-control1">
24              </div>
25            </div>
26          </div>
27        </div>
28      </div>
29    </div>
30  </div>
```

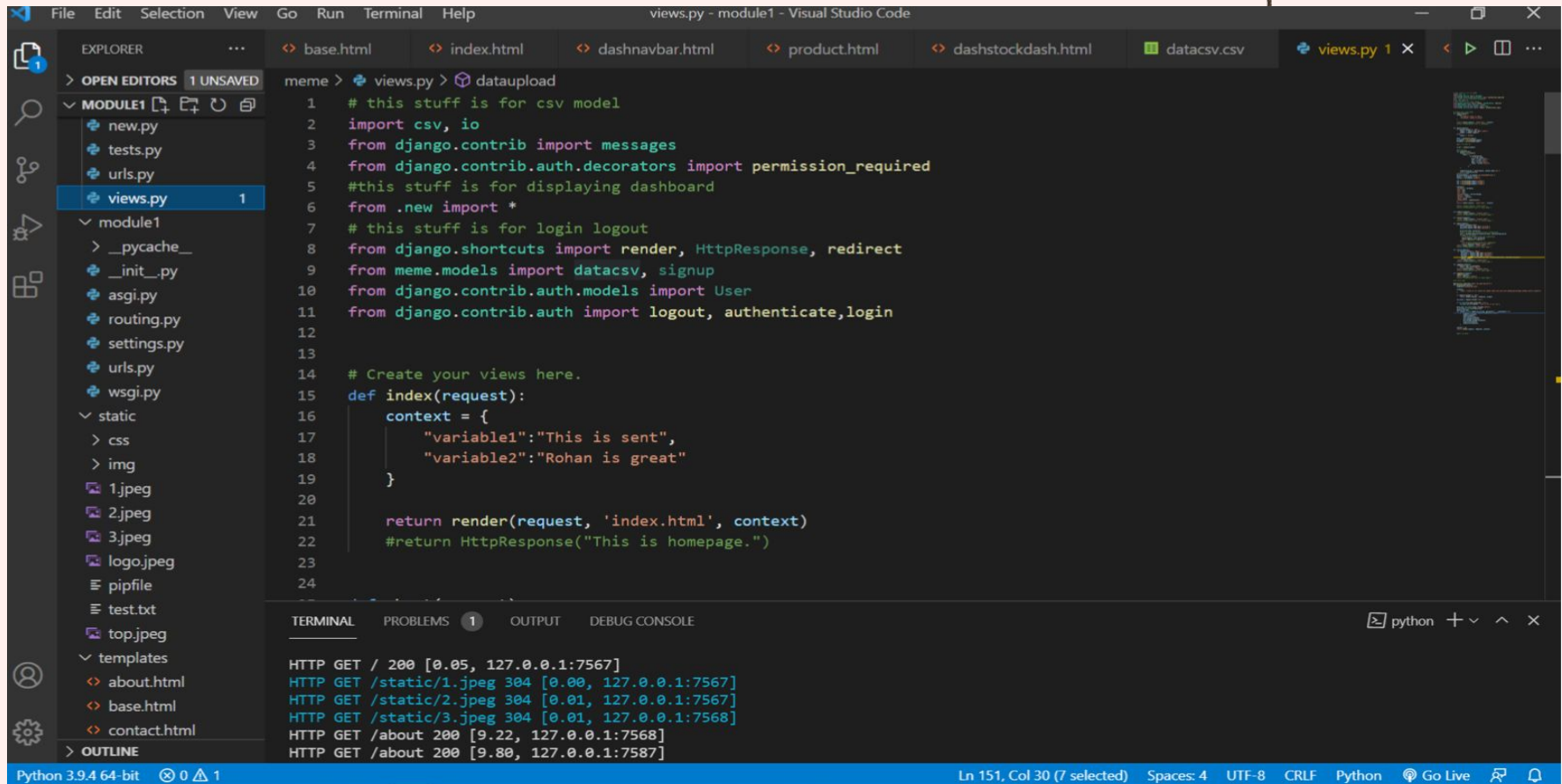
TERMINAL

```
HTTP GET / 200 [0.05, 127.0.0.1:7567]
HTTP GET /static/1.jpeg 304 [0.00, 127.0.0.1:7567]
HTTP GET /static/2.jpeg 304 [0.01, 127.0.0.1:7567]
HTTP GET /static/3.jpeg 304 [0.01, 127.0.0.1:7568]
HTTP GET /about 200 [9.22, 127.0.0.1:7568]
HTTP GET /about 200 [9.80, 127.0.0.1:7587]
```

Python 3.9.4 64-bit 0 1

Ln 5, Col 1 Spaces: 2 UTF-8 CRLF Django HTML Go Live

Code:



The image shows a Visual Studio Code editor window titled "views.py - module1 - Visual Studio Code". The Explorer sidebar on the left shows a project structure with "MODULE1" containing files like new.py, tests.py, urls.py, and views.py (which is selected and has a "1" next to it). Below "module1" are files like __pycache__, __init__.py, asgi.py, routing.py, settings.py, urls.py, wsgi.py, and a "static" directory with subdirectories "css" and "img", and files like 1.jpeg, 2.jpeg, 3.jpeg, logo.jpeg, pipfile, test.txt, top.jpeg, and a "templates" directory with files like about.html, base.html, and contact.html. The main editor area shows the content of views.py, which includes imports for csv, io, django.contrib.messages, django.contrib.auth.decorators, permission_required, and .new, and functions for index(request) and dataupload. The index function returns a render call for 'index.html' with a context dictionary containing 'variable1' and 'variable2'. The terminal at the bottom shows HTTP GET requests and responses, including a 200 status for the root and 304 status for static files.

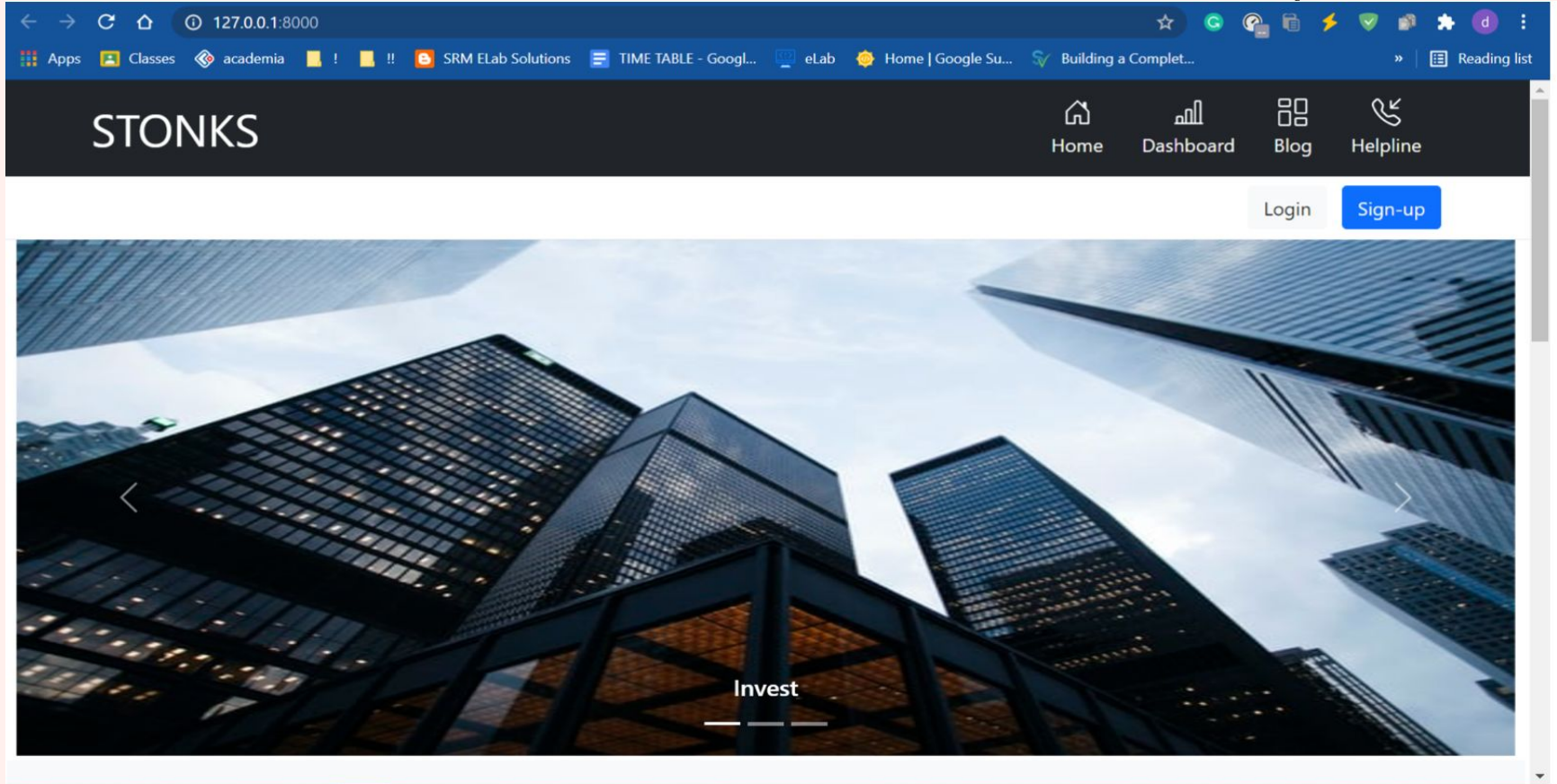
```
meme > views.py > dataupload
1 # this stuff is for csv model
2 import csv, io
3 from django.contrib import messages
4 from django.contrib.auth.decorators import permission_required
5 #this stuff is for displaying dashboard
6 from .new import *
7 # this stuff is for login logout
8 from django.shortcuts import render, HttpResponseRedirect, redirect
9 from meme.models import datacsv, signup
10 from django.contrib.auth.models import User
11 from django.contrib.auth import logout, authenticate, login
12
13
14 # Create your views here.
15 def index(request):
16     context = {
17         "variable1": "This is sent",
18         "variable2": "Rohan is great"
19     }
20
21     return render(request, 'index.html', context)
22     #return HttpResponseRedirect("This is homepage.")
23
24
```

TERMINAL

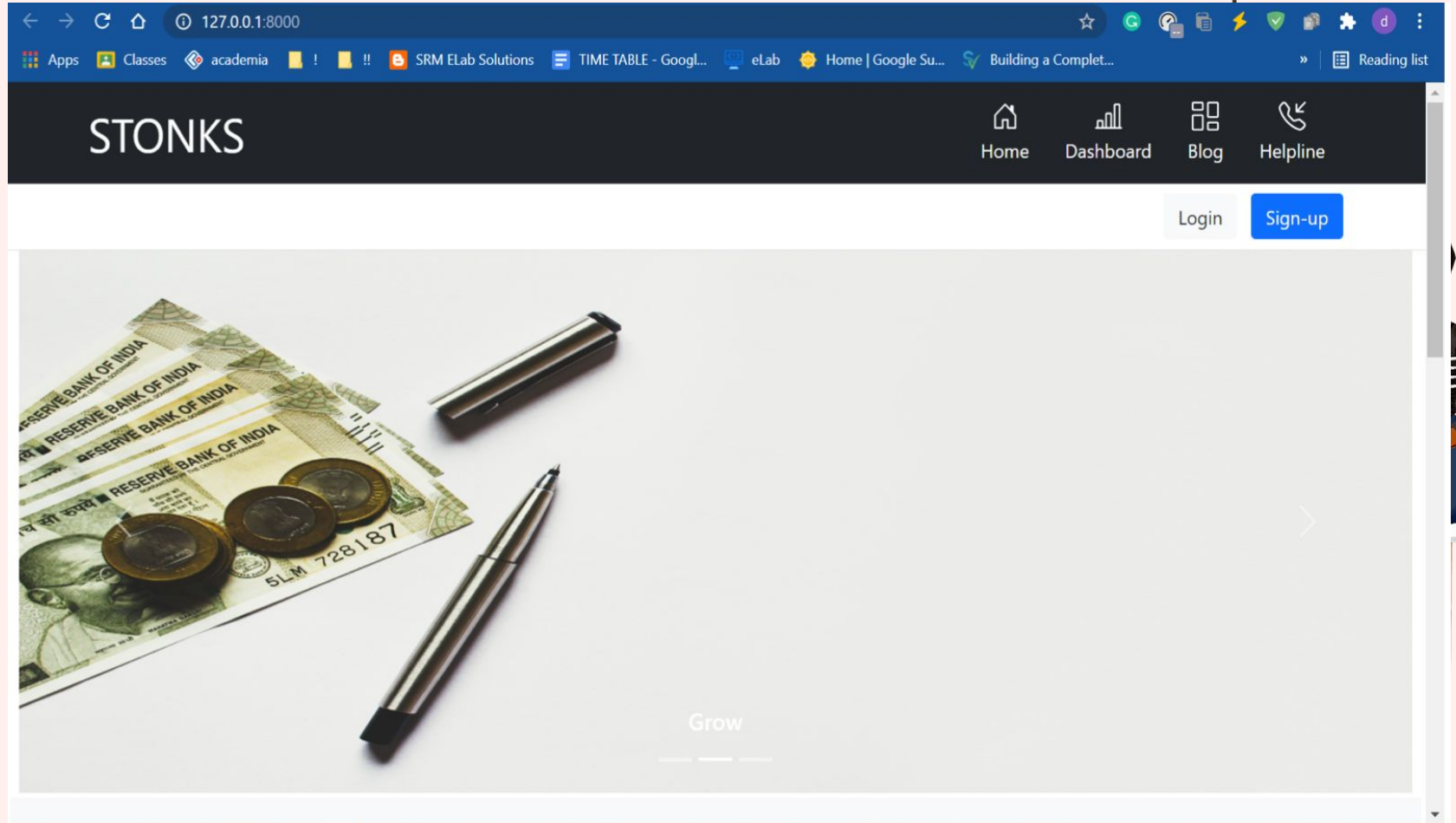
```
HTTP GET / 200 [0.05, 127.0.0.1:7567]
HTTP GET /static/1.jpeg 304 [0.00, 127.0.0.1:7567]
HTTP GET /static/2.jpeg 304 [0.01, 127.0.0.1:7567]
HTTP GET /static/3.jpeg 304 [0.01, 127.0.0.1:7568]
HTTP GET /about 200 [9.22, 127.0.0.1:7568]
HTTP GET /about 200 [9.80, 127.0.0.1:7587]
```

Python 3.9.4 64-bit | 0 | 1 | Ln 151, Col 30 (7 selected) | Spaces: 4 | UTF-8 | CRLF | Python | Go Live

Frontend



Frontend



Frontend

STONKS



Home



Dashboard



Blog



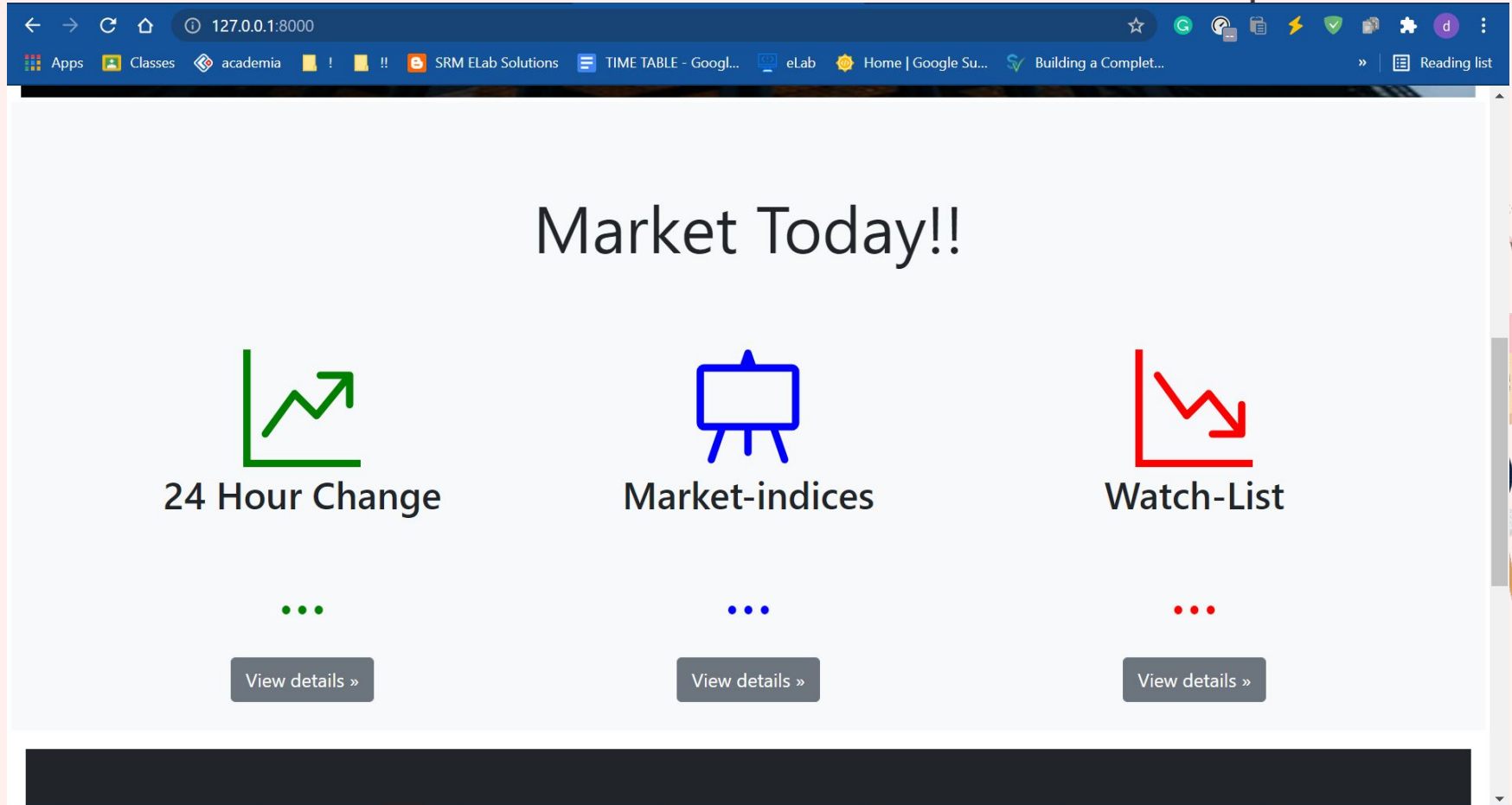
Helpline

Login

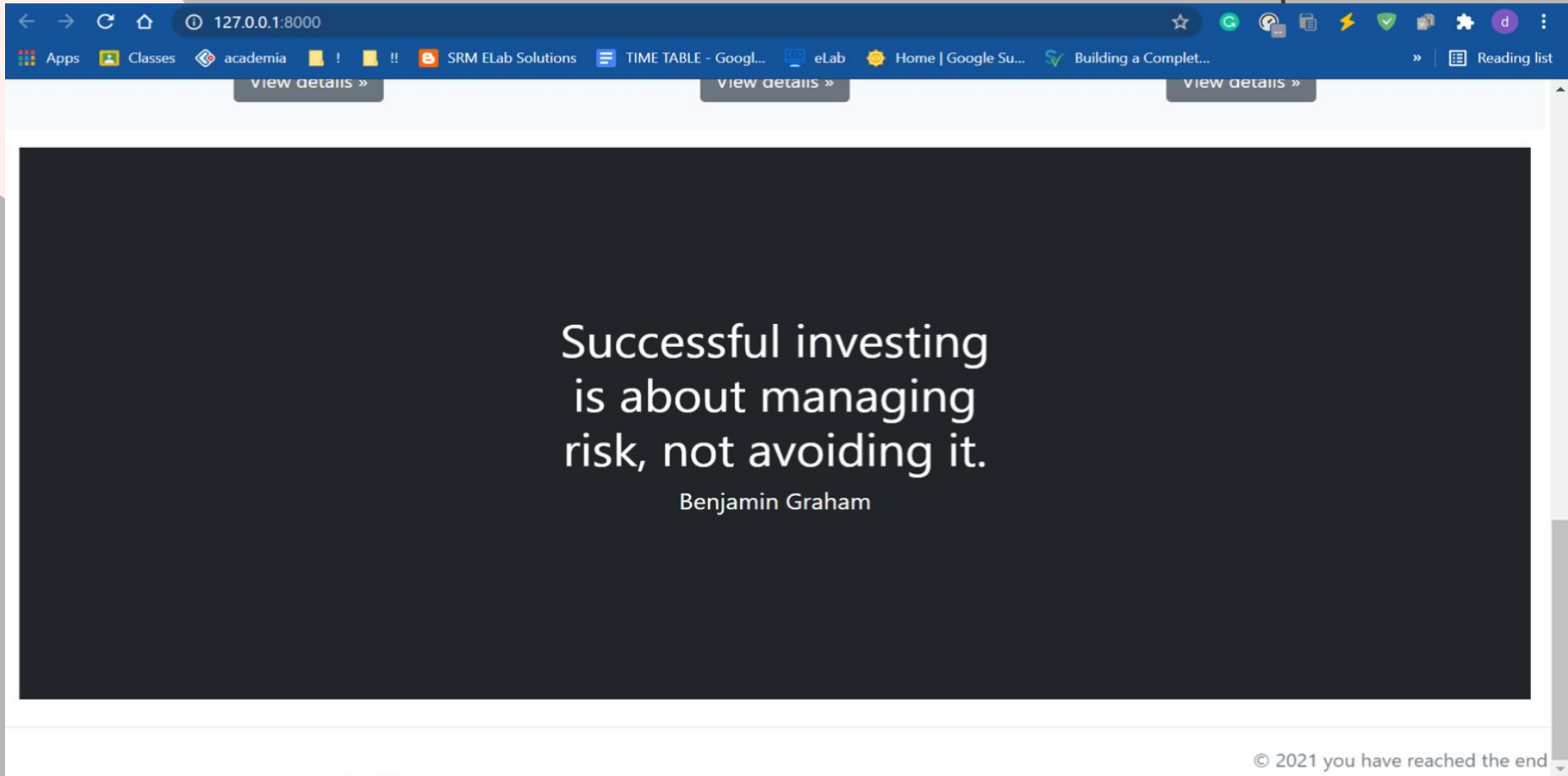
Sign-up



Frontend



Frontend



Frontend

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/login'. The browser's tab bar contains several tabs, including 'Apps', 'Classes', 'academia', 'SRM ELab Solutions', 'TIME TABLE - Googl...', 'eLab', 'Home | Google Su...', and 'Building a Complet...'. The page header is dark blue with the 'STONKS' logo on the left and navigation links 'Home', 'Dashboard', 'Blog', and 'Helpline' on the right. The main content area has a white background. On the left, the text 'Log-in for awesomeness!!' is displayed in a large, bold, black font, with the tagline 'Investments made easy.' below it. On the right, there is a light gray login form. The form contains two input fields: 'User-name' with the value 'dipali' and 'Password' with masked characters '*****'. Below these fields is a blue 'Log-in' button. At the bottom of the form, a line of text reads: 'By clicking Log-in, you agree to the terms of use.'

127.0.0.1:8000/login

Apps Classes academia SRM ELab Solutions TIME TABLE - Googl... eLab Home | Google Su... Building a Complet... Reading list

STONKS

Home Dashboard Blog Helpline

Log-in for awesomeness!!

Investments made easy.

User-name
dipali

Password

Log-in

By clicking Log-in, you agree to the terms of use.



Sign-up for awesomeness!!

Investments made easy.

Username

dipali

Password

.....

☐ Remember me

Sign-up

By clicking Sign-up, you agree to the terms of use.

Implementation of Module 2

Code of Module 2

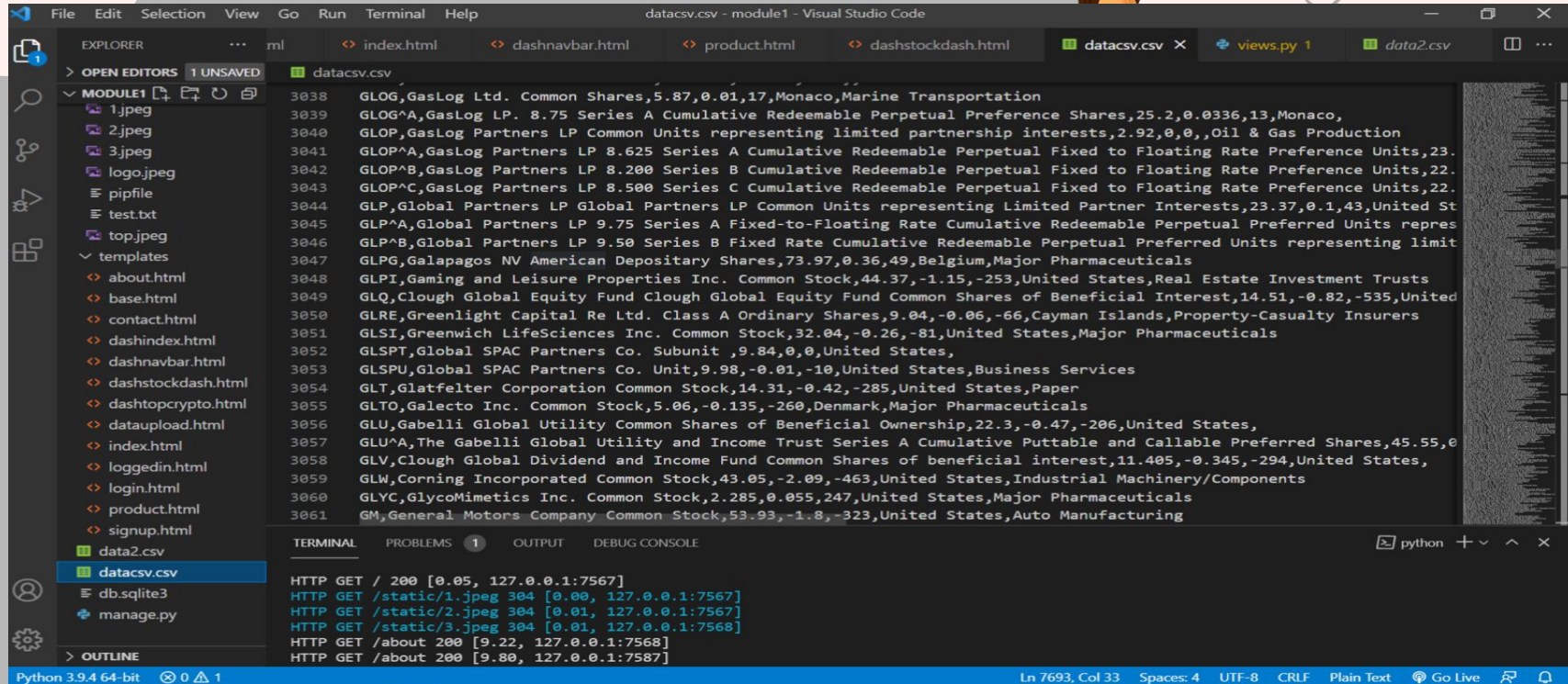
The code and the data for back-end can be found in the following link:

<https://drive.google.com/drive/u/0/folders/1d114nrIDAKcCiGxYD5l6gC7bBieDlIsB>

The data of around 9000 companies in stock-market:



The Data of around 7000 companies in Stock Market



The screenshot displays the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with files like `1.jpeg`, `2.jpeg`, `3.jpeg`, `logo.jpeg`, `pipfile`, `test.txt`, `top.jpeg`, and a `templates` folder. The main editor area is open to `datacsv.csv`, which contains a list of company names and their stock data, such as `GLOG, GasLog Ltd. Common Shares, 5.87, 0.01, 17, Monaco, Marine Transportation`. The bottom panel features a terminal window showing a series of HTTP GET requests and responses, including `HTTP GET / 200 [0.05, 127.0.0.1:7567]` and `HTTP GET /static/1.jpeg 304 [0.00, 127.0.0.1:7567]`. The status bar at the bottom indicates the Python 3.9.4 64-bit environment.

```
File Edit Selection View Go Run Terminal Help
datacsv.csv - module1 - Visual Studio Code

EXPLORER
ml
index.html
dashnavbar.html
product.html
dashstockdash.html
datacsv.csv
views.py
data2.csv

OPEN EDITORS 1 UNSAVED
MODULE1
1.jpeg
2.jpeg
3.jpeg
logo.jpeg
pipfile
test.txt
top.jpeg
templates
about.html
base.html
contact.html
dashindex.html
dashnavbar.html
dashstockdash.html
dashstockdash.html
dataupload.html
index.html
loggedin.html
login.html
product.html
signup.html
data2.csv
datacsv.csv
db.sqlite3
manage.py

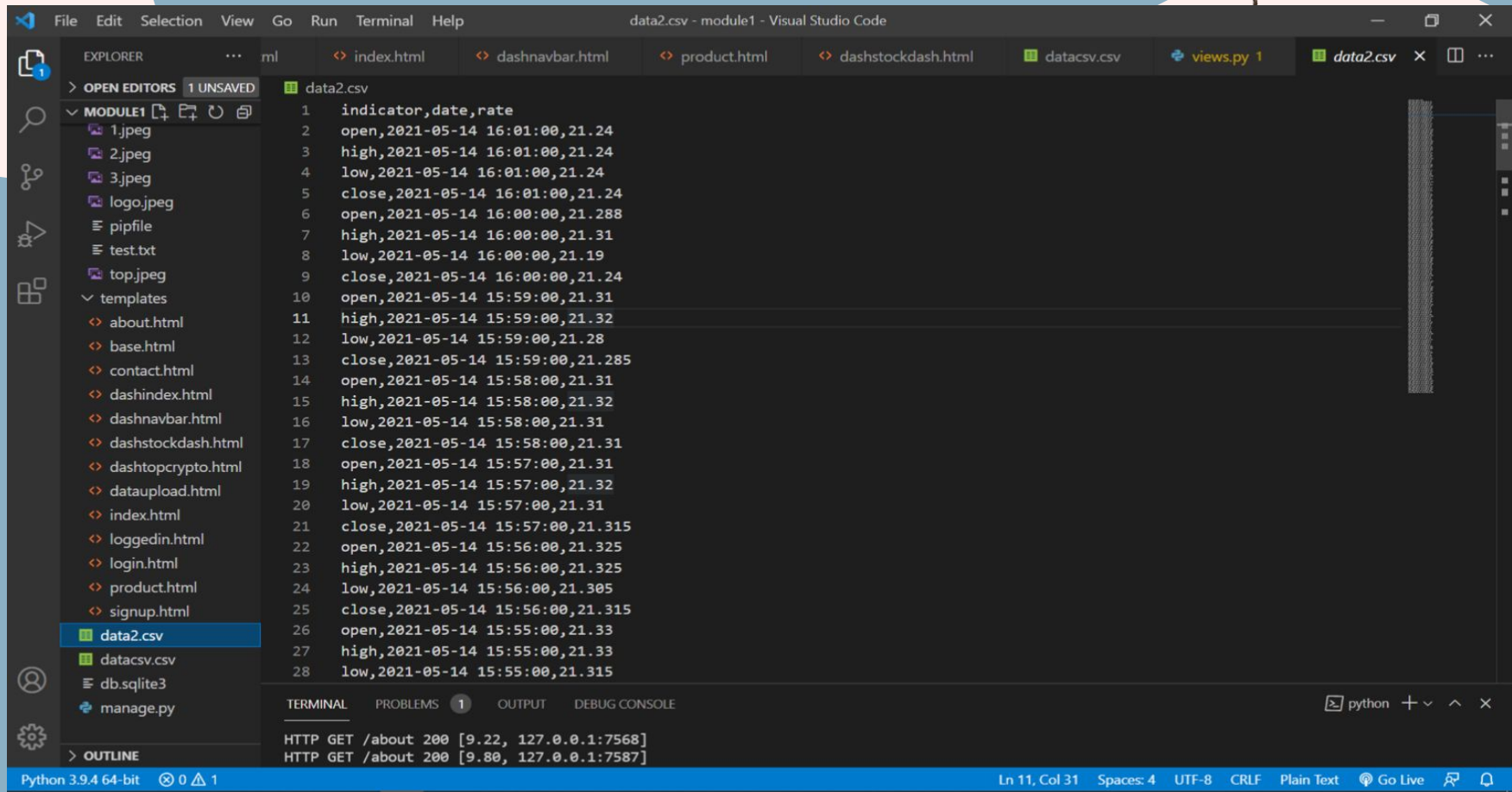
OUTLINE

3038 GLOG, GasLog Ltd. Common Shares, 5.87, 0.01, 17, Monaco, Marine Transportation
3039 GLOG^A, GasLog LP. 8.75 Series A Cumulative Redeemable Perpetual Preference Shares, 25.2, 0.0336, 13, Monaco,
3040 GLOP, GasLog Partners LP Common Units representing limited partnership interests, 2.92, 0.0, Oil & Gas Production
3041 GLOP^A, GasLog Partners LP 8.625 Series A Cumulative Redeemable Perpetual Fixed to Floating Rate Preference Units, 23.
3042 GLOP^B, GasLog Partners LP 8.200 Series B Cumulative Redeemable Perpetual Fixed to Floating Rate Preference Units, 22.
3043 GLOP^C, GasLog Partners LP 8.500 Series C Cumulative Redeemable Perpetual Fixed to Floating Rate Preference Units, 22.
3044 GLP, Global Partners LP Global Partners LP Common Units representing limited partnership interests, 23.37, 0.1, 43, United St
3045 GLP^A, Global Partners LP 9.75 Series A Fixed-to-Floating Rate Cumulative Redeemable Perpetual Preferred Units repres
3046 GLP^B, Global Partners LP 9.50 Series B Fixed Rate Cumulative Redeemable Perpetual Preferred Units representing limit
3047 GLPG, Galapagos NV American Depositary Shares, 73.97, 0.36, 49, Belgium, Major Pharmaceuticals
3048 GLPI, Gaming and Leisure Properties Inc. Common Stock, 44.37, -1.15, -253, United States, Real Estate Investment Trusts
3049 GLQ, Clough Global Equity Fund Clough Global Equity Fund Common Shares of Beneficial Interest, 14.51, -0.82, -535, United
3050 GLRE, Greenlight Capital Re Ltd. Class A Ordinary Shares, 9.04, -0.06, -66, Cayman Islands, Property-Casualty Insurers
3051 GLSI, Greenwich LifeSciences Inc. Common Stock, 32.04, -0.26, -81, United States, Major Pharmaceuticals
3052 GLSPT, Global SPAC Partners Co. Subunit, 9.84, 0.0, United States,
3053 GLSPU, Global SPAC Partners Co. Unit, 9.98, -0.01, -10, United States, Business Services
3054 GLT, Glatfelter Corporation Common Stock, 14.31, -0.42, -285, United States, Paper
3055 GLTO, Galecto Inc. Common Stock, 5.06, -0.135, -260, Denmark, Major Pharmaceuticals
3056 GLU, Gabelli Global Utility Common Shares of Beneficial Ownership, 22.3, -0.47, -206, United States,
3057 GLU^A, The Gabelli Global Utility and Income Trust Series A Cumulative Puttable and Callable Preferred Shares, 45.55, 0
3058 GLV, Clough Global Dividend and Income Fund Common Shares of Beneficial Interest, 11.405, -0.345, -294, United States,
3059 GLW, Corning Incorporated Common Stock, 43.05, -2.09, -463, United States, Industrial Machinery/Components
3060 GLYC, GlycoMimetics Inc. Common Stock, 2.285, 0.055, 247, United States, Major Pharmaceuticals
3061 GM, General Motors Company Common Stock, 53.93, -1.8, -323, United States, Auto Manufacturing

TERMINAL
python
[0.05, 127.0.0.1:7567]
HTTP GET / 200 [0.05, 127.0.0.1:7567]
HTTP GET /static/1.jpeg 304 [0.00, 127.0.0.1:7567]
HTTP GET /static/2.jpeg 304 [0.01, 127.0.0.1:7567]
HTTP GET /static/3.jpeg 304 [0.01, 127.0.0.1:7568]
HTTP GET /about 200 [9.22, 127.0.0.1:7568]
HTTP GET /about 200 [9.80, 127.0.0.1:7587]

Python 3.9.4 64-bit 0 1 Ln 7693, Col 33 Spaces: 4 UTF-8 CRLF Plain Text Go Live
```

The Live data of highs and lows of each stock:



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'MODULE1' with various files including images, text files, and HTML files. The code editor displays the contents of 'data2.csv', which contains stock market data. The data is organized into columns for indicator, date, time, and rate. The terminal at the bottom shows HTTP GET requests and responses.

```
data2.csv
1 indicator,date,rate
2 open,2021-05-14 16:01:00,21.24
3 high,2021-05-14 16:01:00,21.24
4 low,2021-05-14 16:01:00,21.24
5 close,2021-05-14 16:01:00,21.24
6 open,2021-05-14 16:00:00,21.288
7 high,2021-05-14 16:00:00,21.31
8 low,2021-05-14 16:00:00,21.19
9 close,2021-05-14 16:00:00,21.24
10 open,2021-05-14 15:59:00,21.31
11 high,2021-05-14 15:59:00,21.32
12 low,2021-05-14 15:59:00,21.28
13 close,2021-05-14 15:59:00,21.285
14 open,2021-05-14 15:58:00,21.31
15 high,2021-05-14 15:58:00,21.32
16 low,2021-05-14 15:58:00,21.31
17 close,2021-05-14 15:58:00,21.31
18 open,2021-05-14 15:57:00,21.31
19 high,2021-05-14 15:57:00,21.32
20 low,2021-05-14 15:57:00,21.31
21 close,2021-05-14 15:57:00,21.315
22 open,2021-05-14 15:56:00,21.325
23 high,2021-05-14 15:56:00,21.325
24 low,2021-05-14 15:56:00,21.305
25 close,2021-05-14 15:56:00,21.315
26 open,2021-05-14 15:55:00,21.33
27 high,2021-05-14 15:55:00,21.33
28 low,2021-05-14 15:55:00,21.315
```

TERMINAL

```
HTTP GET /about 200 [9.22, 127.0.0.1:7568]
HTTP GET /about 200 [9.80, 127.0.0.1:7587]
```

Python 3.9.4 64-bit 0 1 Ln 11, Col 31 Spaces: 4 UTF-8 CRLF Plain Text Go Live

Code for backend -

The screenshot displays the Visual Studio Code interface with a Django project named 'models.py' open. The Explorer sidebar on the left shows the project structure, including 'module1' and its subdirectories 'migrations', 'static', and 'media'. The 'models.py' file is selected in the Explorer. The main editor area shows the code for 'models.py', which includes a 'signup' model and a 'datacsv' model. The 'signup' model has fields for 'username', 'password', and 'checkbox'. The 'datacsv' model has fields for 'symbol', 'name', 'last_sale', 'net_change', 'percentage_change', 'country', and 'industry'. The bottom status bar shows the Python version (3.9.4 64-bit) and the current file (models.py).

```
meme > models.py > signup
1  from django.db import models
2
3  # Create your models here.
4  class signup(models.Model):
5      username = models.CharField(max_length=150)
6      password = models.CharField(max_length=50)
7      checkbox = models.BooleanField()
8
9
10 class datacsv(models.Model):
11     symbol=models.CharField(max_length=100)
12     name=models.CharField(max_length=100)
13     last_sale=models.FloatField()
14     net_change=models.FloatField()
15     percentage_change=models.FloatField()
16     country=models.CharField(max_length=1000)
17     industry=models.CharField(max_length=1000)
18
```

Python 3.9.4 64-bit 0 1

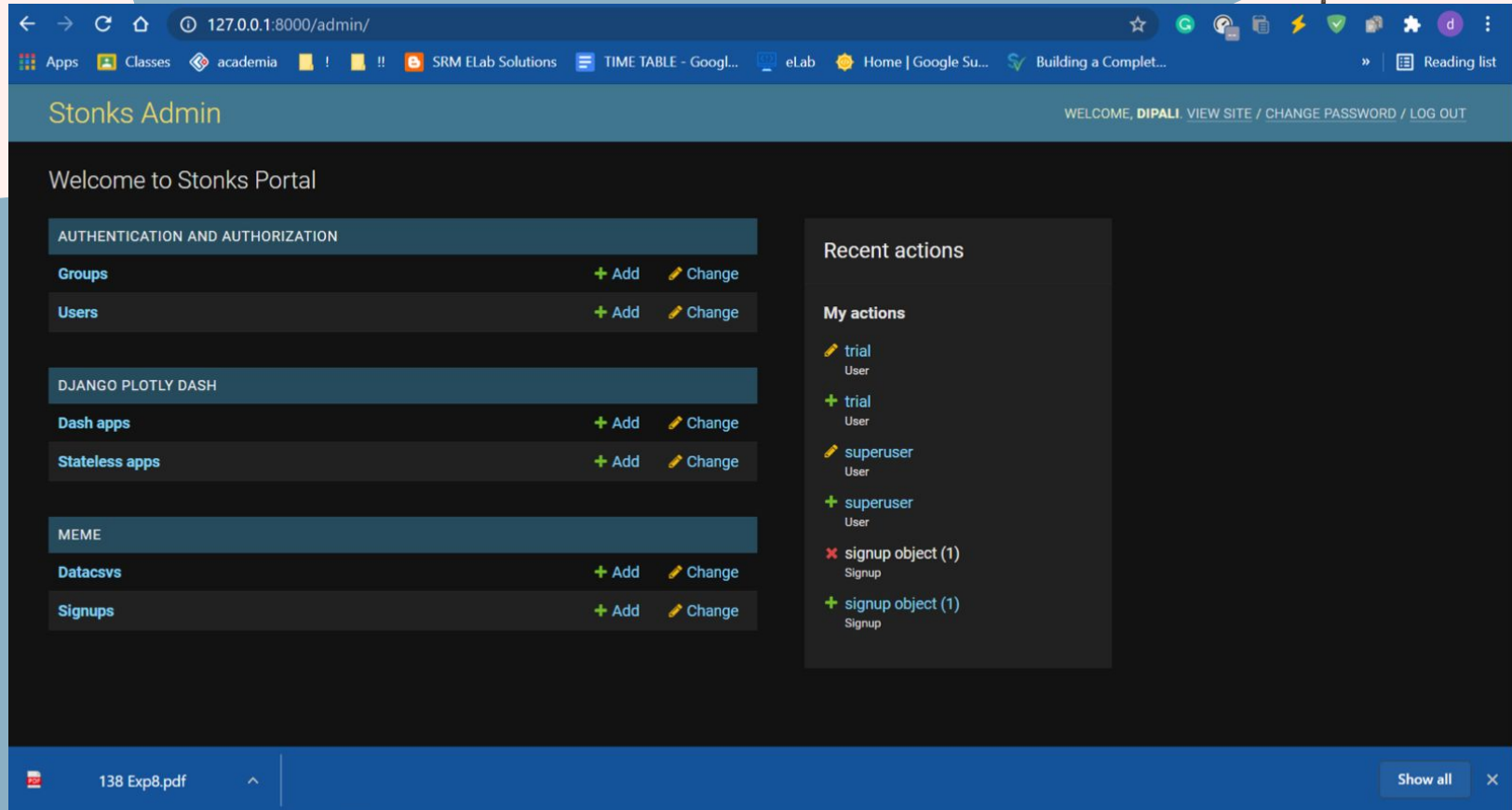
Code for backend -

```
34 moredata = pricechange(symbol)
35
36 #get a fricken df
37
38 ts_df = candles(symbol)
39
40 #PlotlyGraph
41 def candlestick():
42     figure = go.Figure(
43         data = [
44             go.Candlestick(
45                 x = ts_df.index,
46                 high = ts_df['high'],
47                 low = ts_df['low'],
48                 open = ts_df['open'],
49                 close = ts_df['close'],
50             )
51         ]
52     )
53
54     candlestick_div = plot.figure, output_type='div')
55     return candlestick_div
56
57 #endPlotlyGraph
58 percentchange = pricedata['priceChangePercent']
59 buyers = pricedata['askQty']
60 sellers = pricedata['bidQty']
61
62 eth = pricechange(symbol='ETHUSD')
```

python 3.9.4 64-bit

Ln 151, Col 30 (7 selected) Spaces: 4 UTF-8 CRLF Python Go Live

Result of Module 2



The screenshot shows a web browser window displaying the 'Stonks Admin' dashboard. The browser's address bar shows the URL '127.0.0.1:8000/admin/'. The dashboard has a dark blue header with the title 'Stonks Admin' and a welcome message 'WELCOME, DIPALI' with links for 'VIEW SITE', 'CHANGE PASSWORD', and 'LOG OUT'. The main content area is divided into two columns. The left column contains three sections: 'AUTHENTICATION AND AUTHORIZATION' with links for 'Groups' and 'Users' (each with '+ Add' and 'Change' buttons); 'DJANGO PLOTLY DASH' with links for 'Dash apps' and 'Stateless apps' (each with '+ Add' and 'Change' buttons); and 'MEME' with links for 'Datacsys' and 'Signups' (each with '+ Add' and 'Change' buttons). The right column contains a 'Recent actions' section with a list of actions: 'trial User' (with a pencil icon), '+ trial User' (with a plus icon), 'superuser User' (with a pencil icon), '+ superuser User' (with a plus icon), 'signup object (1) Signup' (with a red 'x' icon), and '+ signup object (1) Signup' (with a plus icon). At the bottom of the browser window, a taskbar shows a PDF file named '138 Exp8.pdf' and a 'Show all' button.

127.0.0.1:8000/admin/

Apps Classes academia SRM Elab Solutions TIME TABLE - Googl... eLab Home | Google Su... Building a Complet... Reading list

Stonks Admin

WELCOME, DIPALI [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Welcome to Stonks Portal

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change
Users	+ Add	Change

DJANGO PLOTLY DASH

Dash apps	+ Add	Change
Stateless apps	+ Add	Change

MEME

Datacsys	+ Add	Change
Signups	+ Add	Change

Recent actions

My actions

- [trial](#)
User
- [+ trial](#)
User
- [superuser](#)
User
- [+ superuser](#)
User
- [✗ signup object \(1\)](#)
Signup
- [+ signup object \(1\)](#)
Signup

138 Exp8.pdf ^

Show all ×

Result of Module 2

The screenshot displays the Stonks Admin web application. The browser address bar shows the URL `127.0.0.1:8000/admin/auth/user/`. The page header includes the title "Stonks Admin" and a welcome message for "DIPALI" with links for "VIEW SITE", "CHANGE PASSWORD", and "LOG OUT". The breadcrumb trail indicates the current location: "Home > Authentication and Authorization > Users".

The left sidebar contains a navigation menu with the following sections:

- AUTHENTICATION AND AUTHORIZATION**
 - Groups (+ Add)
 - Users (+ Add)
- DJANGO PLOTLY DASH**
 - Dash apps (+ Add)
 - Stateless apps (+ Add)
- MEME**
 - Datacsvs (+ Add)
 - Signups (+ Add)

The main content area is titled "Select user to change" and features a search bar and a table of users. The table has columns for USERNAME, EMAIL ADDRESS, FIRST NAME, LAST NAME, and STAFF STATUS. Three users are listed: "dipali" (staff status Yes), "superuser" (staff status No), and "trial" (staff status No). The "Action" dropdown is set to "Go", and the text "0 of 3 selected" is displayed. A "3 users" summary is shown at the bottom of the table.

On the right side, there is a "FILTER" panel with the following options:

- By staff status**
 - All
 - Yes
 - No
- By superuser status**
 - All
 - Yes
 - No
- By active**
 - All
 - Yes
 - No

The bottom of the interface shows a file explorer with "138 Exp8.pdf" and a "Show all" button.

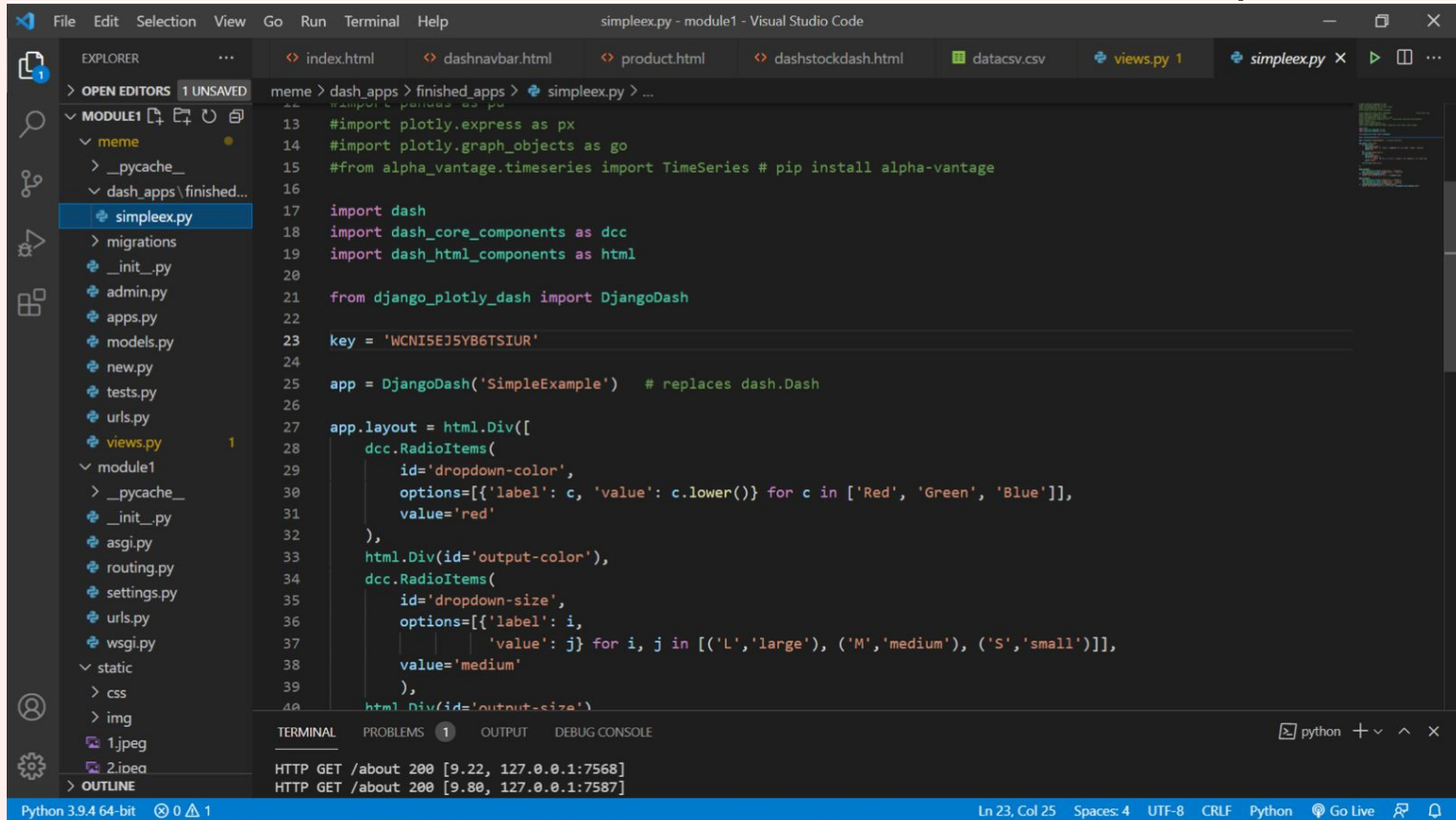
Implementation of Module 3

Code of Module 3:

Module 3 (Live API parsing to show live graphs) was implemented using the following code: Link for detailed code:
<https://drive.google.com/drive/u/0/folders/1d114nrIDAKcCiGxYD5l6gC7bBieDlIsB>



Code -

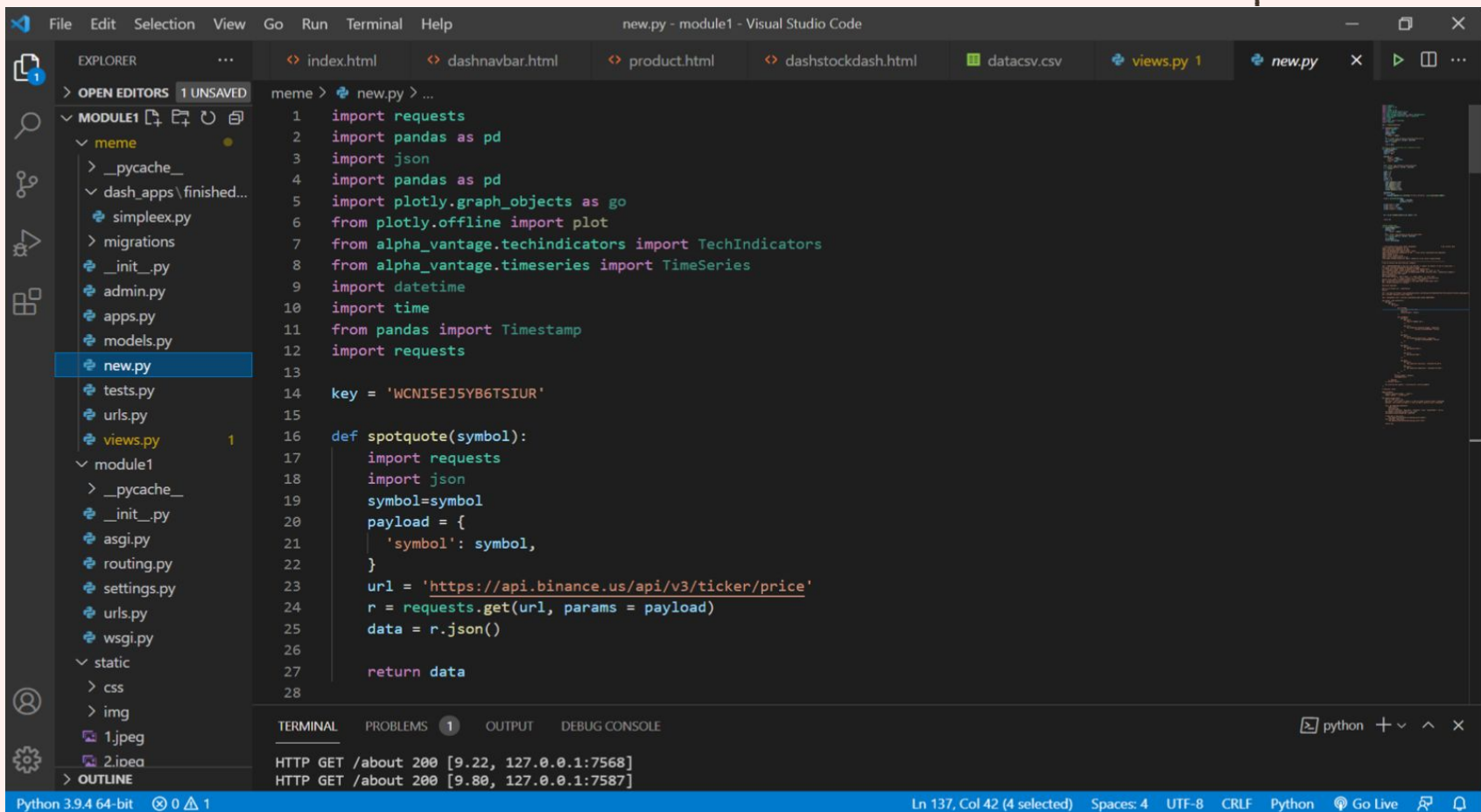


```
13 #import plotly.express as px
14 #import plotly.graph_objects as go
15 #from alpha_vantage.timeseries import TimeSeries # pip install alpha-vantage
16
17 import dash
18 import dash_core_components as dcc
19 import dash_html_components as html
20
21 from django_plotly_dash import DjangoDash
22
23 key = 'WCNI5EJ5YB6TSIUR'
24
25 app = DjangoDash('SimpleExample') # replaces dash.Dash
26
27 app.layout = html.Div([
28     dcc.RadioItems(
29         id='dropdown-color',
30         options=[{'label': c, 'value': c.lower()} for c in ['Red', 'Green', 'Blue']],
31         value='red'
32     ),
33     html.Div(id='output-color'),
34     dcc.RadioItems(
35         id='dropdown-size',
36         options=[{'label': i,
37                   'value': j} for i, j in [('L','large'), ('M','medium'), ('S','small')]],
38         value='medium'
39     ),
40     html.Div(id='output-size')
```

Python 3.9.4 64-bit 0 1

Ln 23, Col 25 Spaces: 4 UTF-8 CRLF Python Go Live

Code -

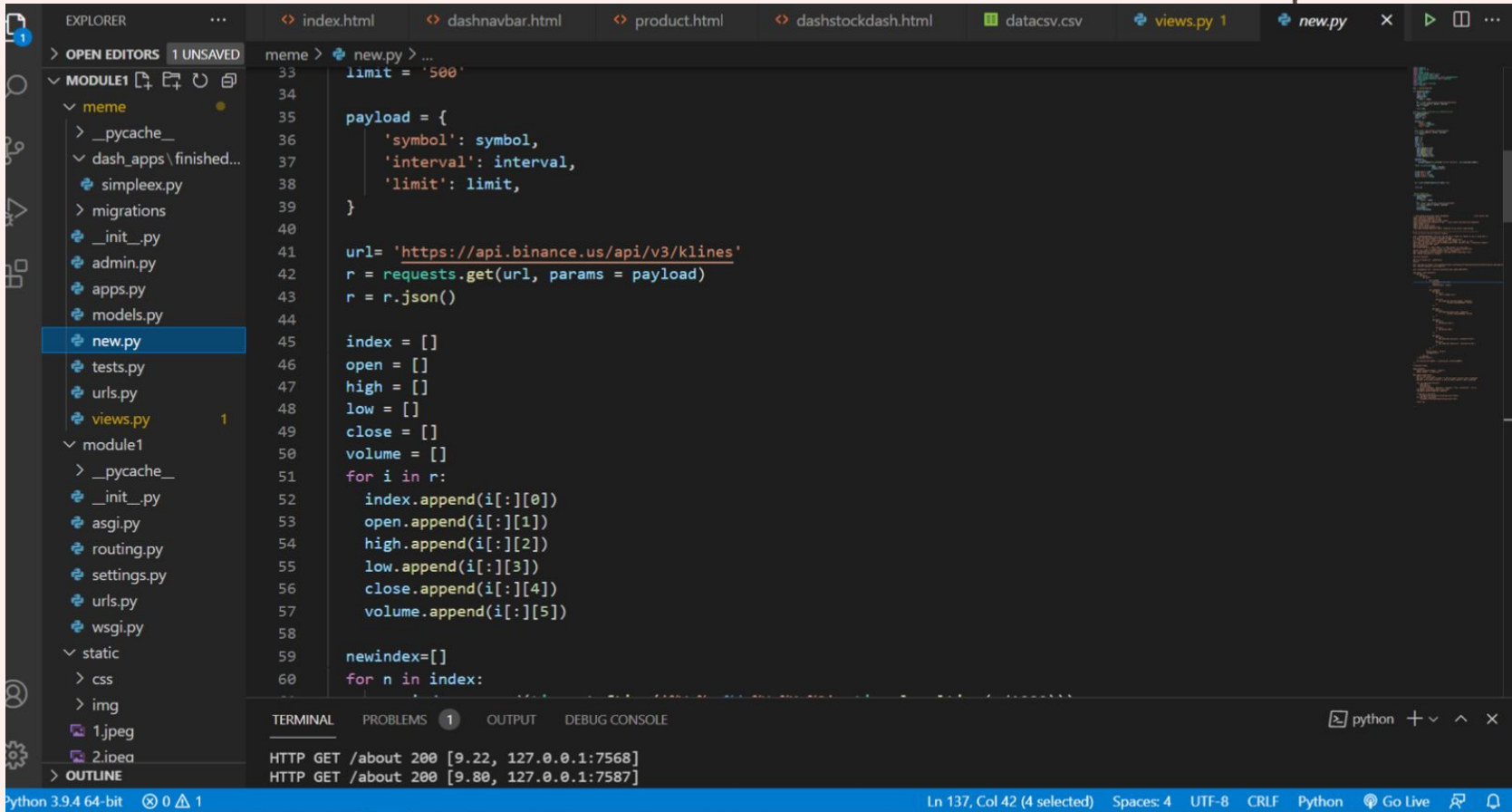


The screenshot shows the Visual Studio Code interface with a Python file named `new.py` open. The file is part of a Dash application project structure. The code in `new.py` is as follows:

```
meme > new.py > ...
1  import requests
2  import pandas as pd
3  import json
4  import pandas as pd
5  import plotly.graph_objects as go
6  from plotly.offline import plot
7  from alpha_vantage.techindicators import TechIndicators
8  from alpha_vantage.timeseries import TimeSeries
9  import datetime
10 import time
11 from pandas import Timestamp
12 import requests
13
14 key = 'WCNI5EJ5YB6TSIUR'
15
16 def spotquote(symbol):
17     import requests
18     import json
19     symbol=symbol
20     payload = {
21         'symbol': symbol,
22     }
23     url = 'https://api.binance.us/api/v3/ticker/price'
24     r = requests.get(url, params = payload)
25     data = r.json()
26
27     return data
28
```

The interface also shows the Explorer sidebar with the project structure, the Terminal panel at the bottom, and the status bar at the bottom indicating Python 3.9.4 64-bit.

Code -

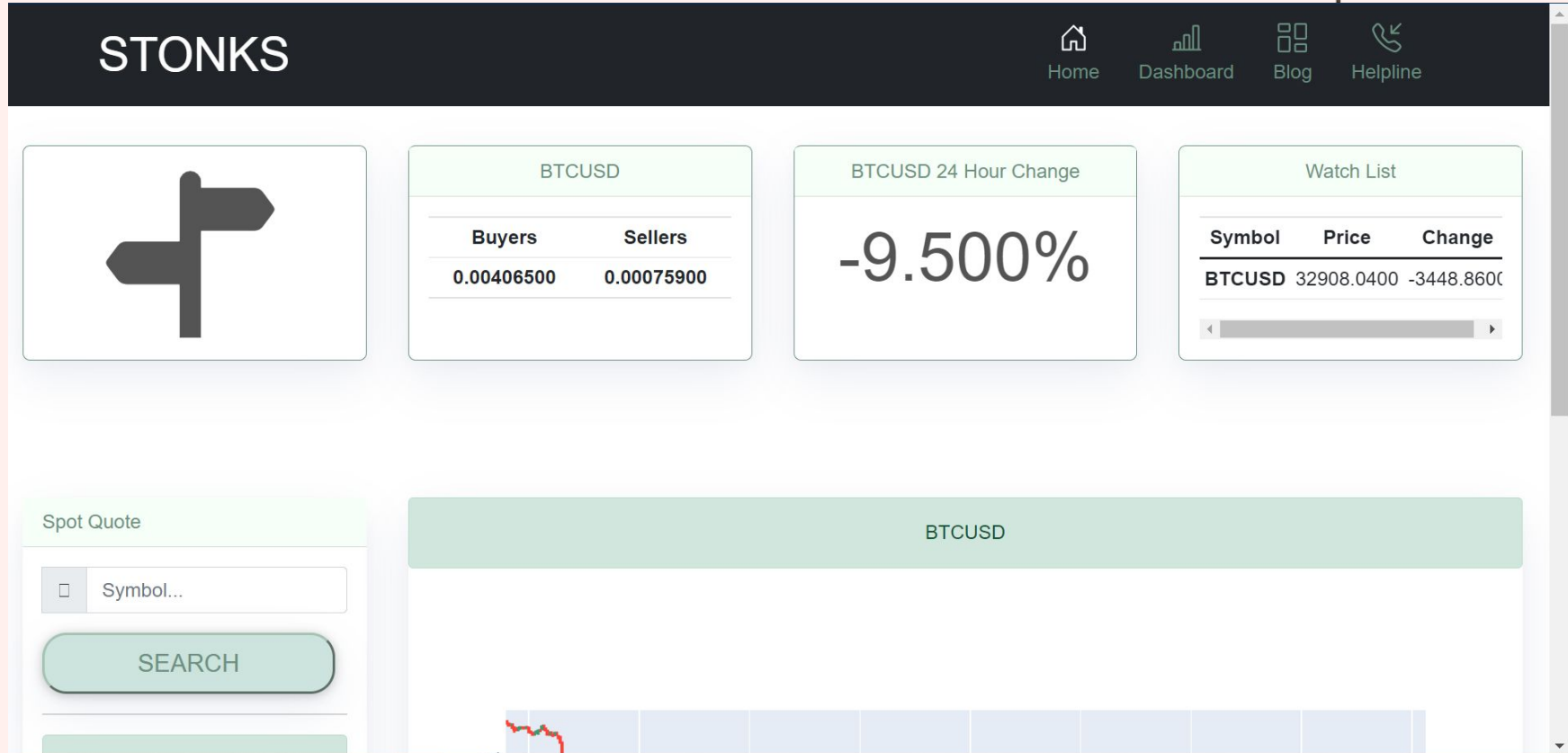


```
33 limit = '500'
34
35 payload = {
36     'symbol': symbol,
37     'interval': interval,
38     'limit': limit,
39 }
40
41 url= 'https://api.binance.us/api/v3/klines'
42 r = requests.get(url, params = payload)
43 r = r.json()
44
45 index = []
46 open = []
47 high = []
48 low = []
49 close = []
50 volume = []
51 for i in r:
52     index.append(i[0])
53     open.append(i[1])
54     high.append(i[2])
55     low.append(i[3])
56     close.append(i[4])
57     volume.append(i[5])
58
59 newindex=[]
60 for n in index:
```

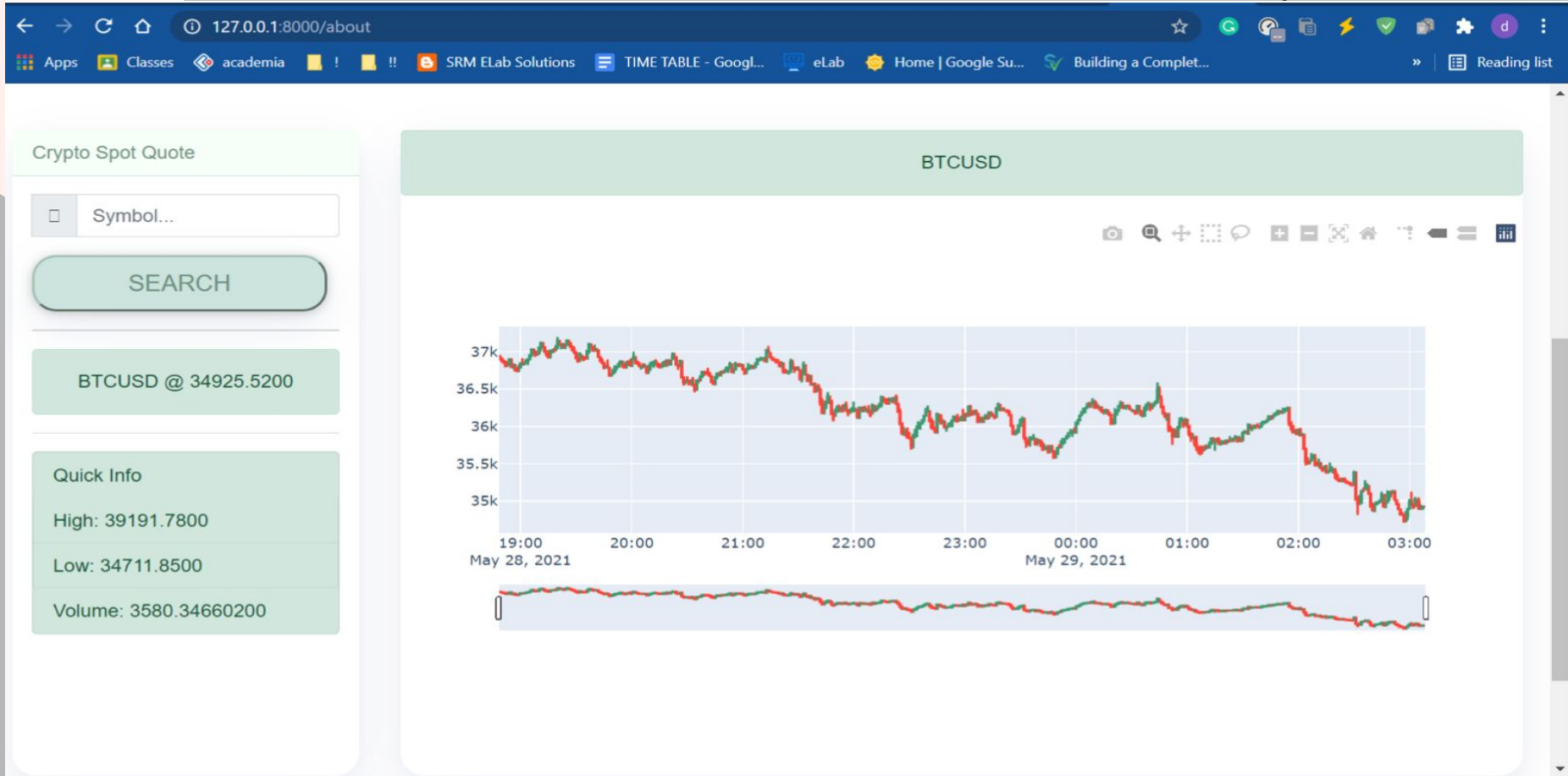
Python 3.9.4 64-bit 0 1

Ln 137, Col 42 (4 selected) Spaces: 4 UTF-8 CRLF Python Go Live

Result of Module 3:



Result of Module 3:



Master Test Plan and Test cases



Scope of Testing

Functional: The following modules need to be tested:

1. Front-end
2. Back-end(migrations)
3. Live API parsing.

Non-Functional: The non-functional modules are:

1. Blog page
2. Helpline page



Types of Testing, Methodology and Tools

Category	Methodology	Tools Required
Functional Requirements	Manual	Django Frame



Functional Test Cases

Test ID#	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome
1.	Verify user registration	Accept Valid username on page#1	<ol style="list-style-type: none">1. User clicks on user registration link2. Enter the username on the box	User should be taken to login page	successful
2.	Verify live data parsi	Parse API using Alpha-vantage API.	View data, and display it into a candlestick graph.	Live data	successful



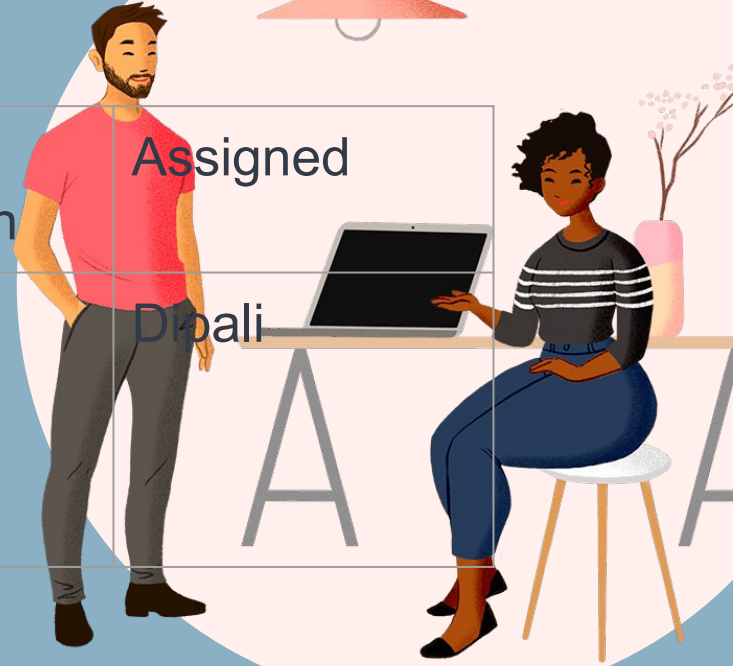
Non-Functional Test cases

Test ID(#)	Test Scenario	Execution Steps	Expected outcome	Test cases	Actual outcome
1	Blog Page	Execution using the following command: -python manage.py runserver	Working blog page	nil	Working blog page
2	Helpline Page	Execution using command: Python manage.py runserver	Working helpline page	nil	Working helpline pag



Defect Log

Requirement #	Defect ID #	Defect Description	Assigned
M1R1	1	Favourites button for signed in users	Dipali



Test Report

Category	Progress Against Plan	Status
Functional Testing	Green	In-Progress
Non-Functional Testing	Green	In-Progress

Functional	Test Case Coverage	Status
Module ID 1	80%	In-Progress
Module ID 2	70%	In-Progress
Module ID 3	85%	In-Progress



Test Report Reference

1. <https://www.pmi.org/>



Thankyou

