

PROJECT REPORT

ns03444

IT4530

Contents

1	Product Description	4
2	Web Scraping	5
2.1	Web Scraping: Stock Performance Data	5
2.1.1	Connecting to the Data	5
2.1.2	Scraping Ticker Symbols.....	6
2.1.3	Scraping the Names and Prices	8
2.1.4	Storing Stock Performance Data	9
2.2	Web Scraping: Bloomberg News Article Data.....	10
2.2.1	Connecting to the Data	11
2.2.2	Scraping the title.....	12
2.2.3	Scraping the description and publication date.....	12
2.2.4	Scraping the hyperlink.....	12
2.2.5	Looping Through the Latest Sector News Items.....	12
2.2.6	Storing the article data.....	13
2.3	Web Scraping: Barron's News Article Data.....	14
2.3.1	Scraping the Title and Hyperlink	15
2.3.2	Scraping the description and publication date.....	15
2.3.3	Looping Through the Latest Sector News Items.....	16
2.3.4	Storing Barron's Article Data	16
2.4	Web Scraping: Google Search Results.....	16
2.4.1	Scraping the Data	17
2.4.2	Storing the data	18
3	Midterm Summary.....	18
4	The Django Framework	19
4.1	Hosting	19
4.2	Creating a Django Project.....	19
4.3	Creating the News App	21
4.3.1	Developing the first View.....	22
4.3.2	Implementing the web scraper function into the view.....	23
4.3.3	Introducing Templates with Bootstrap	25
4.3.4	Using Django Context Processers	29
4.3.5	Creating a Django Model.....	31

4.3.6	Populating the Database.....	37
4.3.7	A different approach to populating the database.....	43
4.4	Designing the News page	47
4.4.1	Formatting the home page	48
4.4.2	Aligning page content with Bootstrap	50
4.4.3	Bootstrap Colors.....	52
4.4.4	Bootstrap Navigation Bars.....	55
4.5	Final Output.....	57
4.6	Creating the Stock App	58
4.6.1	Creating a Django Model.....	58
4.6.2	Populating the Database.....	60
4.6.3	Developing the view.....	62
4.7	Designing the Stock page	67
4.7.1	Table Formatting	70
4.8	Final Output.....	71
5	Data Vizualization	71
5.1	Creating the app	71
5.2	Introducing Bokeh.....	72
5.2.1	Building the Chart	75
5.2.2	Integrating Bokeh & Django.....	81
5.2.3	Building the template.....	83
5.3	Final Output.....	88
5.4	Matplotlib & Pandas	88
5.4.1	Context for the Analysis	88
5.4.2	Collecting the data.....	88
5.4.3	Building the Charts.....	90
5.5	Matplotlib & Yfinance	96
5.5.1	Collecting the data.....	96
5.5.2	Building the charts.....	98
5.6	Integrating Matplotlib & Django	102
5.6.1	Static file configuration	102
5.6.2	Creating the template.....	104
5.7	Final output.....	106

6	Project Deployment	107
6.1	Setting up Github	107
6.2	Github to Python Anywhere	113
6.2.1	Connecting to the Repository.....	115
6.2.2	Creating the web app	116
6.3	Final output.....	122
6.4	Scheduling Tasks	124
6.4.1	Modifying the tasks	124
6.4.2	Scheduling the tasks	127
6.4.3	Testing the Scheduler	128
7	Reflection	134

1 Product Description

This application is a content-aggregator built primarily with Python. It provides users specifically interested in the latest news for the Information Technology sector of the stock market to stay updated by aggregating the most recent new articles from 3 popular websites and displaying them on one webpage. Users will also be able analyze the top 25 stocks/stock prices in the technology industry. It will encompass different types of data visualizations as well as an in-depth financial analysis on 3 largest wireless network providers.

2 Web Scraping

2.1 Web Scraping: Stock Performance Data

The first portion of my project consisted of using Python to scrape each stock's ticker symbol, name, and current price from a table of the top 25 technology stocks listed on:
https://finance.yahoo.com/sector/ms_technology?.tsrc=fin-srch.

yahoo! finance <small>PREMIUM</small>		Be a bull among bears										Advanced tools to navigate volatile markets				Try it free*	
yahoo! finance		Search for news, symbols or companies										Nicholas		2 Mail			
Finance Home	Watchlists	My Portfolio	Screners	Premium 	Markets	News	Personal Finance		Videos		Industries	...	 Premium + Try it free				
 AAPL	Apple Inc.			119.25	-2.17	-1.7%	39.042M	109.948M		1.995T	32.24	55.15		145.09			
 MSFT	Microsoft Corporation			232.82	+1.22	+0.53%	8.787M	29.554M		1.754T	34.67	132.52		246.13			
 TSM	Taiwan Semiconductor Manufacturing Company Limited			118.41	-2.39	-1.9%	4.477M	11.982M		548.552B	52.87	42.70		142.20			
 NVDA	NVIDIA Corporation			492.60	-5.86	-1.18%	2.642M	7.856M		302.734B	70.77	180.68		614.90			
 INTC	Intel Corporation			61.53	+0.78	+1.29%	6.160M	38.095M		248.899B	12.40	43.61		65.11			
 ASML	ASML Holding N.V.			527.05	+0.01	0.00%	250.603K	804.145		212.822B	75.47	191.25		608.71			
 ORCL	Oracle Corporation			71.60	+1.63	+2.33%	7.196M	11.754M		210.616B	21.64	39.71		72.71			
 ADBE	Adobe Inc.			439.75	-1.08	-0.24%	961.608K	2.657M		209.479B	40.41	256.13		536.88			
 CSCO	Cisco Systems, Inc.			47.60	+1.35	+2.91%	7.751M	19.8M		200.514B	19.87	30.40		49.34			
 CRM	salesforce.com, inc.			214.17	+3.41	+1.62%	2.439M	8.208M		196.413B	48.86	115.29		284.50			
 AVGO	Broadcom Inc.			444.00	-6.14	-1.36%	694.050K	1.834M		180.738B	51.20	155.67		495.14			
 ACN	Accenture plc			255.37	+5.86	+2.35%	533.124K	2.013M		167.994B	31.28	137.15		271.18			
 TXN	Texas Instruments Incorporated			168.14	+0.20	+0.12%	765.981K	4.56M		153.882B	28.02	93.08		181.80			
 SAP	SAP SE			124.65	+1.57	+1.28%	261.984K	959.331		145.296B	30.07	90.90		169.30			
 QCOM	QUALCOMM Incorporated			128.87	-0.88	-0.68%	2.354M	9.918M		145.113B	21.84	58.00		167.94			
 SHOP	Shopify Inc.			1,151.20	+20.19	+1.79%	587.175K	1.433M		139.895B	440.83	306.30		1,486.75			
 SNE	Sony Corporation			102.74	-1.40	-1.34%	239.033K	1.046M		126.628B	21.58	50.94		118.50			
 IBM	International Business Machines Corporation			124.21	+1.38	+1.12%	1.304M	6.515M		110.904B	19.91	90.56		130.89			
 INTU	Intuit Inc.			387.11	+4.90	+1.28%	203.728K	1.395M		105.742B	58.93	187.68		423.74			
 AMAT	Applied Materials, Inc.			113.64	+0.19	+0.17%	1.267M	7.711M		103.558B	26.99	36.64		145.50			
 UBER	Uber Technologies, Inc.			55.69	-0.15	+0.27%	4.216M	22.278M		102.822B	N/A	13.71		64.00			
 MU	Micron Technology, Inc.			88.69	-0.24	-0.27%	5.182M	17.949M		98.555B	33.23	31.13		95.79			
 SQ	Square, Inc.			218.00	+1.56	+0.72%	3.603M	10.438M		97.688B	488.36	32.33		281.19			
 NOW	ServiceNow, Inc.			488.35	+1.39	+0.29%	414.432K	1.43M		95.058B	821.53	238.93		598.37			
 AMD	Advanced Micro Devices, Inc.			77.89	-0.63	-0.81%	12.847M	42.894M		93.659B	37.46	36.75		99.23			
Show 25 rows 										 c. Prev	Next 5 						

Figure 1.1-1-Yahoo Finance Stock Data Table

2.1.1 Connecting to the Data

To begin the process, I had import and explore the basic nature and syntax of the “BeautifulSoup” and “requests” Python libraries to the “WebScraper1_tickers” script. The “requests” library is one of the most common libraries used for making HTTP requests in Python. The “requests” library and the “get()” method, combined, are used to send an HTTP request and receive a “response object” in return. The object contains the source code of the website and other descriptive data about the site (encoding, status, etc.). Then, by using the “content” method, the source code can be assigned to a variable. In the code shown below, the content variable now contains the source code of the website. However, since the content variable is of the data type “bytes”, Python is still unable to do very much with the scrambled, unorganized source code. This is where the “Beautiful Soup” library comes into play. By creating a “BeautifulSoup object” and an “HTML parser”, Python can process a copy of the source code data and return the HTML code using specific elements, tags, containers, classes, etc. in a very efficient way.



```

1  import requests
2  from bs4 import BeautifulSoup
3  import csv
4
5  url = requests.get("https://finance.yahoo.com/sector/ms_technology?.tsrc=fin-srch")
6  c = url.content
7  print(c)
8  soup = BeautifulSoup(c, "html.parser")

```

Figure 1.1.1-1-Creating Response Object

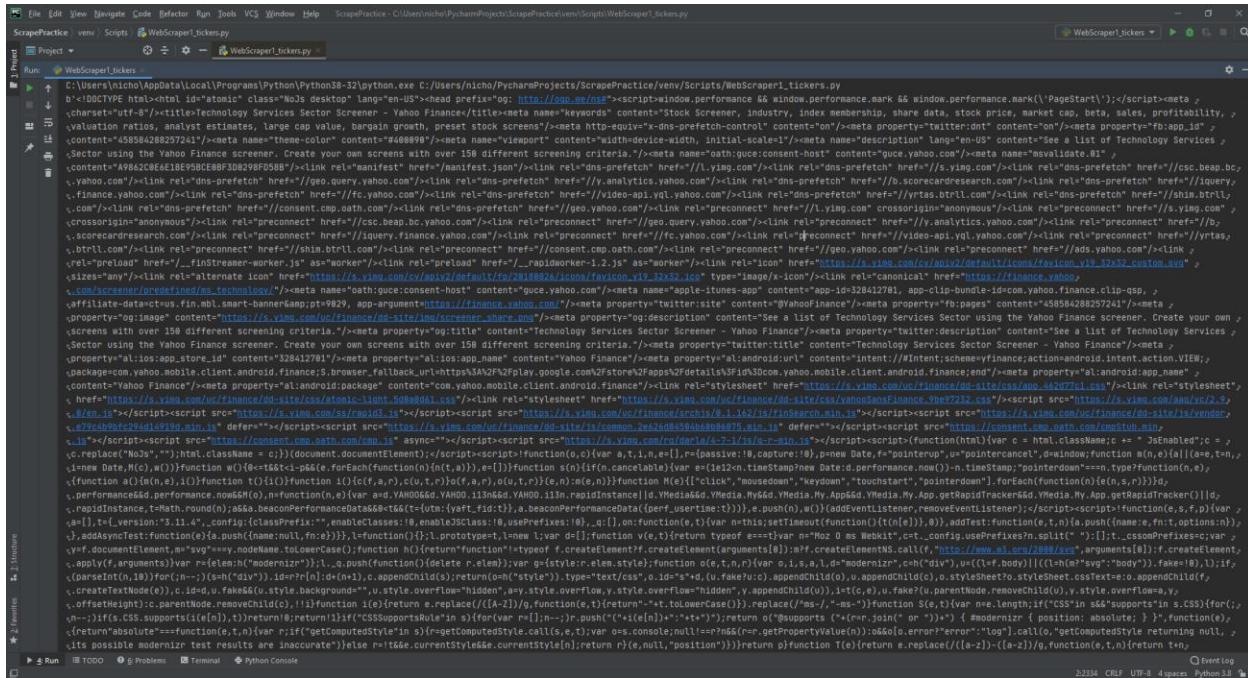


Figure 1.1.1-2 Printing Response Object

2.1.2 Scraping Ticker Symbols

By visiting the website through Google Chrome and using the “inspection” developer tool, I examined the overall structure of the website’s HTML code. Furthermore, since no obvious pattern was recognized, I used the “inspect element” developer tool to take a closer look at where the first ticker symbol was located (“APPL”). I discovered that the string for this ticker was stored in an `<a>` tag, but I discovered that the next ticker symbol in the list did not share the same attribute “id”. Therefore, I knew this was not the element I wanted. So, I found the parent element was a `<div>` tag. However, due to the odd parent-child relationship, there were more than one `<div>` tags with the same attribute “id”, so I had to analyze each one to determine which one contained the data I needed. After printing all the items in each of the `<div>` tags that shared this id, I found that the first `<div>` contained the correct data (figure 1.1.6 - hence the [0] on line 12). By using the `find_all` method for `<a>` tags with a certain “class” name, I was able to extract the HTML code that contained the Unicode text for each ticker symbol into a list. However, the data was still in HTML syntax while I only needed the plain text. So, I created a new list and used a for loop to iterate through the old list and used the “text” method to solely extract the ticker symbol text and append it to the new list.

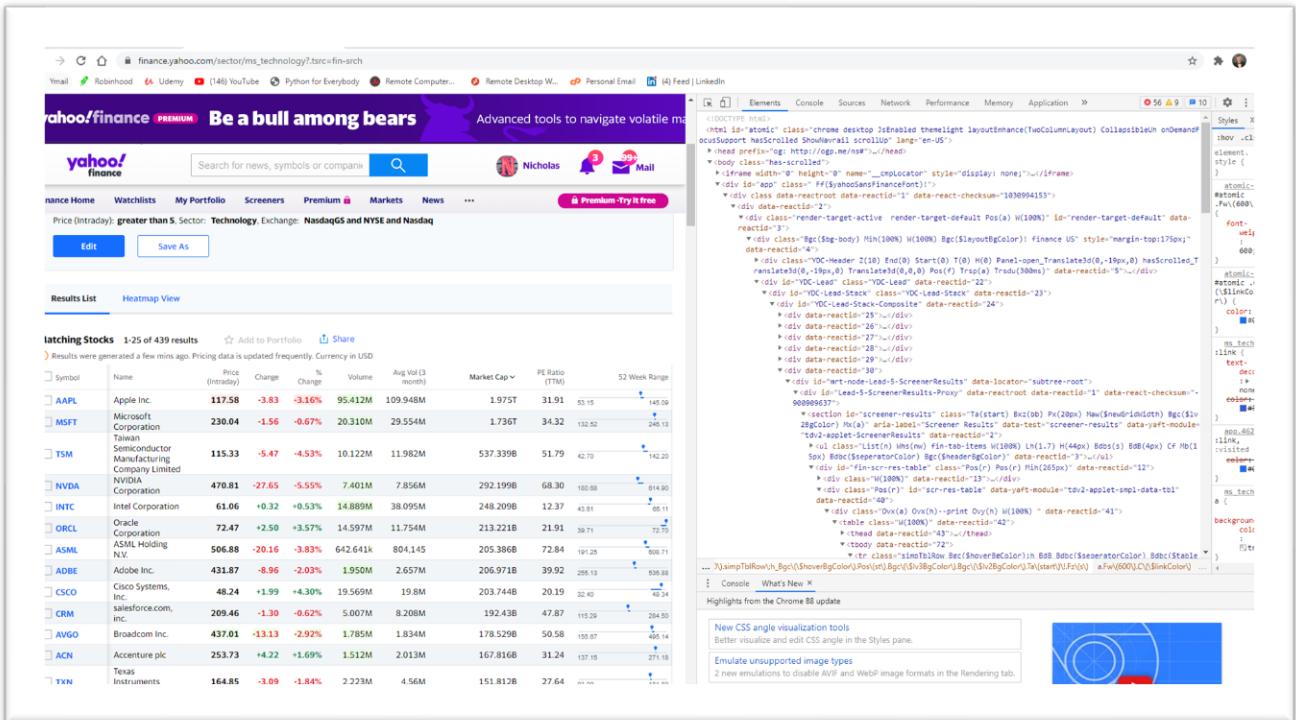


Figure 1.1.2-1 Using Google Chrome's Inspection Tool

```

▶ <label class="Ta(c) Pos(r) Va(tb) Pend(5px) D(n)--print" data-reactid="75">...</label>
  <a href="/quote/AAPL?p=AAPL" title="Apple Inc." class="Fw(600) C($linkColor)" data-reactid="79">AAPL</a> == $0
  <div class="W(3px) Pos(a) Start(100%) T(0) H(100%) Bg($pfColumnFakeShadowGradient) Pe(n) Pend(5px)" data-reactid="80"></div>
</td>

```

Figure 1.1.2-2 Inspecting the 'AAPL' ticker Element

```

table = soup.find_all("div", {"id": "fin-scr-res-table"})
print(table[0])

```

```


| Symbol | Name                                               | Price (Intraday) | Change | % Change | Volume   | Avg Vol (3 month) | Market Cap | PE Ratio (TTM) | 52 Week Range   |
|--------|----------------------------------------------------|------------------|--------|----------|----------|-------------------|------------|----------------|-----------------|
| AAPL   | Apple Inc.                                         | 117.58           | -3.83  | -3.16%   | 95,412M  | 109,948M          | 1,975T     | 31.91          | 53.15 - 140.09  |
| MSFT   | Microsoft Corporation                              | 230.04           | -1.56  | -0.67%   | 20,310M  | 29,554M           | 1,736T     | 34.32          | 122.52 - 246.13 |
| TSM    | Semiconductor Manufacturing Company Limited Taiwan | 115.33           | -5.47  | -4.53%   | 10,122M  | 11,982M           | 537,339B   | 51.79          | 42.70 - 142.20  |
| NVDA   | NVIDIA Corporation                                 | 470.81           | -27.65 | -5.55%   | 7,401M   | 7,856M            | 292,199B   | 68.30          | 180.00 - 614.90 |
| INTC   | Intel Corporation                                  | 61.06            | +0.32  | +0.53%   | 14,889M  | 38,095M           | 248,209B   | 12.37          | 43.91 - 65.11   |
| ORCL   | Oracle Corporation                                 | 72.47            | +2.50  | +3.57%   | 14,597M  | 11,754M           | 213,218B   | 21.91          | 38.71 - 77.57   |
| ASML   | ASML Holding N.V.                                  | 506.88           | -20.16 | -3.83%   | 642,641K | 804,145           | 205,386B   | 72.84          | 191.29 - 609.71 |
| ADBE   | Adobe Inc.                                         | 431.87           | -8.96  | -2.03%   | 1,950M   | 2,657M            | 206,971B   | 39.92          | 265.13 - 538.03 |
| CSCO   | Cisco Systems, Inc.                                | 48.24            | +1.99  | +4.30%   | 19,569M  | 19.8M             | 203,744B   | 20.19          | 32.40 - 49.34   |
| CRM    | Salesforce.com, inc.                               | 209.46           | -1.30  | -0.62%   | 5,007M   | 8,208M            | 192,438B   | 47.87          | 115.29 - 294.90 |
| AVGO   | Broadcom Inc.                                      | 437.01           | -13.13 | -2.92%   | 1,785M   | 1,834M            | 178,529B   | 50.58          | 155.97 - 495.14 |
| ACN    | Accenture plc                                      | 253.73           | +4.22  | +1.69%   | 1,512M   | 2,013M            | 167,816B   | 31.24          | 137.10 - 271.18 |
| TXN    | Texas Instruments                                  | 164.85           | -3.09  | -1.84%   | 2,223M   | 4,56M             | 151,812B   | 27.64          | 44.44 - 144.44  |


```

Figure 1.1.2-3 Finding the correct div item.

The screenshot shows the PyCharm IDE interface. The top window displays the code for `WebScraper_tickers.py`, which uses BeautifulSoup to parse a webpage and extract a list of stock tickers. The bottom window shows the terminal output, which lists 24 stock tickers: AAPL, MSFT, TSM, NVDA, INTC, ORCL, ASML, ADBE, CSCO, CRM, AVGO, ACN, TXN, SAP, QCOM, SHP, SNE, IBM, INTU, UBER, AMAT, MU, SO, NOW, and AMD. The terminal also indicates that the process finished with an exit code of 0.

Figure 2.1.2-4 Creating & Printing the Ticker List

2.1.3 Scraping the Names and Prices

Since the data for the ticker symbols, names, and prices were all located in the first list of lists in the “table” variable I previously created all I had to do was find the individual elements that contained the Unicode text for the names and prices, as I did for the ticker symbols. Then, repeat the same process of looping through the list of html code, extracting the text and appending it to a new list. However, one exception came in the process of extracting the prices. While I did not have trouble extracting the string value for the price, I did have to convert the data type to float() during this iteration so the list would contain numeric values.

```
▼<td colspan="1" class="Va(m) Ta(start) Px(10px) Fz(s)" aria-label="Name" data-reactid="81"> == $0
  <!-- react-text: 82 -->
  "Apple Inc."
  <!-- /react-text -->
</td>
```

Figure 1.1.3-1 Inspecting the 'Apple Inc.' element.

```
▼<td colspan="1" class="Va(m) Ta(end) Pstart(20px) Fw(600) Fz(s)" aria-label="Price (Intraday)" data-reactid="83">
  <span class="Trsdu(0.3s)" data-reactid="84">121.42</span> == $0
</td>
```

Figure 1.1.3-2 Inspecting Apple's 'Intraday Price' element.

```
27 |     prices = table[0].find_all("td", {"aria-label": "Price (Intraday)"})
28 |     price_lst = []
29 |     for price in prices:
30 |         price = price.text
31 |         price = float(price.replace(',', '.'))
32 |         price_lst.append(price)
```

Figure 1.1.3-3 Converting the price data type.

Figure 1.1.3-4 All values stored into lists, respectively.

2.1.4 Storing Stock Performance Data

After storing all the data into python lists, I needed to find a way to save the data .csv file to later be imported into a database or elsewhere, if needed. Therefore, I created a dictionary and stored the ticker symbol as the dictionary keys and appended the name and price to a new list and stored the new list in the dictionary value. After the data was properly stored in the dictionary, I used the csv library to write the key, value pairs to .csv file.

```
35     dct={}
36     for i in range(len(ticker_lst)):
37         lst=[]
38         lst.append(name_lst[i])
39         lst.append(price_lst[i])
40         dct[ticker_lst[i]] = lst
41
42     w = csv.writer(open("stockData.csv", "w"))
43     for key, val in dct.items():
44         w.writerow([key, val])
45         write = csv.writer(f)
46         write.writerow(ticker_lst)
47         write.writerow(name_lst)
48         write.writerow(price_lst)
```

Figure 1.1.4-1 Storing lists in dictionary and writing to csv file.

```
1 AAPL,"['Apple Inc.', 121.42]"
2 MSFT,"['Microsoft Corporation', 231.6]"
3
4 TSM,"['Taiwan Semiconductor Manufacturing Company Limited', 120.8]"
5
6 NVDA,"['NVIDIA Corporation', 498.46]"
7
8 INTC,"['Intel Corporation', 60.74]"
9
10 ASML,"['ASML Holding N.V.', 527.04]"
11
12 ADBE,"['Adobe Inc.', 440.83]"
13
14 ORCL,"['Oracle Corporation', 69.97]"
15
16 CSCO,"['Cisco Systems, Inc.', 46.25]"
17
18 CRM,"['salesforce.com, inc.', 210.76]"
19
20 AVGO,"['Broadcom Inc.', 450.14]"
21
22 ACN,"['Accenture plc', 249.51]"
23
24 TXN,"['Texas Instruments Incorporated', 167.94]"
25
26 QCOM,"['QUALCOMM Incorporated', 129.75]"
27
28 SAP,"['SAP SE', 123.08]"
29
30 SHOP,"['Shopify Inc.', 1131.01]"
31
32 SNE,"['Sony Corporation', 104.14]"
33
34 IBM,"['International Business Machines Corporation', 122.83]"
35
36 INTU,"['Intuit Inc.', 382.21]"
37
38 AMAT,"['Applied Materials, Inc.', 113.45]"
```

Figure 1.1.4-2 StockData.csv view.

2.2 Web Scraping: Bloomberg News Article Data

The next process of my project dealt with scraping the 3 most recent news articles from 3 different websites. The first website I chose to scrape were the three articles stored in the “Latest Sector News” content block of <https://www.bloomberg.com/markets/sectors/information-technology>. Furthermore, I needed to scrape and store the title, publication date, link to the article, and description of each article. After gaining some experience from scraping the stock market data, I learned a more efficient way for looping through the HTML elements and parsing the data which I will explain.

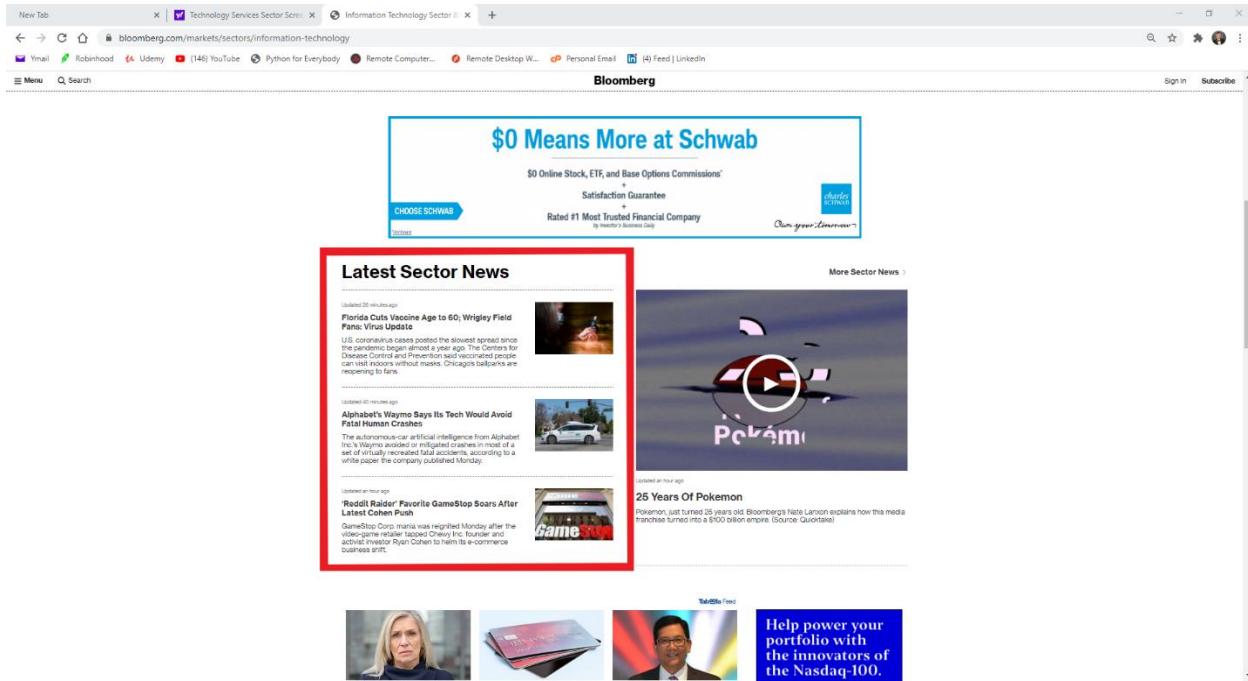


Figure 1.2-1 Bloomberg's latest technology sector news

2.2.1 Connecting to the Data

First, I would like to note that no new libraries were needed for this process, however, I did import the `html5lib` library into the “`bloombergScraper.py`” file to use as my Beautiful Soup HTML parser because it seemed to work more accurately for all the websites. The reason `html5lib` worked more effectively was because it allows you to include “request headers” to give the request context and return a more accurate response. To begin with, I visited the link mentioned above through Google Chrome and used the inspection tool to locate the `<div>` tags containing *all* the elements I needed to iterate through and extract. I found that the recent article section consisted of 3 `<div>` tags with matching class names, indicating the containers for the 3 articles. Since only the elements I want to iterate through are stored in `content` variable, I can use a single for loop to extract the title, date, description, and link to article.

```
bloombergScraper.py
1  from urllib.request import Request, urlopen
2  from bs4 import BeautifulSoup
3  import requests
4
5  link = "https://www.bloomberg.com/markets/sectors/information-technology"
6  req = Request(link, headers={'User-Agent': 'Mozilla/5.0'})
7  webpage = urlopen(req).read()
8
9  with requests.Session() as c:
10    soup = BeautifulSoup(webpage, 'html5lib')
11    content = soup.find_all('div', attrs={'class': 'sector-related-content__item'})
```

Figure 1.2.1-1 Creating Response & Beautiful Soup Objects.

```
► <div class="sector-related-content__item">...</div> == $0
► <div class="sector-related-content__item">...</div>
► <div class="sector-related-content__item">...</div>
```

Figure 1.2.1-2 Latest Sector News <div> tags.

2.2.2 Scraping the title

To scrape the title from the Bloomberg website, I used the inspection tool in google chrome and found that the text for the title was found in an <a> tag. However, I had an issue with referencing the <a> tags by their class names. Since there were no other <a> tags in each of the 3 <div> tags containing a hyperlink, I used the find method for the <a> tags where a hyperlink was present. Last, I added the text method to the end to return the Unicode string for each of the three titles.

```
for item in content:
    title = item.find('a', href=True).text
```

2.2.3 Scraping the description and publication date

Next, to scrape the publication date I repeated the process of locating the element tags containing the time, which were actual <time> tags. Since there was only one-time tag in each article container there was no need to define special attributes. This was also the case with description except it was in a tag instead of <time>. So, I just defined the time and description variables inside of the for loop and used the “find” method and text methods to return the data in the correct format for each article.

```
time = item.find('time').text
des = item.find('span').text
```

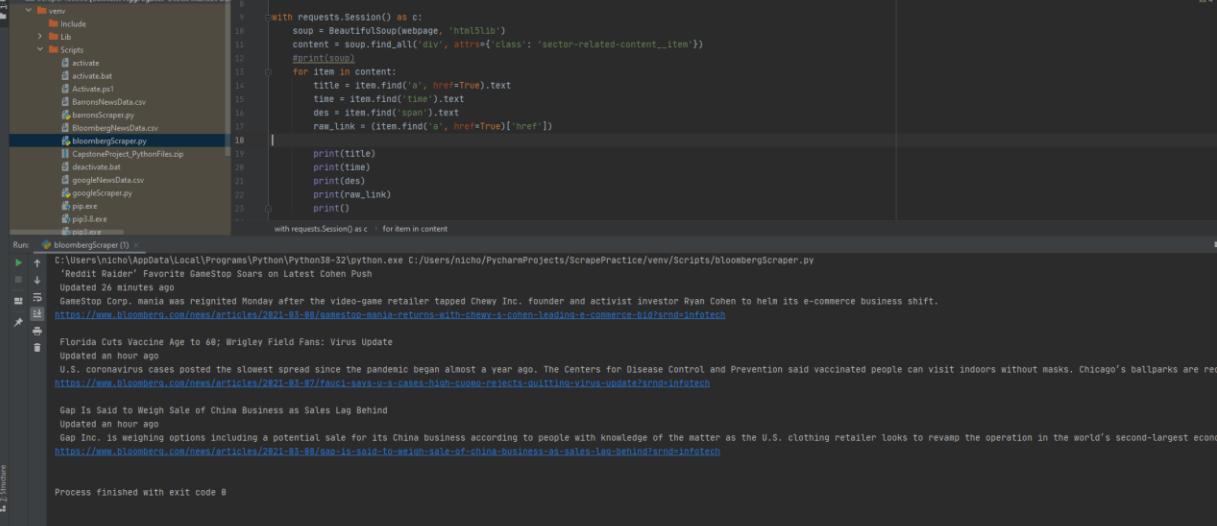
2.2.4 Scraping the hyperlink

Lastly, I scraped the hyperlinks to the articles which forced me to take a different approach than any of the previous data I had scraped. The reason for this being that, previously, I have only had to scrape data and store it as textual/string data, except for the stock prices. However, to scrape an actual link, I used the same find method that I used for the title except, instead of using the text method, I used the ['href'] method, which returns the value in the “href”(the hyperlink to the article.)

```
raw_link = (item.find('a', href=True)['href'])
```

2.2.5 Looping Through the Latest Sector News Items

After defining the previously mentioned variables and placing them inside the for loop, each item is carefully extracted and ready to be stored in a proper data frame or format.



The screenshot shows the PyCharm IDE interface with the following details:

- File Structure:** The project is named "ScrapePractice" and contains a "Scripts" directory with files like "activate", "activate.bat", "Activate.ps1", "BaronsNewData.csv", "baronsScrape.py", "BloombergNewData.csv", and "bloombergScrape.py".
- Code Editor:** The file "bloombergScrape.py" is open, showing Python code for web scraping. It uses the requests library to get a webpage,BeautifulSoup to parse it, and loops through items to extract title, time, and description.
- Run Tab:** The "bloombergScrape (1)" tab is selected, showing the command "C:/Users/nicho/AppData/Local/Programs/Python/Python38-32/python.exe C:/Users/nicho/PycharmProjects/ScrapePractice/venv/Scripts/bloombergScrape.py" and the output of the script running. The output includes news articles from Bloomberg about GameStop and Gap.
- Bottom Status Bar:** Shows "Process finished with exit code 0".
- Bottom Right:** Event log tab with 161 entries.

Figure 1.2.5-1 Scraping and Printing Bloomberg News Article items.

2.2.6 Storing the article data

Lastly, I used the csv library and string formatting to store the data in a csv file. However, on the first attempt I realized there was an error somewhere when I opened the csv file and saw that not all data was stored in the appropriate columns. After a combination of debugging and researching, I found that this error was being caused by commas in the actual strings being used as the delimiter to create the “comma separated file”. So, I used the replace() function to replace any commas in the title, description or time to be replaced with a blank space.

A	B
DBS Cuts Bonus of CEO	Senior Managers After Profit Decline
Atari Setting Up Crypto Casino to Tap Into Nostalgia and NFTs	Updated 2 hours ago
Chinese Beauty App Becomes First Major Company to Buy Ether	Updated 3 hours ago

Figure 1.2.6-1 Data Stored incorrectly.

Figure 1.2.6-2 Updated for loop.

	A	B
1	DBS Cuts Bonus of CEO Senior Managers After Profit Decline	Updated an hour ago
2	Atari Setting Up Crypto Casino to Tap Into Nostalgia and NFTs	Updated 2 hours ago
3	Chinese Beauty App Becomes First Major Company to Buy Ether	Updated 3 hours ago

Figure 1.2.6-3 Data Stored Correctly.

2.3 Web Scraping: Barron's News Article Data

The next website I chose to scrape News article data from was the 3 most recent news articles found at https://www.barrons.com/topics/technology?mod=BOL_TOPNAV. Connecting to the Barron's data was the same as connecting to the Bloomberg data, other than changing the URL to request the appropriate address. However, the elements on this website were a bit more embedded, which required more detailed uses of the find method.

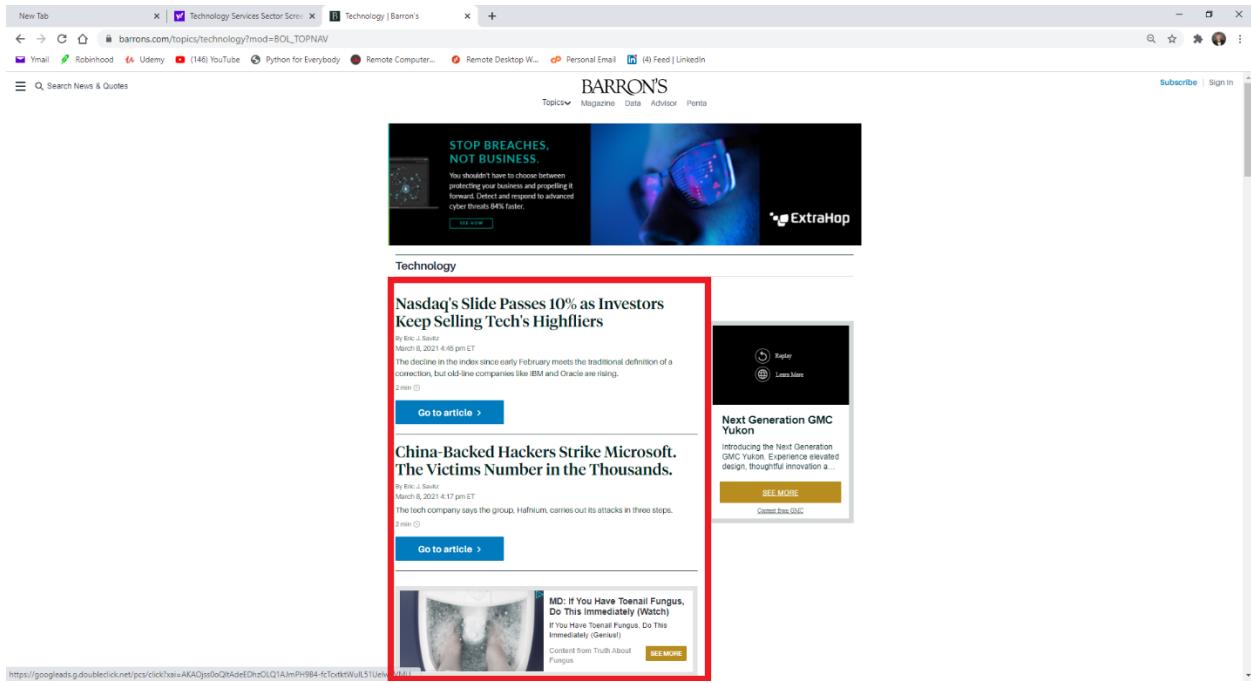


Figure 1.3-1 Barron's Latest Technology Sector News

2.3.1 Scraping the Title and Hyperlink

First off, I repeated the process of visiting the website through Google Chrome and inspecting the elements that contained each of the items I wanted for individual articles. After locating the `<div>` tag with the correct class name, I noticed it was pulling every single article container on the page, while I only needed the data from the top 3. Therefore, I set my “for” loop to only loop through the first 3 items. Then, the title was very simple because it was the only `<a>` tag inside of each of the `<div>` tags, similarly to the Bloomberg data. The hyperlink also followed the same scraping technique used to scrape the Bloomberg data.

```
for item in content[0:4]:
    title = item.find('a').text
    raw_link = (item.find('a', href=True)['href'])
```

2.3.2 Scraping the description and publication date

Since the description and date were more embedded in the container, it did take some more specifications to properly extract data from the correct elements. Unlike the title, the description was not stored in the only tag of its kind, therefore, unique class attributes had to be identified. Furthermore, the time for each article was even more deeply embedded. Not only was the time stored inside of a child `<div>`, but it was also stored inside of another `<p>` tag. So, both elements as well as their class attributes had to be defined in the find function.

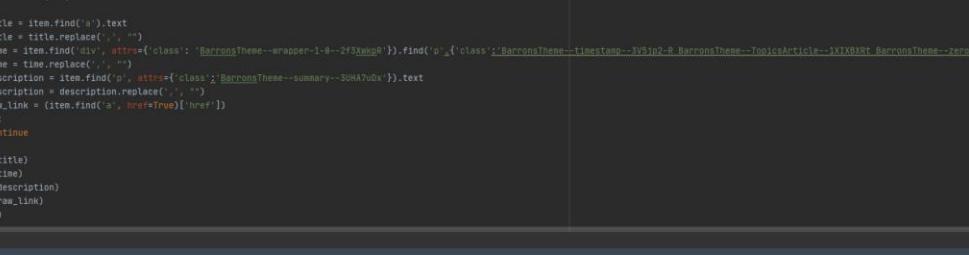
```
time =
item.find('div', attrs={'class': 'BarronsTheme--wrapper-1-0--2f3XwkpR'}).find('p', {'class': 'BarronsTheme--timestamp--3V5jp2-R BarronsTheme--TopicsArticle--1XIXBXRt BarronsTheme--zero-margin-timestamp--YIWqgHaW'}).text

description =
```

```
item.find('p', attrs={'class':'BarronsTheme--summary--3UHA7uDx'}).text
```

2.3.3 Looping Through the Latest Sector News Items

After defining the previously mentioned variables and placing them inside the for loop, each item is carefully extracted and ready to be stored in a proper data frame or format. However, since I encountered the .csv issue with storing items in the incorrect column, I went ahead and used the replace() function to be cautious.



The screenshot shows the PyCharm IDE interface with the following details:

- File Menu:** File, edit, view, Navigate, Code, Refactor, Run, Tools, Window, Help.
- Project Bar:** ScrapePractice, venv, Scripts, barronsScraper.py.
- Code Editor:** The script file `barronsScraper.py` is open, containing Python code to scrape news articles from Barron's. The code uses BeautifulSoup to parse HTML content and extract titles, times, descriptions, and links.
- Run Tab:** Shows the command run: `C:/Users/nicho/AppData/Local/Programs/Python/Python38-32/python.exe C:/Users/nicho/PycharmProjects/ScrapePractice/venv/Scripts/barronsScraper.py`. The output shows several news articles:

 - Nasdaq's Slide Passes 10% as Investors Keep Selling Tech's Highfliers
 - March 8 2021 9:45 pm ET
 - The decline in the index since early February meets the traditional definition of a correction but old-line companies like IBM and Oracle are rising.
 - <https://www.barrons.com/articles/nasdaq-slide-passes-10-as-investors-keep-selling-techs-high-fliers-5141525993#refsec-technology>

- China-Backed Hackers Strike Microsoft. The Victims Number in the Thousands.
- March 8 2021 9:17 pm ET
- The tech company says the group Hafnium carries out its attacks in three steps.
- <https://www.barrons.com/articles/thousands-of-businesses-hit-by-hack-of-microsoft-exchange-5141527254#refsec-technology>
- Bumble Wall Street Is Just Not That Into You
- March 8 2021 6:38 pm ET
- Analysts have begun coverage of the dating app company. The verdict so far is mixed on whether to buy its shares.
- <https://www.barrons.com/articles/bumble-stock-is-getting-a-very-mixed-review-from-wall-street-analysts-5161526877#refsec-technology>

- Bottom Status Bar:** PEP 8: W391 blank line at end of file, TODO, Problems, Terminal, Python Console, Run, PyCharm 2020.2.5 available, Update...

Figure 1.3.3-1 Scraping and Printing Barron's News Article items.

2.3.4 Storing Barron's Article Data

Since I went ahead and updated each variable in the for loop using the replace function for any commas, the data properly stored in a csv file.

A	B
1 Nasdaq's Slide Passes 10% as Investors Keep Selling Tech's Highfliers	March 8 2021 9:45 pm ET
2 China-Backed Hackers Strike Microsoft. The Victims Number in the Thousands.	March 8 2021 9:17 pm ET
3 Bumble Wall Street Is Just Not That Into You	March 8 2021 6:38 pm ET

Figure 1.3.4-1 Barron's data stored correctly.

2.4 Web Scraping: Google Search Results

Instead of choosing a different news company website, I decided to scrape the top 3 articles posted within the last 24 hours from the Google Search engine “News” feature using the search query “technology stocks”. The URL I am sending a request to is

https://www.google.com/search?q=technology+stocks&tbo=nws&source=lnl&tbs=qdr:d&sa=X&ved=0ahUKEwjs8JXwpJ_vAhURLs0KHRryCsEQpwUIKQ&biw=1920&bih=937&dpr=1

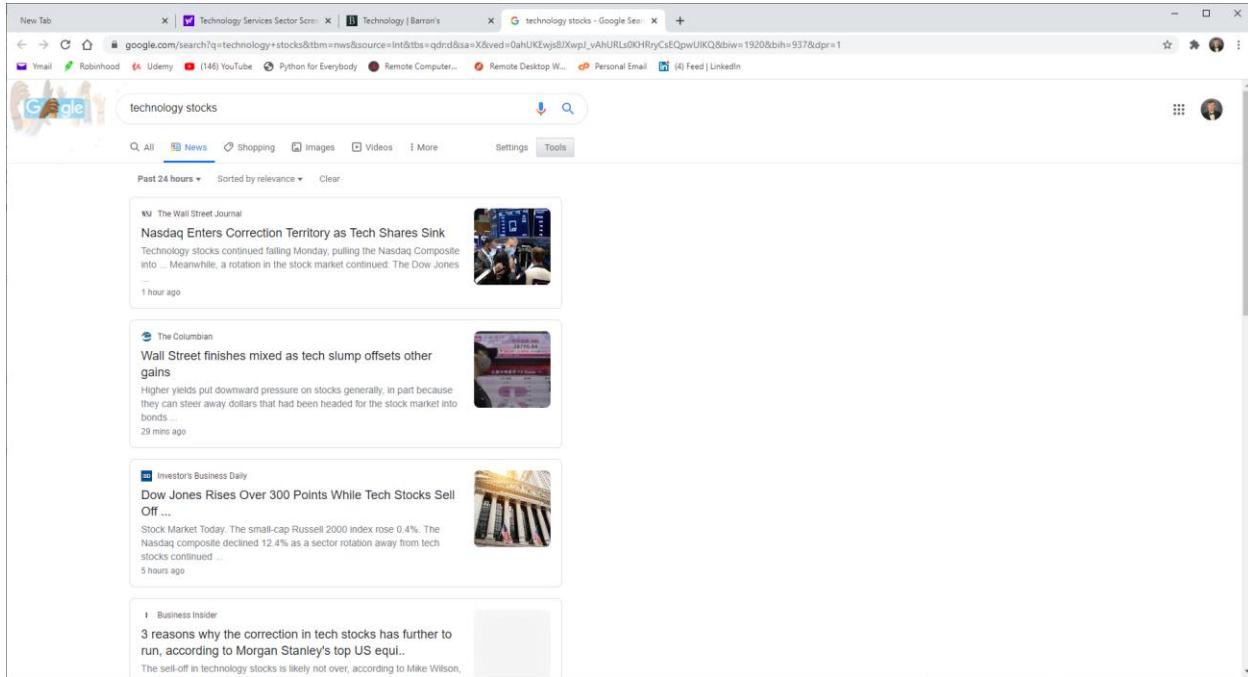


Figure 1.3-1 Google news search results for: technology stocks.

2.4.1 Scraping the Data

I took the exact same approach to locating the correct `<div>` container to iterate through as I did the previous two times. Moreover, scraping the items was also very similar to the other websites, with a few additional steps. The additional steps involved separating the time and description to two separate values by using the `split()` function to separate the data on the bullet point.

```
googleScraper (1) ×
C:\Users\nicho\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/nicho/PycharmProjects/Scr
Nasdaq Enters Correction Territory as Tech Shares Sink
1 hour ago · Technology stocks continued falling Monday pulling the Nasdaq Composite into ... Meanwhil
https://www.wsj.com/articles/global-stock-markets-dow-update-03-08-2021-11615192681

Dow Jones Rises Over 300 Points While Tech Stocks Sell Off ...
6 hours ago · Stock Market Today. The small-cap Russell 2000 index rose 0.4%. The Nasdaq composite dec
https://www.investors.com/market-trend/stock-market-today/dow-jones-rises-over-400-points-while-tech-s

Wall Street finishes mixed as tech slump offsets other gains
43 mins ago · Higher yields put downward pressure on stocks generally in part because they can steer a
https://www.columbian.com/news/2021/mar/08/global-shares-mixed-oil-prices-up-technology-stocks-fall/
```

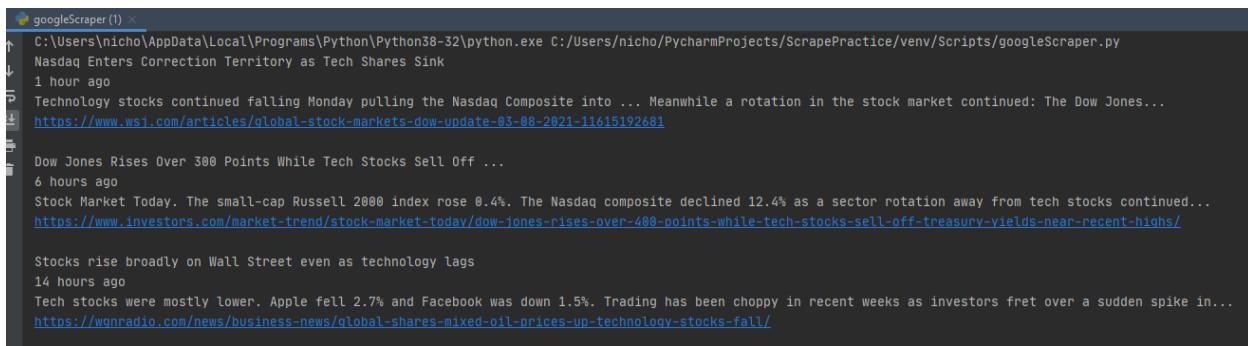
Figure 1.4.1-1 Incorrectly extracting time and description.

```

14     for item in content[0:3]:
15         title = item.find('div', attrs={'class': 'BNeawe vvjwJb AP7Wnd'}).text
16         title = title.replace('\n', '')
17
18         raw_link = (item.find('a', href=True)['href'])
19         link = (raw_link.split("/url?q=")[1]).split('&sa=U&')[0]
20
21         description = item.find('div', attrs={'class': 'BNeawe s3v9rd AP7Wnd'}).text
22         description = description.replace('\n', '')
23         time = description.split(' · ')[0]
24         intro = description.split(' · ')[1]
25
26         print(title)
27         print(time)
28         print(intro)
29
30         print(link)
31         print()

```

Figure 1.4.1-2 Using the split function to correct extraction.



```

googleScraper () ×
C:\Users\nicho\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/nicho/PycharmProjects/ScrapePractice/venv/Scripts/googleScraper.py
↑ Nasdaq Enters Correction Territory as Tech Shares Sink
↓ 1 hour ago
Technology stocks continued falling Monday pulling the Nasdaq Composite into ... Meanwhile a rotation in the stock market continued: The Dow Jones...
https://www.wsj.com/articles/global-stock-markets-dow-update-03-08-2021-11615192681

Dow Jones Rises Over 300 Points While Tech Stocks Sell Off ...
6 hours ago
Stock Market Today. The small-cap Russell 2000 index rose 0.4%. The Nasdaq composite declined 12.4% as a sector rotation away from tech stocks continued...
https://www.investors.com/market-trend/stock-market-today/dow-jones-rises-over-400-points-while-tech-stocks-sell-off-treasury-yields-near-recent-highs/

Stocks rise broadly on Wall Street even as technology lags
14 hours ago
Tech stocks were mostly lower. Apple fell 2.7% and Facebook was down 1.5%. Trading has been choppy in recent weeks as investors fret over a sudden spike in...
https://wgnradio.com/news/business-news/global-shares-mixed-oil-prices-up-technology-stocks-fall/

```

Figure 1.4.1-3 Correctly extracting time and description

2.4.2 Storing the data

I used the exact same method for storing the data in a csv file as in the previous two examples. Since the replace function was already included, the data stored correctly.

	A	B
1	Nasdaq Enters Correction Territory as Tech Shares Sink	1 hour ago
2	Dow Jones Rises Over 300 Points While Tech Stocks Sell Off ...	6 hours ago
3	Stocks rise broadly on Wall Street even as technology lags	14 hours ago

Figure 1.4.2-1 Data stored Correctly.

3 Midterm Summary

Now that I have explored the art of web scraping and have successfully found a way to collect the data in the correct format whenever I need, it is time to introduce Django. Django is an extremely powerful front-end framework for Python. I will use this framework to store, manage, and display my content in an organized website format.

4 The Django Framework

4.1 Hosting

Before I began using the Django framework, I switched from the “Pycharm” IDE to the “Atom” text editor. I found that my project would be more compatible, portable, and dependable by creating a virtual environment and testing my project files in the command prompt. However, I did still test the project/run specific files in PyCharm periodically.

```
F:\>cd CapstoneFolder
F:\CapstoneFolder>python -m venv virtual_env
F:\CapstoneFolder>virtual_env\Scripts\activate.bat
(virtual_env) F:\CapstoneFolder>pip install django
Collecting django
  Downloading Django-3.2-py3-none-any.whl (7.9 MB)
    |████████████████████████████████| 7.9 MB 6.4 MB/s
Collecting asgiref<4,>=3.3.2
  Downloading asgiref-3.3.4-py3-none-any.whl (22 kB)
Collecting pytz
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
Collecting sqlparse>0.2.2
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)
Installing collected packages: asgiref, pytz, sqlparse, django
```

Figure 3.1-1- Creating Virtual Environment

4.2 Creating a Django Project

After installing Django and other necessary dependencies, and an extensive period of trial and error exploring the functionality of Django, I found that the framework operates on a very simple three “MVT” architecture which consist of, the Model, View, and Template files. A Django project, like any website, can contain multiple applications that are used for different functionalities, and each Django application contains the “MVT” architecture. The Model is a Python script used to create, modify, or delete database tables in the SQLite3 database that comes with a Django project by default. The View is Python file that enables you to specify the logic in which you want to display/render data to the webpage. The template is a folder that is created conventionally to store all the HTML/CSS files that make up the front-end of the website. The following screenshots contain the simple commands I ran to create my Django project and successfully start the Django server.

```
F:\CapstoneFolder>virtual_env\Scripts\activate.bat
(virtual_env) F:\CapstoneFolder>django-admin startproject capstone_project
(virtual_env) F:\CapstoneFolder>cd capstone_project
(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

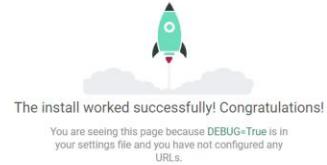
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

Django Version 3.2, using settings 'capstone_project.settings'
Starting development server at http://127.0.0.1:8000
Quit the server with CTRL-BREAK.
```

Information Technology Sector | Python Django Project - Learn to | python - Django model "doesnt" | Applications | Django document | css - How to indent a DIV? - Sta | Django: the Web framework for | +

Ymail Robinhood Udemy (146) YouTube Python for Everybody Remote Computer... Remote Desktop W... Personal Email (4) Feed LinkedIn CIS-429 Chapter 3 F...

django View release notes for Django 3.1



Django Documentation | Tutorial: A Polling App | Django Community

Figure 3.2-1 Django server running.



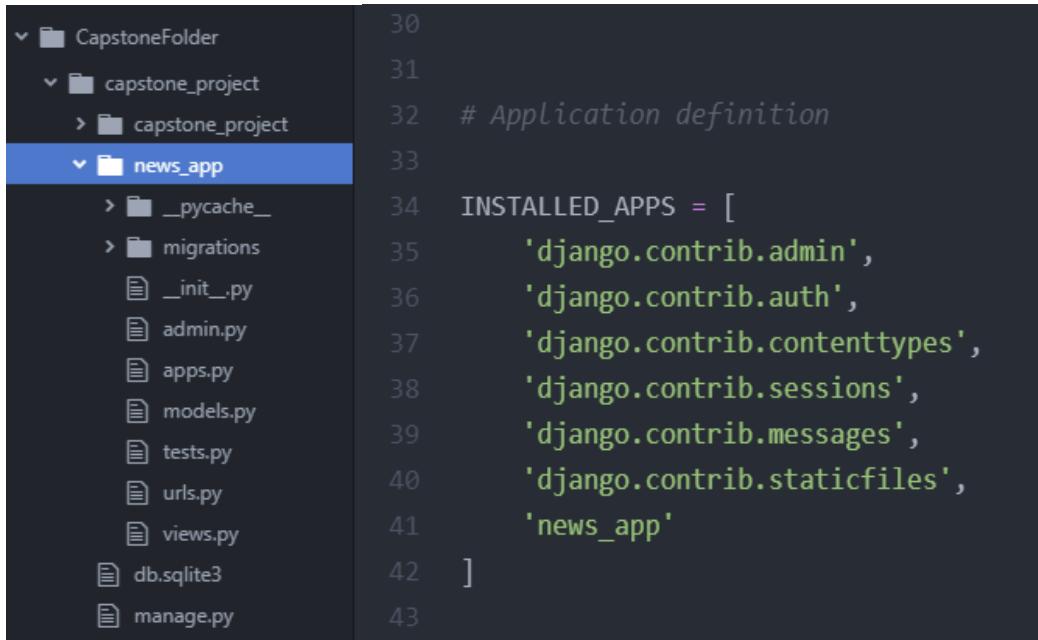
Figure 3.2-2- Django project directory created as 'capstone project'.

4.3 Creating the News App

The next process involved is creating an application (websites can contain multiple applications). The application was also created in the command prompt as demonstrated below.

```
(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py startapp news_app
(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 3.3-1 creating the news application.



```

30
31
32 # Application definition
33
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41     'news_app'
42 ]
43

```

Figure 3.3-2- news app file directory created. Figure 3.3-3 – configuring project settings with new application.

4.3.1 Developing the first View

In Django, the views script works as a function or class to send a request and return a response. However, before anything could be requested or returned, I had to configure my project “urls.py” file to point to my function in the “views.py” file inside my new application. By using the `HttpResponse` function, the views file can return a response displaying the data in the browser straight from the python script rather than querying from elsewhere. The following screenshots show my initial experience/success with displaying data.

```

1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 def index(request):
5     return HttpResponse('<h1>My first HTTP Response</h1>')
6
7 # Create your views here.
8

```

Figure 3.3.1-1 creating first view containing the `index` function.

```

1 from django.conf.urls import url
2 from news_app import views
3 urlpatterns = [
4     url(r'^$', views.index, name='index'),
5
6 ]
7

```

Figure 3.3.1-2 - creating the `news_app/urls.py` file.

```

16  from django.contrib import admin
17  from django.conf.urls import url, include
18  from news_app import views
19
20  urlpatterns = [
21      url(r'', include('news_app.urls')),
22      url(r'^admin/', admin.site.urls),
23  ]
24

```

Figure 3.3.1-3 Configuring project urls.py file with app urls.py.

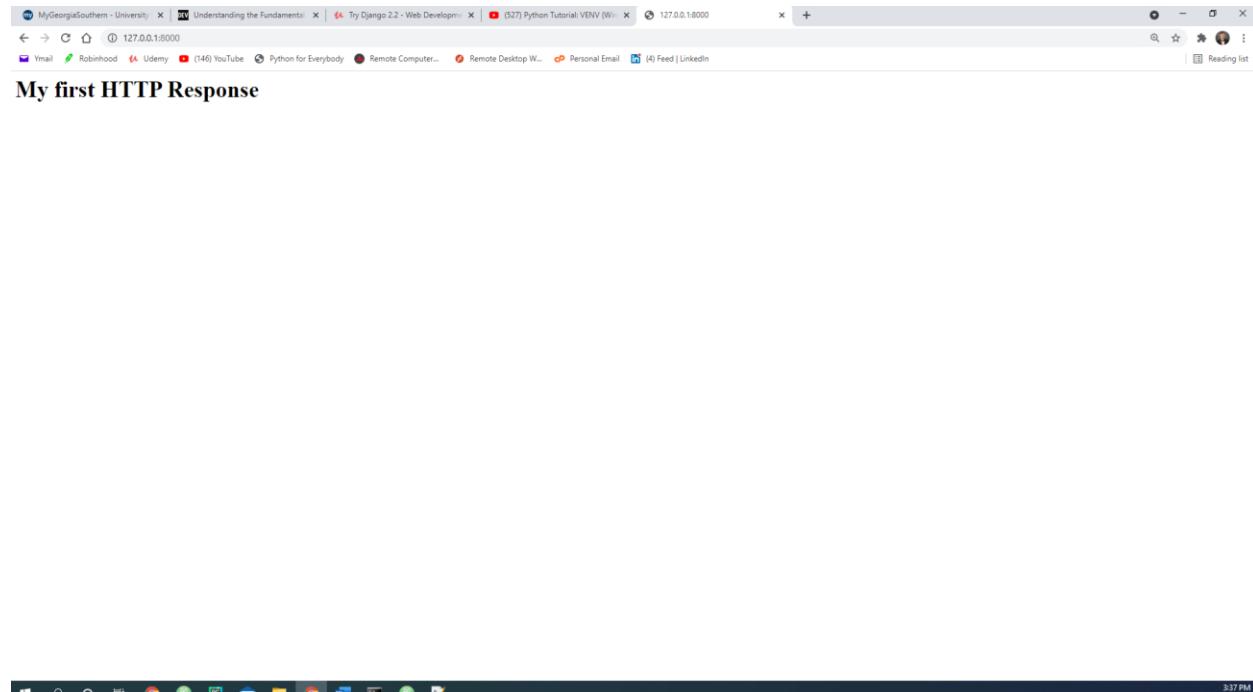


Figure 3.3.1-4 -My first HTTP response my first view

4.3.2 Implementing the web scraper function into the view

My next goal is to use the `HttpResponse` function to list items in the browser that my web scraper function generates. The first step is to replace the “index” function in the `views` file to the new function “`news_scrape`”, then reconfigure my `news_app/urls.py` file from pointing to “`views.index`” to “`views.news_scrape`”.

```

(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py runserver
atching for file changes with StatReloader
Performing system checks...

Exception in thread django-main-thread:
Traceback (most recent call last):
  File "C:\Users\nicho\AppData\local\Programs\Python\Python38-32\lib\threading.py", line 932, in _bootstrap_inner
    self.run()
  File "C:\Users\nicho\AppData\local\Programs\Python\Python38-32\lib\threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\utils\autoreload.py", line 64, in wrapper
    fn(*args, **kwargs)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\management\commands\runserver.py", line 118, in inner_run
    self.check(display_error=error_on_warning)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\management\base.py", line 419, in check
    all_issues = checks.run_checks()
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\checks\registry.py", line 76, in run_checks
    new_errors = check(app_configs=app_configs, databases=databases)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\checks\urls.py", line 13, in check_url_config
    return check_resolver(resolver)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\checks\urls.py", line 23, in check_resolver
    return check_method()
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 412, in check
    for pattern in self.url_patterns:
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\utils\functional.py", line 48, in __get__
    res = instance._dict_[self.name] = self.func(instance)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 598, in url_patterns
    patterns = getattr(self.urlconf_module, "urlpatterns", self.urlconf_module)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\utils\functional.py", line 48, in __get__
    res = instance._dict_[self.name] = self.func(instance)
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 591, in urlconf_module
    return import_module(self.urlconf_name)
  File "C:\Users\nicho\AppData\local\Programs\Python\Python38-32\lib\importlib\_init_.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "F:\frozen importlib._bootstrap", line 1014, in _gcd_import
    _bootstrap._load(name[level-1],globals,locals,fromlist)
  File "F:\frozen importlib._bootstrap", line 975, in _load
    _bootstrap._find_and_load_unlocked
  File "F:\frozen importlib._bootstrap", line 671, in _load_unlocked
  File "F:\frozen importlib._bootstrap_external", line 783, in _exec_module
  File "F:\frozen importlib._bootstrap", line 219, in _call_with_frames_removed
  File "F:\CapstoneFolder\capstone_project\capstone_project\urls.py", line 21, in <module>
    from . import views
  File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\conf.py", line 34, in include
    urlconf_module = import_module(uniconf_module)
  File "C:\Users\nicho\AppData\local\Programs\Python\Python38-32\lib\importlib\_init_.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "F:\frozen importlib._bootstrap", line 1014, in _gcd_import
    _bootstrap._load(name[level-1],globals,locals,fromlist)
  File "F:\frozen importlib._bootstrap", line 975, in _load
    _bootstrap._find_and_load_unlocked
  File "F:\frozen importlib._bootstrap", line 671, in _load_unlocked
  File "F:\frozen importlib._bootstrap_external", line 783, in _exec_module
  File "F:\frozen importlib._bootstrap", line 219, in _call_with_frames_removed
  File "F:\CapstoneFolder\capstone_project\news_app\urls.py", line 4, in <module>
    url(r'', views.index, name='Index')
AttributeError: module 'news_app.views' has no attribute 'index'

```

Figure 3.3.2-1 ERROR did not reconfigure urls.py.

TypeError at /

news_scrape() takes 0 positional arguments but 1 was given

Request Method: GET
 Request URL: http://127.0.0.1:8000/
 Django Version: 3.2
 Exception Type: TypeError
 Exception Value: news_scrape() takes 0 positional arguments but 1 was given
 Exception Location: F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\handlers\base.py, line 181, in _get_response
 Python Executable: F:\CapstoneFolder\virtual_env\Scripts\python.exe
 Python Version: 3.8.5
 Python Path: ['F:\\CapstoneFolder\\capstone_project', 'C:\\Users\\nicho\\AppData\\Local\\Programs\\Python\\Python38-32\\python38.zip', 'C:\\Users\\nicho\\AppData\\Local\\Programs\\Python\\Python38-32\\DLLs', 'C:\\Users\\nicho\\AppData\\Local\\Programs\\Python\\Python38-32\\lib', 'C:\\Users\\nicho\\AppData\\Local\\Programs\\Python\\Python38-32', 'F:\\CapstoneFolder\\virtual_env', 'F:\\CapstoneFolder\\virtual_env\\lib\\site-packages']
 Server time: Wed, 07 Apr 2021 20:24:16 +0000

Traceback [Switch to copy-and-paste view](#)

```

F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\handlers\exception.py, line 47, in inner
  47.     response = get_response(request)
  ► Local vars
F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\handlers\base.py, line 181, in _get_response
  181.     response = wrapped_callback(request, *callback_args, **callback_kwargs)
  ► Local vars

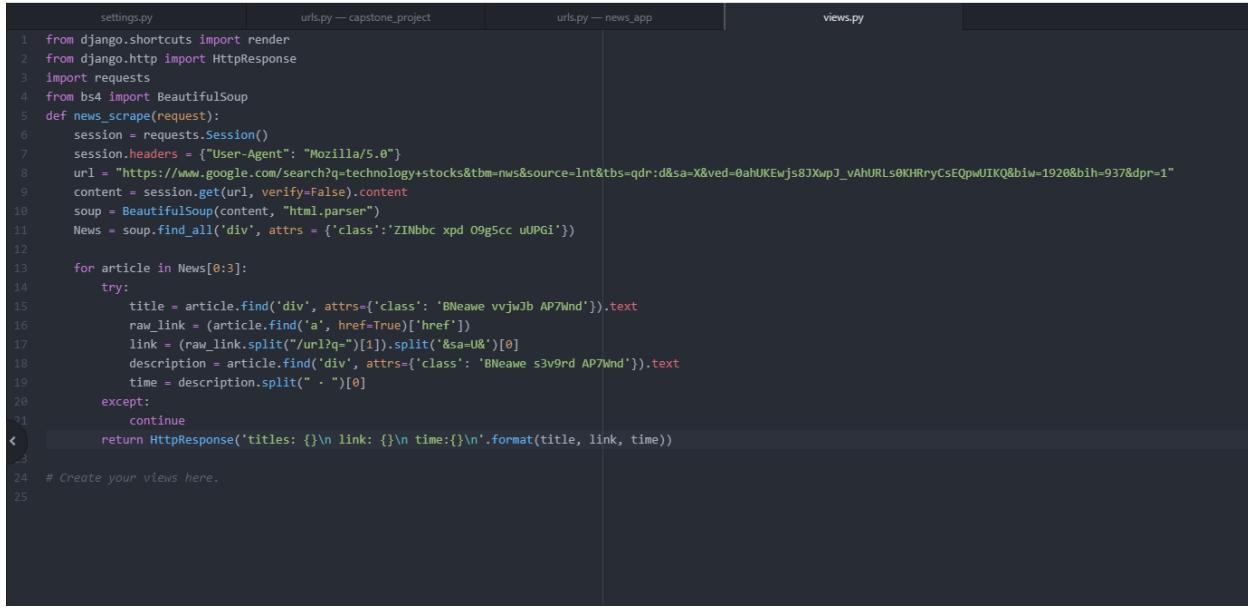
```

Request information

USER	[unable to retrieve the current user]
GET	No GET data
POST	No POST data
FILES	No FILES data
COOKIES	Variable Value csrf_token: '880d4f92d616PyU2qBMsu599Pg1DQHtHzy4152nZFH5xmQeKz77eZ0ZKV16mJU' sessionid: '3xrdmekmffof8f246nsh4kj1qf9ddzwz'
META	Variable Value

4:24 PM

Figure 3.3.2-2-ERROR- forgetting to use "requests" parameter in new function



```

1  from django.shortcuts import render
2  from django.http import HttpResponseRedirect
3  import requests
4  from bs4 import BeautifulSoup
5  def news_scrape(request):
6      session = requests.Session()
7      session.headers = {"User-Agent": "Mozilla/5.0"}
8      url = "https://www.google.com/search?q=technology+stocks&tbo=nws&source=lnt&tbs=qdr:d&sa=X&ved=0ahUKEwjs8JXwpJ_vAhURLs0KHRryCsEOpwUIKQ&biw=1920&bih=937&dpr=1"
9      content = session.get(url, verify=False).content
10     soup = BeautifulSoup(content, "html.parser")
11     News = soup.find_all('div', attrs = {'class':'ZINbbc xpd O9g5cc uUPGi'})
12
13     for article in News[0:3]:
14         try:
15             title = article.find('div', attrs={'class': 'BNeawe vvjwJb AP7Wnd'}).text
16             raw_link = (article.find('a', href=True)['href'])
17             link = (raw_link.split("/url?q=")[1]).split('&sa=U&t')[-1]
18             description = article.find('div', attrs={'class': 'BNeawe s3v9rd AP7Wnd'}).text
19             time = description.split(" · ")[0]
20         except:
21             continue
22     return HttpResponseRedirect('titles: {} \n link: {} \n time: {} \n'.format(title, link, time))
23
24 # Create your views here.
25

```

Figure 3.3.2-3 successfully listed titles in browser using `HttpResponse`.

Next, I repeated the process for the other two web scraping scripts by placing them inside the `news_scrape` function and successfully rendered data to the browser with both. I have been extensively researching the proper way to get/request data in Django, and found that “`render`”, “`redirect`”, and “`HttpResponse`” are the most commonly used methods. However, unless your application is extremely small, using the “`HttpResponse`” method can get quite repetitive, as well as extensive. Since I was aware that my website would contain multiple different pages (HTML templates), I chose to use the `render` method for requesting specific data for HTML templates.

4.3.3 Introducing Templates with Bootstrap

As previously mentioned, since I intended on creating multiple webpages inside my website, it is conventional to create a “`templates`” folder inside my Django project directory (`capstone_project`), which will contain all of my static HTML/CSS files. However, after the new folder has been created, I had to configure the Django project “`settings.py`” file to look for my html files in the “`templates`” directory. Next, I created my first template “`index.html`”, and will be using the “`Bootstrap`” for my front-end framework.

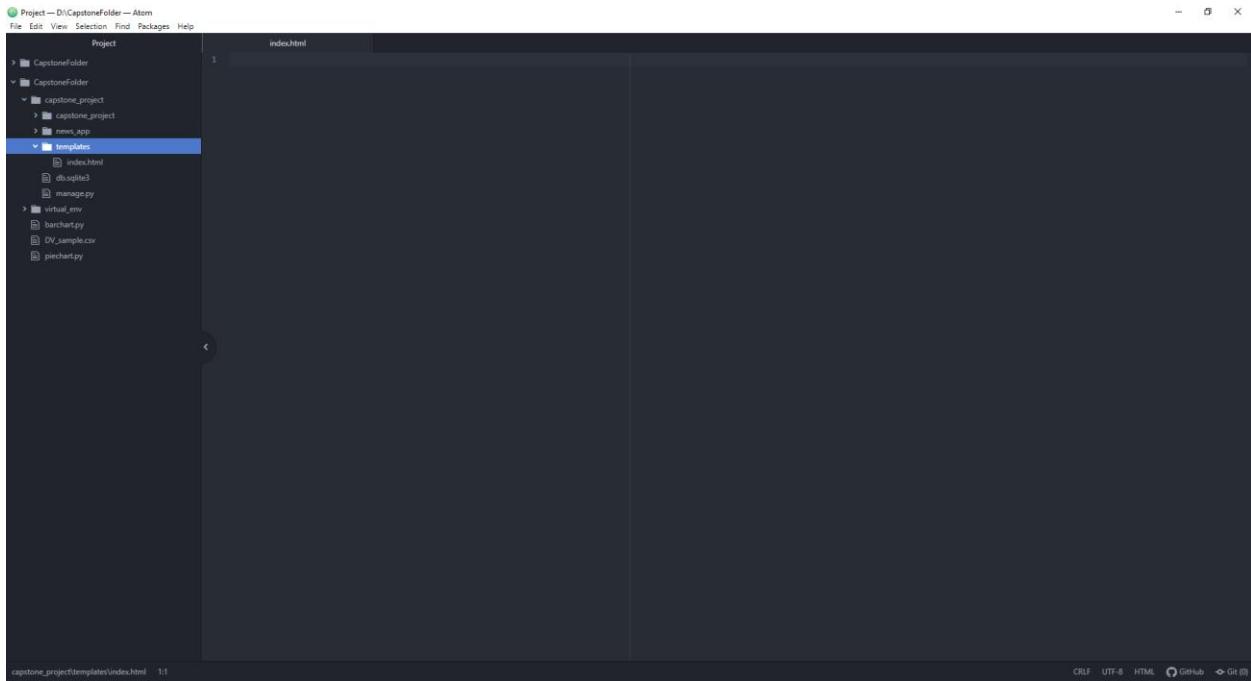


Figure 3.3.3-1 creating first template inside the new directory for templates.

Bootstrap is the most popular CSS Framework for developing responsive and mobile-first websites. It's an open-source framework that contains numerous pre-created templates for users to copy and customize to their liking. For starters, I simply went their [website](#), and copied/pasted their start template to my index template.

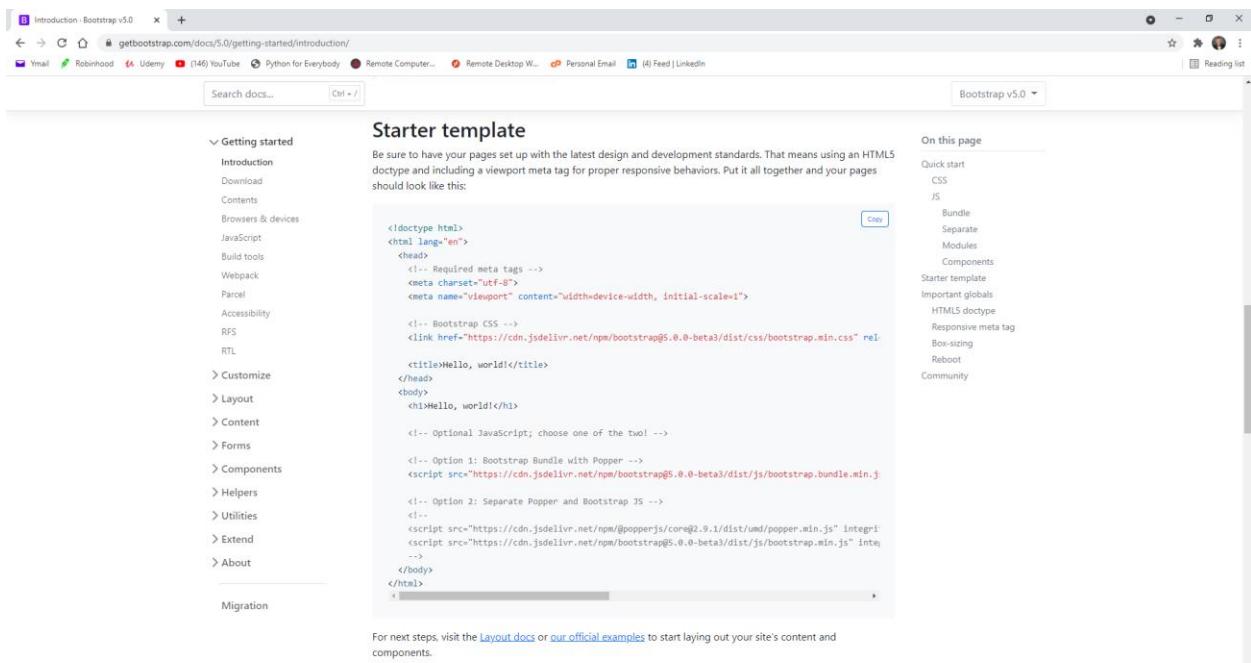


Figure 3.3.3-2- Copying the Bootstrap starter template

```

index.html — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project
  CapstoneFolder
    CapstoneProject
      capstone_project
        __pycache__
          __init__.py
          asgi.py
          settings.py
          urls.py
          wsgi.py
        news_app
          __pycache__
          migrations
          __init__.py
          admin.py
          apps.py
          models.py
          tests.py
          urls.py
          views.py
        templates
          index.html
          db.sqlite3
          manage.py
      virtual_env
        batcsh.py
        DV_sample.csv
        picsh.py
  index.html
  settings.py
  index.html
  views.py

```

```

1  !DOCTYPE html
2  <html lang="en"
3  <head>
4  <!-- Required meta tags -->
5  <meta charset="utf-8"
6  <meta name="viewport" content="width=device-width, initial-scale=1"
7
8  </head>
9  <body>
10 <h1>Hello, world!</h1>
11 </body>
12
13
14
15  <!-- Optional JavaScript; choose one of the two -->
16  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js" integrity="sha384-JE0xhjzj2d7u95t/8v4/8+X1+q5Q7PbOoF1qr5Q1IqPiqEh8fJ17PbIwv9yZ5O5P4" crossorigin="anonymous">
17  </script>
18  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js" integrity="sha384-JE0xhjzj2d7u95t/8v4/8+X1+q5Q7PbOoF1qr5Q1IqPiqEh8fJ17PbIwv9yZ5O5P4" crossorigin="anonymous">
19  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js" integrity="sha384-JE0xhjzj2d7u95t/8v4/8+X1+q5Q7PbOoF1qr5Q1IqPiqEh8fJ17PbIwv9yZ5O5P4" crossorigin="anonymous">
20  </script>
21
22
23
24
25
26  </body>
27
28
29
30
31
32
33
34
35
36
37
38
39
39

```

Figure 3.3.3-2 index.html with title "hello world"

The next step in the process is to implement the “render” function inside my views.py file in order to return the new “index.html” file in the browser.

```

Introduction · Bootstrap v5.0
TemplateDoesNotExist at / 127.0.0.1:8000
Ymail Robinhood Udemy (146) YouTube Python for Everybody Remote Computer... Remote Desktop W... Personal Email (4) Feed | LinkedIn Reading list

TemplateDoesNotExist at /
index.html

Request Method: GET
Request URL: http://127.0.0.1:8000/
Django Version: 3.2
Exception Type: TemplateDoesNotExist
Exception Value: index.html
Exception Location: F:\CapstoneFolder\virtual_env\lib\site-packages\django\template\loader.py line 19, in get_template
Python Executable: F:\CapstoneFolder\virtual_env\Scripts\python.exe
Python Version: 3.8.5
Python Path: ['F:\CapstoneFolder\CapstoneProject', 'C:\Users\linc01\AppData\Local\Programs\Python\Python38-32\python38.zip', 'C:\Users\linc01\AppData\Local\Programs\Python\Python38-32\DLLs', 'C:\Users\linc01\AppData\Local\Programs\Python\Python38-32\lib', 'C:\Users\linc01\AppData\Local\Programs\Python\Python38-32\lib\site-packages', 'F:\CapstoneFolder\virtual_env\lib\site-packages']
Server time: Sat, 10 Apr 2021 01:29:48 +0000

Template-loader postmortem

Django tried loading these templates, in this order:
Using engine django:
• django.template.loaders.app_directories.Loader: F:\CapstoneFolder\virtual_env\lib\site-packages\django\contrib\admin\templates\index.html (Source does not exist)
• django.template.loaders.app_directories.Loader: F:\CapstoneFolder\virtual_env\lib\site-packages\django\contrib\auth\templates\index.html (Source does not exist)

Traceback Switch to copy-and-paste view

F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\handlers\exception.py line 47, in inner
 47.     response = get_response(request)
▶ Local vars

F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\handlers\base.py line 181, in _get_response
 181.     response = wrapped_callback(request, *callback_args, **callback_kwargs)
▶ Local vars

F:\CapstoneFolder\CapstoneProject\news_app\views.py line 25, in index
 25.     return render(request, "index.html")
▶ Local vars

F:\CapstoneFolder\virtual_env\lib\site-packages\django\shortcuts.py line 19, in render
 39.     content = loader.render_to_string(template_name, context, request, using=using)
▶ Local vars

```

Figure 3.3.3-3 ERROR- "index.html" template does not exist.

The error message above is being displayed because the server cannot find my templates. This is because I did not properly configure the settings.py file to point to the “templates” directory. By creating the variable shown on line 17 below, and then adding to the variable “TEMPLATES” list. Django then

knows to look inside the project directory for a folder named “templates”, for rendering HTML files. This is all credited to Django’s unique “MVT” paradigm.

Figure 3.3.3-4 adding new variable to project templates

After making the previous changes, and reloading the webpage, the “index.html” file containing the bootstrap stater template was successfully rendered.

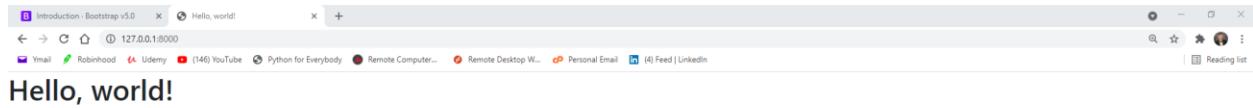
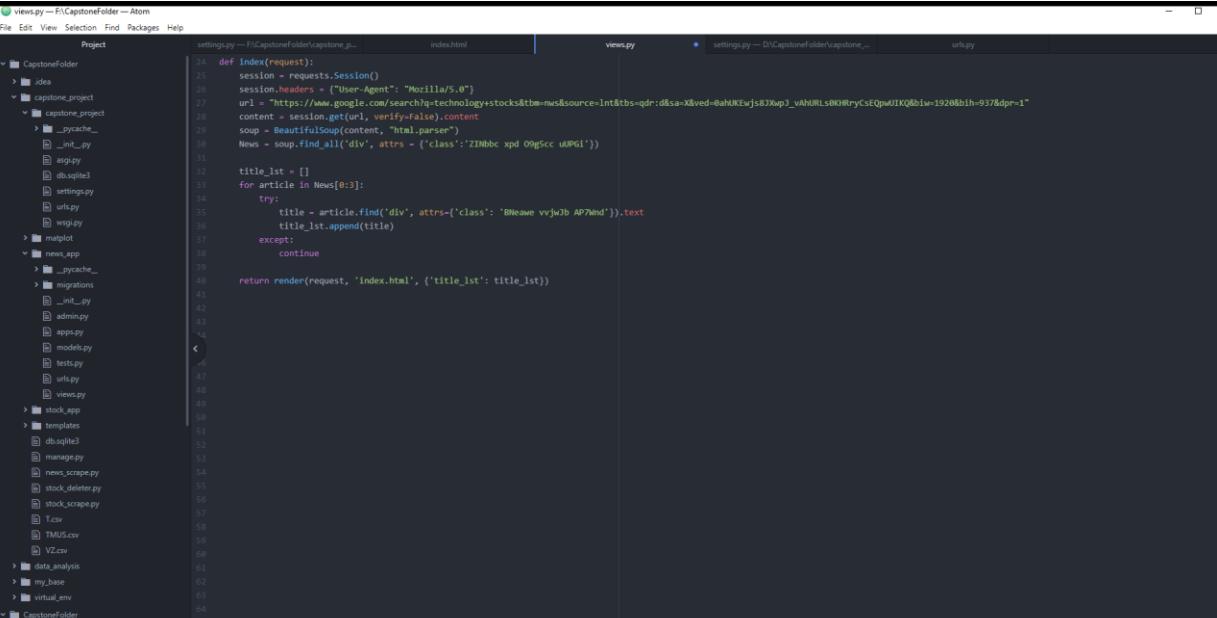


Figure 3.3.3-4 Rendering my first HTML template.

4.3.4 Using Django Context Processors

One of the most powerful, useful features of Django are their context processors. These processors are written in a language of their own and enable users to uniquely populate web pages with data generated by the views when a template is rendered with a request. Essentially, connecting python's backend capabilities with front-end language capabilities. This unique language consists of two main operators within its syntax: { % block content % } for using python logic inside HTML files, and { {{content}} } for printing data generated from the python function in the views.py file. For experimental purposes I will place the titles from one of my scraping functions inside the views file, append them to a list, place proper context processors in the index.html webpage and attempt displaying the titles in the browser.



views.py — F:\CapstoneFolder\capstone_p... File Edit View Selection Find Packages Help

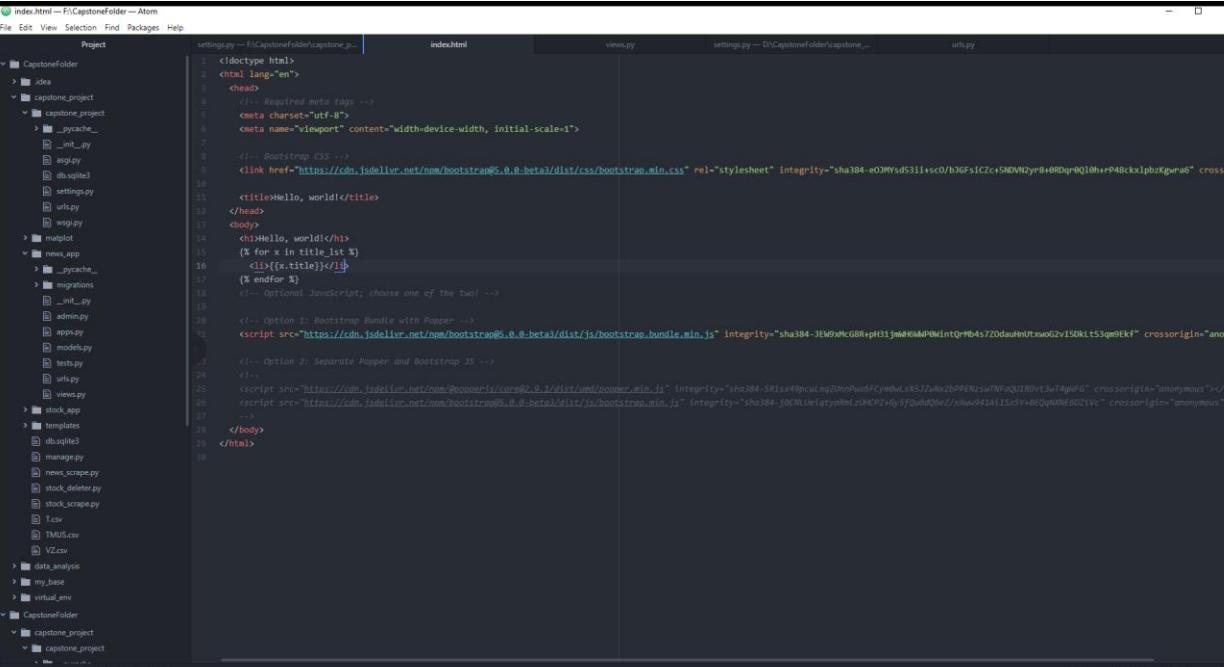
Project

- CapstoneFolder
- idea
- capstone_project
- capstone_project
 - __pycache__
 - __init__.py
 - asgi.py
 - db.sqlite3
 - settings.py
 - urls.py
 - wsgi.py
 - matplotlib
 - news_app
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - stock_app
 - templates
 - db.sqlite3
 - manage.py
 - news_scrape.py
 - stock_deleter.py
 - stock_scrape.py
 - T.csv
 - TMUS.csv
 - VZ.csv
 - data_analysis
 - my_base
 - virtual_env
- CapstoneFolder
- capstone_project
- capstone_project
 - __init__.py

settings.py — F:\CapstoneFolder\capstone_p... index.html views.py settings.py — D:\CapstoneFolder\capstone_p... urls.py

```
def index(request):  
    session = requests.Session()  
    session.headers = {"User-Agent": "Mozilla/5.0"}  
    url = "https://www.google.com/search?q=technology+stocks&tbo=msa&source=lnrt&tbs=qdr:d&sa=X&ved=0ahUKEwjs8Jxap7_vtbhURLs0K3HRYcSQm4UIK0fbivw-1910&hl=hi&q=37&dpr=1"  
    content = session.get(url, verify=False).content  
    soup = BeautifulSoup(content, "html.parser")  
    news = soup.find_all('div', attrs = {'class':'ZINbbc xpd 09gHc c u0PGI'})  
  
    title_lst = []  
    for article in news[0:3]:  
        try:  
            title = article.find('div', attrs={'class': 'BNeawe vvjwB AP7Hnd'}).text  
            title_lst.append(title)  
        except:  
            continue  
  
    return render(request, 'index.html', {'title_lst': title_lst})
```

Figure 3.3.4-1 placing titles in a list in views.py.



index.html --> CapstoneFolder -- Atom

File Edit View Selection Find Packages Help

Project

- CapstoneFolder
 - idea
 - capstone_project
 - __init__.py
 - index.html
 - views.py
 - settings.py
 - urls.py
 - wsgi.py
 - mapplot
 - news_app
 - __init__.py
 - migrations
 - __init__.py
 - 0001_initial.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - stock_app
 - __init__.py
 - manage.py
 - news_scrape.py
 - stock_delete.py
 - stock_scrape.py
 - T.csv
 - TMUS.csv
 - VZ.csv
 - data_analysis
 - my_base
 - virtual_env
- CapstoneFolder
 - capstone_project
 - __init__.py

settings.py --> CapstoneFolder\capstone_project__init__.py

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <!-- Required meta tags -->
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7
8     <!-- Bootstrap CSS -->
9     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOHY5d3l1IqE4uQMyd3l1i+sc0/b3Gf1CzCz15DvN2y8+08Dq0Q10h+rP48ck1pbzXgurab" crossorigin="anonymous">
10
11   <title>Hello, world!</title>
12 </head>
13 <body>
14   <h1>Hello, world!</h1>
15   <% for x in title_list %>
16     <div>{{x.title}}</div>
17   <% endfor %>
18
19   <!-- Optional JavaScript; choose one of the two! -->
20
21   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-3E9g6K6phw31JqF01jM8kK4xVdqI6XzqVfZJGv0o6PvQ9L94pZ81PZUmZGqtUw8" crossorigin="anonymous">
22   </script>
23
24   <!-- Option 2: Separate Popper and Bootstrap JS -->
25
26   <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js" integrity="sha384-S81xa0OpcaLmQ2UmPwz9Y3v2Q3tD6IYUZq9iYmE7nR80wWzLMqfz62pWZD91r" crossorigin="anonymous">
27   </script>
28   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js" integrity="sha384-jOCNlUe/qyuhmzibk9Tz+g5Qd0d0eZ/syav941A115x5Y+0QqXNNE6DZlVc" crossorigin="anonymous">
29   </script>
30 </body>
```

index.html --> CapstoneFolder\capstone_project\index.html

views.py --> CapstoneFolder\capstone_project__init__.py

urls.py --> CapstoneFolder\capstone_project__init__.py

index.html --> CapstoneFolder\capstone_project\index.html

1626

CRLF UTF-8 HTML GitHub Git (8)

Figure 3.3.4-2 Using context Processors in index.html.



Figure 3.3.4-3 displaying the list of titles

After testing all scrape functions as demonstrated above, I ran into issues with displaying the most recent data as well as having to constantly reboot the server to update the content. Also, I anticipated that I would need to find a different technique in order to separate the article's content by *website*. Eventually, I realized this was because I had not yet utilized the “Model” phase of the Django architecture which involves storing/managing the data in relational database tables (SQLite3).

4.3.5 Creating a Django Model

On the instance of creating a Django application, inside the new application that is automatically created, there is a “models.py” file that is used for creating tables in the “SQLite3.db” database that comes with the Django project by default. In the following procedure I will create 3 database tables (one for each website) inside the “news_app/models.py” file, and then run the necessary commands to physically create the tables in the database.

Project — D:\CapstoneFolder — Atom

File Edit View Selection Find Packages Help

Project

models.py — F:\CapstoneFolder\capstone_p... models.py — D:\CapstoneFolder\capstone_p... admin.py

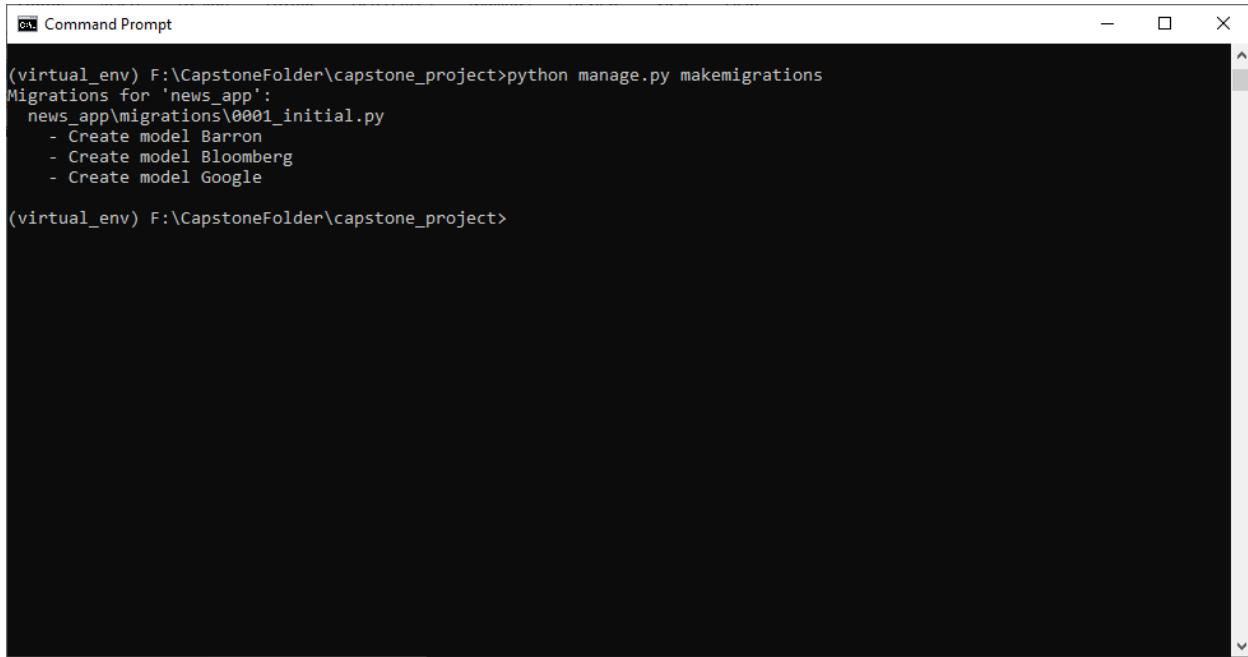
```
1  from django.db import models
2
3  # Create your models here.
4  class Barron(models.Model):
5      title = models.CharField(max_length=300)
6      time = models.CharField(max_length=300)
7      link = models.URLField()
8      def __str__(self):
9          return self.title
10
11 class Bloomberg(models.Model):
12     title = models.CharField(max_length=300)
13     time = models.CharField(max_length=300)
14     link = models.URLField()
15     def __str__(self):
16         return self.title
17
18
19 class Google(models.Model):
20     title = models.CharField(max_length=300)
21     time = models.CharField(max_length=300)
22     link = models.URLField()
23     def __str__(self):
24         return self.title
25
```

capstone_project\news_app\models.py 21/44

CRLF UTF-8 Python GitHub Git (0)

Figure 3.3.5-1 Creating Database Tables (models) for each website.

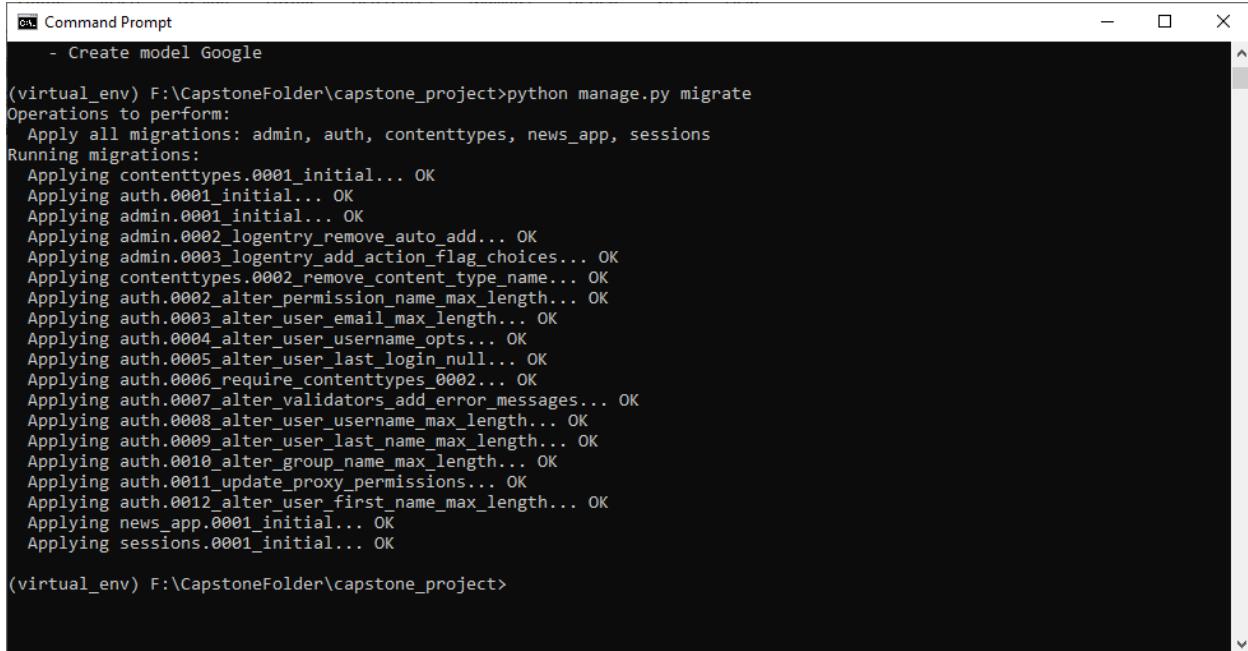
The code shown above contains 3 classes, each one represents a table for a different website, and each table contains 3 fields. The 1st field is created to store the article's title, the 2nd is for the time the article was posted, and the last field is a unique “URL field” that will be used to store the hyperlink to the actual article. Next, I will make my migrations in the command prompt to physically initialize and create the tables in the database and create a “super user” account. Then, by adding “/admin” to the end of my local IP address and logging in with the superuser credentials I previously created, I will be able to manage and view the tables through the administrator panel (GUI).



```
(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py makemigrations
Migrations for 'news_app':
  news_app\migrations\0001_initial.py
    - Create model Barron
    - Create model Bloomberg
    - Create model Google

(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 3.3.5-2 Making Migrations



```
- Create model Google

(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, news_app, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying news_app.0001_initial... OK
  Applying sessions.0001_initial... OK

(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 3.3.5-3 New tables successfully created

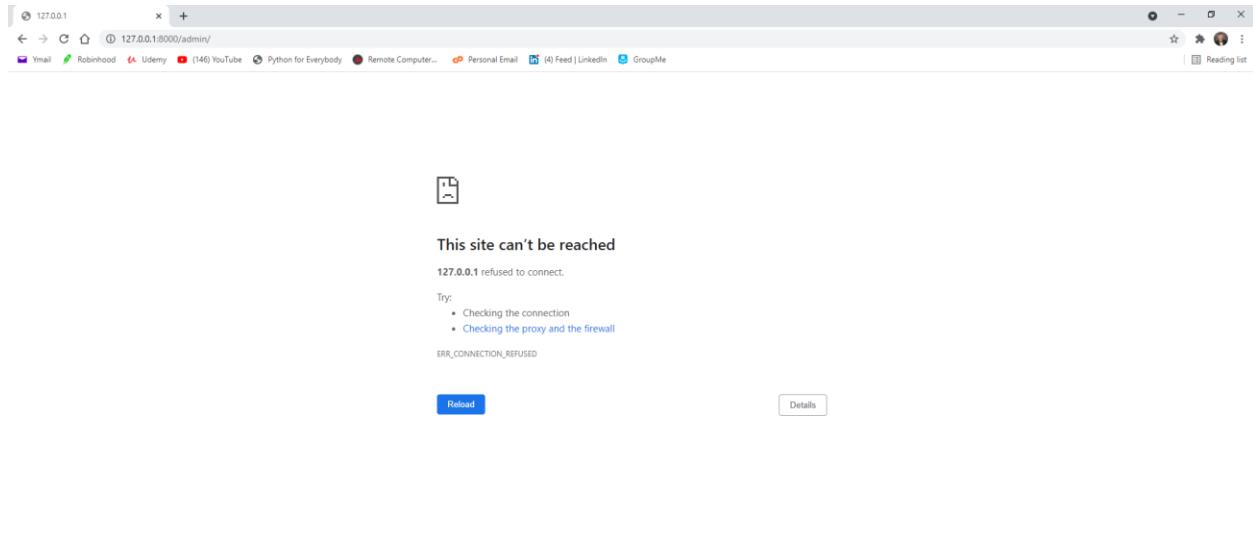
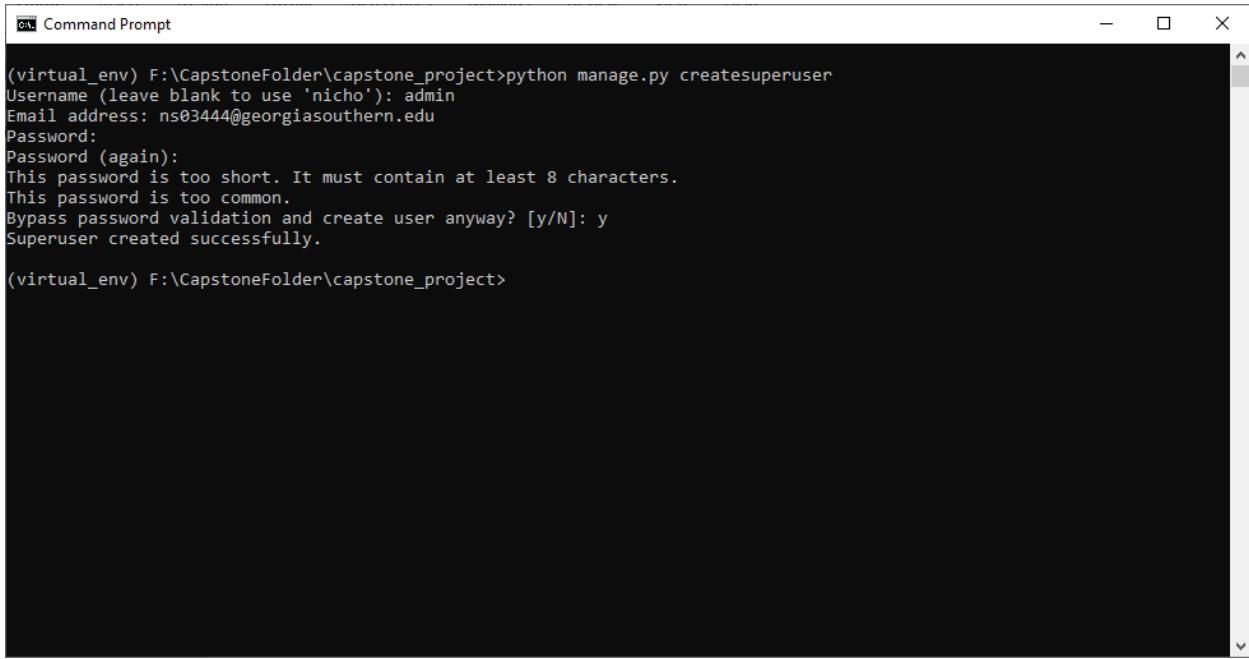


Figure 3.3.5-4 ERROR – can not log into admin panel



Figure 3.3.5-5 ERROR - admin URL returns same has webpage “index.html” URL returns.

I concluded that the previous two errors were raised due to not correctly defining URL paths and also because I did not create a super user account which is needed to log into the panel.



```
Command Prompt

(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py createsuperuser
Username (leave blank to use 'nicho'): admin
Email address: ns03444@georgiasouthern.edu
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 3.3.5-6 Creating super user for admin log in.

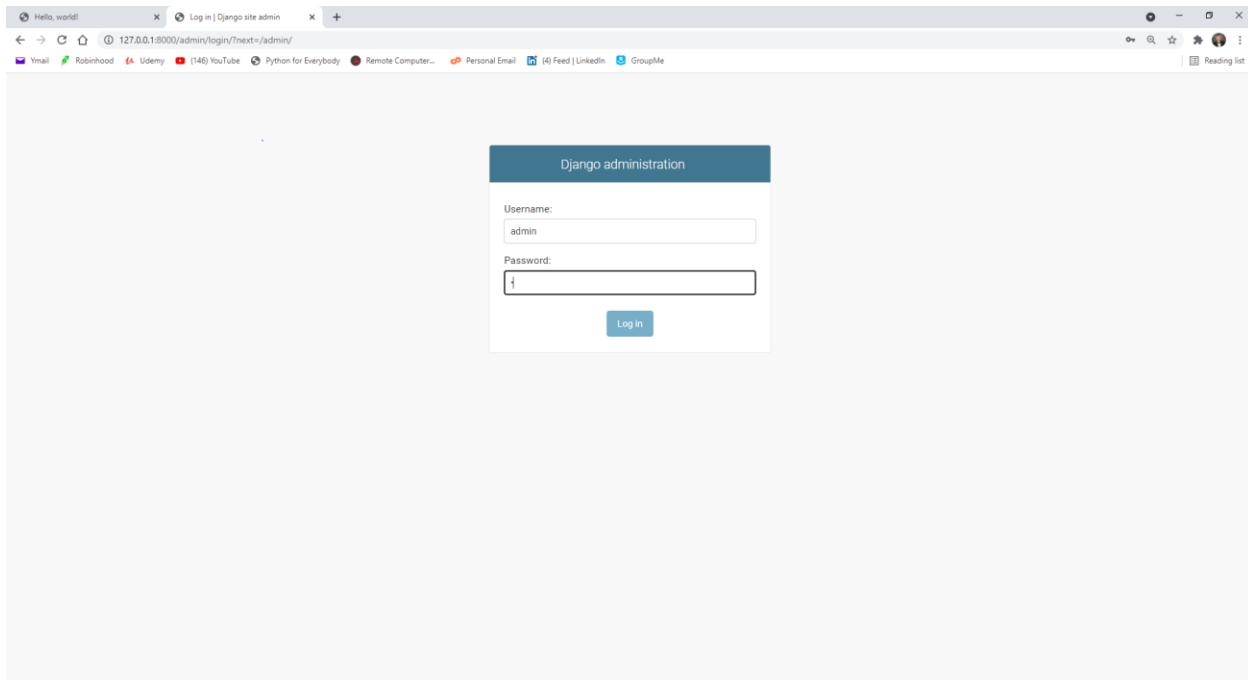


Figure 3.3.5-7 Logging into admin panel.

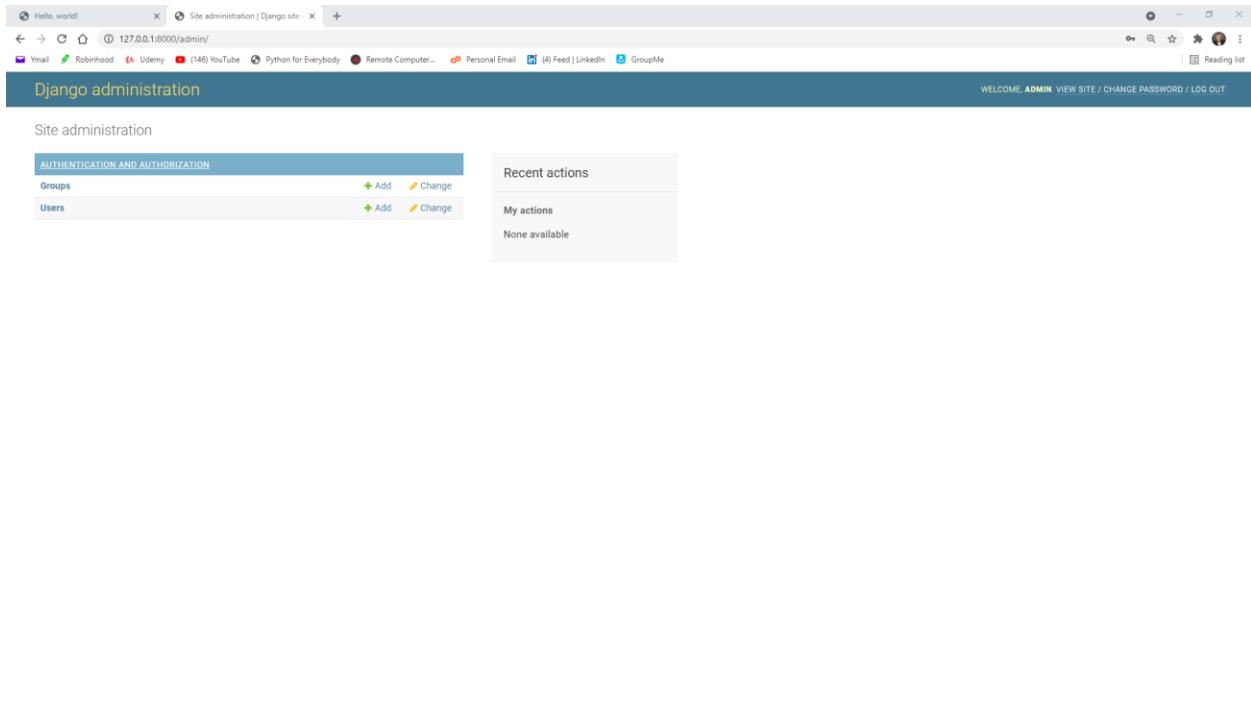


Figure 3.3.5-7 *ERROR - no tables in admin panel.*

After I finally managed to successfully log into the admin panel, the tables previously created were not busy. This was not expected, however, I realized I forgot to register the tables/models in the “news_app/admin.py” file.

File Edit View Selection Find Packages Help

Project

models.py

```
1 from django.contrib import admin
2 from news_app.models import Barron, Bloomberg, Google
3
4 admin.site.register(Barron)
5 admin.site.register(Bloomberg)
6 admin.site.register(Google)
7
8 # Register your models here.
9
10
```

settings.py

urls.py

admin.py

capstone_project

- __init__.py
- asgi.py
- db.sqlite3
- settings.py
- urls.py
- wsgi.py

news_app

- __init__.py
- asgi.py
- db.sqlite3
- manage.py
- news_scrape.py
- stock_deleter.py
- stock_scrape.py
- T.csv
- TMJS.csv
- V2.csv

data_analysis

my_base

virtual_env

CapstoneFolder

capstone_project

- __init__.py
- asgi.py
- db.sqlite3
- settings.py
- urls.py
- wsgi.py

news_app

- __init__.py

capstone_project/news_app/admin.py 8:1 (6, 149)

CRLF UTF-8 Python GitHub Git (0)

Figure 3.3.5-9 Registering models in admin.py file.

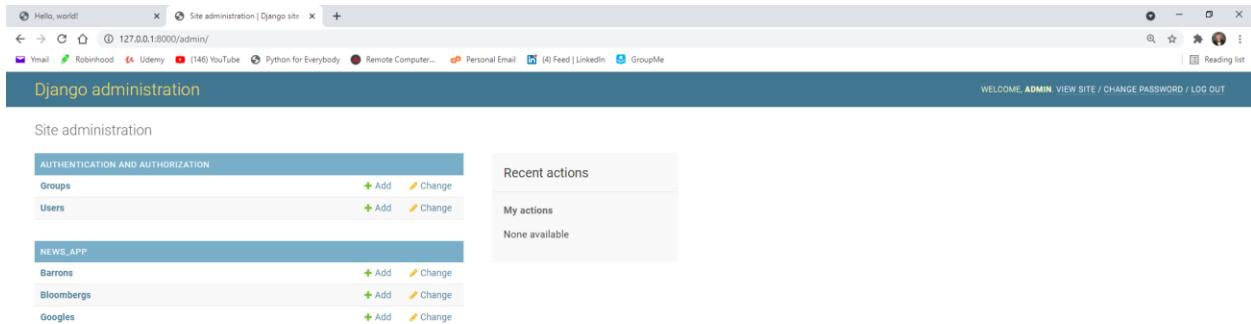


Figure 3.3.5-10 refreshed page, tables visible.

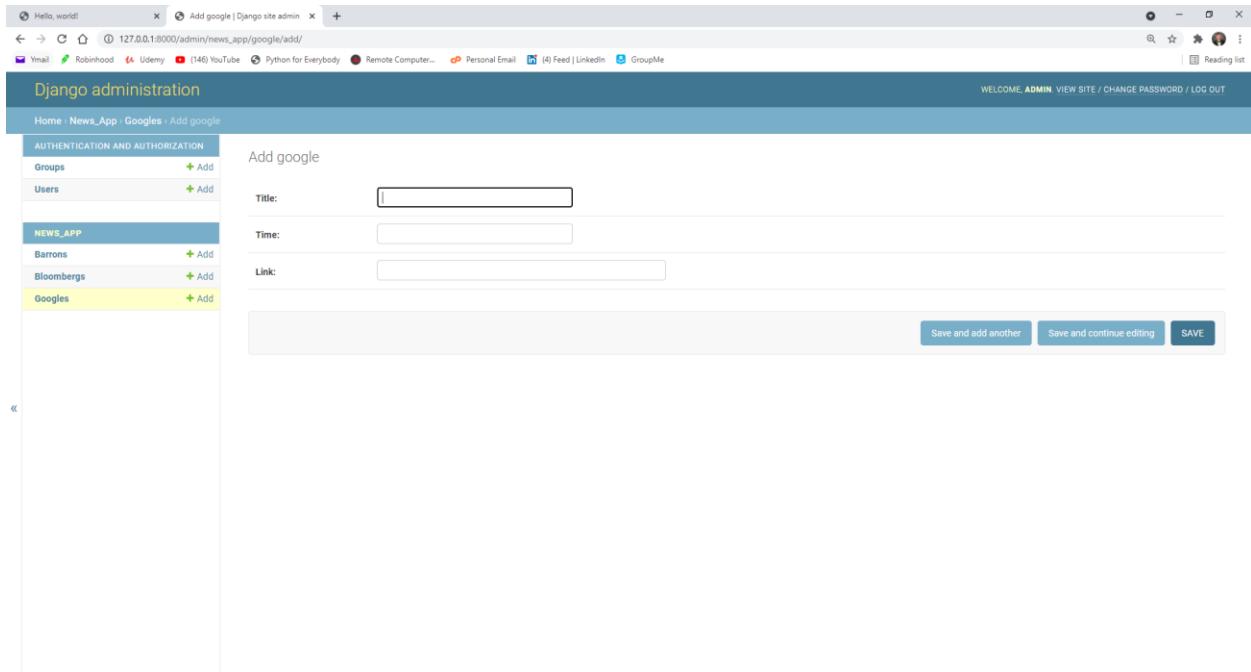
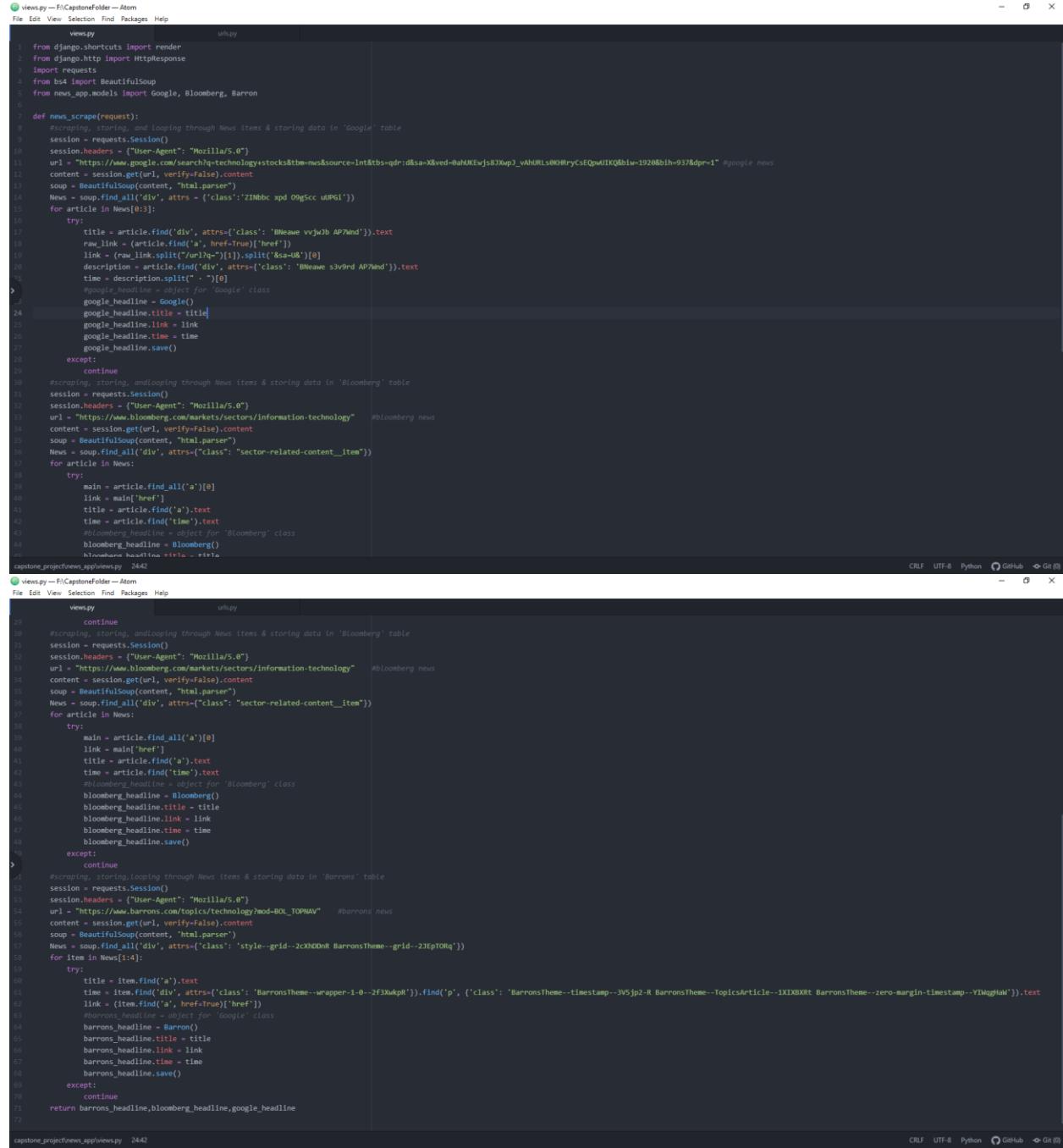


Figure 3.3.5-11 Exploring tables and insert feature through admin panel.

4.3.6 Populating the Database

Now that the tables are created, I will combine my scrape functions and store them inside a single function in the “views.py” file. Also, there will be additional code added to the end of each for loop in the function, to store the data in the appropriate tables. The next function in the “views.py” file called, “news_lst” will contain the request parameter and the context variables for which the items will be passed

to the template (index.html). However, before creating the next function, I will run the server to test my news_scrape function. If there are items in the database, then it has worked successfully.



The image shows two side-by-side code editors in the Atom text editor. Both editors have tabs for 'views.py' and 'urls.py'. The left editor contains the 'news_scrape' function for Google news, and the right editor contains the 'news_scrape' function for Bloomberg news. Both functions use requests.Session() to get URLs, BeautifulSoup to parse the content, and a custom class (e.g., Google_headline, Bloomberg_headline) to store the title, link, and time. The Bloomberg function also handles Barron's news by continuing the loop. The code is well-structured with comments explaining the scraping logic.

```

views.py -- F:\CapstoneFolder -- Atom
File Edit View Selection Find Packages Help
views.py          urls.py
1  from django.shortcuts import render
2  from django.http import HttpResponse
3  import requests
4  from bs4 import BeautifulSoup
5  from news_app.models import Google, Bloomberg, Barron
6
7  def news_scrape(request):
8      #scraping, storing, and looping through News items & storing data in 'Google' table
9      session = requests.Session()
10     session.headers = {"User-Agent": "Mozilla/5.0"}
11     url = "https://www.google.com/search?q=technology+stocks&tbs=qdr:d&sa=X&ved=0ahUKEwjs8J0up1_vAhURLs00RrycEqn0IKQ&hl=en-GB&pr=1" #google news
12     content = session.get(url, verify=False).content
13     soup = BeautifulSoup(content, "html.parser")
14     News = soup.find_all('div', attrs = {'class': 'ZINbbc xpd O9gGc c wUgPi'})
15     for article in News[0:3]:
16         try:
17             title = article.find('div', attrs={'class': 'BNeawe vvjwJb AP7Wnd'}).text
18             raw_link = (article.find('a', href=True))['href']
19             link = (raw_link.split('/url?q=')[1]).split('&sa=U&')[0]
20             description = article.find('div', attrs={'class': 'BNeawe s3v9rd AP7Wnd'}).text
21             time = description.split(' · ')[0]
22             #google_headline = object for 'Google' class
23             google_headline = Google()
24             google_headline.title = title
25             google_headline.link = link
26             google_headline.time = time
27             google_headline.save()
28         except:
29             continue
30     #scraping, storing, and looping through News items & storing data in 'Bloomberg' table
31     session = requests.Session()
32     session.headers = {"User-Agent": "Mozilla/5.0"}
33     url = "https://www.bloomberg.com/markets/sectors/information-technology" #Bloomberg news
34     content = session.get(url, verify=False).content
35     soup = BeautifulSoup(content, "html.parser")
36     News = soup.find_all('div', attrs={"class": "sector-related-content__item"})
37     for article in News:
38         try:
39             main = article.find_all('a')[0]
40             link = main['href']
41             title = article.find('a').text
42             time = article.find('time').text
43             #bloomberg_headline = object for 'Bloomberg' class
44             bloomberg_headline = Bloomberg()
45             bloomberg_headline.title = title
46             bloomberg_headline.link = link
47             bloomberg_headline.time = time
48             bloomberg_headline.save()
49         except:
50             continue
51     #scraping, storing, and looping through News items & storing data in 'Barrons' table
52     session = requests.Session()
53     session.headers = {"User-Agent": "Mozilla/5.0"}
54     url = "https://www.barrons.com/topics/technology?mod=B01_TOPNAV" #barrons news
55     content = session.get(url, verify=False).content
56     soup = BeautifulSoup(content, "html.parser")
57     News = soup.find_all('div', attrs={"class": "style--grid--2cXh0dnR BarronsTheme--grid--23EpT0Rq"})
58     for item in News[1:]:
59         try:
60             title = item.find('a').text
61             time = item.find('div', attrs={'class': 'BarronsTheme--wrapper-1-0-2f3XuKpI'}).find('p', {'class': 'BarronsTheme--timestamp-3V5jp2-R BarronsTheme--TopicsArticle--1XIXBXxt BarronsTheme--zero-margin-timestamp--YIwqgIAh'}).text
62             link = (item.find('a', href=True))['href']
63             #barrons_headline = object for 'Google' class
64             barrons_headline = Barron()
65             barrons_headline.title = title
66             barrons_headline.link = link
67             barrons_headline.time = time
68             barrons_headline.save()
69         except:
70             continue
71     return barrons_headline,bloomberg_headline,google_headline
72

```

Figure 3.3.6-1 - storing scrape functions inside news_scrape().

After running the server, I received the following server errors, as well as some unusual functionality with the database tables. After logging into the admin panel, I realized each table had data inserted an uneven number of times, as well, as storing identical items. I believe something is not working properly with the call of my “news_scrape” function. Furthermore, just moments later I noticed that number of items in each table is increasing as the server continues running.

Python Django Project - Learn to ... | Select google to change | Django | AttributeError at /

AttributeError at /

'tuple' object has no attribute 'get'

Request Method: GET
 Request URL: http://127.0.0.1:8000/
 Django Version: 3.2
 Exception Type: AttributeError
 Exception Value: "'tuple' object has no attribute 'get'"
 Exception Location: F:\CapstoneFolder\virtual_env\lib\site-packages\django\middleware\clickjacking.py, line 26, in process_response
 Python Executable: F:\CapstoneFolder\virtual_env\Scripts\python.exe
 Python Version: 3.8.5
 Python Path: ['F:\\CapstoneFolder\\virtual_env\\site-packages\\django\\core\\handlers\\exception.py', line 47, in inner
 47. response = get_response(request)
 ▶ Local vars
 F:\\CapstoneFolder\\virtual_env\\lib\\site-packages\\django\\utils\\deprecation.py, line 119, in __call__
 119. response = self.process_response(request, response)
 ▶ Local vars
 F:\\CapstoneFolder\\virtual_env\\lib\\site-packages\\django\\middleware\\clickjacking.py, line 26, in process_response
 26. if response.get('X-Frame-Options') is not None:
 ▶ Local vars

Server time: Mon, 12 Apr 2021 17:02:01 +0000

Traceback [Switch to copy-and-paste view](#)

```
F:\\CapstoneFolder\\virtual_env\\lib\\site-packages\\django\\core\\handlers\\exception.py, line 47, in inner
  47.     response = get_response(request)
▶ Local vars
F:\\CapstoneFolder\\virtual_env\\lib\\site-packages\\django\\utils\\deprecation.py, line 119, in __call__
  119.     response = self.process_response(request, response)
▶ Local vars
F:\\CapstoneFolder\\virtual_env\\lib\\site-packages\\django\\middleware\\clickjacking.py, line 26, in process_response
  26.     if response.get('X-Frame-Options') is not None:
▶ Local vars
```

Request information

USER admin
GET No GET data
POST No POST data
FILES No FILES data
COOKIES Variable Value

Figure 3.3.6-2 ERROR - 'tuple object has no get attribute'.

Python Django Project - Learn to ... | Select google to change | Django | AttributeError at /

127.0.0.1:8000/admin/news_app/google/

Authentication and Authorization

Groups + Add

Users + Add

News App

Barrons + Add

Bloomberg + Add

Googles + Add

Select google to change

Action: — Go 0 of 21 selected

GOOGLE

Is Microsoft (MSFT) Outperforming Other Computer and Technology ...

Has Micron Technology (MU) Outpaced Other Computer and ...

11 Information Technology Stocks Moving in Monday's Pre-Market ...

Investing in China tech stocks is partly a bet on Beijing's regulatory attitude, says NYU's Damodaran

Has Micron Technology (MU) Outpaced Other Computer and ...

11 Information Technology Stocks Moving in Monday's Pre-Market ...

Is Microsoft (MSFT) Outperforming Other Computer and Technology ...

Has Micron Technology (MU) Outpaced Other Computer and ...

11 Information Technology Stocks Moving in Monday's Pre-Market ...

Investing in China tech stocks is partly a bet on Beijing's regulatory attitude, says NYU's Damodaran

Has Micron Technology (MU) Outpaced Other Computer and ...

11 Information Technology Stocks Moving in Monday's Pre-Market ...

Is Microsoft (MSFT) Outperforming Other Computer and Technology ...

Has Micron Technology (MU) Outpaced Other Computer and ...

11 Information Technology Stocks Moving in Monday's Pre-Market ...

Is Microsoft (MSFT) Outperforming Other Computer and Technology ...

Has Micron Technology (MU) Outpaced Other Computer and ...

11 Information Technology Stocks Moving in Monday's Pre-Market ...

21 googles

Figure 3.3.6-3 google table with 21 items.

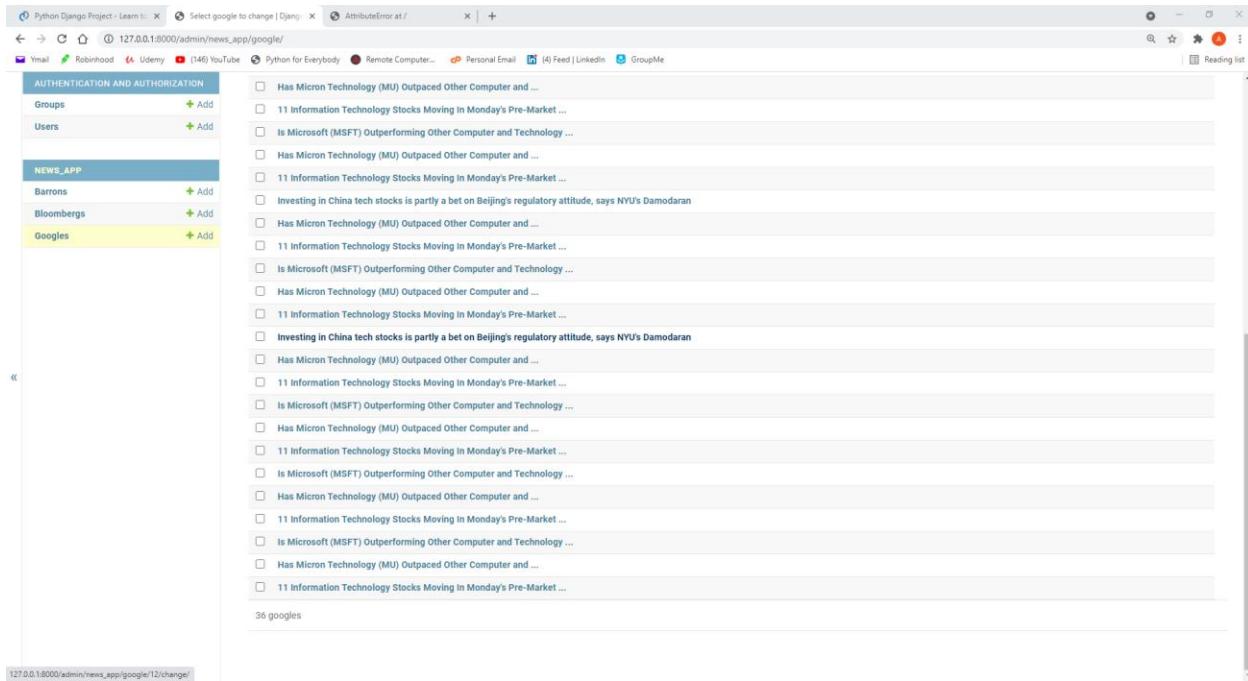


Figure 3.3.6-4 Google table seconds later with 36 items

Luckily, I can use the Django's admin feature to clear all the items from the database.

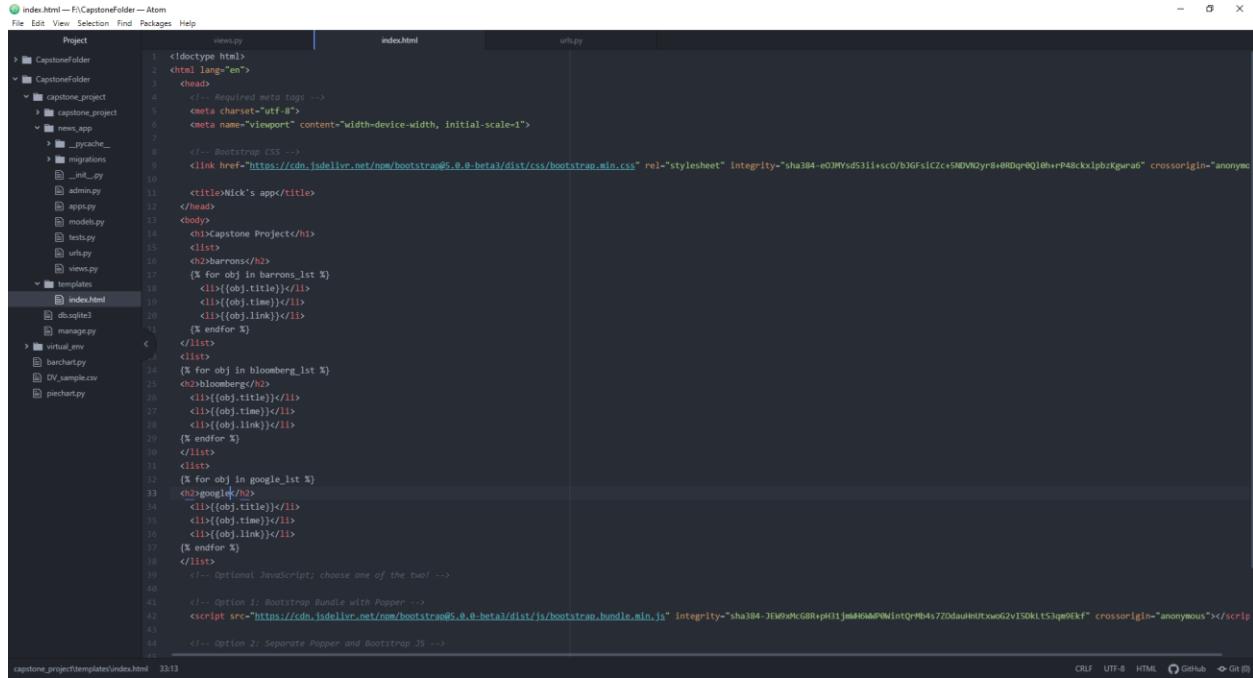
Next, I will create the “news_list” function in the “views.py” file and try passing some of the items to the template using context processors and hopefully it clears up some issues with the first function.

```

74
75 def news_list(request):
76     barrons_headline = Barron.objects.all()
77     barrons_headline = Bloomberg.objects.all()
78     google_headline = Google.objects.all()
79     context = {
80         'barrons_list': barrons_headline,
81         'bloomberg_list': bloomberg_headline,
82         'google_list': google_headline,
83     }
84     return render(request, "news/home.html", context)
85

```

Figure 3.3.6-5 Creating the “news_list()” function in the view.



The screenshot shows the Atom code editor with the following file structure:

- Project: CapstoneFolder
- File: index.html
- Views: views.py
- Index: index.html
- URLs: urls.py

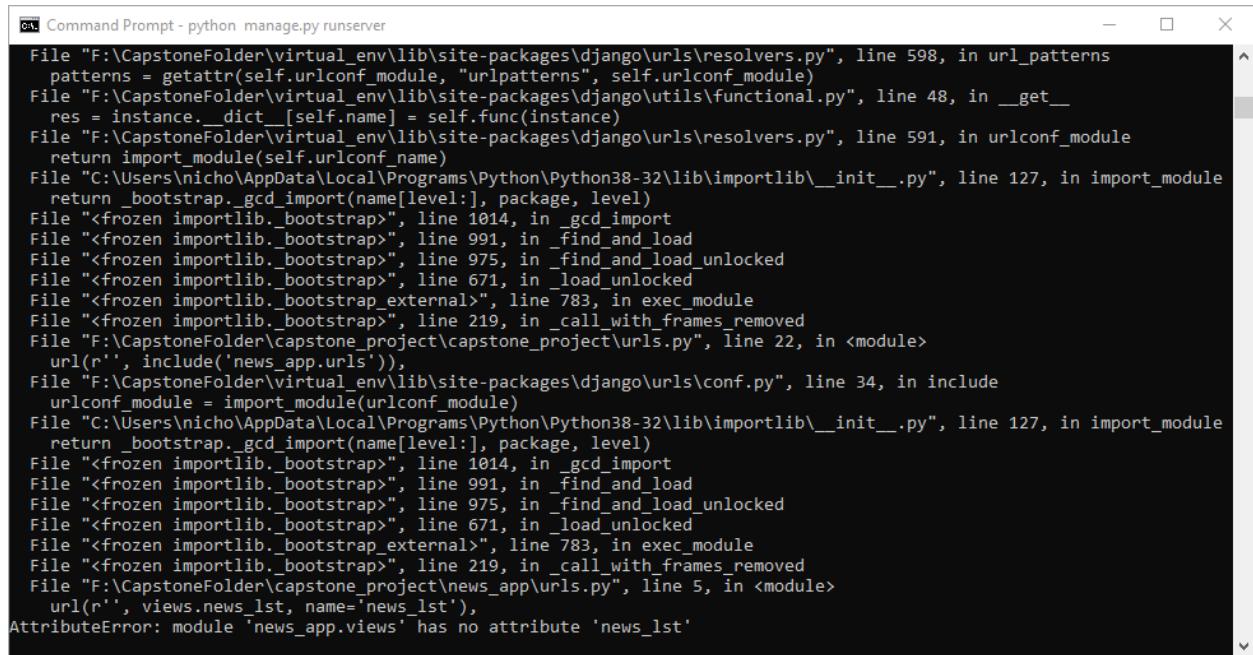
The content of index.html is as follows:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53li+sc0/bJsf1CE+5NDVlyZyrR8+0RDq0Q1h+rP48cxLpbzKgurub" crossorigin="anonymous">
<title>Nick's app</title>
</head>
<body>
    <h1>Capstone Project</h1>
    <ul>
        <li><h2>Barclays</h2>
            <% for obj in barclays_lst %>
                <li>{{obj.title}}</li>
                <li>{{obj.time}}</li>
                <li>{{obj.link}}</li>
            <% endfor %>
            </li>
        <li><h2>Bloomberg</h2>
            <% for obj in bloomberg %>
                <li>{{obj.title}}</li>
                <li>{{obj.time}}</li>
                <li>{{obj.link}}</li>
            <% endfor %>
            </li>
        <li><h2>Google</h2>
            <% for obj in google_lst %>
                <li>{{obj.title}}</li>
                <li>{{obj.time}}</li>
                <li>{{obj.link}}</li>
            <% endfor %>
            </li>
    <%-- Optional JavaScript; choose one of the two -->
    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG6HqtlwGI1/IT8z0v5B+Y1z24xJ14r49a6BfzWf2y35WzZ7F4" crossorigin="anonymous"></script>
    <!-- Option 2: Separate Popper and Bootstrap JS -->

```

Figure 3.3.6-6 Injecting article items in template using context processors



The screenshot shows the Command Prompt with the following error message:

```

File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 598, in url_patterns
    patterns = getattr(self.urlconf_module, "urlpatterns", self.urlconf_module)
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\utils\functional.py", line 48, in __get__
    res = instance.__dict__[self.name] = self.func(instance)
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 591, in urlconf_module
    return import_module(self.urlconf_name)
File "C:\Users\nicho\AppData\Local\Programs\Python\Python38-32\lib\importlib\_init_.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
File "<frozen importlib._bootstrap>", line 1014, in _gcd_import
File "<frozen importlib._bootstrap>", line 991, in _find_and_load
File "<frozen importlib._bootstrap>", line 975, in _find_and_load_unlocked
File "<frozen importlib._bootstrap>", line 671, in _load_unlocked
File "<frozen importlib._bootstrap_external>", line 783, in exec_module
File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
File "F:\CapstoneFolder\capstone_project\capstone_project\urls.py", line 22, in <module>
    url(r'', include('news_app.urls')),
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\conf.py", line 34, in include
    urlconf_module = import_module(urlconf_module)
File "C:\Users\nicho\AppData\Local\Programs\Python\Python38-32\lib\importlib\_init_.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
File "<frozen importlib._bootstrap>", line 1014, in _gcd_import
File "<frozen importlib._bootstrap>", line 991, in _find_and_load
File "<frozen importlib._bootstrap>", line 975, in _find_and_load_unlocked
File "<frozen importlib._bootstrap>", line 671, in _load_unlocked
File "<frozen importlib._bootstrap_external>", line 783, in exec_module
File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
File "F:\CapstoneFolder\capstone_project\news_app\urls.py", line 5, in <module>
    url(r'', views.news_lst, name='news_lst'),
AttributeError: module 'news_app.views' has no attribute 'news_lst'

```

Figure 3.3.6-7 ERROR – typo: misspelled function name in urls.py file



Capstone Project
barrons

Figure 3.3.6-8 Template rendered in browser, no items displayed.

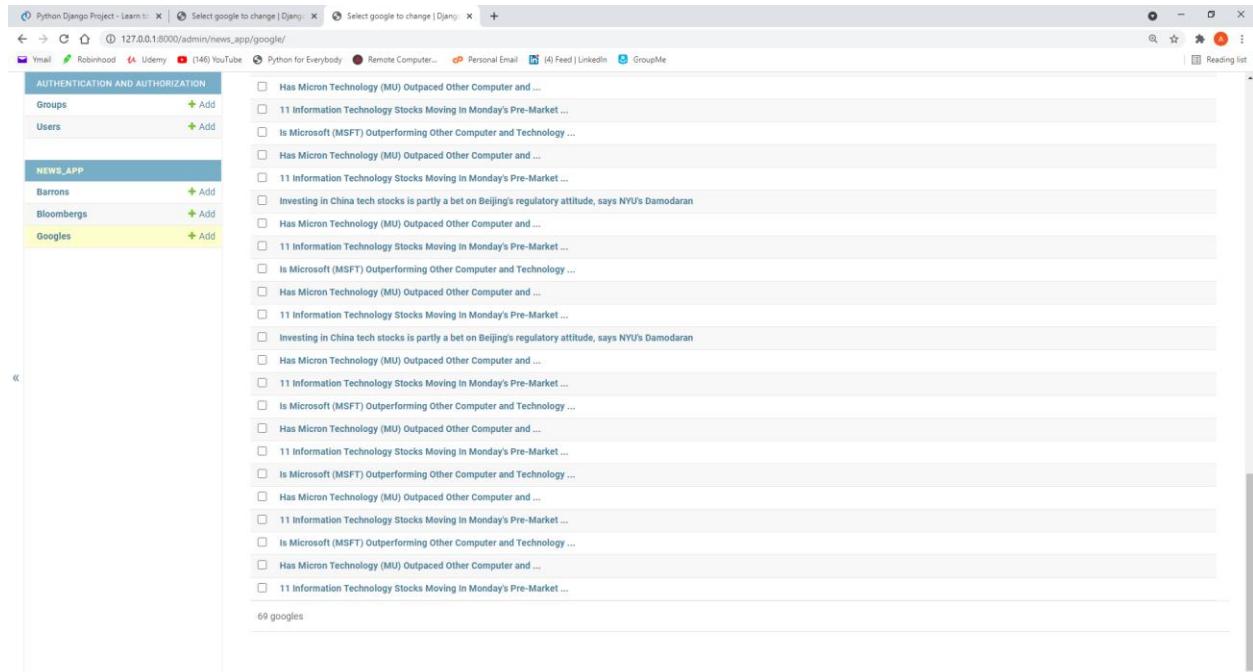
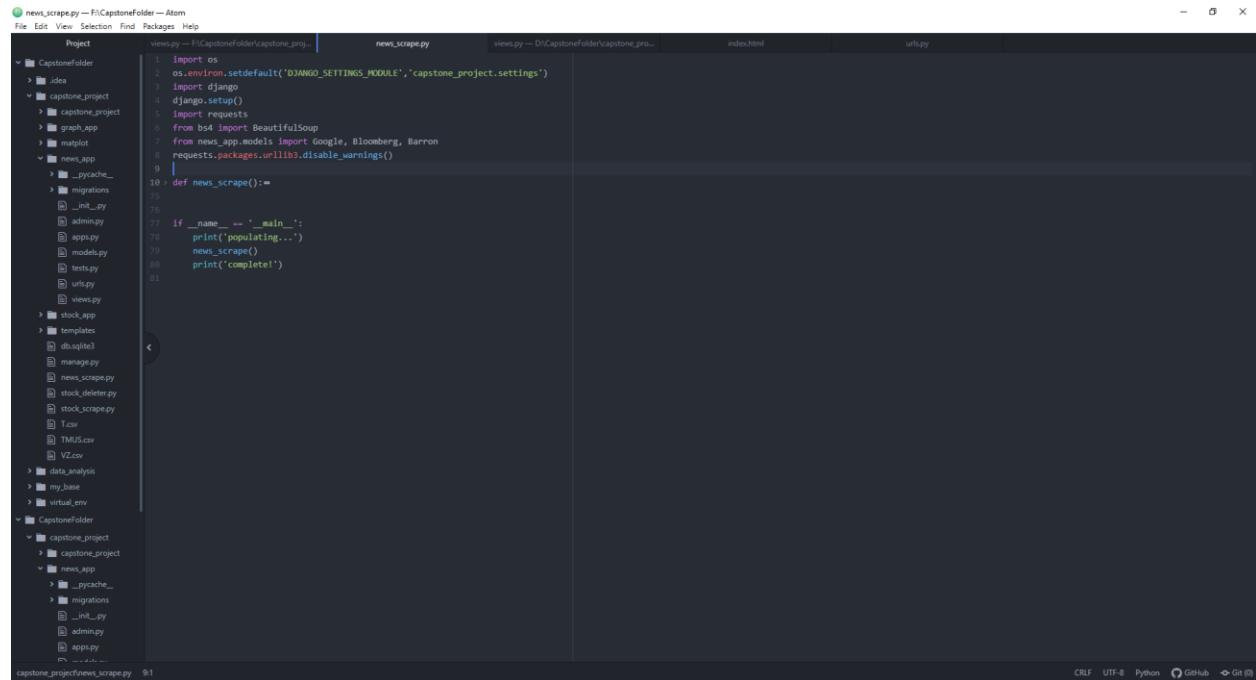


Figure 3.3.6-9 database continuing to populate while server is running.

I concluded that this must be happening because of Django's file directory structure and that my "views.py" file is calling the "news_scrape" function every time there is an http request in the back end. Therefore, I must find a different process for populating my database.

4.3.7 A different approach to populating the database.

After conducting some research to solve the issue I had been faced with, I realized the issue may be starting in my "views.py" file. Since this file is being ran (requested) while the server is running, I found that you can create a new python file inside the Django project directory (outside of application directory) to contain the "news_scrape" function along with some additional dependencies. This file runs as a normal python file inside the command prompt (inside the virtual environment) and is used to populate the database. The if statement at the end of the file is used to tell the python interpreter to check the module's source code to ensure that this is the main program to run.



The screenshot shows the Atom code editor with two tabs open: "news_scrape.py" and "views.py".

news_scrape.py content:

```
1  import os
2  os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'capstone_project.settings')
3
4  import django
5  django.setup()
6
7  import requests
8  from bs4 import BeautifulSoup
9  from news_app.models import Google, Bloomberg, Barron
10
11  requests.packages.urllib3.disable_warnings()
12
13  def news_scrape():
14
15      if __name__ == '__main__':
16          print('populating...')
17          news_scrape()
18          print('complete!')
```

views.py content:

```
1  from django.http import HttpResponse
2
3  def index(request):
4      return HttpResponse("Hello, world. You're at the polls index.")
```

Figure 3.3.7-1 Creating the new file named "news_scrape.py" to populate database.

The screenshot shows the Atom code editor with two tabs open: 'views.py' and 'news_scrape.py'. The 'views.py' tab contains the following code:

```

1  from django.shortcuts import render
2  from news_app.models import Barron, Bloomberg, Google
3
4  def index(request):
5      barrons_lst = Barron.objects.all()
6      bloomberg_lst = Bloomberg.objects.all()
7      google_lst = Google.objects.all()
8
9      context = {
10          'barrons_lst': barrons_lst,
11          'bloomberg_lst': bloomberg_lst,
12          'google_lst': google_lst,
13      }
14
15      return render(request, 'index.html', context)

```

The 'news_scrape.py' tab is partially visible. The file structure on the left shows a project named 'CapstoneFolder' with subfolders like 'idea', 'capstone_project', 'news_app', 'migrations', and 'stock_app'. The 'news_app' folder contains 'models.py', 'tests.py', 'urls.py', and 'views.py'.

Figure 3.3.7-2 Editing and renaming function in `views.py` as “`index()`”.

The screenshot shows a Command Prompt window with the following text:

```

Command Prompt

(virtual_env) F:\CapstoneFolder\capstone_project>python news_scrape.py
populating...
complete!

(virtual_env) F:\CapstoneFolder\capstone_project>

```

Figure 3.3.7-3 Successfully running “`news_scrape.py`”.

Now that my new script has ran successfully without any errors, it is time to log back into the admin panel and check the tables, then check the browser for “`index.html`” containing the new items.

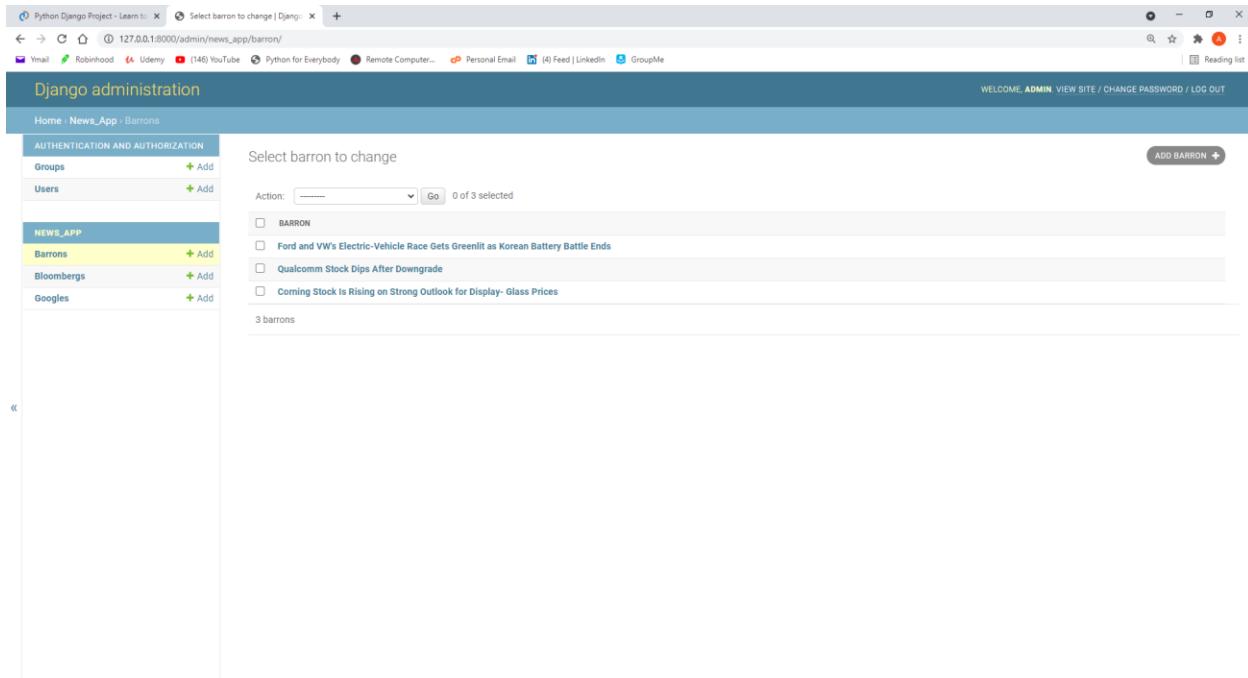


Figure 3.3.7-4 Populating Barron table.

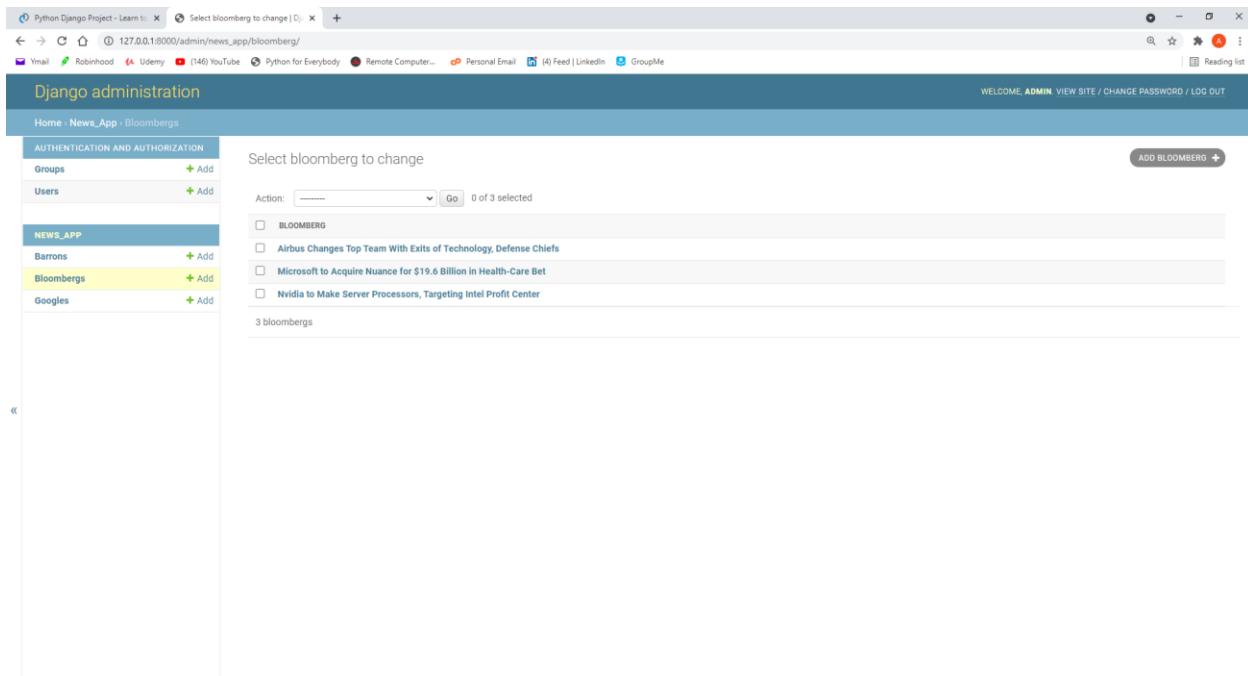


Figure 3.3.7-5 Populating Bloomberg table

Figure 3.3.7-6 Populating Google table

Capstone Project

barrons

- Corning Stock Is Rising on Strong Outlook for Display- Glass Prices
- April 12, 2021 5:26 pm ET
- <https://www.barrons.com/articles/corning-stock-is-rising-on-strong-outlook-for-display-glass-prices-51618248401?refsec=technology>
- Qualcomm Stock Dips After Downgrade
- April 12, 2021 5:22 pm ET
- <https://www.barrons.com/articles/qualcomm-stock-dips-after-downgrade-51618248135?refsec=technology>
- Ford and VW's Electric-Vehicle Race Gets Greenlit as Korean Battery Battle Ends
- April 12, 2021 5:17 pm ET
- <https://www.barrons.com/articles/how-the-end-to-a-south-korean-battery-battle-paves-the-way-for-ford-and-volkswagens-electric-vehicle-race-51618238766?refsec=technology>

bloomberg

- Nvidia to Make Server Processors, Targeting Intel Profit Center
- Updated 26 minutes ago
- <https://www.bloomberg.com/news/articles/2021-04-12/nvidia-to-make-central-processing-units-going-after-intel?srnd=infotech>

bloomberg

- Microsoft to Acquire Nuance for \$19.6 Billion in Health-Care Bet
- Updated an hour ago
- <https://www.bloomberg.com/news/articles/2021-04-12/microsoft-buys-nuance-for-19-7-billion-in-bet-on-health-care?srnd=infotech>

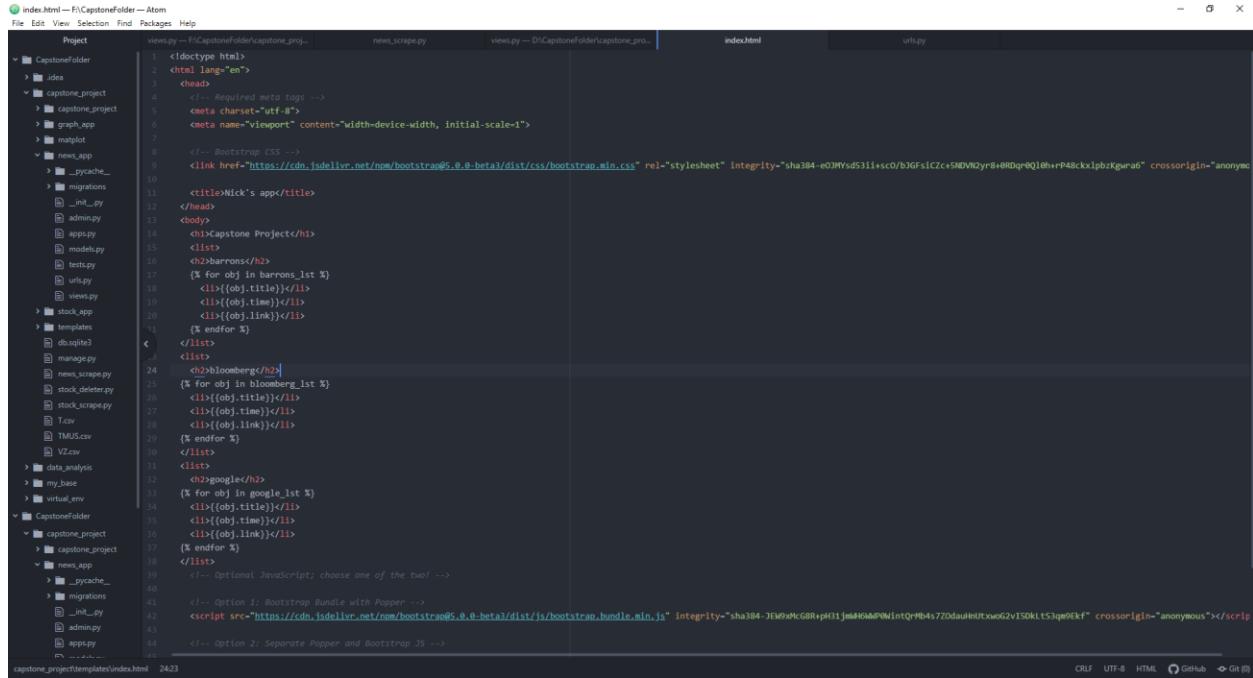
google

- 11 Information Technology Stocks Moving In Monday's Pre-Market ...
- 5 hours ago
- <https://www.benzinga.com/pre-market-outlook/21/04/20575933/11-information-technology-stocks-moving-in-mondays-pre-market-session>

google

- 12 Information Technology Stocks Moving In Monday's Intraday ...

Figure 3.3.7-7 News items displayed in browser, unformatted.

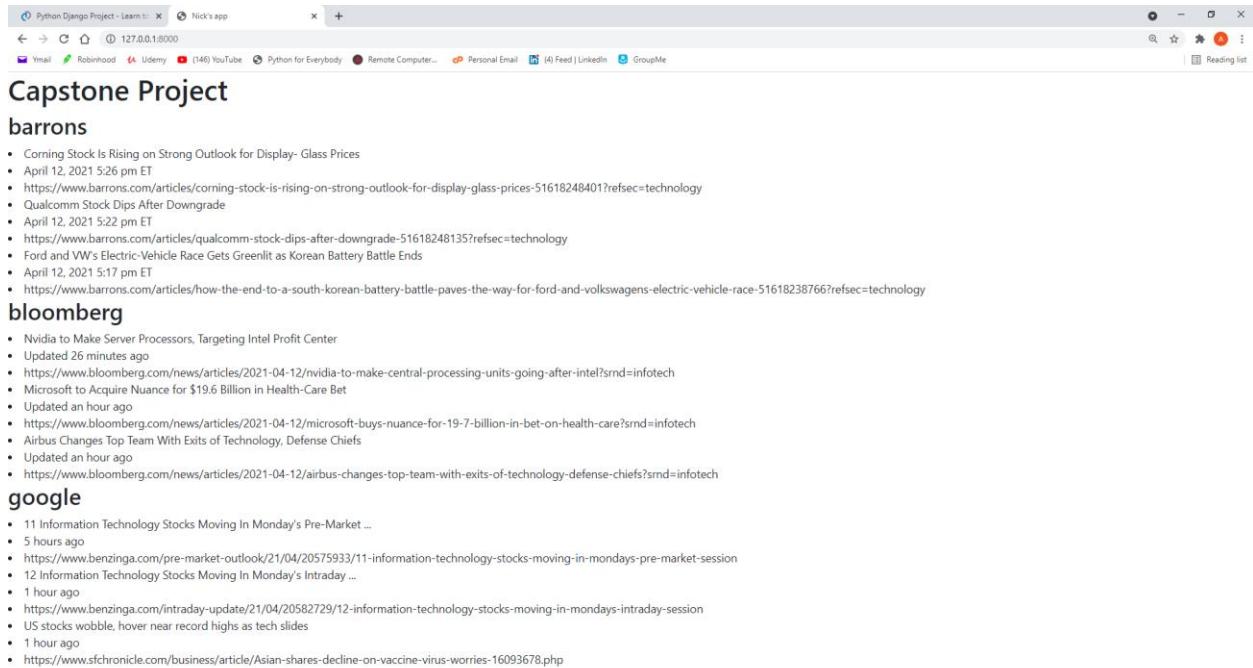


```

index.html — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project
views.py — F:\CapstoneFolder\capstone_pro... news_script.py — D:\CapstoneFolder\capstone_pro...
index.html
urls.py
index.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <!-- Required meta tags -->
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7
8   <!-- Bootstrap CSS -->
9   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOHYsD31i+sc0/bJGf51CZ+5NDvZyr8+0RDr0Q16hrP48cx1pbzKgurub" crossorigin="anonymous"/>
10
11   <title>Nick's app</title>
12 </head>
13 <body>
14   <h1>Capstone Project</h1>
15   <ul>
16     <li>barrows</li>
17     <% for obj in barrows_lst %>
18       <li>{obj.title}</li>
19       <li>{obj.time}</li>
20       <li>{obj.link}</li>
21     <% endfor %>
22   </ul>
23   <ul>
24     <li>bloomberg</li>
25     <% for obj in bloomberg_lst %>
26       <li>{obj.title}</li>
27       <li>{obj.time}</li>
28       <li>{obj.link}</li>
29     <% endfor %>
30   </ul>
31   <ul>
32     <li>google</li>
33     <% for obj in google_lst %>
34       <li>{obj.title}</li>
35       <li>{obj.time}</li>
36       <li>{obj.link}</li>
37     <% endfor %>
38   </ul>
39   <!-- Optional JavaScript; choose one of the two -->
40   <!-- Option 1: Bootstrap Bundle with Popper -->
41   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW0eGGB+ph0ijmHOdP0WnIntQmB4572daultxwG2v150kL53q9Eki" crossorigin="anonymous"></script>
42   <!-- Option 2: Separate Popper and Bootstrap JS -->

```

Figure 3.3.7-8 Moved titles outside of for loop to enhance formatting.



Capstone Project

barrons

- Corning Stock Is Rising on Strong Outlook for Display- Glass Prices
- April 12, 2021 5:26 pm ET
- <https://www.barrons.com/articles/coming-stock-is-rising-on-strong-outlook-for-display-glass-prices-51618248401?refsec=technology>
- Qualcomm Stock Dips After Downgrade
- April 12, 2021 5:22 pm ET
- <https://www.barrons.com/articles/qualcomm-stock-dips-after-downgrade-51618248135?refsec=technology>
- Ford and VW's Electric-Vehicle Race Gets Greenlit as Korean Battery Battle Ends
- April 12, 2021 5:17 pm ET
- <https://www.barrons.com/articles/how-the-end-to-a-south-korean-battery-battle-paves-the-way-for-ford-and-volkswagens-electric-vehicle-race-51618238766?refsec=technology>

bloomberg

- Nvidia to Make Server Processors, Targeting Intel Profit Center
- Updated 26 minutes ago
- <https://www.bloomberg.com/news/articles/2021-04-12/nvidia-to-make-central-processing-units-going-after-intel?srnd=infotech>
- Microsoft to Acquire Nuance for \$19.6 Billion in Health-Care Bet
- Updated an hour ago
- <https://www.bloomberg.com/news/articles/2021-04-12/microsoft-buys-nuance-for-19-7-billion-in-bet-on-health-care?srnd=infotech>
- Airbus Changes Top Team With Exits of Technology, Defense Chiefs
- Updated an hour ago
- <https://www.bloomberg.com/news/articles/2021-04-12/airbus-changes-top-team-with-exits-of-technology-defense-chiefs?srnd=infotech>

google

- 11 Information Technology Stocks Moving In Monday's Pre-Market ...
- 5 hours ago
- <https://www.benzinga.com/pre-market-outlook/21/04/20575933/11-information-technology-stocks-moving-in-mondays-pre-market-session>
- 12 Information Technology Stocks Moving In Monday's Intraday ...
- 1 hour ago
- <https://www.benzinga.com/intraday-update/21/04/20582729/12-information-technology-stocks-moving-in-mondays-intraday-session>
- US stocks wobble, hover near record highs as tech slides
- 1 hour ago
- <https://www.sfgate.com/business/article/Asian-shares-decline-on-vaccine-virus-worries-16093678.php>

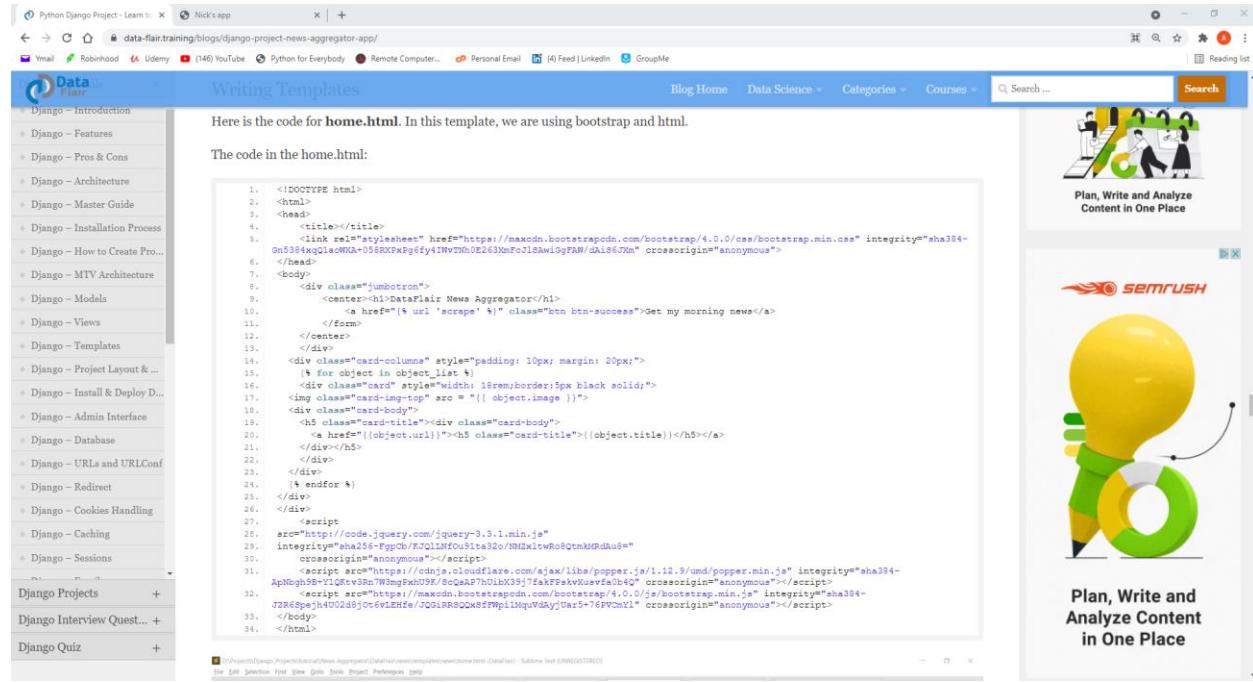
Figure 3.3.7-9 **MILESTONE ACHIEVED – RENDERED ALL ITEMS TO TEMPLATE IN SEPARATE LISTS FROM DATABASE.

4.4 Designing the News page

Now that the Django project has been created and the model, view, template system is properly functioning, I still need to make some front-end adjustments to make the webpage more presentable. Although I was fortunate enough to find an example template very similar to my need, I will have to do some digging in the bootstrap documentation to make all the adjustments needed.

4.4.1 Formatting the home page

Now that I have properly rendered the database objects to the template in the correct logic, it is time to explore bootstrap's excellent documentation to add some design features to the homepage. Luckily, without having to look very long, I found a bootstrap template specifically used for news articles being web scraped from a Django tutorial on the Data Flair website ([Link](#)).



The screenshot shows a browser window with the Data Flair website. The URL is `data-flair.training/blogs/django-project-news-aggregator-app/`. The page title is "Writing Templates". The content area displays a list of news articles with cards. On the right, there is a large graphic of a lightbulb and text that reads "Plan, Write and Analyze Content in One Place". The Sublime Text 3 interface is visible at the bottom of the browser window.

```
1.  !DOCTYPE html
2.  <html>
3.  <head>
4.  <title></title>
5.  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5544xqQlaXWQXpPgdy4IWvTNhOE263XmFcJ1SAwlgFAM+dA186Xm" crossorigin="anonymous">
6.  </head>
7.  <body>
8.  <div class="jumbotron">
9.  <center><h1>DataFlair News Aggregator</h1>
10. <a href="{% url 'scrape' %}" class="btn btn-success">Get my morning news</a>
11. </center>
12. </div>
13. <div class="card-columns" style="padding: 10px; margin: 20px;">
14. {% for object in object_list %}
15. <div class="card" style="width: 18rem; border: 5px black solid;">
16. 
18. <h5 class="card-title"><div class="card-body">
19. <a href="{{ object.url }}><h5 class="card-title">{{ object.title }}</h5></a>
20. </div></h5>
21. </div>
22. </div>
23. {% endfor %}
24. </div>
25. </div>
26. </div>
27. <script>
28. src="https://code.jquery.com/jquery-3.3.1.min.js"
29. integrity="sha256-FkpChX/K2L1Nfou1ta32c/NMU1twRo8QtmKMRdAu8="
30. crossorigin="anonymous"></script>
31. <script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh8YVidRIn7Sk3fD7tfWhIcMqPCx4Uf8tduK2e8DgkL+oXq5v8Q=" crossorigin="anonymous"></script>
32. <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6spejhTU02d506xLifc/2Q1RR8QCK6ffMpl1mquWshyJax5+6FNCwY1" crossorigin="anonymous"></script>
33. </body>
34. </html>
```

Figure 3.4.1-1 Data flair tutorial template.

After coping and pasting the html code shown above, I will edit the context processors to match the variables I created previously.

Figure 3.4.1-2 Editing the data flair template for my context processors

Python Django Project - Learn to ... NoReverseMatch at / 127.0.0.1:8000

Ymail Robinhood Udemy (146) YouTube Python for Everybody Remote Computer... Personal Email (4) Feed | LinkedIn GroupMe

Reading list

NoReverseMatch at /

Reverse for 'scrape' not found. 'scrape' is not a valid view function or pattern name.

Request Method: GET
Request URL: http://127.0.0.1:8000/
Django Version: 3.2
Exception Type: NoReverseMatch
Exception Value: Reverse for 'scrape' not found. 'scrape' is not a valid view function or pattern name.
Exception Location: F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py, line 694, in _reverse_with_prefix
Python Execution: F:\CapstoneFolder\virtual_env\Scripts\python.exe
Python Version: 3.8.5
Python Path: ['F:\\CapstoneFolder\\Capstone_Project', 'C:\\Users\\rich0\\AppData\\Local\\Programs\\Python\\Python38-32\\python38.zip', 'C:\\Users\\rich0\\AppData\\Local\\Programs\\Python\\Python38-32\\DLLs', 'C:\\Users\\rich0\\AppData\\Local\\Programs\\Python\\Python38-32\\lib', 'C:\\Users\\rich0\\AppData\\Local\\Programs\\Python\\Python38-32', 'F:\\CapstoneFolder\\virtual_env', 'F:\\CapstoneFolder\\virtual_env\\lib\\site-packages']
Server time: Mon, 12 April 2021 18:27:46 +0000

Error during template rendering

In template F:\CapstoneFolder\capstone_project\templates\index.html. Error at line 10

Reverse for 'scrape' not found. 'scrape' is not a valid view function or pattern name.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5BfPw2lxWQqQ0QlaoXKA+058RXPxpgfY4IuVTHhE263XmFcJLSw10gFA/dai567Xm" crossorigin="anonymous">
6 </head>
7 <body>
8   <div class="jumbotron">
9     <center>hi DataJair News Aggregator</h1>
10    <a href="{% url 'scrape' %}" class="btn btn-success">Get my morning news</a>
11  </form>
12 </center>
13 </div>
14 <h2>Barons News</h2>
15 <div class="card-columns" style="padding: 10px; margin: 20px;">
16  {% for object in barrons_list %}<br>
17  <div class="card" style="width: 10rem; border: 5px black solid;">
18  <div class="card-body">
19  <h5 class="card-title"><div class="card-body">
20    <a href="{% url 'link' %}"><h5 class="card-title">{{object.title}}</h5></a>
```

Figure 3.4.1-3 *ERROR – did not make all necessary changes to data flair template*

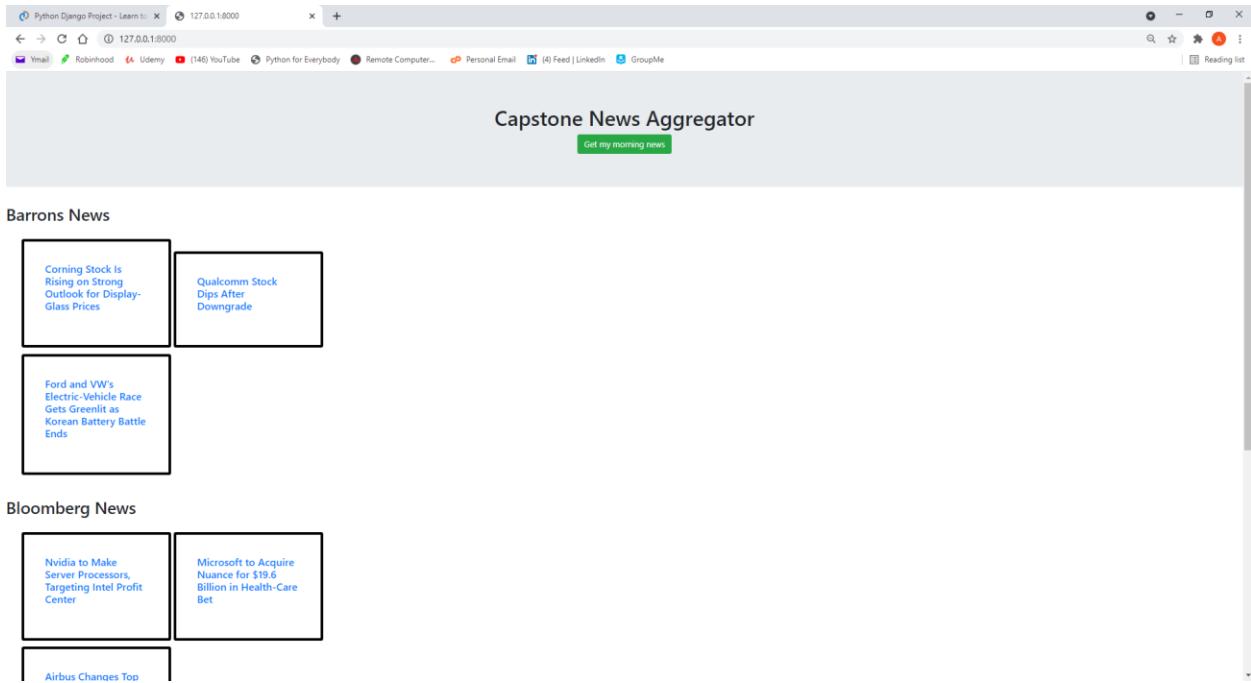


Figure 3.4.1-4 items returned with new features (cards, push button, header, etc.)

The next process just consists of exploring the bootstrap documentation to make small front-end adjustments and customizations to display the content to my personal pleasing.

4.4.2 Aligning page content with Bootstrap

To begin the process, I will focus on the main elements that I want to manipulate and search Bootstrap's documentation related to those elements. The first element or group of elements that I want to focus on are the "card" elements containing the article titles. Furthermore, since the template I used from Data Flair used "card-columns" I will first attempt to display the boxes horizontally in the middle of the webpage.

index.html — F:\CapstoneFolder\capstone_project\templates — F:\CapstoneFolder — Atom

Project

- CapstoneFolder
- CapstoneFolder
 - capstone_project
 - capstone_project
 - news_app
 - templates
 - index.html
 - virtual_env
 - barclaypy
 - OV_sample.csv
 - piechart.py

index.html

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Capstone News Aggregator</title>
5      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5B4XqQJhK4mW9QG0e6Pnq4mQnqIgGfAu/dAIS6Xm" crossorigin="anonymous">
6  </head>
7  <body>
8      <div class="jumbotron">
9          <center>Capstone News Aggregator</center>
10         <a href="#" class="btn btn-success">Get my morning news</a>
11     </form>
12 </div>
13 <div>
14     <h2>Barons News</h2>
15     <div class="card-columns" style="padding: 10px; margin: 20px;">
16         <% for object in barons_list %>
17             <div class="card" style="width: 10rem; border: 5px black solid;">
18                 <div class="card-body">
19                     <h5 class="card-title"><div class="card-body">
20                         <a href="{{object.link}}><h5 class="card-title">{{object.title}}</h5></a>
21                     </div></h5>
22                 </div>
23             </div>
24         <% endfor %>
25     </div>
26     <h2>Bloomberg News</h2>
27     <div class="card-columns" style="padding: 10px; margin: 20px;">
28         <% for object in bloomberg_list %>
29             <div class="card" style="width: 10rem; border: 5px black solid;">
30                 <div class="card-body">
31                     <h5 class="card-title"><div class="card-body">
32                         <a href="{{object.link}}><h5 class="card-title">{{object.title}}</h5></a>
33                     </div></h5>
34                 </div>
35             </div>
36         <% endfor %>
37     </div>
38     <h2>Google News</h2>
39     <div class="card-columns" style="padding: 10px; margin: 20px;">
40         <% for object in google_list %>
41             <div class="card" style="width: 10rem; border: 5px black solid;">
42                 <div class="card-body">
43                     <h5 class="card-title"><div class="card-body">
44                         <a href="{{object.link}}><h5 class="card-title">{{object.title}}</h5></a>
45                     </div></h5>
46                 </div>
47             </div>
48         <% endfor %>
49     </div>

```

capstone_project\templates\index.html 13

CR LF UTF-8 HTML GitHub Git

Figure 3.4.2-1 “index.html” using card columns.

Central Authentication Service | Cards - Bootstrap | +

getbootstrap.com/docs/4.0/components/card/

Ymail Robinhood Udemy YouTube Python for Everybody Remote Computer... Personal Email (4) Feed | LinkedIn GroupMe

There's a newer version of Bootstrap 4!

B Home Documentation Examples Themes Expo Blog

Search...

Getting started

Layout

Content

Components

Alerts

Badge

Breadcrumb

Buttons

Button group

Card

Carousel

Collapse

Dropdowns

Forms

Input group

Jumbotron

Modal

Navs

Navbar

Pagination

Popovers

Progress

Scrollspy

Tooltips

Cards

Bootstrap's cards provide a flexible and extensible content container with multiple variants and options.

 Multiply your marketing smarts with Mailchimp. Try for free. ads via Carbon

About

A card is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options. If you're familiar with Bootstrap 3, cards replace our old panels, wells, and thumbnails. Similar functionality to those components is available as modifier classes for cards.

Example

Cards are built with as little markup and styles as possible, but still manage to deliver a ton of control and customization. Built with flexbox, they offer easy alignment and mix well with other Bootstrap components. They have no margin by default, so use spacing utilities as needed.

Below is an example of a basic card with mixed content and a fixed width. Cards have no fixed width to start, so they'll naturally fill the full width of its parent element. This is easily customized with our various sizing options.

About Example Content types Body Titles, text, and links Images List groups Kitchen sink Header and footer Sizing Using grid markup Using utilities Using custom CSS Text alignment Navigation Images Image caps Image overlays Card styles Background and color Border Mixins utilities Card layout Card groups Card decks Card columns

Figure 3.4.2-2 Bootstrap card documentation.

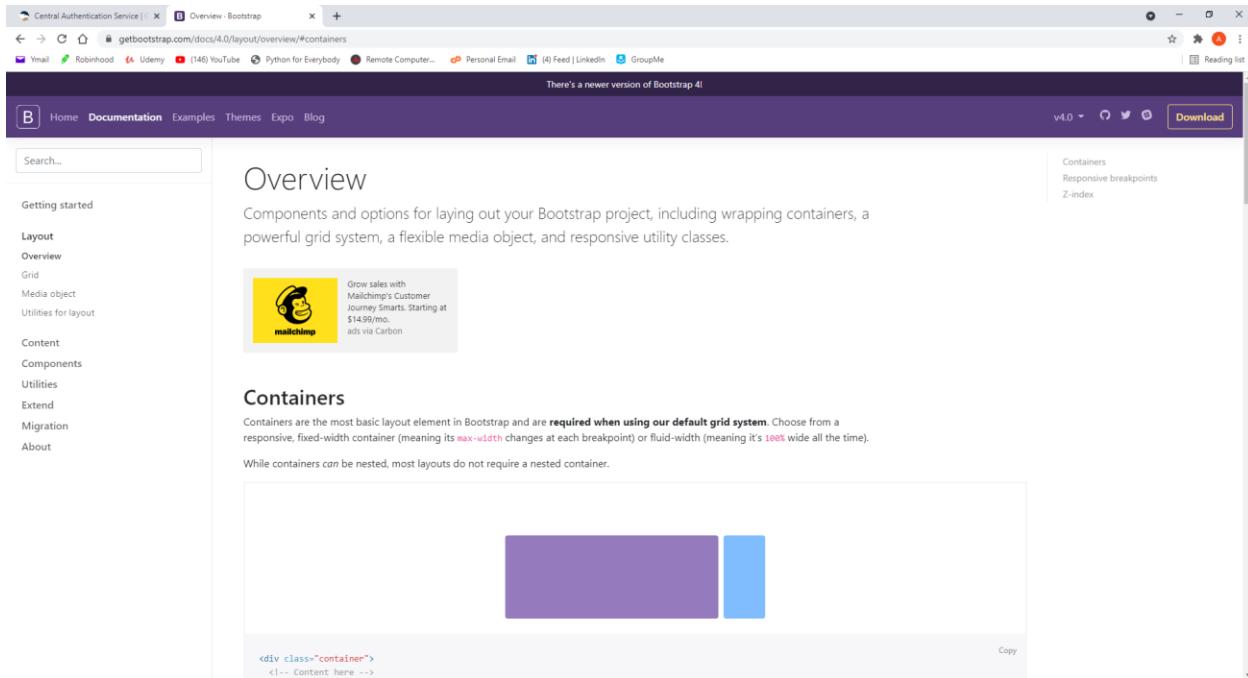


Figure 3.4.2-4 Bootstrap containers documentation.

By changing the card-column “div” elements to a row and container div elements, the cards are moved towards the center of the page in a horizontal manner, shown below.

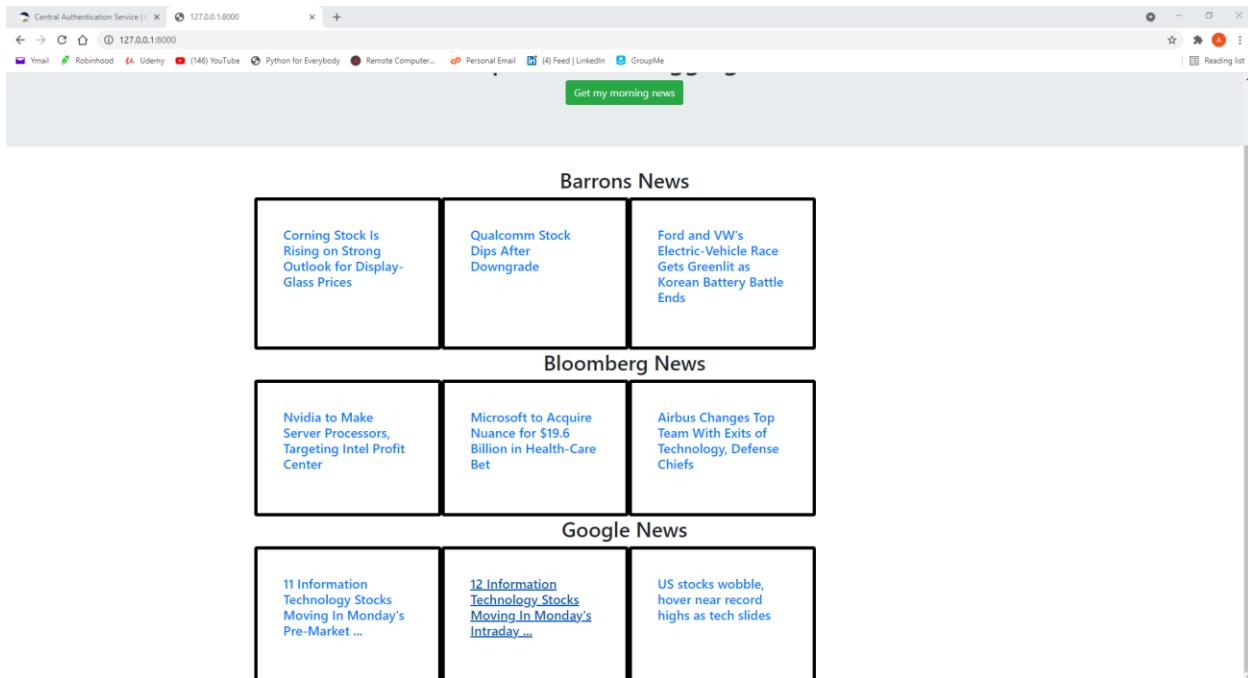


Figure 3.4.2-5 Removing card-columns to display rows.

4.4.3 Bootstrap Colors

The next objective is to add some color and spacing to the elements. In bootstrap, elements can be altered by adding css attributes in the “style” or “class” attributes of the element.

```
<div class="card bg-dark" style="width: 25rem; border: 2px blue solid; margin: 0 auto;">
```

Figure 3.4.3-1 Style attribute to customize elements.

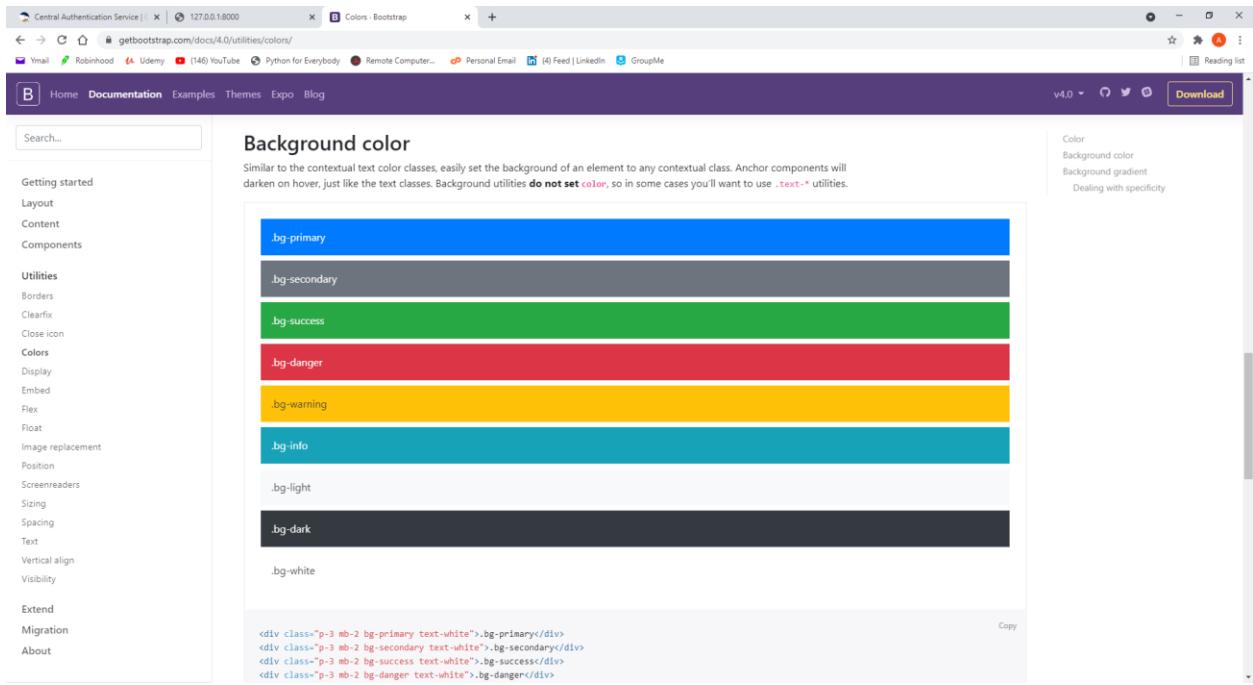


Figure 3.4.3-2 Bootstrap color documentation.

```
<body class='bg-secondary'>
```

Figure 3.4.3-3 Altering HTML body by changing class to Bootstrap background color.

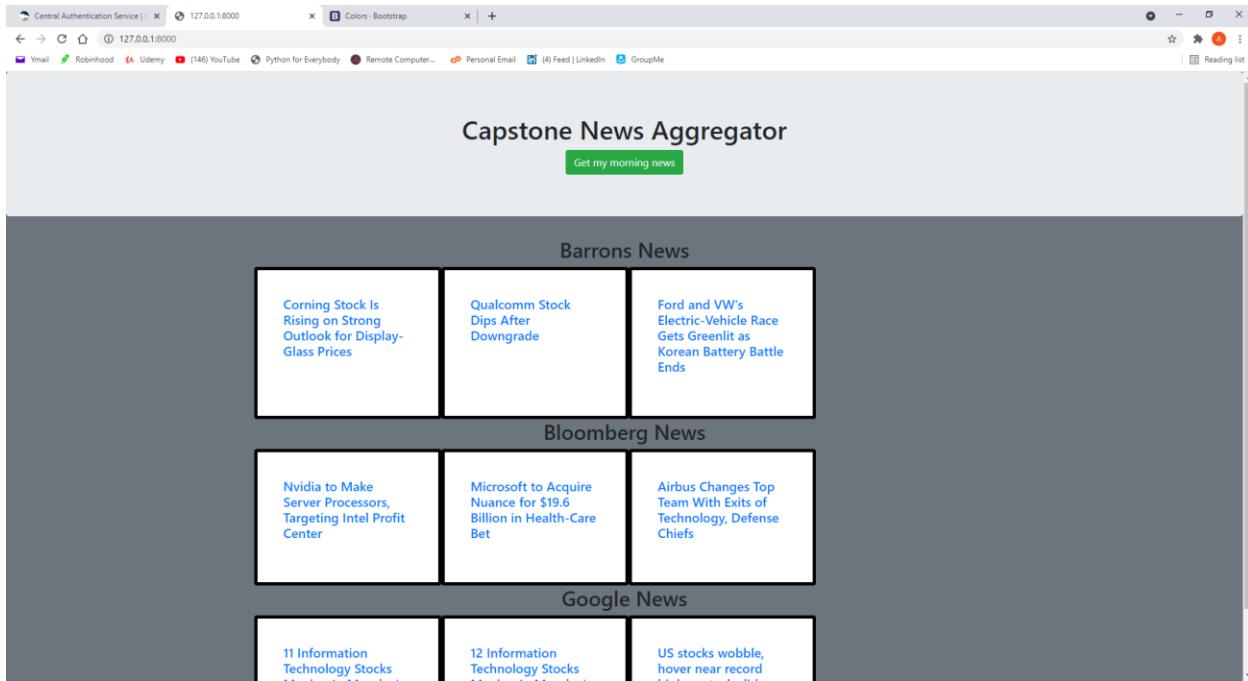


Figure 3.4.3-4 Color and layout formatting continued.

Next, I will make additional size and formatting (margin, width, etc.) changes to further align the content.

```

index.html — F:\CapstoneFolder\capstone_project\templates — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project Welcome index.html — F:\CapstoneFolder\capstone_p... views.py
index.html — F:\CapstoneFolder\capstone_p... views.py
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Bmb6uPwQ2LcHsE985c2sYsYcmeWfLUqizoqzo9FEQFVjzNIDp8GqZ9r+hQQpgg==">
    <title>Capstone News Aggregator</title>
  </head>
  <body class="p-3 mb-2 bg-secondary text-white">
    <center>
      <div class="jumbotron">
        <h1>Capstone News Aggregator</h1>
      </div>
    </center>
    <div class="container" style="width: 100%">
      <div class="row" style="width: 100%">
        <center><h3>Barron's Latest Technology Sector News</h3></center>
        <% for object in barrons_lst %>
          <div class="card" style="width: 25rem; border: 2px solid black; margin: 0 auto;">
            <div class="card-body">
              <a href="#"><h5 class="card-title">{{object.title}}</h5></a>
              <p class="text-dark">{{object.time}}</p>
            </div>
          </div>
        <% endfor %>
      </div>
      <div class="container" style="width: 100%">
        <div class="row" style="width: 100%">
          <center><h3>Google's Latest Technology Sector News</h3></center>
          <% for object in google_lst %>
            <div class="card" style="width: 25rem; border: 2px solid black; margin: 0 auto;">
              <div class="card-body">
                <h5 class="card-title">{{object.title}}</h5>
              </div>
            </div>
          <% endfor %>
        </div>
      </div>
    </div>
  </body>

```

Figure 3.4.3-5 Adding additional customizations to “index.html”

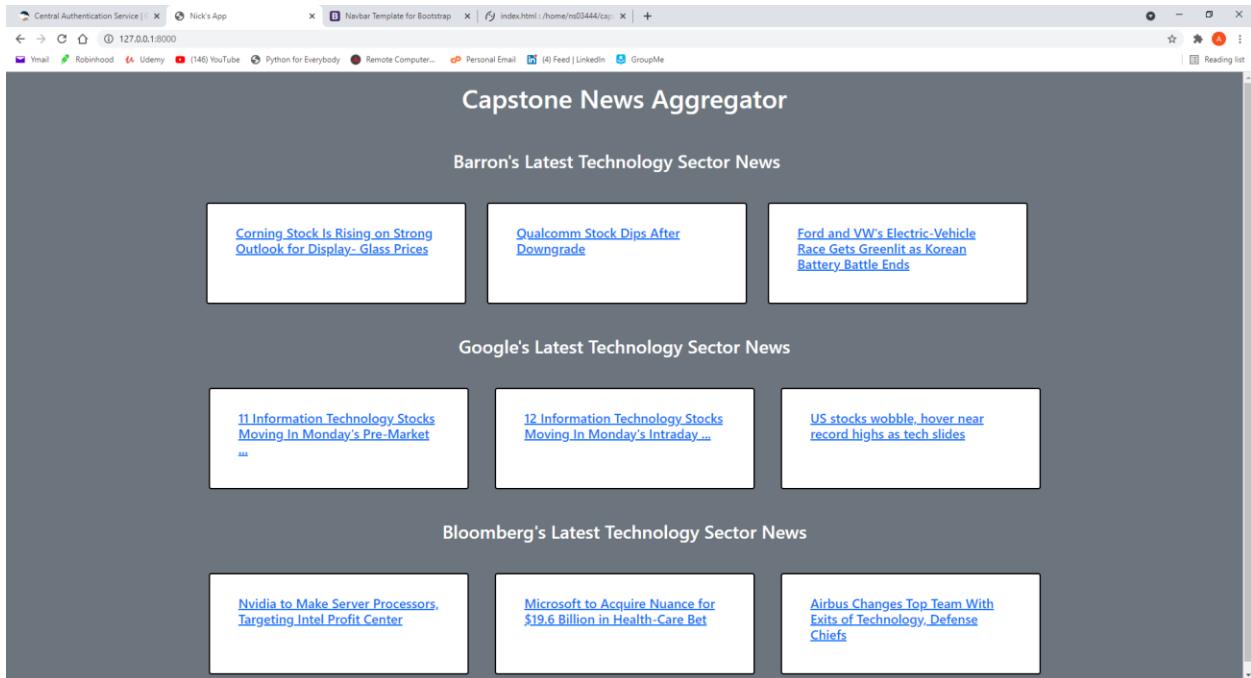


Figure 3.4.3-6 formatting “index.html” output.

****NOTE:** the ‘Get my Morning News’ button was accidentally deleted but will be added back in the next section.

4.4.4 Bootstrap Navigation Bars

Although my webpage only consists of one template right now, I will need a navbar in the future so users can navigate between pages easily. Luckily, Bootstrap has a variety of sample header templates to choose from and customize however you please.

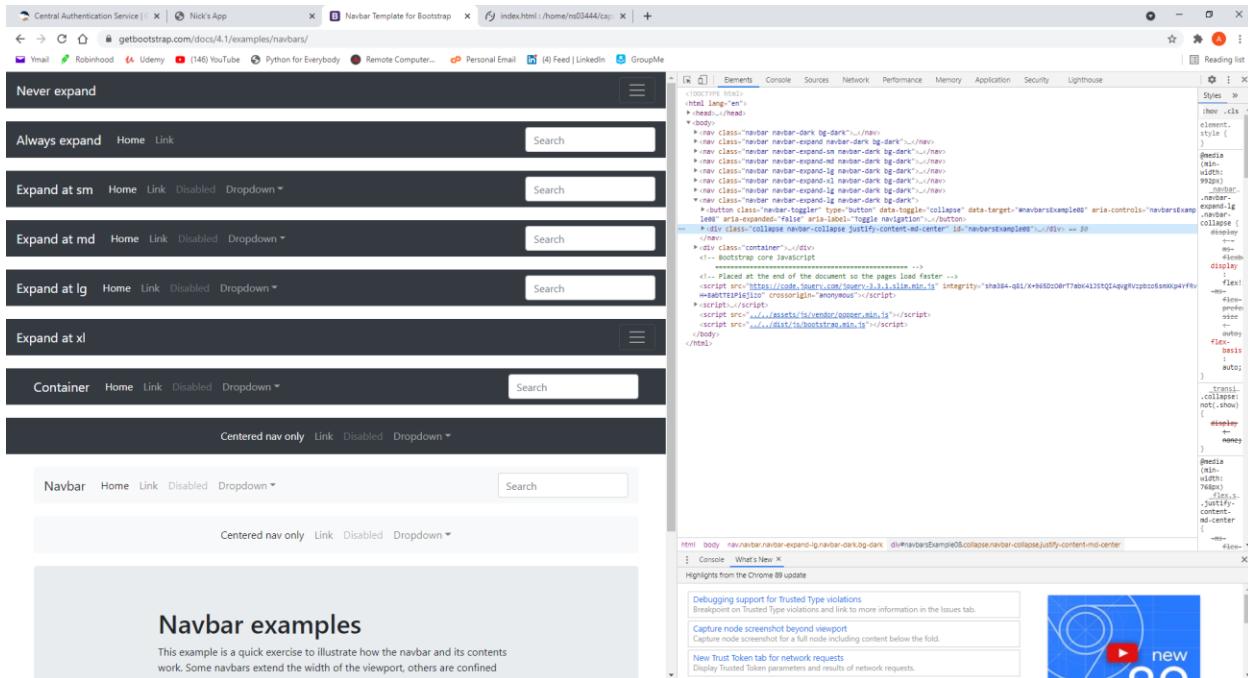


Figure 3.4.4-1

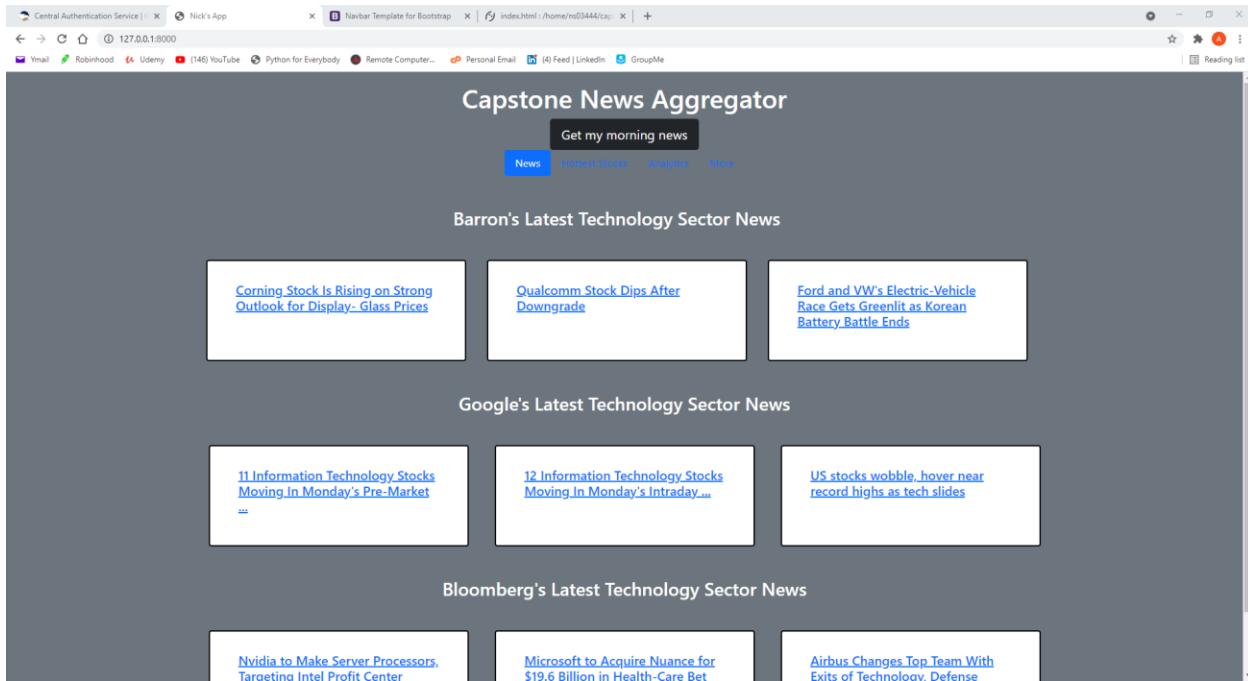
After finding a navbar that I favored, I inspected the html code of the webpage and copied/pasted the elements inside my header. Next, I will change the names to something more closely related to website. At this point, the navbar links will be static.

```

index.html — F:\CapstoneFolder\capstone_project\templates — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project Welcome index.html — D:\CapstoneFolder\user\apcme_3... index.html — F:\CapstoneFolder\capstone_p...
index.html — F:\CapstoneFolder\capstone_p...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <!-- Required meta tags -->
5   <meta charset="utf-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7   <!-- Bootstrap CSS -->
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-BIViRQW4qKXnVqWZUaDfTfHdPp6t37Vt4jw4Iu6D86Z9zqC233vqK9QjxuJGtjz30" crossorigin="anonymous">
9   <title>Nick's App</title>
10 </head>
11 <body class="p-3 mb-2 bg-secondary text-white">
12 <center>
13   <div class="jumbotron">
14     <h1>Capstone News Aggregator</h1>
15     <a href="#"><button type="button" class="btn btn-lg btn-dark">Get my morning news</button></a>
16   <header class="d-flex justify-content-center py-3">
17     <ul class="nav nav-pills">
18       <li class="nav-item"><a href="#" class="nav-link active">News</a></li>
19       <li class="nav-item"><a href="#" class="nav-link">Hottest Stocks</a></li>
20       <li class="nav-item"><a href="#" class="nav-link">Analytics</a></li>
21       <li class="nav-item"><a href="#" class="nav-link">More</a></li>
22     </ul>
23   </header>
24 </div>
25 </center>
26 <div class="container" style="width: 100%">
27   <div class="row" style="width: 100%">
28     <center><h3 style="margin: 3rem;">Barron's Latest Technology Sector News</h3></center>
29     <% for object in barrons_lst %>
30       <div class="card" style="width: 25rem; border: 2px black solid; margin: 0 auto;">
31         <div class="card-body">
32           <h5 class="card-title"><div class="card-body">
33             <a href="{{object.link}}><h5 class="card-title">{{object.title}}</h5></a>
34             <p class="text-dark">{{object.time}}</p>
35           </div></h5>
36         </div>
37       </div>
38     <% endfor %>
39   </div>

```

Figure 3.4.4-2 Adding a navigation bar to “index.html”.



****NOTE:** the article time was not being displayed in the previous sections due to a typo in my 'news_scrape.py' file. The bug was fixed, and the time was added.

4.5 Final Output

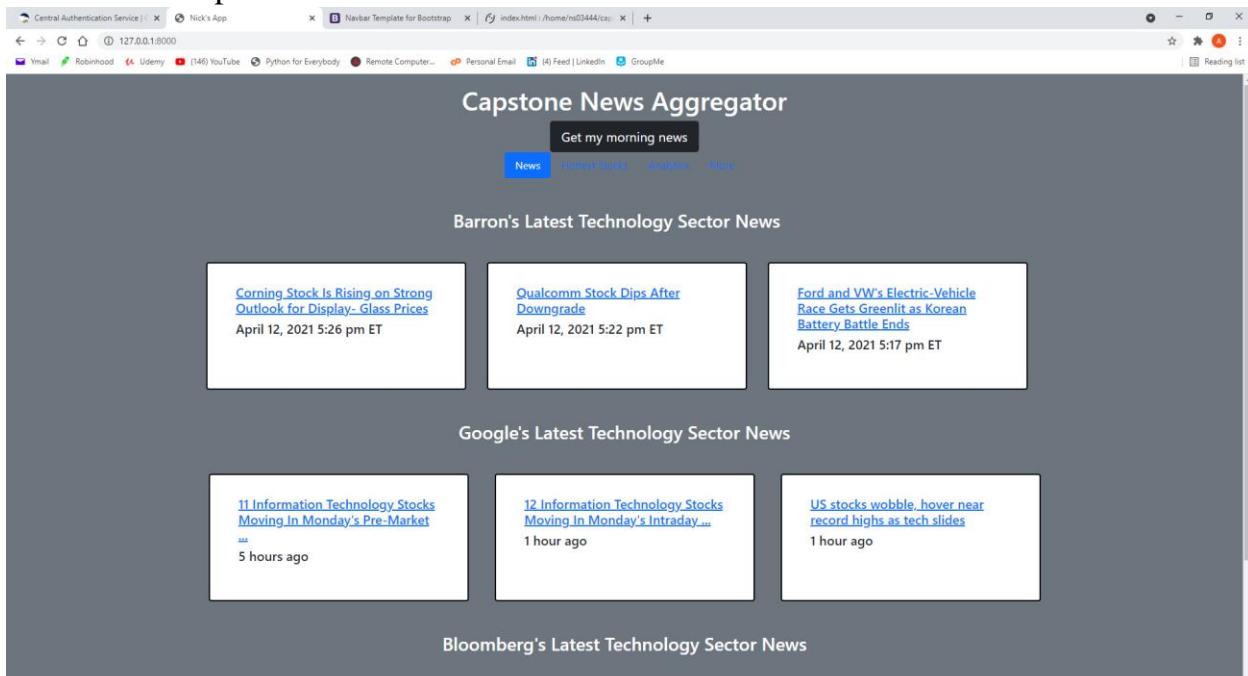
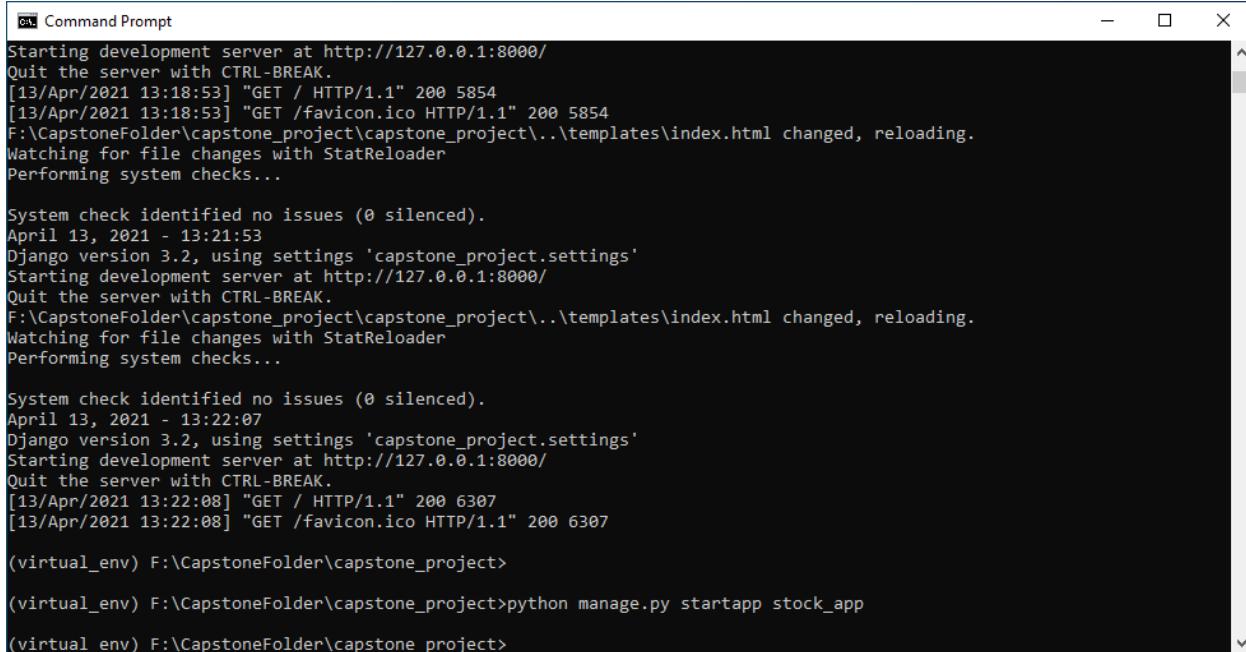


Figure 3.5-1 Final output for "index.html".

4.6 Creating the Stock App

Now that I have a much better understanding of how content can be injected into html templates and, the process of rendering templates in the browser, querying database objects, etc. I will use the same process I used for the “news_app” to make the “stock_app”, relatively. The stock app will render a web page with a table of the scraped data referenced in section 1.1 (Web Scraping Performance Data). So, displaying the data in tabular form and using the correct data types for the fields will take a different approach opposed to creating the news app. I will start by creating the app via command prompt.



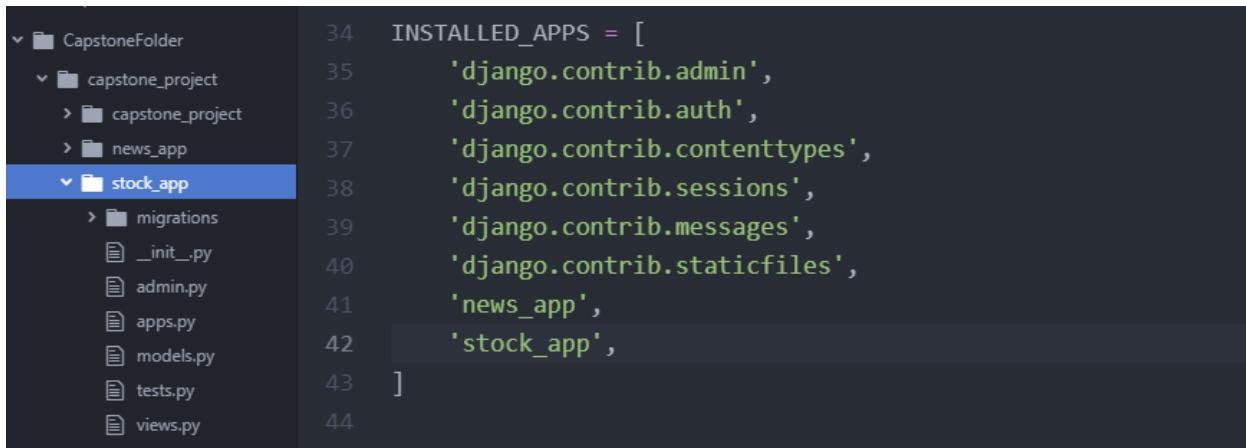
```
Command Prompt
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[13/Apr/2021 13:18:53] "GET / HTTP/1.1" 200 5854
[13/Apr/2021 13:18:53] "GET /favicon.ico HTTP/1.1" 200 5854
F:\CapstoneFolder\capstone_project\capstone_project..\templates\index.html changed, reloading.
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 13, 2021 - 13:21:53
Django version 3.2, using settings 'capstone_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
F:\CapstoneFolder\capstone_project\capstone_project..\templates\index.html changed, reloading.
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 13, 2021 - 13:22:07
Django version 3.2, using settings 'capstone_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[13/Apr/2021 13:22:08] "GET / HTTP/1.1" 200 6307
[13/Apr/2021 13:22:08] "GET /favicon.ico HTTP/1.1" 200 6307

(virtual_env) F:\CapstoneFolder\capstone_project>
(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py startapp stock_app
(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 3.6-1 Creating the stock application.

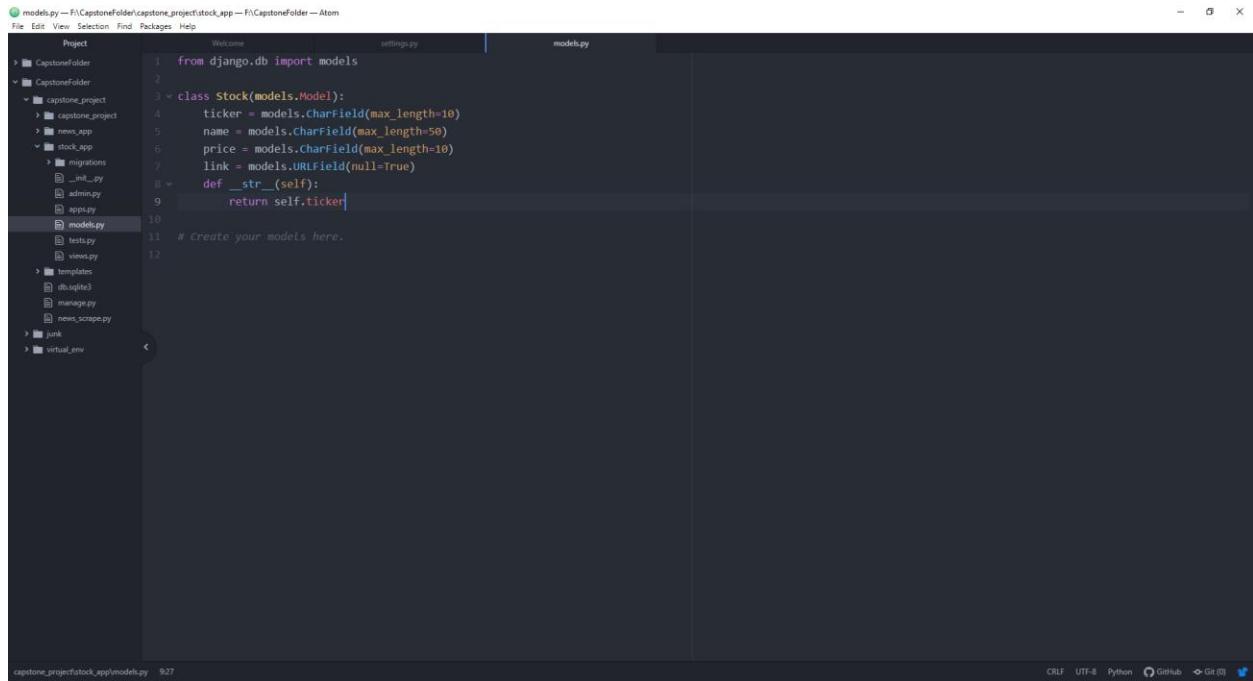


```
CapstoneFolder
  capstone_project
    capstone_project
    news_app
    stock_app
      migrations
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      views.py
      INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'news_app',
        'stock_app',
      ]
```

Figure 3.6-2 Configuring Django project settings for new application.

4.6.1 Creating a Django Model

Although creating this application will endure many of the procedures used previously, I can develop the app in a more efficient manner now since I am no longer experimenting with Django functionalities. Therefore, I will begin by creating a Django model “Stock” with fields to store the ticker, name, current price, and the link to that specific stock on finance.yahoo.com.



models.py — F:\CapstoneFolder\capstone_project\stock_app — F:\CapstoneFolder — Atom

File Edit View Selection Find Packages Help

Project

models.py

```
1 from django.db import models
2
3 class Stock(models.Model):
4     ticker = models.CharField(max_length=10)
5     name = models.CharField(max_length=50)
6     price = models.CharField(max_length=10)
7     link = models.URLField(null=True)
8     def __str__(self):
9         return self.ticker
10
11 # Create your models here.
12
```

settings.py

models.py

templates

db.sqlite3

manage.py

news_scrape.py

junk

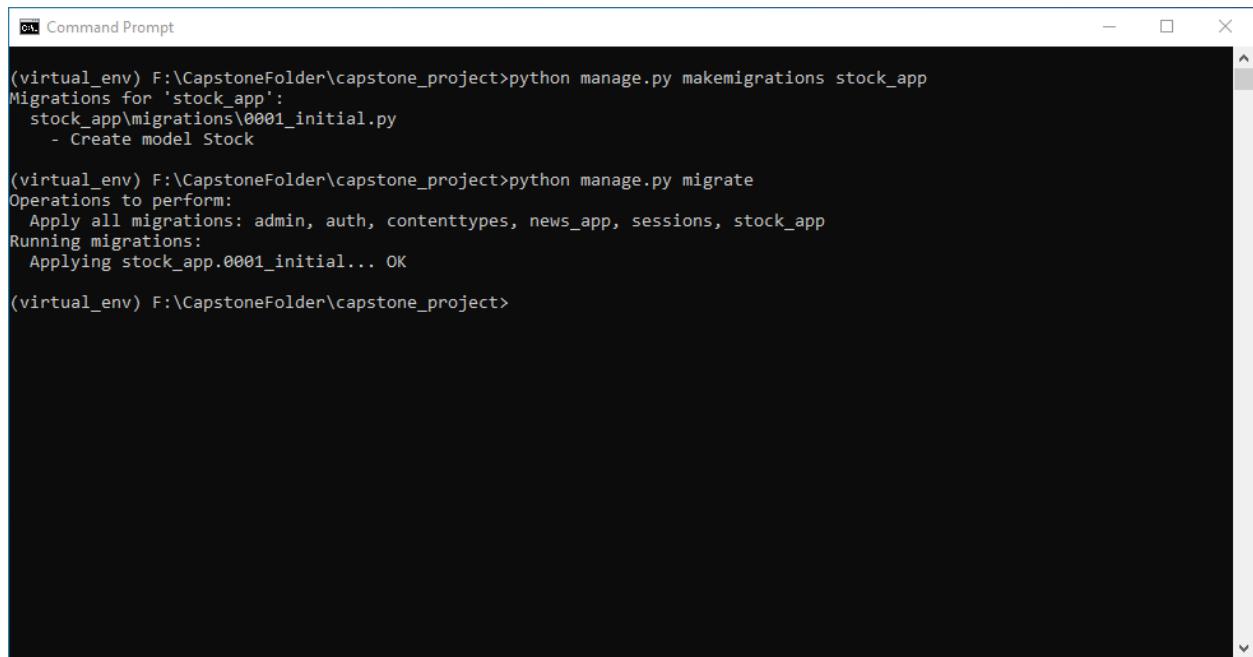
virtual_env

capstone_project\stock_app\models.py 9:27

CRLF UTF-8 Python GitHub Git

This screenshot shows the Atom code editor with the file 'models.py' open. The code defines a 'Stock' model with fields for 'ticker', 'name', 'price', and 'link'. The 'name' and 'price' fields are of type 'CharField' with a maximum length of 50. The 'link' field is of type 'URLField' and can be null. The model is registered with the database using the __str__ method. The code editor interface includes a sidebar for the project structure and a status bar at the bottom.

Figure 3.6.1-1 Creating the database table for stock performance data.



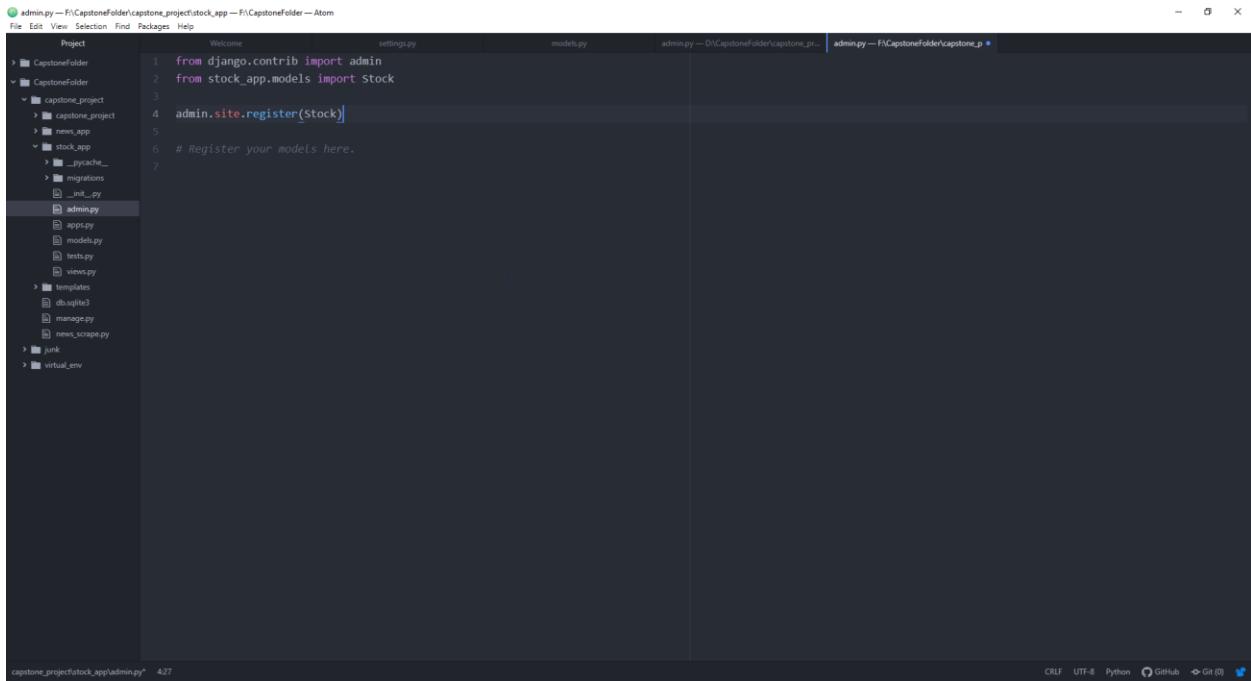
Command Prompt

```
(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py makemigrations stock_app
Migrations for 'stock_app':
  stock_app\migrations\0001_initial.py
    - Create model Stock

(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, news_app, sessions, stock_app
Running migrations:
  Applying stock_app.0001_initial... OK

(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 3.6.1-2 Applying migrations for new database table, “Stock”.



```

admin.py -- F:\CapstoneFolder\capstone_project\stock_app -- F:\CapstoneFolder -- Atom
File Edit View Selection Find Packages Help
Project
  CapstoneFolder
    CapstoneFolder
      capstone_project
        capstone_project
        news_app
        stock_app
          __pycache__
          migrations
          __init__.py
          admin.py
        apps.py
        models.py
        tests.py
        views.py
      templates
        db.sqlite3
        manage.py
        news_scrape.py
    junk
    virtual_env
  admin.py
  settings.py
  models.py
  admin.py -- D:\CapstoneFolder\capstone_pr...
  admin.py -- F:\CapstoneFolder\capstone_p...

```

admin.py -- F:\CapstoneFolder\capstone_project\stock_app -- F:\CapstoneFolder -- Atom

```

1 from django.contrib import admin
2 from stock_app.models import Stock
3
4 admin.site.register(Stock)
5
6 # Register your models here.
7

```

admin.py

settings.py

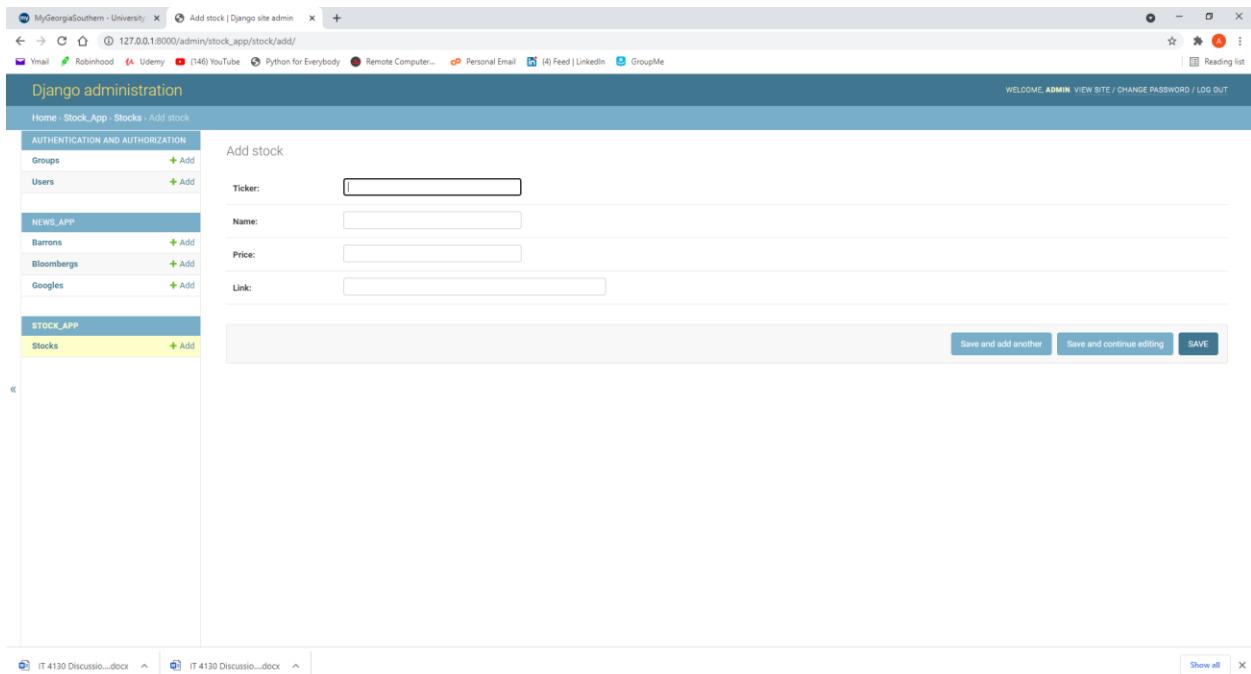
models.py

admin.py -- D:\CapstoneFolder\capstone_pr...

admin.py -- F:\CapstoneFolder\capstone_p...

CRLF UTF-8 Python GitHub Git (8) Twitter

Figure 3.6.1-3 Registering “Stock” model.



MyGeorgiaSouthern - University 127.0.0.1:8000/admin/stock_app/stock/add/

Ymail Robinhood Udemy YouTube Python for Everybody Remote Computer... Personal Email (4) Feed | LinkedIn GroupMe

Django administration

Home - Stock_App - Stocks - Add stock

WELCOME ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

NEWS_APP

Barrons + Add

Bloomberg + Add

Googles + Add

STOCK_APP

Stocks + Add

Add stock

Ticker:

Name:

Price:

Link:

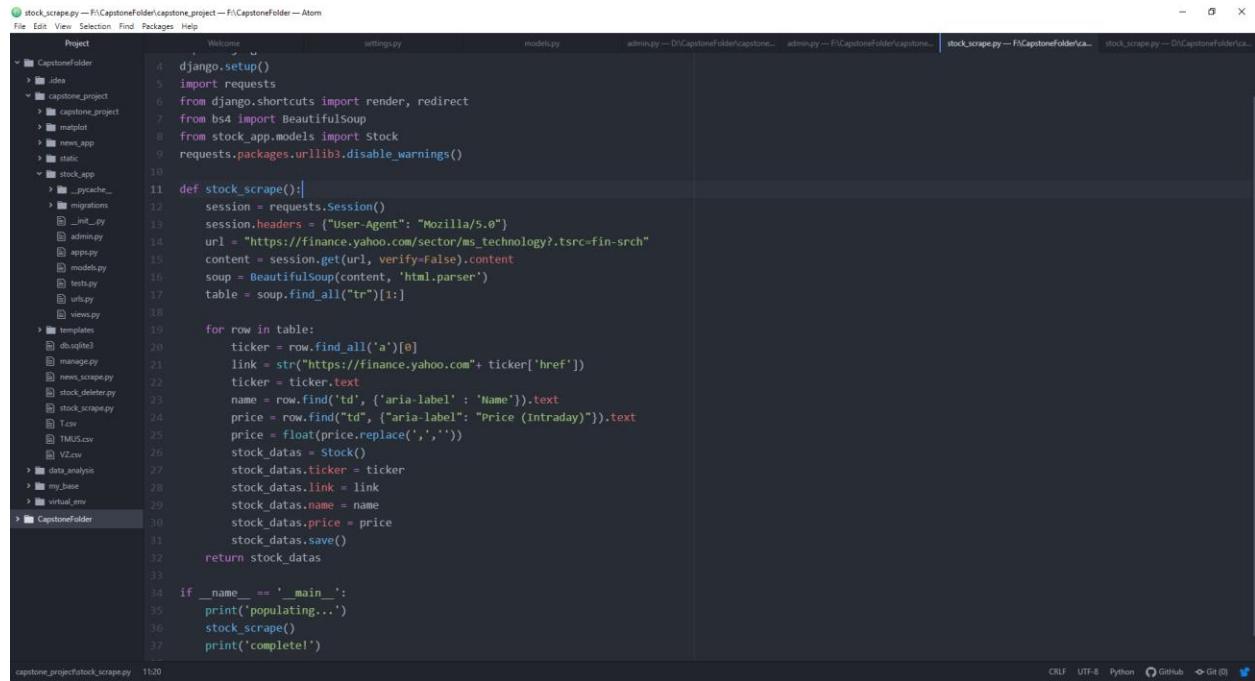
Save and add another Save and continue editing SAVE

IT 4130 Discussion.docx IT 4130 Discussion.docx Show all

Figure 3.6.1-4 New table created successfully and accessed through admin panel.

4.6.2 Populating the Database

Since finding a successful method for properly inserting scraped data into the news app models was not intuitive, I will use the same process to populate the “Stock” table. So, I will start by creating a file “stock_scrape.py” and a function “stock_scrape” that will contain the code for scraping the data as well as storing in the model.

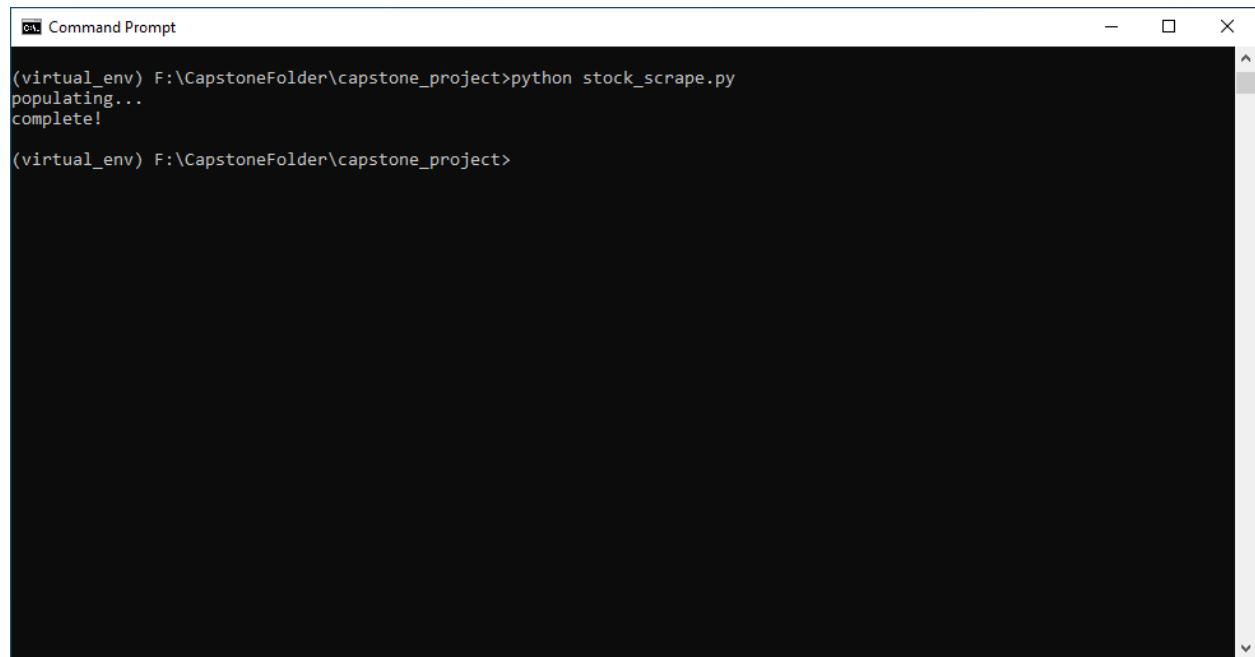


stock_scrape.py — F:\CapstoneFolder\capstone_project — F:\CapstoneFolder — Atom

```
Project Welcome settings.py models.py admin.py — D:\CapstoneFolder\capstone... admin.py — F:\CapstoneFolder\capstone... stock_scrape.py — F:\CapstoneFolder\ca... stock_scrape.py — D:\CapstoneFolder\ca...  
stock_scrape.py  
4 django.setup()  
5 import requests  
6 from django.shortcuts import render, redirect  
7 from bs4 import BeautifulSoup  
8 from stock_app.models import Stock  
9 requests.packages.urllib3.disable_warnings()  
10  
11 def stock_scrape():  
12     session = requests.Session()  
13     session.headers = {"User-Agent": "Mozilla/5.0"}  
14     url = "https://finance.yahoo.com/sector/ms_technology?.tsrc=fin-srch"  
15     content = session.get(url, verify=False).content  
16     soup = BeautifulSoup(content, 'html.parser')  
17     table = soup.find_all("tr")[1:]  
18  
19     for row in table:  
20         ticker = row.find_all('a')[0]  
21         link = str("https://finance.yahoo.com"+ ticker['href'])  
22         ticker = ticker.text  
23         name = row.find('td', {'aria-label': 'Name'}).text  
24         price = row.find("td", {"aria-label": "Price (Intraday)".text})  
25         price = float(price.replace(',', ''))  
26         stock_datas = Stock()  
27         stock_datas.ticker = ticker  
28         stock_datas.link = link  
29         stock_datas.name = name  
30         stock_datas.price = price  
31         stock_datas.save()  
32  
33     return stock_datas  
34  
35 if __name__ == '__main__':  
36     stock_scrape()  
37     print('complete!')
```

capstone_project\stock_scrape.py 11:20 CRLF UTF-8 Python GitHub Git

Figure 3.6.2-1 Creating new file named “stock_scrape.py” to populate new table.



Command Prompt

```
(virtual_env) F:\CapstoneFolder\capstone_project>python stock_scrape.py
populating...
complete!

(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 3.6.2-2 Successfully running “stock_scrape.py”

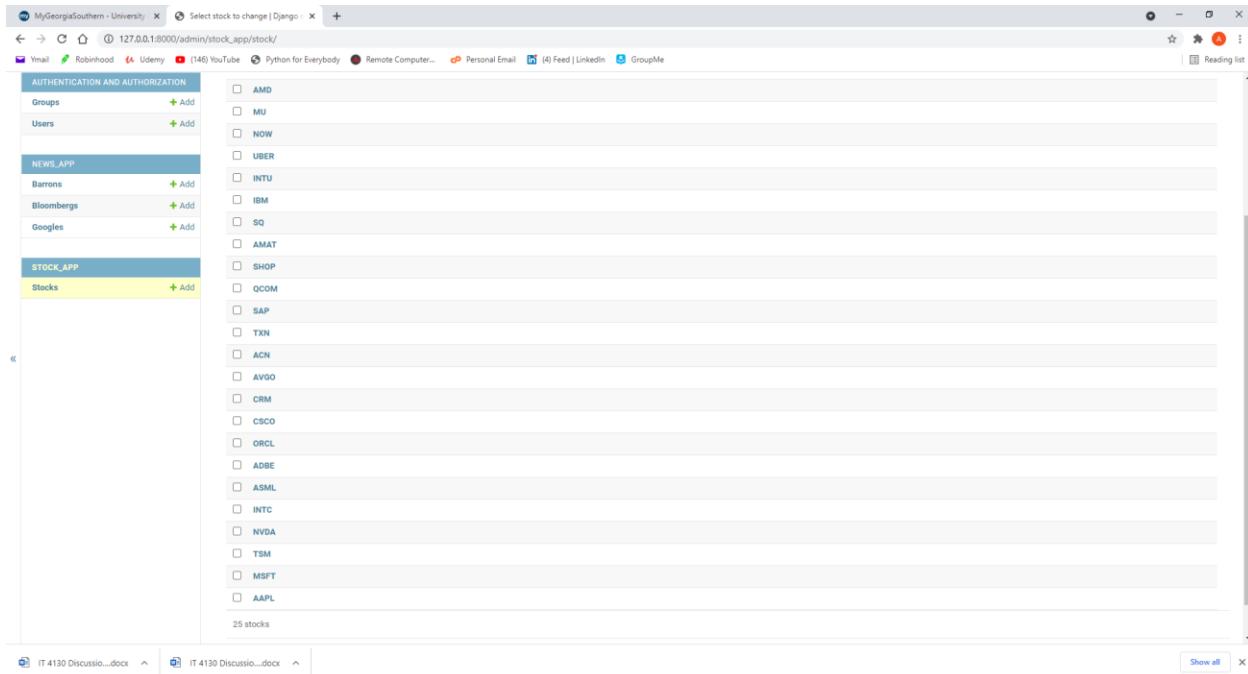


Figure 3.6.2-3 Populating “Stock” table.

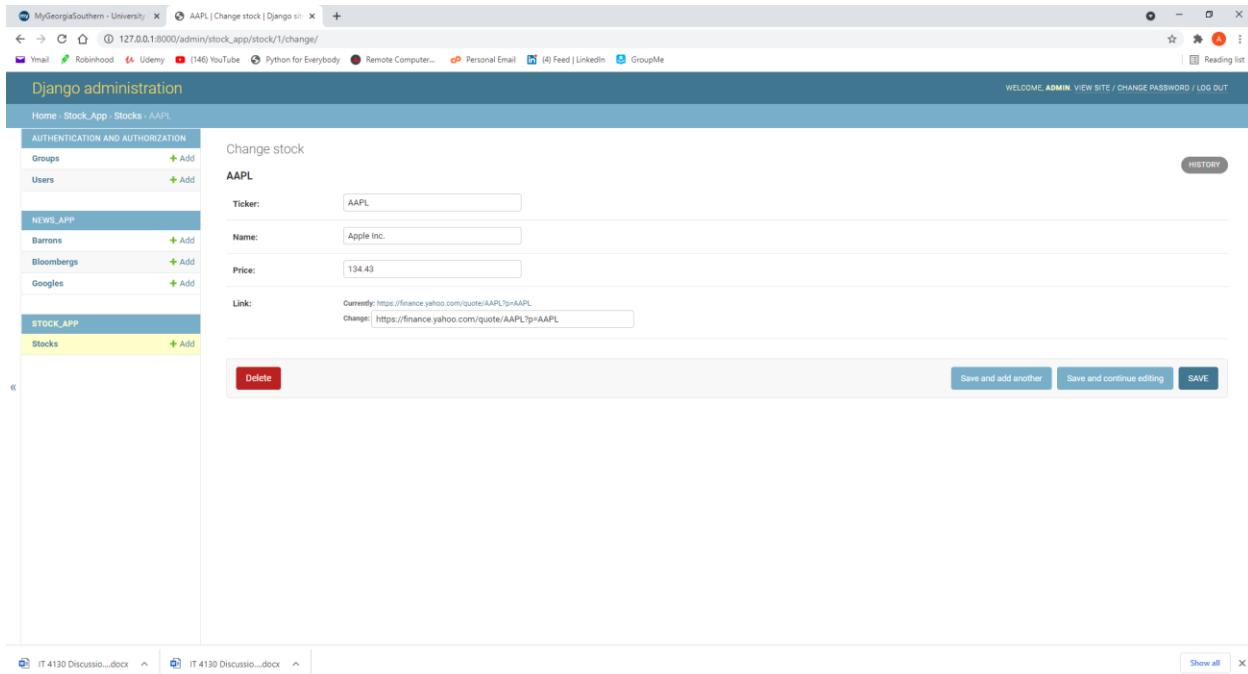


Figure 3.6.2-4 Successfully populating Stock model with correct fields and data types.

4.6.3 Developing the view

The screenshots above demonstrate the process I used to successfully populate the stock app model. Next, I will attempt rendering a new template in the browser that uses context processors to print a list of the objects in the database. I will begin by creating a new template and a function in my “views.py” file to define the context in which the data will be rendered.

Figure 3.6.3-1 “stock_app/views.py”

The screenshot above illustrates sending a request to the Stock model in the database and returning back the objects and rendering the template. Next, I will create a new template, “stock_page.html” and add the navbar from the index page in the header as well as the context processors containing a for loop to display my data in the form of a list.

stock_page.html — F:\CapstoneFolder\capstone_project\templates — F:\CapstoneFolder — Atom

File Edit View Selection Find Packages Help

Project

- stock_page.html — F:\CapstoneFolder\capstone_project\templates — stock_page.html — F:\CapstoneFolder\capstone...
- Welcome
- views.py — F:\CapstoneFolder\capstone_pro...
- views.py — D:\CapstoneFolder\capstone_pro...
- stock_page.html — D:\CapstoneFolder\cap...
- stock_page.html — F:\CapstoneFolder\cap...

```
5 <meta charset="utf-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <!-- Bootstrap CSS -->
8 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-BmbxuPwQaZlc/FVzCnJ7UAYjxM6wuqIj61t..." data-bbox="11 11 988 911" data-label="Text">

9 <title>Nick's App</title>
10 </head>
11 <body class="p-3 mb-2 bg-secondary text-white">
12 <div>
13 <div class="jumbotron">
14 <center><h1>Capstone News Aggregator</h1>
15 <a href="{% url 'index' %}" class="btn btn-lg btn-dark">Get my morning news</a></center>
16 <header class="d-flex justify-content-center py-3">
17 <ul class="nav nav-pills">
18 <li class="nav-item"><a href="/index" class="nav-link">News</a></li>
19 <li class="nav-item"><a href="/stock_page" class="nav-link active">Hottest Stocks</a></li>
20 <li class="nav-item"><a href="#" class="nav-link">Analytics</a></li>
21 <li class="nav-item"><a href="#" class="nav-link">More</a></li>
22 </ul>
23 </header>
24 </div>
25 </div>
26 </div>
27 <h2>Top 25 Tech Stocks</h2>
28 <div>
29 <list>
30 {% for obj in stock_objects %}
31 <li>{{obj.ticker}}</li>
32 <li>{{obj.name}}</li>
33 <li>{{obj.price}}</li>
34 </list>
35 {% endfor %}
36 </div>
37 </body>
38 </html>


```

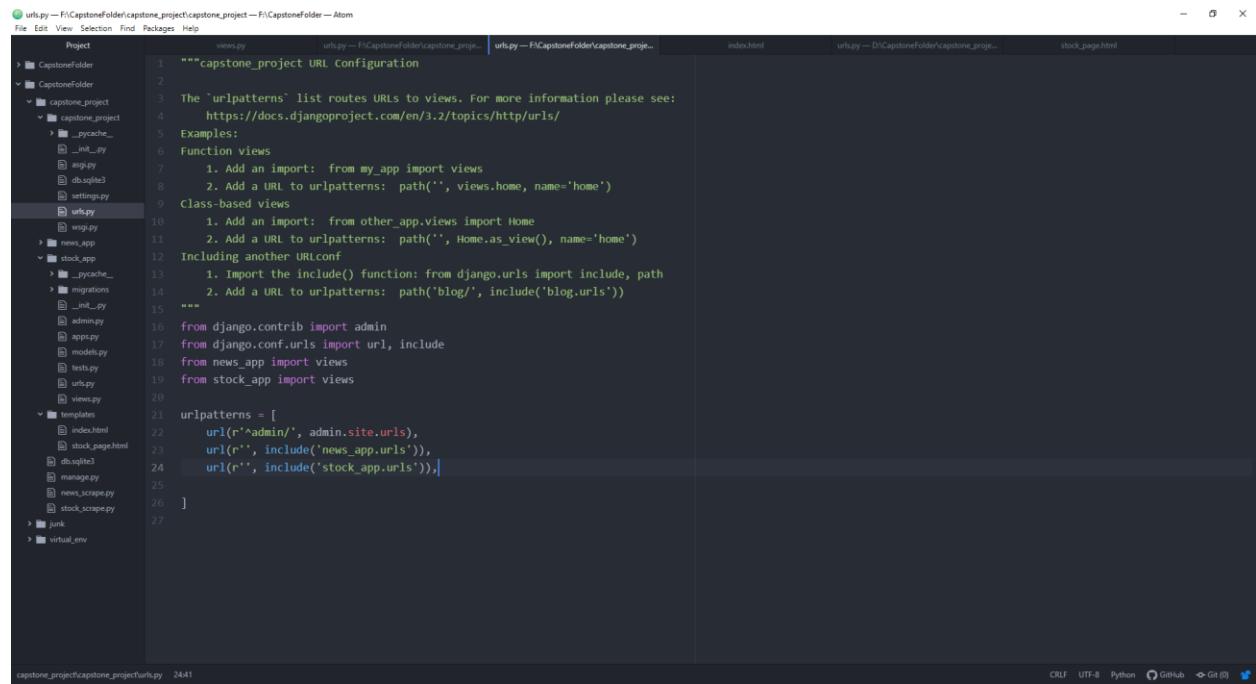
capstone_project\templates\stock_page.html 35/37 (0, 20%)

Nick's App - Google Chrome



Figure 3.6.3-2 “*templates/stock_page.html*”

One of the last things to do before running the server is configuring the application URLs with the project URLs. To do this, I will edit the project “urls.py” file to look for a new “urls.py” file in the stock app directory. After the new file is created, I will give it a path to locate my “stock_page” function in the views file and an extension of “stock_page/” to give the webpage a unique address. Which explains the “href=/stock_page” in the navbar shown above. This will allow the “hottest stocks” link to dynamically follow the link to “stock_page.html” when clicked.

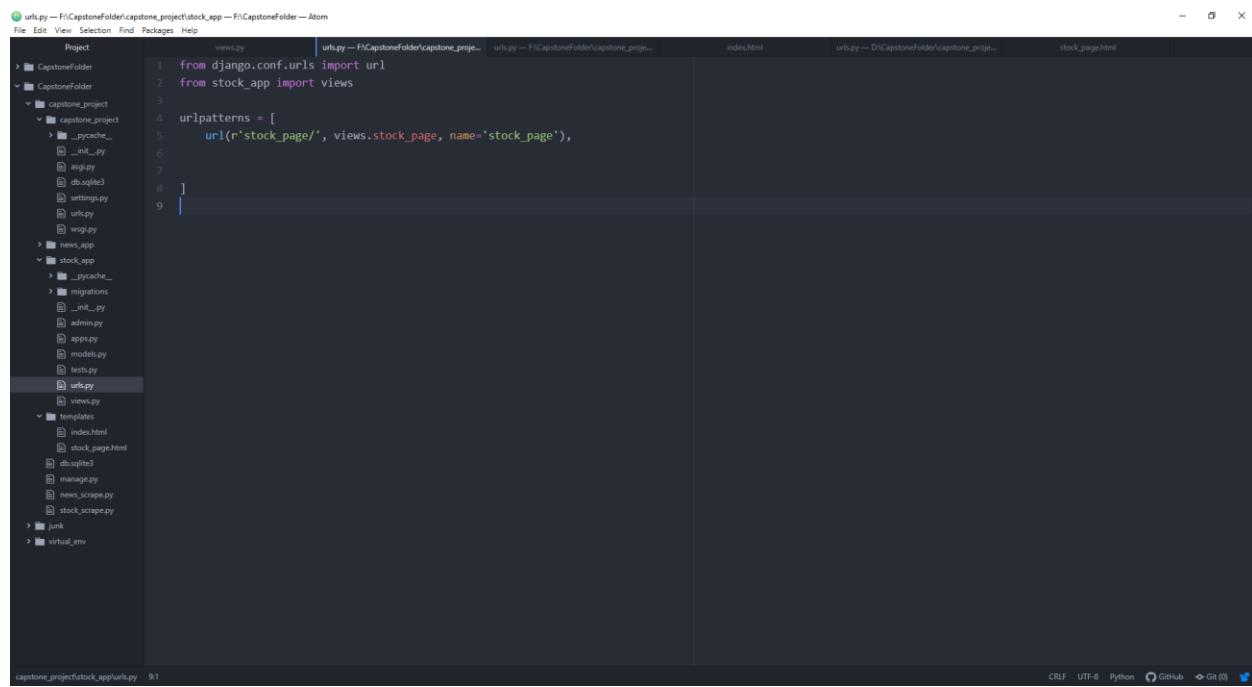


```

1  """capstone_project URL Configuration
2
3  The `urlpatterns` list routes URLs to views. For more information please see:
4      https://docs.djangoproject.com/en/3.2/topics/http/urls/
5
6  Examples:
7      1. Add an import: from my_app import views
8          2. Add a URL to urlpatterns: path('', views.home, name='home')
9
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13
14 Including another URLconf
15
16     1. Import the include() function: from django.urls import include, path
17     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
18
19
20
21 urlpatterns = [
22     url(r'^admin/', admin.site.urls),
23     url(r'', include('news_app.urls')),
24     url(r'', include('stock_app.urls')),]
25
26
27

```

Figure 3.6.3-3 Configuring URLs for the project



```

urls.py -- F:\CapstoneFolder\capstone_project\stock_app -- F:\CapstoneFolder -- Atom
File Edit View Selection Find Packages Help
Project urls.py urls.py -- F:\CapstoneFolder\capstone_project...
> CapstoneFolder index.html urls.py -- F:\CapstoneFolder\capstone_project...
  > capstone_project stock_page.html urls.py -- F:\CapstoneFolder\capstone_project...
    > capstone_project urls.py stock_page.html urls.py -- F:\CapstoneFolder\capstone_project...
      > __pycache__ urls.py
        __init__.py
        asgi.py
        db.sqlite3
        settings.py
        urls.py
        wsgi.py
      news_app
      stock_app
        > __pycache__
          __init__.py
          migrations
            __init__.py
            admin.py
            apps.py
            models.py
            tests.py
            urls.py
            views.py
        templates
          index.html
          stock_page.html
          db.sqlite3
          manage.py
          news_scrape.py
          stock_scrape.py
    junk
  virtual_env

```

capstone_project\stock_app\urls.py 9:1

CR LF UTF-8 Python GitHub Git (0) 

Figure 3.6.3-4 Configuring stock application URLs.

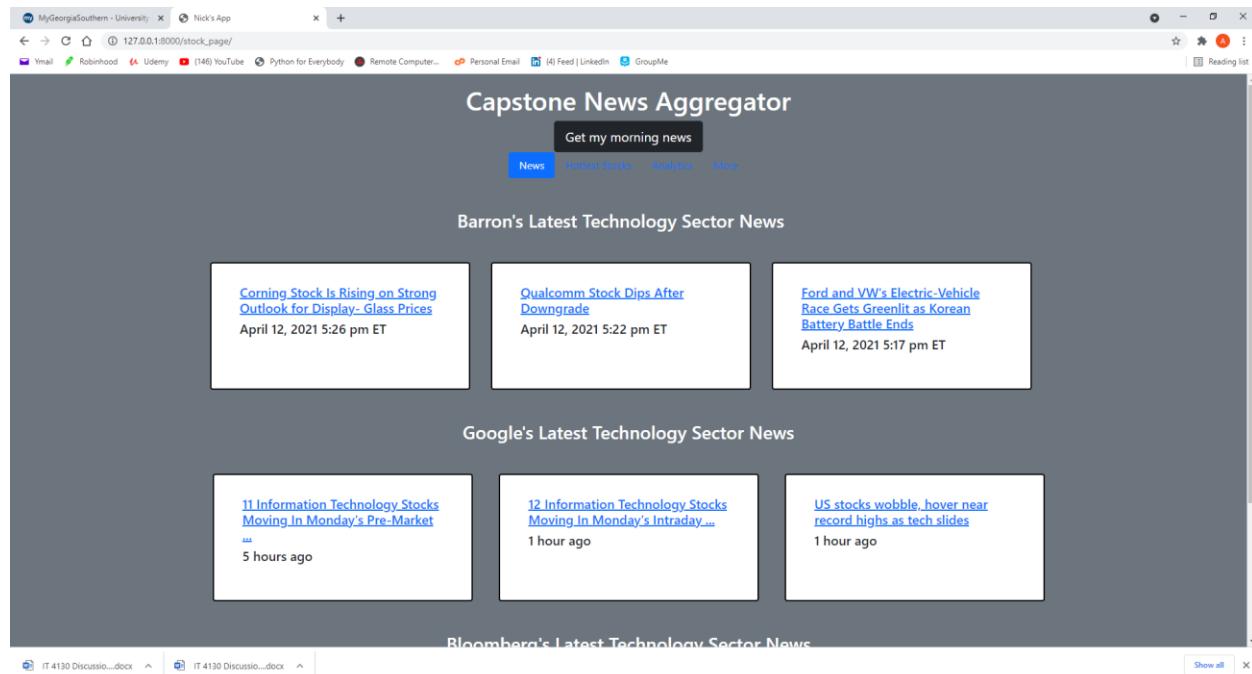
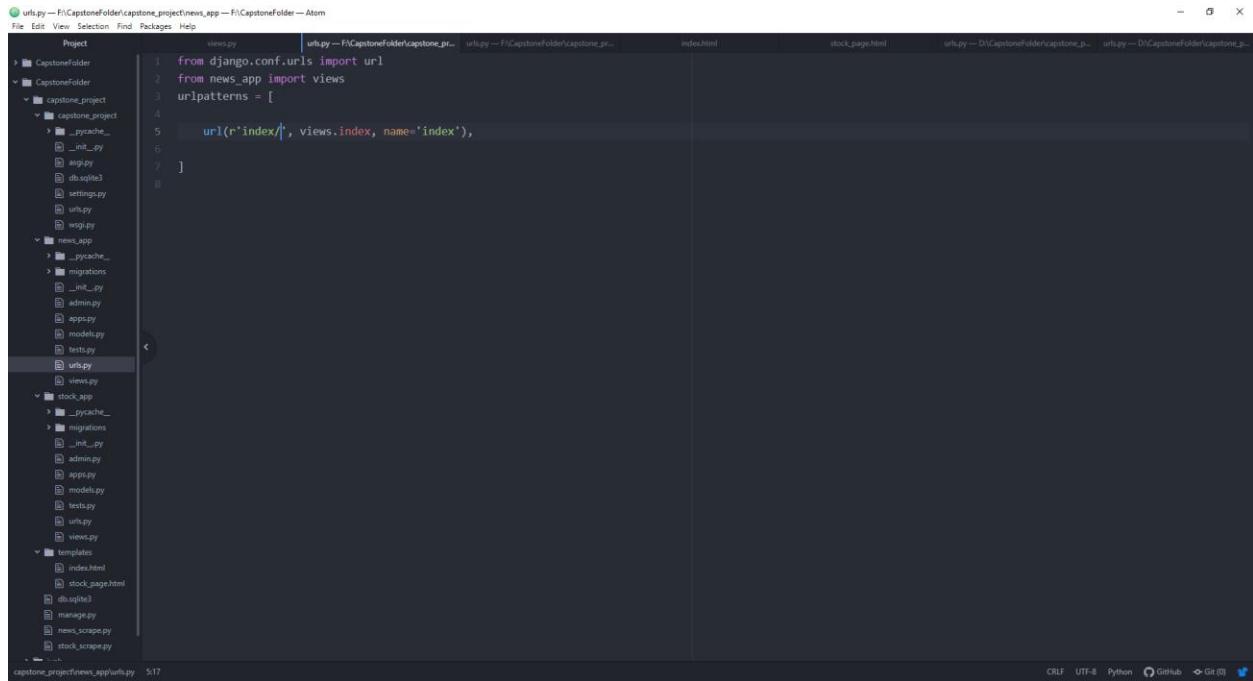


Figure 3.6.3-5 “/stock_page” extension in URL not rendering correct template.

If you notice at the top of the image above, in the address bar, “/stock_page” is added to the end of the local address, however, the index page is being displayed rather than the stock data. After some debugging, I found that my index.html page needed a unique extension as well, therefore it is now located at the local address followed by “/index”.



```

1  from django.conf.urls import url
2  from news_app import views
3
4  urlpatterns = [
5      url(r'^index/$', views.index, name='index'),
6  ]
7
8

```

Figure 3.6.3-6 Configuring unique address for index page.

Now, the “stock_page.html” is being properly rendered when requested, as well as, displaying the stock data as a list. Also, the navbar is actively working, and each page has a unique address that can be seen in the address bar at the top of the images below.

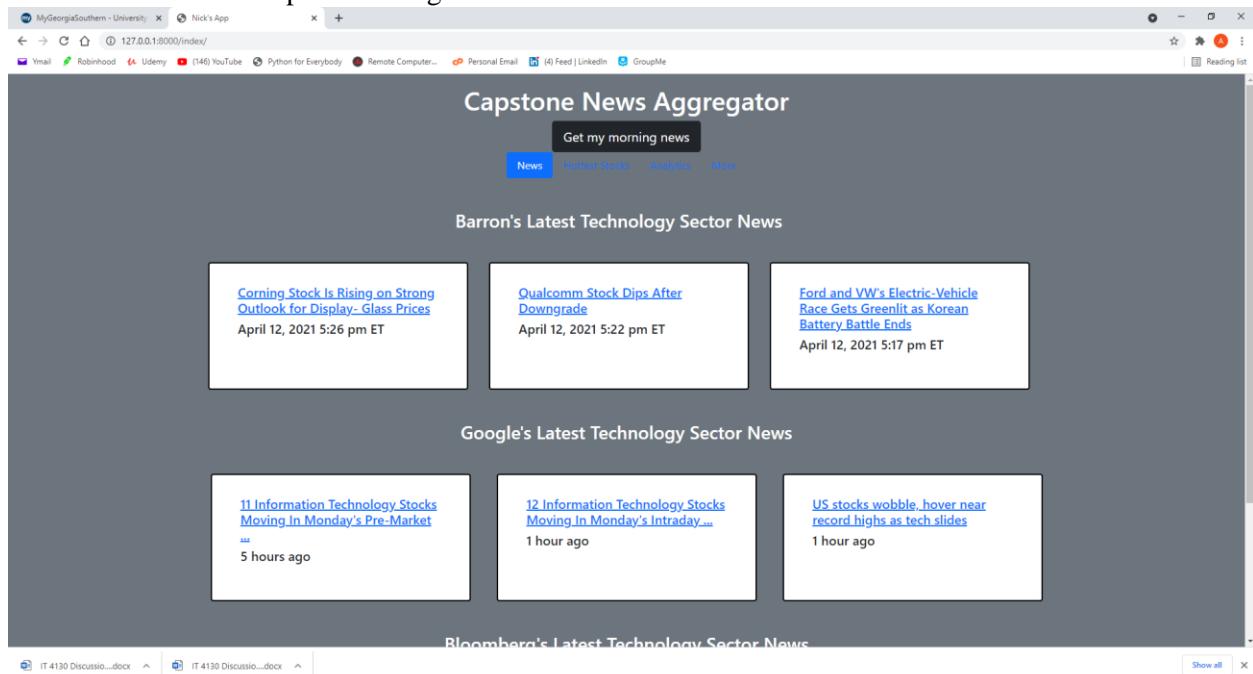


Figure 3.6.3-7 News content being displayed at 127.0.0.1:8000/index/

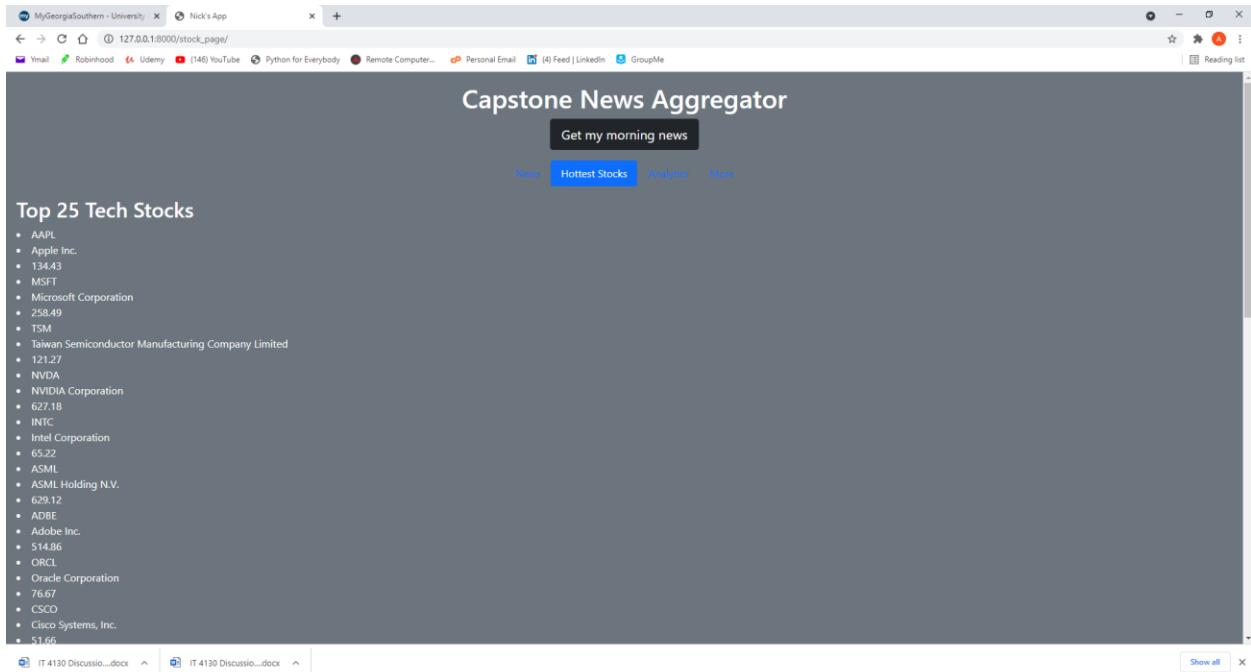


Figure 3.6.3-8 Stock content being displayed at 127.0.0.1:8000/stock_page/

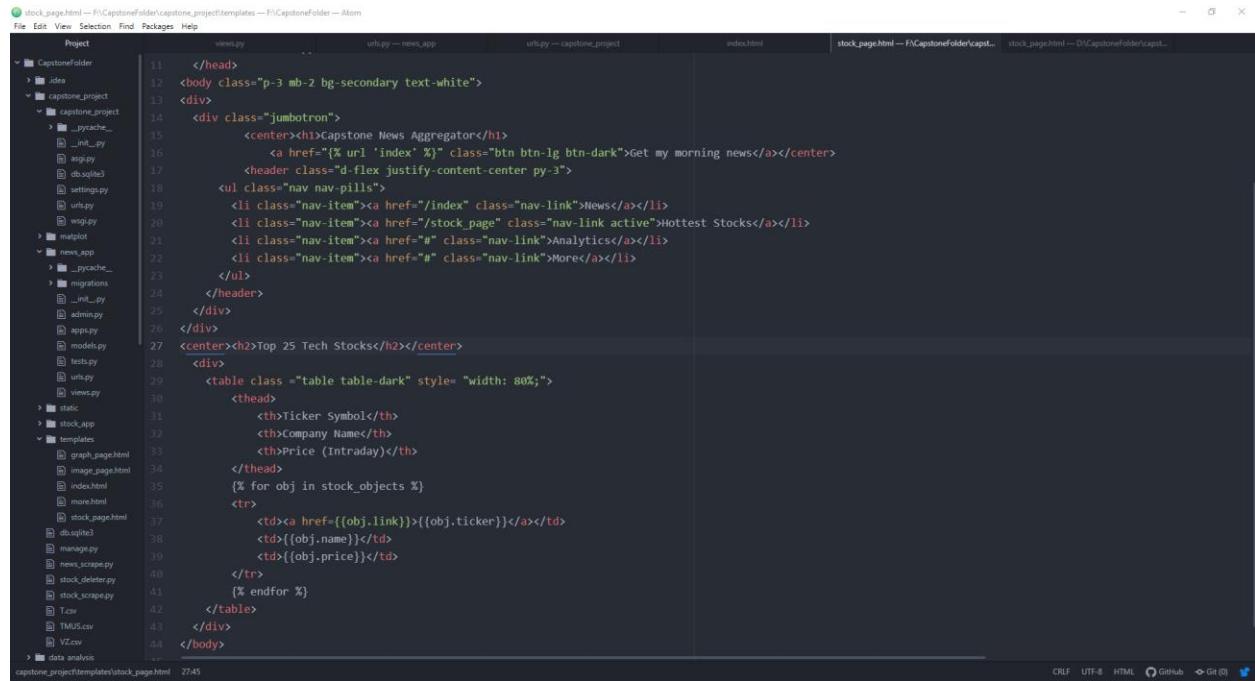
4.7 Designing the Stock page

Now that I have successfully displayed the data in the browser, I will now attempt converting the list of objects into tabular format. Fortunately, when I spent time browsing the Bootstrap documentation for the news article application, I saw that Django offered a variety of table element examples as well as great documentation on customizing them.

The screenshot shows a web browser displaying the Bootstrap 4.0 documentation for tables. The page has a dark header with a logo and navigation links. Below the header, there is a search bar and a sidebar with a navigation menu. The main content area shows the source code for a table with dark rows and columns. A 'Copy' button is located next to the code. To the right of the code, the browser's developer tools are open, showing the DOM structure of the table. The DOM tree includes elements like `<thead>`, `<tr>`, `<th>`, and `<td>`. The developer tools also show the CSS classes being applied, such as `table.table-dark`. The browser's status bar at the bottom shows two tabs: 'IT 4130 Discussion...docx' and 'IT 4130 Discussion...docx'.

Figure 3.7-1 Bootstrap table documentation.

I chose to go with the dark table theme for my table. Next, I will define the table size, headers, and implement context processors in the place of the table data elements (`<td>`) to inject my content into the table. I will also be converting the ticker symbols to hyperlinks by passing the Stock model’s “link” object to the ticker elements “href” attribute.

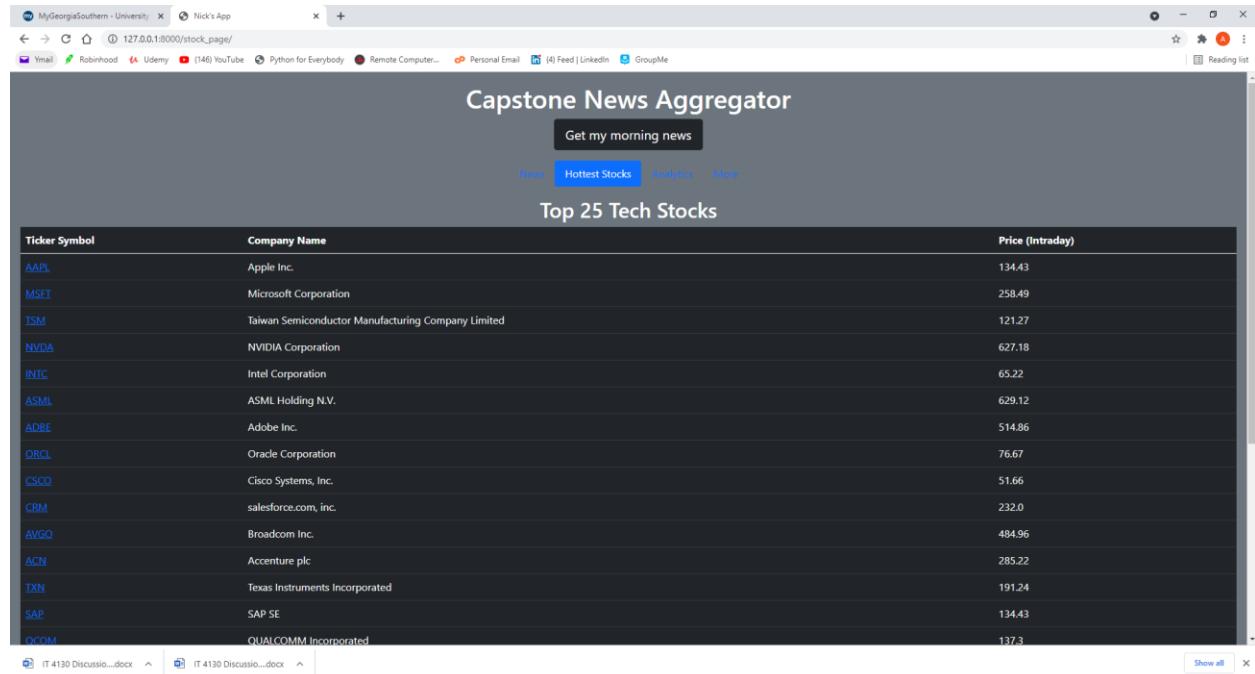


```

11     </head>
12 <body class="p-3 mb-2 bg-secondary text-white">
13 <div>
14     <div class="jumbotron">
15         <center><h1>Capstone News Aggregator</h1>
16         <a href="{% url 'index' %}" class="btn btn-lg btn-dark">Get my morning news</a></center>
17     <header class="d-flex justify-content-center py-3">
18         <ul class="nav nav-pills">
19             <li class="nav-item"><a href="/index" class="nav-link">News</a></li>
20             <li class="nav-item"><a href="/stock_page" class="nav-link active">Hottest Stocks</a></li>
21             <li class="nav-item"><a href="#" class="nav-link">Analytics</a></li>
22             <li class="nav-item"><a href="#" class="nav-link">More</a></li>
23         </ul>
24     </header>
25     </div>
26 </div>
27 <center><h2>Top 25 Tech Stocks</h2></center>
28     <div>
29         <table class="table table-dark" style="width: 80%;">
30             <thead>
31                 <th>Ticker Symbol</th>
32                 <th>Company Name</th>
33                 <th>Price (Intraday)</th>
34             </thead>
35             <tbody>
36                 {% for obj in stock_objects %}
37                     <tr>
38                         <td><a href="{{obj.link}}>{{obj.ticker}}</a></td>
39                         <td>{{obj.name}}</td>
40                         <td>{{obj.price}}</td>
41                     </tr>
42                 {% endfor %}
43             </tbody>
44         </table>
45     </div>
46 </body>

```

Figure3.7-2 creating table element in stock page template.



The screenshot shows a web browser window with the URL 127.0.0.1:8000/stock_page/. The page title is "Capstone News Aggregator". At the top, there is a navigation bar with buttons for "News", "Hottest Stocks" (which is active), "Analytics", and "More". Below the navigation is a section titled "Top 25 Tech Stocks" containing a table with the following data:

Ticker Symbol	Company Name	Price (Intraday)
AAPL	Apple Inc.	134.43
MSFT	Microsoft Corporation	258.49
TSM	Taiwan Semiconductor Manufacturing Company Limited	121.27
NVDA	NVIDIA Corporation	627.18
INTC	Intel Corporation	65.22
ASML	ASML Holding N.V.	629.12
ADBE	Adobe Inc.	514.86
ORCL	Oracle Corporation	76.67
CSCO	Cisco Systems, Inc.	51.66
CRM	salesforce.com, inc.	232.0
AVGO	Broadcom Inc.	484.96
ACN	Accenture plc	285.22
TXN	Texas Instruments Incorporated	191.24
SAP	SAP SE	134.43
QCOM	QUALCOMM Incorporated	137.3

Figure3.7-3 Unformatted table displayed.

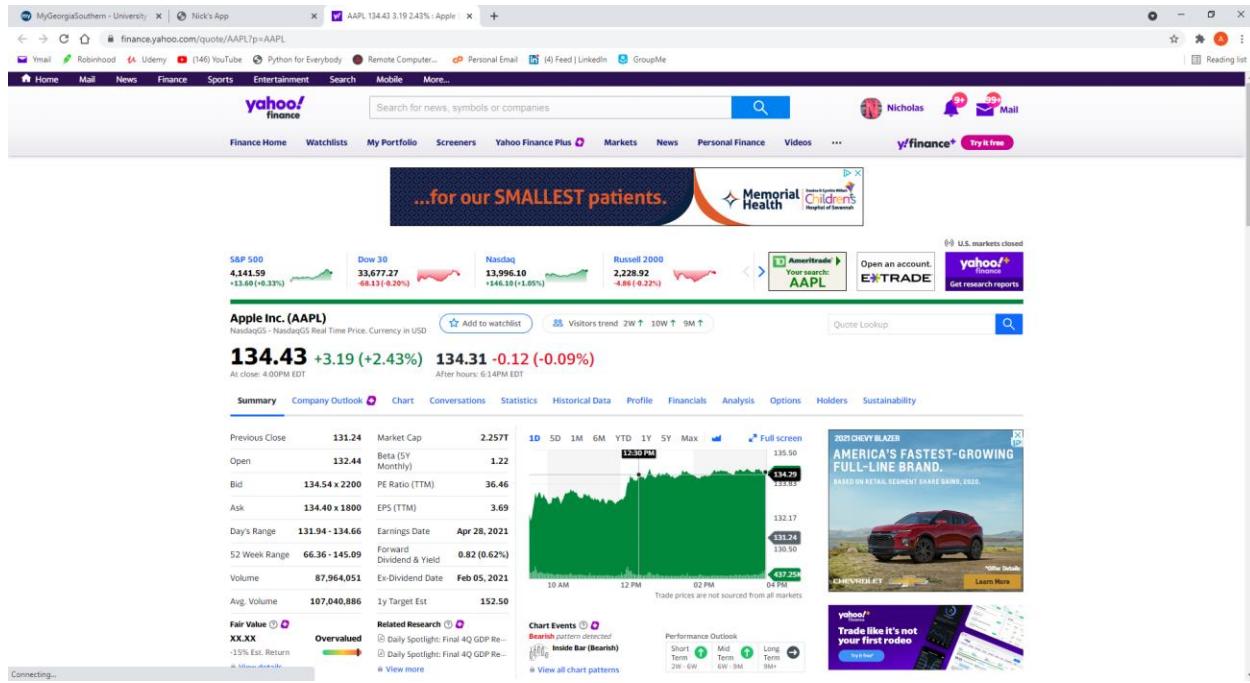


Figure 3.7-4 Ticker hyperlinks working successfully.

4.7.1 Table Formatting

Since the table is currently stretched across the entire page but only contains three columns of data, I will decrease the width and make the proper alignments to center the table.

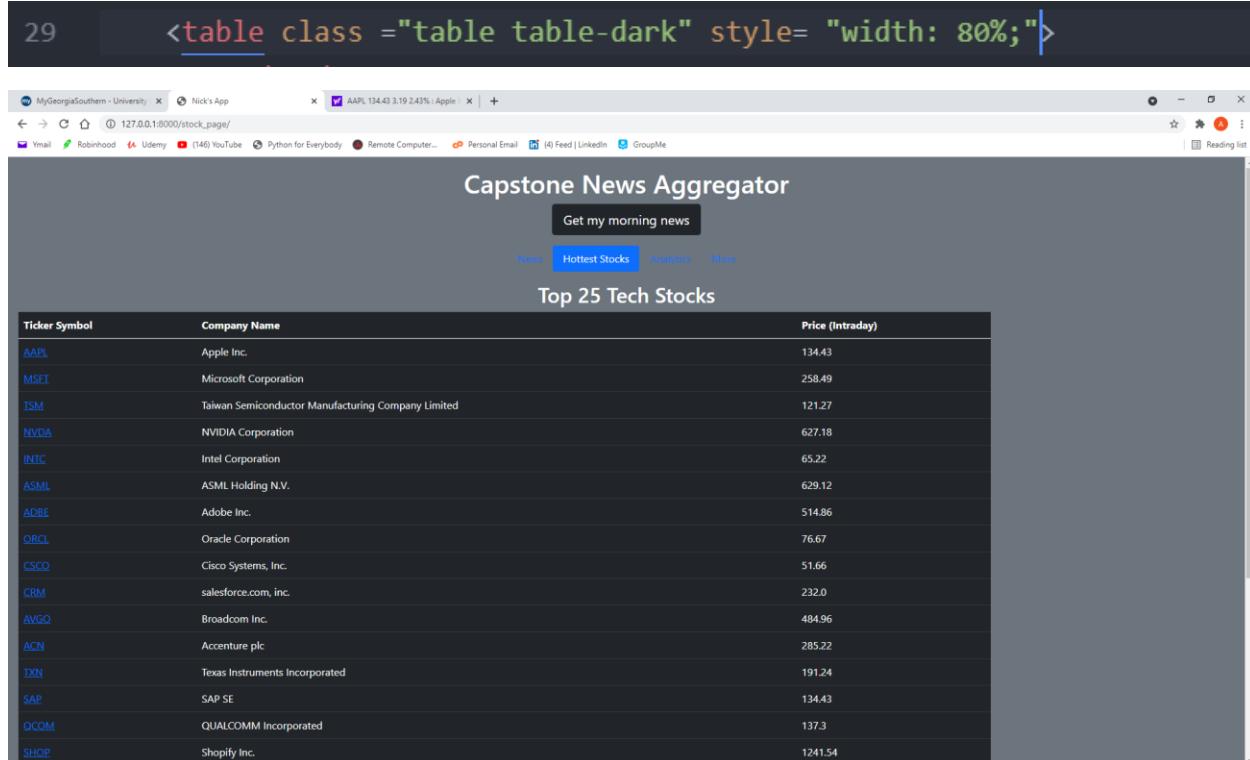


Figure 3.7.1-1 Reducing width of table.

4.8 Final Output

The screenshot shows a web browser window with the title 'Capstone News Aggregator'. Below the title is a button 'Get my morning news'. A navigation bar includes 'News', 'Hottest Stocks' (which is highlighted in blue), 'Analytics', and 'More'. The main content is a table titled 'Top 25 Tech Stocks' with the following data:

Ticker Symbol	Company Name	Price (Intraday)
AAPL	Apple Inc.	134.43
MSFT	Microsoft Corporation	258.49
TSM	Taiwan Semiconductor Manufacturing Company Limited	121.27
NVDA	NVIDIA Corporation	627.18
INTC	Intel Corporation	65.22
ASML	ASML Holding N.V.	629.12
ADBE	Adobe Inc.	514.86
ORCL	Oracle Corporation	76.67
CSCO	Cisco Systems, Inc.	51.66
CRM	salesforce.com, inc.	232.0
AVGO	Broadcom Inc.	484.96
ACN	Accenture plc	285.22
TXN	Texas Instruments Incorporated	191.24
SAP	SAP SE	134.43
QCOM	QUALCOMM Incorporated	137.3
SHOP	Shopify Inc.	1241.54

Figure 2- Centered and fully formatted table containing stock performance data.

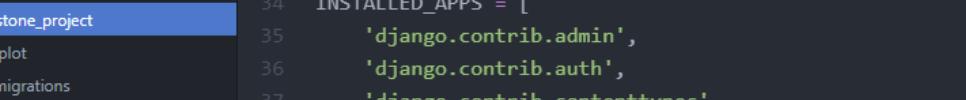
5 Data Vizualization

5.1 Creating the app

```
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nicho>F:
F:>cd CapstoneFolder
F:\CapstoneFolder>virtual_env\Scripts\activate
(virtual_env) F:\CapstoneFolder>cd capstone_project
(virtual_env) F:\CapstoneFolder\capstone_project>python manage.py startapp matplotlib
(virtual_env) F:\CapstoneFolder\capstone_project>
```

Figure 4.1-1 creating matplotlib application.



The image shows a file explorer on the left and a code editor on the right. The file explorer displays a Django project structure with the following contents:

- capstone_project (selected)
- capstone_project (selected)
- matplotlib
- migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - views.py
- news_app
- stock_app

The code editor shows the `INSTALLED_APPS` section of the `settings.py` file, which lists the following applications:

```
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41     'news_app',
42     'stock_app',
43     'matplotlib'
44 ]
```

Figure 4.1-2 New application directory created & configuration for project settings.

5.2 Introducing Bokeh

The next portion of the project deals with using the python modules “Matplotlib” and “Bokeh” to create detailed graphs and plots. However, another key dependency that will be introduced in this section is “pandas_datareader”, which is a sub-package of the pandas library and is used to quickly create a data frame (tabular format) from various internet sources. Although the new application will be named “matplotlib”, I will begin with using Bokeh to create a candlestick charts of the S&P 500 index over various periods of time.

The first step in visualizing any data is collecting the data. Therefore, I will begin by creating the variables to define the time interval of data I want to store for the first graph. Then, assign the “pandas_datareader” function to variable and use the necessary parameters to specify the data source, the ticker symbol, and the start/end dates. The first graph will represent data from the beginning of this year to April 2nd, 2021. I will be testing the graphs on a script stored externally from the rest of my project.

Figure 4.2-1 Collecting data for 1/4/2021 through 4/2/2021 of S&P 500 index

```
Command Prompt

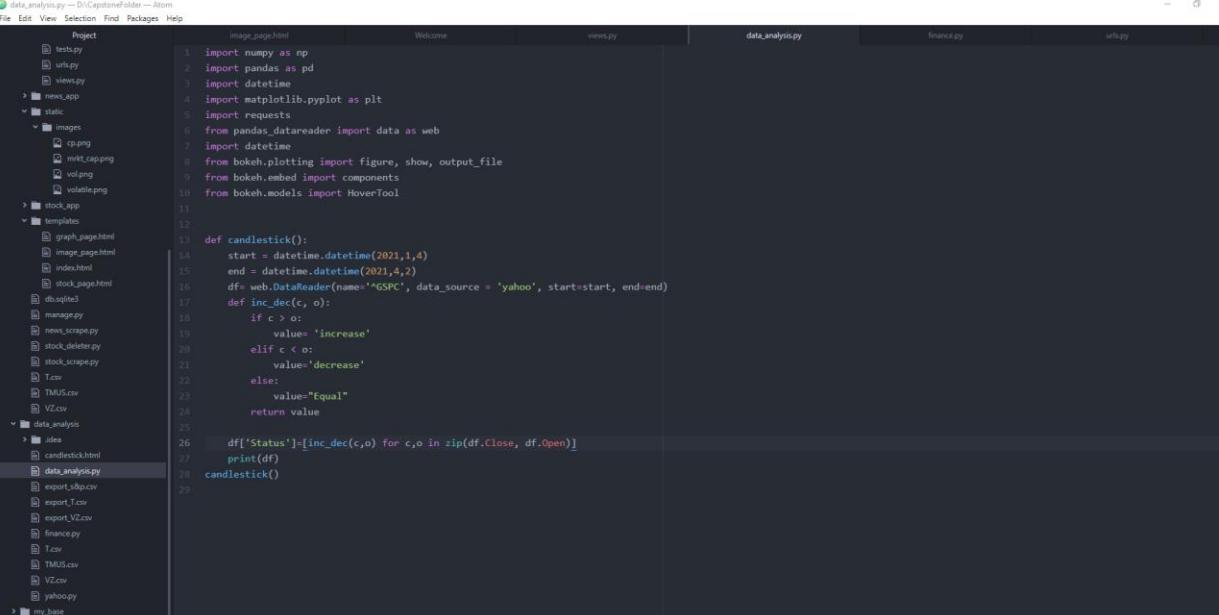
(virtual_env) D:\CapstoneFolder\data_analysis>python data_analysis.py
          High      Low      Open     Close    Volume   Adj Close
Date
2021-01-04  3769.989990  3662.709961  3764.610107  3700.649902  5006680000  3700.649902
2021-01-05  3737.830078  3695.070068  3698.020020  3726.860107  4582620000  3726.860107
2021-01-06  3783.040039  3705.340088  3712.199951  3748.139893  6049970000  3748.139893
2021-01-07  3811.550049  3764.709961  3764.709961  3803.790039  5080370000  3803.790039
2021-01-08  3826.689941  3783.600098  3815.050049  3824.679932  4764180000  3824.679932
...
...
...
2021-03-26  3978.189941  3917.120117  3917.120117  3974.540039  5467850000  3974.540039
2021-03-29  3981.830078  3943.250000  3969.310059  3971.090088  4619840000  3971.090088
2021-03-30  3968.010010  3944.350098  3963.340088  3958.550049  4103570000  3958.550049
2021-03-31  3994.409912  3966.979980  3967.250000  3972.889893  4564980000  3972.889893
2021-04-01  4020.629883  3992.780029  3992.780029  4019.870117  4151240000  4019.870117

[62 rows x 6 columns]

(virtual_env) D:\CapstoneFolder\data_analysis>
```

Figure 4.2-2 S&P 500 data returned in structured format.

Now that I have the data stored in a structured format, I can access specific rows/columns, as well as add, delete, or manipulate the data. The next process may appear a bit unconventional, however, I will create another function inside the candlestick function shown above. This function will contain two parameters, one for the closing price, one for the opening price. Based on whether the price of the index went up, down, or stayed the same on any given day, that row will be given a value of increase, decrease, or equal in a new column called "Status".



data_analysis.py - D:\CapstoneFolder — Atom

File Edit View Selection Find Packages Help

Project

- image_page.html
- Welcome
- views.py
- data_analysis.py
- finance.py
- utils.py

tests.py

urllib

views.py

> news_app

static

- images
- cp.png
- mkt_capping
- volang
- volatile.png

> stock_app

templates

- graph_page.html
- image_page.html
- index.html
- stock_page.html

db.sqlite3

manage.py

news_scrap.py

stock_delete.py

stock_scrap.py

T.csv

TMUS.csv

V2.csv

> data_analysis

idea

candlestick.html

data_analysis.py

export_skp.csv

export_T.csv

export_V2.csv

finance.py

T.csv

TMUS.csv

V2.csv

yahoo.py

my_base

virtual_env

CapstoneFolder

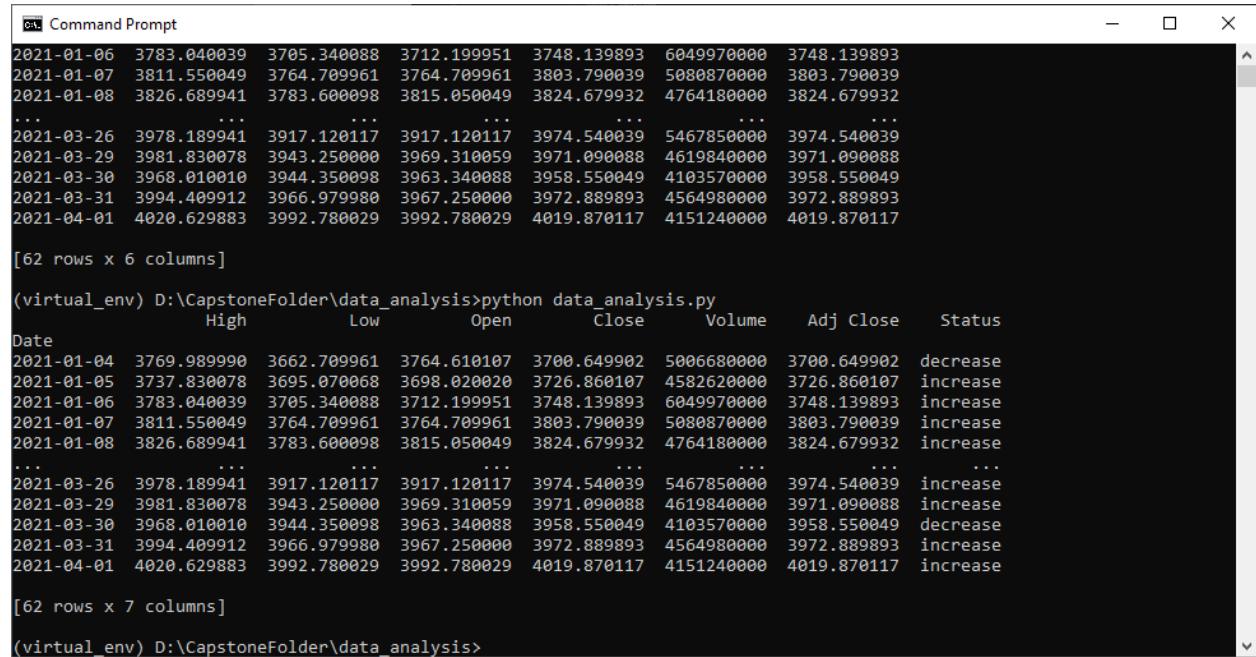
```
1 import numpy as np
2 import pandas as pd
3 import datetime
4 import matplotlib.pyplot as plt
5 import requests
6 from pandas_datareader import data as web
7 import datetime
8 from bokeh.plotting import figure, show, output_file
9 from bokeh.embed import components
10 from bokeh.models import HoverTool
11
12
13 def candlestick():
14     start = datetime.datetime(2021,1,4)
15     end = datetime.datetime(2021,4,2)
16     df= web.DataReader(name='GSPC', data_source = 'yahoo', start=start, end=end)
17     def inc_dec(c, o):
18         if c > o:
19             value='increase'
20         elif c < o:
21             value='decrease'
22         else:
23             value="Equal"
24         return value
25
26     df['Status']=[inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
27     print(df)
28     candlestick()
29
```

data_analysis/data_analysis.py 2686

CRLF UTF-8 Python GitHub < Git (0)

Figure 4.2-3 Creating new column for daily return 'Status'.

In the image above, the last line of code before the print statement initializes a new column and that column's name. Then the “inc_dec” function is called, and the parameters are used by the zip function, which is a powerful python function that allows iteration through two lists at the same time.



```

25
26     df['Status']=[inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
27     df['Middle']=(df.Open+df.Close)/2
28     df['Height']=abs(df.Open - df.Close)
29     print(df)
30     candlestick()
31

```

[62 rows x 7 columns]

(virtual_env) D:\CapstoneFolder\data_analysis>python data_analysis.py

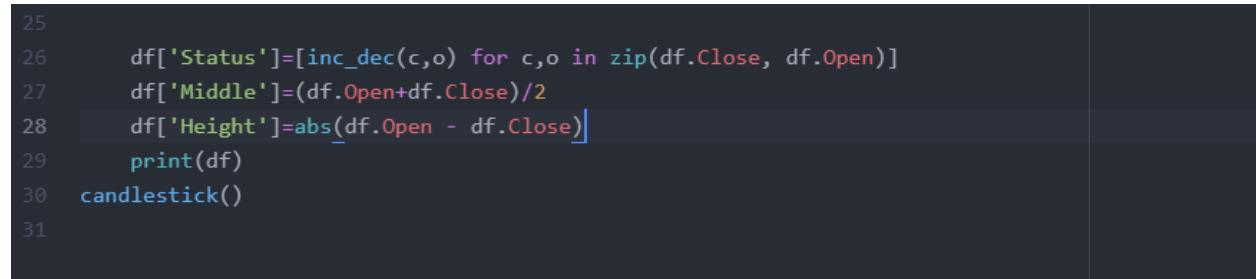
Date	High	Low	Open	Close	Volume	Adj Close	Status
2021-01-04	3769.989990	3662.709961	3764.610107	3700.649902	5006680000	3700.649902	decrease
2021-01-05	3737.830078	3695.070068	3698.020020	3726.860107	4582620000	3726.860107	increase
2021-01-06	3783.040039	3705.340088	3712.199951	3748.139893	6049970000	3748.139893	increase
2021-01-07	3811.550049	3764.709961	3764.709961	3803.790039	5080870000	3803.790039	increase
2021-01-08	3826.689941	3783.600098	3815.050049	3824.679932	4764180000	3824.679932	increase
...
2021-03-26	3978.189941	3917.120117	3917.120117	3974.540039	5467850000	3974.540039	increase
2021-03-29	3981.830078	3943.250000	3969.310059	3971.090088	4619840000	3971.090088	increase
2021-03-30	3968.010010	3944.350098	3963.340088	3958.550049	4103570000	3958.550049	decrease
2021-03-31	3994.409912	3966.979980	3967.250000	3972.889893	4564980000	3972.889893	increase
2021-04-01	4020.629883	3992.780029	3992.780029	4019.870117	4151240000	4019.870117	increase

[62 rows x 7 columns]

(virtual_env) D:\CapstoneFolder\data_analysis>

Figure4.2-4 Status column added to data frame.

Next, I will create two more columns to contain the “daily average price” and another to store the amounts of “daily price variation”. The average price column name will be “Middle”, and the variance will be “Height”. The column names are used for clearer context when building the plot begins, which will be demonstrated shortly.



```

25
26     df['Status']=[inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
27     df['Middle']=(df.Open+df.Close)/2
28     df['Height']=abs(df.Open - df.Close)
29     print(df)
30     candlestick()
31

```

Figure4.2-5 Creating columns for daily: average price & price variance

```
Command Prompt
2021-01-06 3783.040039 3705.340088 3712.199951 3748.139893 6049970000 3748.139893 increase
2021-01-07 3811.550049 3764.709961 3764.709961 3803.790039 5080870000 3803.790039 increase
2021-01-08 3826.689941 3783.600098 3815.050049 3824.679932 4764180000 3824.679932 increase
...
...
2021-03-26 3978.189941 3917.120117 3917.120117 3974.540039 5467850000 3974.540039 increase
2021-03-29 3981.830078 3943.250000 3969.310059 3971.090088 4619840000 3971.090088 increase
2021-03-30 3968.010010 3944.350098 3963.340088 3958.550049 4183570000 3958.550049 decrease
2021-03-31 3994.409912 3966.979980 3967.250000 3972.889893 4564980000 3972.889893 increase
2021-04-01 4020.629883 3992.780029 3992.780029 4019.870117 4151240000 4019.870117 increase
[62 rows x 7 columns]

(virtual_env) D:\CapstoneFolder\data_analysis>python data_analysis.py
      High      Low      Open     Close ...   Adj Close    Status     Middle     Height
Date
2021-01-04 3769.989990 3662.709961 3764.610107 3700.649902 ... 3700.649902 decrease 3732.630005 63.960205
2021-01-05 3737.830078 3695.070068 3698.020020 3726.860107 ... 3726.860107 increase 3712.440063 28.840088
2021-01-06 3783.040039 3705.340088 3712.199951 3748.139893 ... 3748.139893 increase 3730.169922 35.939941
2021-01-07 3811.550049 3764.709961 3764.709961 3803.790039 ... 3803.790039 increase 3784.250000 39.080078
2021-01-08 3826.689941 3783.600098 3815.050049 3824.679932 ... 3824.679932 increase 3819.864990 9.629883
...
...
2021-03-26 3978.189941 3917.120117 3917.120117 3974.540039 ... 3974.540039 increase 3945.830078 57.419922
2021-03-29 3981.830078 3943.250000 3969.310059 3971.090088 ... 3971.090088 increase 3970.200073 1.780029
2021-03-30 3968.010010 3944.350098 3963.340088 3958.550049 ... 3958.550049 decrease 3960.945068 4.790039
2021-03-31 3994.409912 3966.979980 3967.250000 3972.889893 ... 3972.889893 increase 3970.069946 5.639893
2021-04-01 4020.629883 3992.780029 3992.780029 4019.870117 ... 4019.870117 increase 4006.325073 27.090088
[62 rows x 9 columns]

(virtual_env) D:\CapstoneFolder\data_analysis>
```

Now that the data has been collected and properly stored, it is time to begin using Bokeh to build the chart.

5.2.1 Building the Chart

To begin, I will define the size of the Bokeh plot and set the x-axis (horizontal) to hold the dates. Next, I will use Bokeh to build rectangular shapes in the graph. Their height and width will be determined by the daily values calculated and stored in the 3 new columns. I will create one rectangle for “increases” and one for “decreases” (based on value in ‘Status’ column), and attempt displaying the chart.

data_analysis.py - D:\CapstoneFolder -- Atom

File Edit View Selection Find Packages Help

Project

- tests.py
- urls.py
- views.py
- news_app
 - static
 - images
 - cp.png
 - mixt_cap.png
 - volimg
 - variable.png
- stock_app
- templates
 - graph_page.html
 - image_page.html
 - index.html
 - stock_page.html
- db.sqlite3
- manage.py
- news_scrapy
- stock_deleter.py
- stock_scrapy.py
- T.csv
- TMUS.csv
- V2.csv

data_analysis

- ideas
- candlestick.html
- data_analysis.html
- data_analysis.py**
 - export_S&P.csv
 - export_V2.csv
 - export_V2.csv
 - finance.py
 - T.csv
 - TMUS.csv
 - V2.csv
 - yahoo.py
- my_base
- virtual_env

data_analysis/data_analysis.py 35:11

image_page.html

```
1 import numpy as np
2 import pandas as pd
3 import datetime
4 import matplotlib.pyplot as plt
5 import requests
6 from pandas_datareader import data as web
7 import datetime
8 from bokeh.plotting import figure, show, output_file
9 from bokeh.embed import components
10 from bokeh.models import HoverTool
11
12
13 def candlestick():
14     start = datetime.datetime(2021,1,4)
15     end = datetime.datetime(2021,4,2)
16     df= web.DataReader(name='^GSPC', data_source = 'yahoo', start=start, end=end)
17     def inc_dec(c, o):
18         if c > o:
19             value='increase'
20         elif c < o:
21             value='decrease'
22         else:
23             value="Equal"
24         return value
25
26     df['Status']=[inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
27     df['Middle']=(df.Open+df.Close)/2
28     df['Height']=abs(df.Open - df.Close)
29
30     p=figure(x_axis_type='datetime', width=1000, height=500)
31     p.title='S&P 500: Year to Date'
32     hours= 12*60*60*1000
33     p.rect(df.index[df.Status=='increase'], df.Middle[df.Status=='increase'], hours, df.Height[df.Status=='increase'])
34     p.rect(df.index[df.Status=='decrease'], df.Middle[df.Status=='decrease'], hours, df.Height[df.Status=='decrease'])
35     show(p)
36
37 candlestick()
```

Welcome

views.py

data_analysis.py

finance.py

urls.py

Figure 4.2.1-1 creating Bokeh rectangles (components).

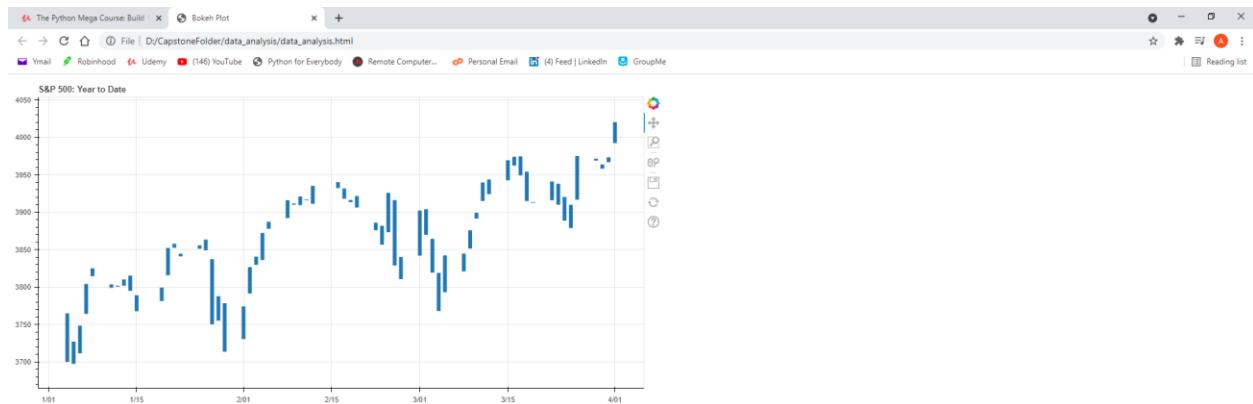


Figure 4.2.1-2 nondetailed candlestick chart displayed.

Although the graph did display the rectangular shapes and axis values correctly, it does not provide a viewer with very insightful detail. Therefore, I will add colors to the increasing (green) and decreasing (red) rectangles, as well as, add a Bokeh segment (line) to represent Highs and Lows.

```

25
26     df['Status']=[inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
27     df['Middle']=(df.Open+df.Close)/2
28     df['Height']=abs(df.Open - df.Close)
29
30     p=figure(x_axis_type='datetime', width=1000, height=500)
31     p.title='S&P 500: Year to Date'
32     hours= 12*60*60*1000
33
34     p.rect(df.index[df.Status=='increase'], df.Middle[df.Status=='increase'],
35             hours, df.Height[df.Status=='increase'], fill_color='green', line_color='black')
36     p.rect(df.index[df.Status=='decrease'], df.Middle[df.Status=='decrease'],
37             hours, df.Height[df.Status=='decrease'], fill_color='red', line_color='black')
38     p.segment(df.index, df.High, df.index, df.Low, color='black')
39     show(p)

```

Figure4.2.1-3 adding red and green colors & segment lines to represent highs/lows.



Figure4.2.1-4 Output after adding colors and segments.

In the image above, the final output of Bokeh's powerful ability to create detailed, interactive visualizations is displayed. Next, I will use the same process to display the same kind of chart except the time interval will be from the beginning of the Covid Pandemic to April 2nd, 2021 (around a year).

```

40     start = datetime.datetime(2020,3,28)
41     end = datetime.datetime(2021,4,2)

```

Figure4.2.1-5 Defining Time interval for Post-Covid price performance.

Next, I will use the exact same script used to create the previous graph, to create this one. However, keep in mind that the start/end date has been altered.

File Edit View Selection Find Packages Help

Project

data_analysis.py

```
 1  import image.Image
 2  import finance
 3
 4  p=Figure(x_axis_type= 'datetime' , width=1000, height=600)
 5  p.title='S&P 500: Year to Date'
 6
 7  hours= 12*60*60*1000
 8
 9  p.rect(df.index[df.Status=='increase'], df.Middle[df.Status=='increase'],
10        hours, df.Height[df.Status=='increase'], fill_color='green', line_color='black')
11  p.rect(df.index[df.Status=='decrease'], df.Middle[df.Status=='decrease'],
12        hours, df.Height[df.Status=='decrease'], fill_color='red', line_color='black')
13  p.segment(df.index, df.High, df.index, df.Low, color='black')
14  show(p)
15
16  start = datetime.datetime(2020,3,28)
17  end = datetime.datetime(2021,4,2)
18
19  df= web.DataReader(name='^GSPC', data_source = 'yahoo', start=start, end=end)
20
21  def inc_dec(c, o):
22
23      if c > o:
24          value= 'increase'
25      elif c < o:
26          value='decrease'
27      else:
28          value="equal"
29
30      return value
31
32
33  df['Status']=inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
34  df['Middle']= (df.Open+df.Close)/2
35  df['Height']=abs(df.Open - df.Close)
36
37  plot=Figure(x_axis_type='datetime', width=1000, height=500)
38  plot.title='S&P 500: Post Covid Pandemic'
39
40  # p.grid.grid_line_alpha=0.3
41
42  hours= 12*60*60*1000
43
44  p.rect(df.index[df.Status=='increase'], df.Middle[df.Status=='increase'],
45        hours, df.Height[df.Status=='increase'], fill_color='green', line_color='black')
46
47  p.rect(df.index[df.Status=='decrease'], df.Middle[df.Status=='decrease'],
48        hours, df.Height[df.Status=='decrease'], fill_color='red', line_color='black')
49  p.segment(df.index, df.High, df.index, df.Low, color='black')
50
51  show(plot)
52
53
54  candlestick()
55
56
57
58
59
60
61
62
63
64
65
66
67
68
```

Figure 4.2.1-6 Creating the Post-Covid candlestick chart.

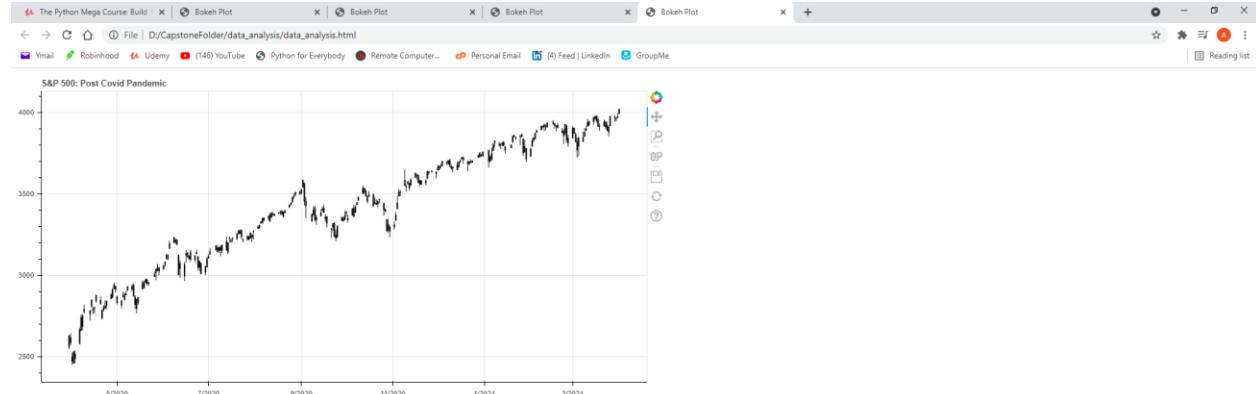


Figure 4.2.1-7 Post-Covid candlestick chart Output.

The new graph properly displayed with correct dates and prices. However, the candlestick chart may not be the best visualization technique for this type of analysis. While there are normally long bars to

represent the candlesticks, the rectangles almost appear as dots. Luckily, this Bokeh plot has the capability of zooming in, as well as some of other interactive functionalities located in the toolbar along the left side of the chart.

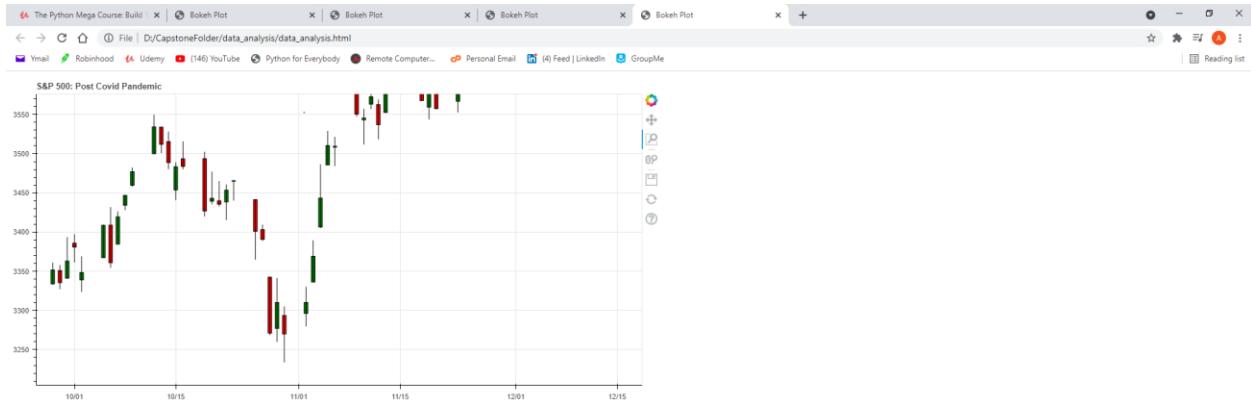


Figure4.2.1-8 Using Bokeh's interactive tools.

Lastly, I will create one more candlestick chart to visualize the S&P 500 index over the past 5 years. I will also use the same process that has been used so far in that I will be changing the start date to begin.

```
66      start = datetime.datetime(2016,4,2)
67      end = datetime.datetime(2021,4,2)
```

Figure4.2.1-9 Defining time interval for 5-year price performance.

```

66     start = datetime.datetime(2016,4,2)
67     end = datetime.datetime(2021,4,2)
68     df= web.DataReader(name='^GSPC', data_source = 'yahoo', start=start, end=end)
69     def inc_dec(c, o):
70         if c > o:
71             value= 'increase'
72         elif c < o:
73             value='decrease'
74         else:
75             value="Equal"
76         return value
77
78     df['Status']=[inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
79     df['Middle']=(df.Open+df.Close)/2
80     df['Height']=abs(df.Open - df.Close)
81     plots=figure(x_axis_type='datetime', width=1000, height=500)
82     plots.title='S&P 500: 2016-present'
83     hours= 12*60*60*1000
84     plots.rect(df.index[df.Status=='increase'], df.Middle[df.Status=='increase'],
85                 hours, df.Height[df.Status=='increase'], fill_color='green', line_color='black')
86
87     plots.rect(df.index[df.Status=='decrease'], df.Middle[df.Status=='decrease'],
88                 hours, df.Height[df.Status=='decrease'], fill_color='red', line_color='black')
89     plots.segment(df.index, df.High, df.index, df.Low, color='black')
90     show(plots)
91
92 candlestick()

```

Figure4.2.1-10 Creating the 5-year period candlestick chart.

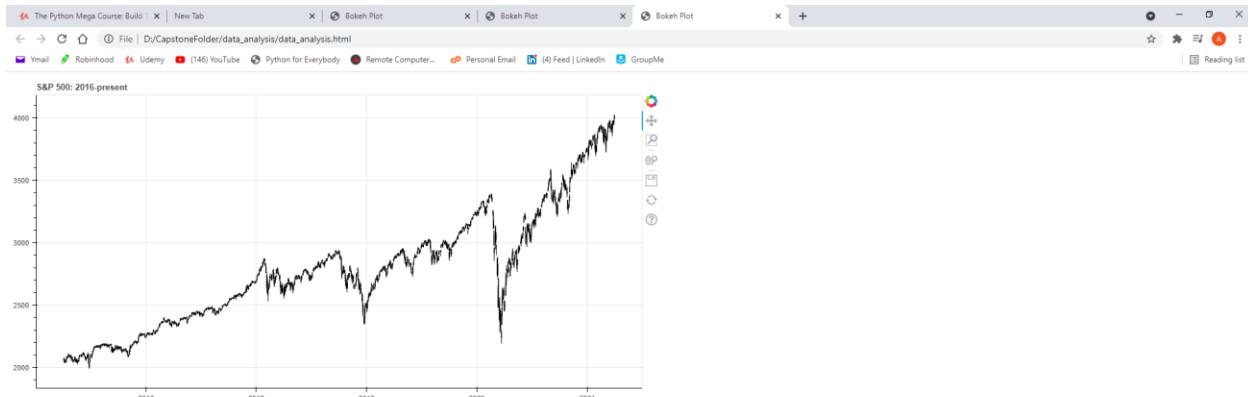


Figure4.2.1-11 5-year period candlestick chart Output.

Like the 2nd graph, the candlesticks are not very vivid on such a large time scale, fortunately, all of the graphs have the zoom capability.

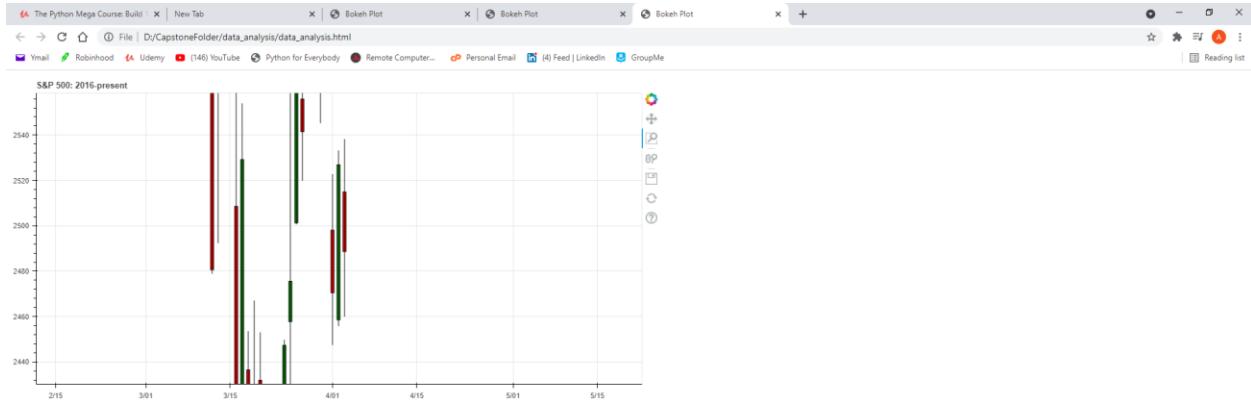
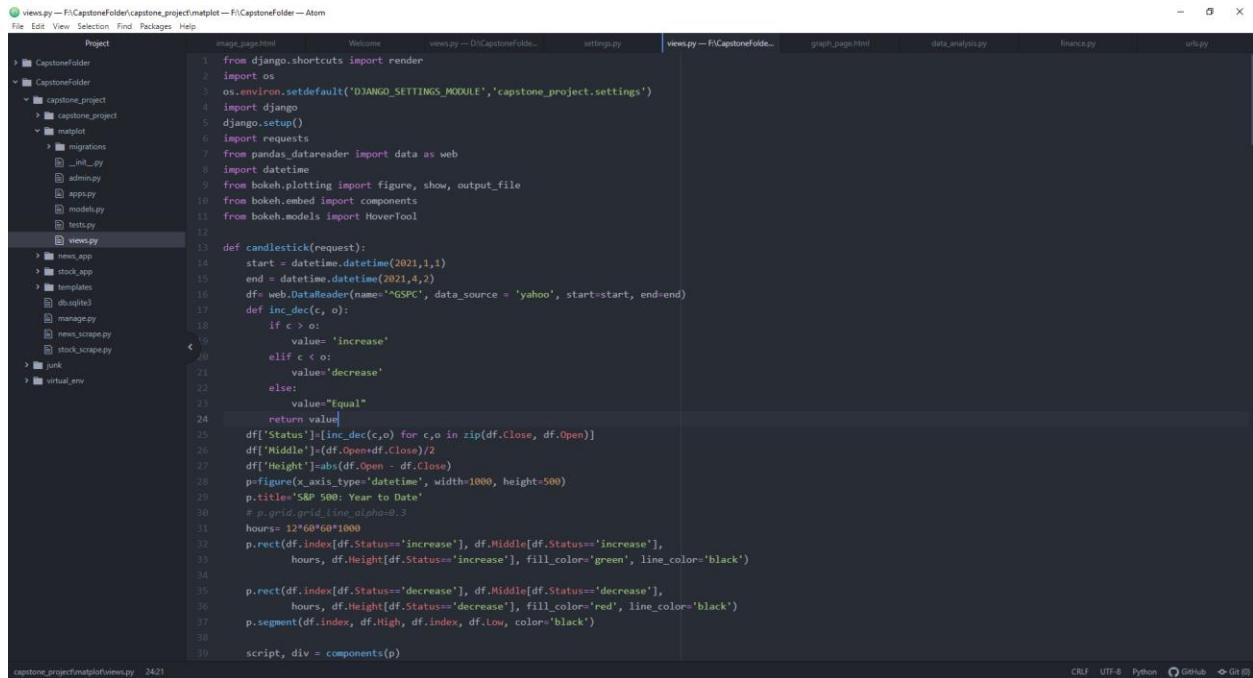


Figure 4.2.1-12 Using Bokeh's interactive tools.

5.2.2 Integrating Bokeh & Django

Now that I have created and modified the charts I will place the candlestick function in the “matplotlib” application’s “views.py” script, and create a new “urls.py” file and configure it with the project urls.py file as I have for the past two applications. Then, I will create a new template to display the graphs called, “graph_page.html” and use context processors to inject the chart/chart objects into the new template.



```

views.py — F:\CapstoneFolder\capstone_project\matplotlib — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project image.py.html Welcome views.py — D:\CapstoneFolde... settings.py | views.py — F:\Capstonefolde... graph_page.html data_analysis.py finance.py urls.py
views.py
1  from django.shortcuts import render
2  import os
3  os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'capstone_project.settings')
4  import django
5  django.setup()
6  import requests
7  from pandas_datareader import data as web
8  import datetime
9  from bokeh.plotting import figure, show, output_file
10 from bokeh.embed import components
11 from bokeh.models import HoverTool
12
13 def candlestick(request):
14     start = datetime.datetime(2021,1,1)
15     end = datetime.datetime(2021,4,2)
16     df= web.DataReader(name='^GSPC', data_source = 'yahoo', start=start, end=end)
17     def inc_dec(c, o):
18         if c > o:
19             value='increase'
20         elif c < o:
21             value='decrease'
22         else:
23             value="Equal"
24         return value
25     df['Status']= [inc_dec(c,o) for c,o in zip(df.Close, df.Open)]
26     df['Middle']=(df.Open+df.Close)/2
27     df['Height']=abs(df.Open - df.Close)
28     p=figure(x_axis_type='datetime', width=1000, height=500)
29     p.title='S&P 500: Year to Date'
30     # p.grid.grid_line_alpha=0.3
31     hours= 12*60*60*1000
32     p.rect(df.index[df.Status=='increase'], df.Middle[df.Status=='increase'],
33            hours, df.Height[df.Status=='increase'], fill_color='green', line_color='black')
34
35     p.rect(df.index[df.Status=='decrease'], df.Middle[df.Status=='decrease'],
36            hours, df.Height[df.Status=='decrease'], fill_color='red', line_color='black')
37     p.segment(df.index, df.High, df.index, df.Low, color='black')
38
39     script, div = components(p)
40
script, div = components(p)

```

Figure4.2.2-1 Placing “candlestick” function inside matplotlib/views.py.

At the end of each plot objects (“p”), I will create two new variable for each plot, these variables will contain the “components” function which takes the plot object itself as a parameter. This is the way Bokeh stores and transports objects created in the plot object (rectangles, segments, etc.).

```

32     p.rect(df.index[df.Status=='increase'], df.Middle[df.Status=='increase'],
33            hours, df.Height[df.Status=='increase'], fill_color='green', line_color='black')
34
35     p.rect(df.index[df.Status=='decrease'], df.Middle[df.Status=='decrease'],
36            hours, df.Height[df.Status=='decrease'], fill_color='red', line_color='black')
37     p.segment(df.index, df.High, df.index, df.Low, color='black')
38
39     script, div = components(p)
40

```

Figure4.2.2-2 Creating script, div variables to store plot components.

Next, I will give the last two charts their own set of unique variables to store the components of the plots (similarly to the script, div variables). These variables names will be used as context processors in the template. Therefore, they will all go in the render/request argument along with the new graph template that I will create next.

```

92     script1, div1 = components(p1)
93
94     return render(request, 'graph_page.html', {'script':script, 'div':div, 'scripts': scripts, 'divs': divs,'script1':script1, 'div1':div1})
95

```

Figure4.2.2-3 Rendering all 3 plot objects with context variables in the candlestick function.

5.2.3 Building the template

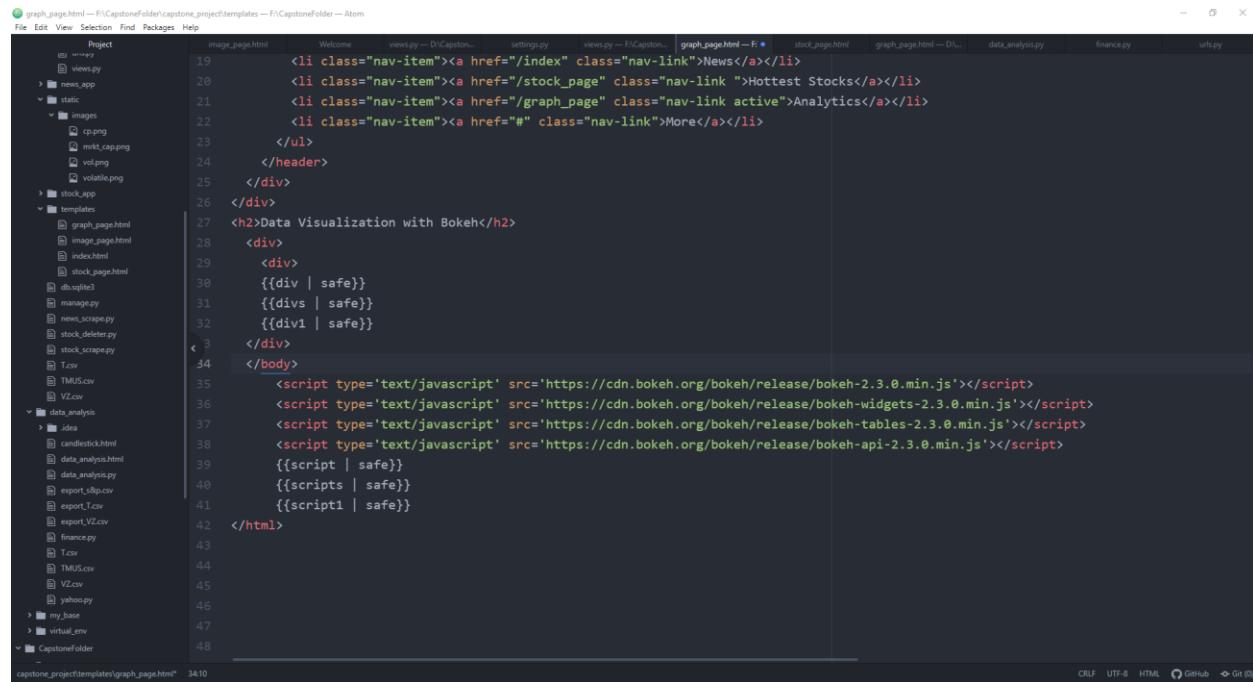
I will begin by creating a new html file in the templates directory, named “graph_page.html”. After a little research, I found that embedding a Bokeh plot inside HTML code requires a “`<div>`” element in the HTML body and a “`<script>`” element at the end of the template that contains a JavaScript file. The image below was taken from Bokeh’s official library of documentation ([link](#)) and does an excellent job of explaining the method. I will be changing the “`x.y.z`” at the end of the scripts to my current Bokeh version (2.3.0) and pass the needed context processors.

You can insert or template this script and its companion `<div>` in an HTML document and, when the script executes, your plot replaces the `<div>`.

For this to work you first need to load BokehJS, either locally or from a content delivery network (CDN). To load BokehJS from a CDN, add the following lines to your HTML text or template with the appropriate version replacing the `x.y.z`:

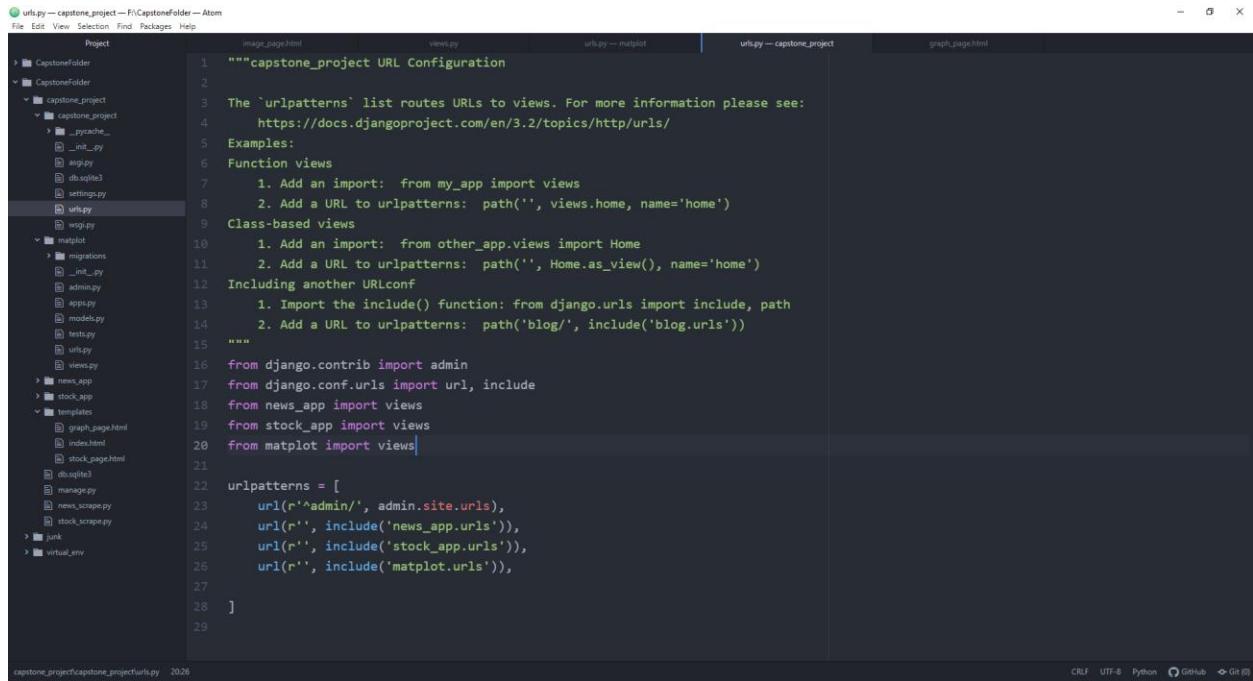
```
<script src="https://cdn.bokeh.org/bokeh/release/bokeh-x.y.z.min.js"
crossorigin="anonymous"></script>
<script src="https://cdn.bokeh.org/bokeh/release/bokeh-widgets-x.y.z.min.js"
crossorigin="anonymous"></script>
<script src="https://cdn.bokeh.org/bokeh/release/bokeh-tables-x.y.z.min.js"
crossorigin="anonymous"></script>
```

Figure4.2.3-1 Bokeh documentation for embedding plots in HTML templates.



```
graph_page.html — F:\CapstoneFolder\capstone_project\templates — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project
  image_page.html
  views.py
  views.py — D:\Capstone...
  settings.py
  views.py — F:\Capstone...
  graph_page.html — F:\●
  stock_page.html
  graph_page.html — D:\...
  data_analysis.py
  finance.py
  urls.py
graph_page.html
19  <li class="nav-item"><a href="/index" class="nav-link">News</a></li>
20  <li class="nav-item"><a href="/stock_page" class="nav-link ">Hottest Stocks</a></li>
21  <li class="nav-item"><a href="/graph_page" class="nav-link active">Analytics</a></li>
22  <li class="nav-item"><a href="#" class="nav-link">More</a></li>
23  </ul>
24  </header>
25  </div>
26  </div>
27  <h2>Data Visualization with Bokeh</h2>
28  <div>
29  <div>
30  {{div | safe}}
31  {{divs | safe}}
32  {{div1 | safe}}
33  </div>
34  </div>
35  <script type='text/javascript' src='https://cdn.bokeh.org/bokeh/release/bokeh-2.3.0.min.js'></script>
36  <script type='text/javascript' src='https://cdn.bokeh.org/bokeh/release/bokeh-widgets-2.3.0.min.js'></script>
37  <script type='text/javascript' src='https://cdn.bokeh.org/bokeh/release/bokeh-tables-2.3.0.min.js'></script>
38  <script type='text/javascript' src='https://cdn.bokeh.org/bokeh/release/bokeh-api-2.3.0.min.js'></script>
39  {{script | safe}}
40  {{scripts | safe}}
41  {{script1 | safe}}
42  </html>
43
44
45
46
47
48
```

Figure4.2.3-2 Placing script elements & context processors in graph template.



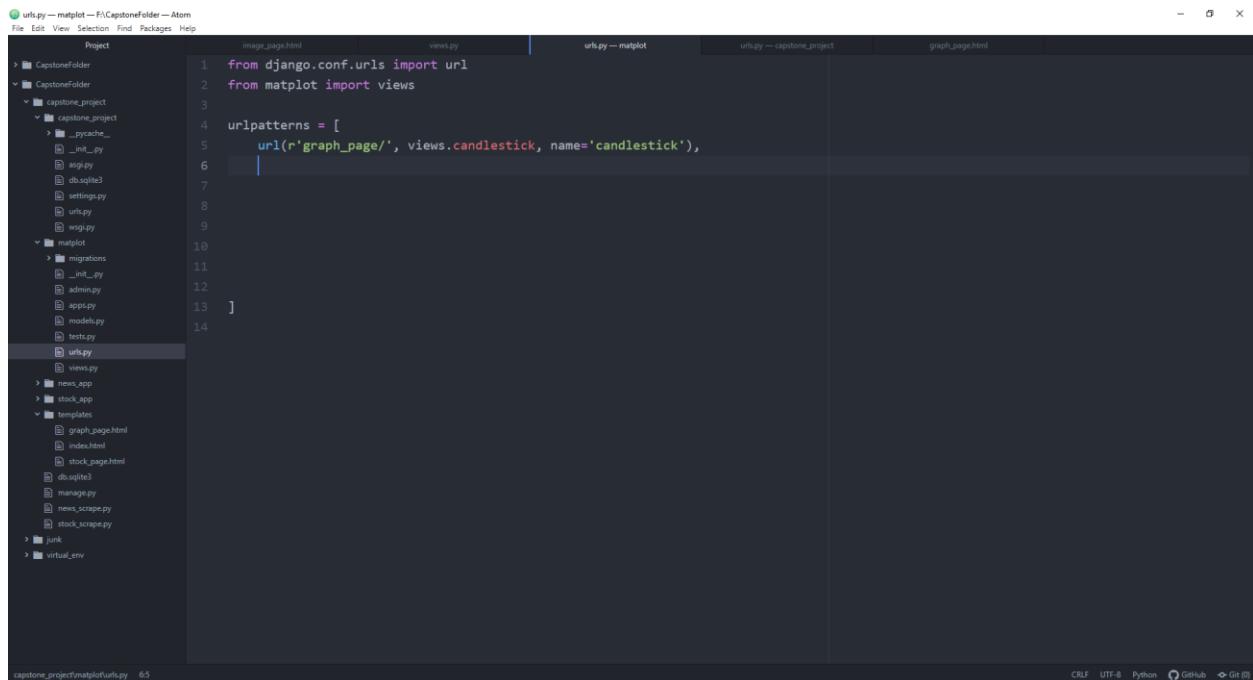
```

urls.py — capstone_project — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project
> CapstoneFolder
  CapstoneFolder
    capstone_project
      capstone_project
        __init__.py
        asgi.py
        db.sqlite3
        settings.py
        urls.py
        wsgi.py
      migrations
        __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
    news_app
    stock_app
    templates
      graph_page.html
      index.html
      stock_page.html
      db.sqlite3
      manage.py
      news_scrape.py
      stock_scrape.py
  junk
  virtual_env

1 """capstone_project URL Configuration
2
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.2/topics/http/urls/
5 Examples:
6     Function views
7         1. Add an import: from my_app import views
8             2. Add a URL to urlpatterns: path('', views.home, name='home')
9 Class-based views
10    1. Add an import: from other_app.views import Home
11        2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12 Including another URLconf
13     1. Import the include() function: from django.urls import include, path
14         2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16
17 from django.contrib import admin
18 from django.urls import url, include
19 from news_app import views
20 from stock_app import views
21 from matplotlib import views
22
23 urlpatterns = [
24     url(r'^admin/', admin.site.urls),
25     url(r'', include('news_app.urls')),
26     url(r'', include('stock_app.urls')),
27     url(r'', include('matplotlib.urls')),
28 ]
29

```

Figure 4.2.3-3 Configuring project URLs.



```

urls.py — matplotlib — F:\CapstoneFolder — Atom
File Edit View Selection Find Packages Help
Project
> CapstoneFolder
  CapstoneFolder
    capstone_project
      capstone_project
        __init__.py
        asgi.py
        db.sqlite3
        settings.py
        urls.py
        wsgi.py
      migrations
        __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
    news_app
    stock_app
    templates
      graph_page.html
      index.html
      stock_page.html
      db.sqlite3
      manage.py
      news_scrape.py
      stock_scrape.py
  junk
  virtual_env

1 from django.conf.urls import url
2 from matplotlib import views
3
4 urlpatterns = [
5     url(r'graph_page/', views.candlestick, name='candlestick'),
6 ]
7
8
9
10
11
12
13 ]
14

```

Figure 4.2.3-4 Configuring matplotlib application's URLs.

```
Command Prompt
new_errors = check(app_configs=app_configs, databases=databases)
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\checks\urls.py", line 13, in check_url_config
    return check_resolver(resolver)
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\core\checks\urls.py", line 23, in check_resolver
    return check_method()
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 412, in check
    for pattern in self.url_patterns:
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\utils\functional.py", line 48, in __get__
    res = instance.__dict__[self.name] = self.func(instance)
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 598, in url_patterns
    patterns = getattr(self.urlconf_module, "urlpatterns", self.urlconf_module)
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\utils\functional.py", line 48, in __get__
    res = instance.__dict__[self.name] = self.func(instance)
File "F:\CapstoneFolder\virtual_env\lib\site-packages\django\urls\resolvers.py", line 591, in urlconf_module
    return import_module(self.urlconf_name)
File "C:\Users\nicho\AppData\Local\Programs\Python\Python38-32\lib\importlib\_init_.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
File "<frozen importlib._bootstrap>", line 1014, in _gcd_import
File "<frozen importlib._bootstrap>", line 991, in _find_and_load
File "<frozen importlib._bootstrap>", line 975, in _find_and_load_unlocked
File "<frozen importlib._bootstrap>", line 671, in _load_unlocked
File "<frozen importlib._bootstrap_external>", line 783, in exec_module
File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
File "F:\CapstoneFolder\capstone_project\capstone_project\urls.py", line 20, in <module>
    from matplotlib import views
File "F:\CapstoneFolder\capstone_project\matplotlib\views.py", line 7, in <module>
    from pandas_datareader import data as web
ModuleNotFoundError: No module named 'pandas_datareader'

(virtual_env) F:\CapstoneFolder\capstone_project>cd F:\CapstoneFolder
```

Figure4.2.3-5 ERROR – module not found.

```
Command Prompt
(virtual_env) F:\CapstoneFolder>pip install pandas_datareader
Collecting pandas_datareader
  Using cached pandas_datareader-0.9.0-py3-none-any.whl (107 kB)
Requirement already satisfied: requests>=2.19.0 in f:\capstonefolder\virtual_env\lib\site-packages (from pandas_datareader) (2.25.1)
Collecting pandas>=0.23
  Downloading pandas-1.2.4-cp38-cp38-win32.whl (8.2 MB)
    |██████████| 8.2 MB 3.2 MB/s
Collecting lxml
  Using cached lxml-4.6.3-cp38-cp38-win32.whl (3.2 MB)
Requirement already satisfied: chardet<5,>=3.0.2 in f:\capstonefolder\virtual_env\lib\site-packages (from requests>=2.19.0->pandas_datareader) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in f:\capstonefolder\virtual_env\lib\site-packages (from requests>=2.19.0->pandas_datareader) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in f:\capstonefolder\virtual_env\lib\site-packages (from requests>=2.19.0->pandas_datareader) (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in f:\capstonefolder\virtual_env\lib\site-packages (from requests>=2.19.0->pandas_datareader) (2020.12.5)
Requirement already satisfied: pytz>=2017.3 in f:\capstonefolder\virtual_env\lib\site-packages (from pandas>=0.23->pandas_datareader) (2021.1)
Requirement already satisfied: numpy>=1.16.5 in f:\capstonefolder\virtual_env\lib\site-packages (from pandas>=0.23->pandas_datareader) (1.20.2)
Requirement already satisfied: python-dateutil>=2.7.3 in f:\capstonefolder\virtual_env\lib\site-packages (from pandas>=0.23->pandas_datareader) (2.8.1)
Requirement already satisfied: six>=1.5 in f:\capstonefolder\virtual_env\lib\site-packages (from python-dateutil>=2.7.3->pandas>=0.23->pandas_datareader) (1.15.0)
Installing collected packages: pandas, lxml, pandas-datareader
Successfully installed lxml-4.6.3 pandas-1.2.4 pandas-datareader-0.9.0
WARNING: You are using pip version 20.1.1; however, version 21.0.1 is available.
You should consider upgrading via the 'f:\capstonefolder\virtual_env\scripts\python.exe -m pip install --upgrade pip' command.

(virtual_env) F:\CapstoneFolder>
```

Figure4.2.3-7 Installing panda data-reader dependency.

```

from matplotlib import views
File "F:\CapstoneFolder\capstone_project\matplotlib\views.py", line 9, in <module>
    from bokeh.plotting import figure, show, output_file
ModuleNotFoundError: No module named 'bokeh'

```

Figure4.2.3-8 ERROR – no module named ‘Bokeh’.

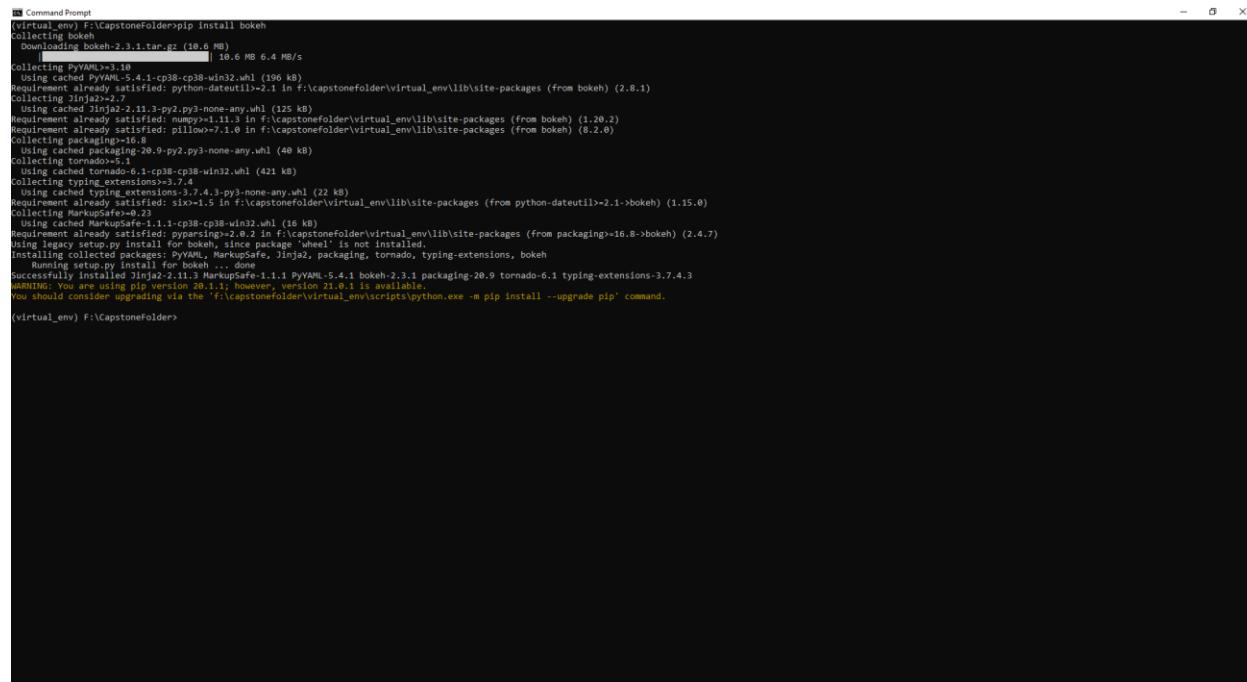


Figure4.2.3-9 Installing Bokeh.

After downloading all the required dependencies, which should have been done at the beginning of this phase, I will restart the server and use this URL to point to the graph template:

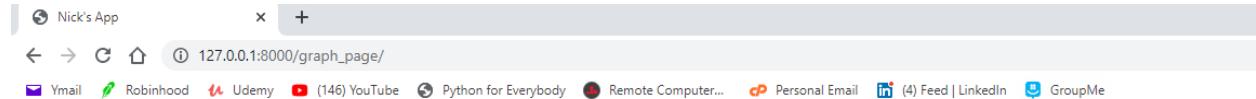


Figure4.2.3-10 Requesting the graph template.

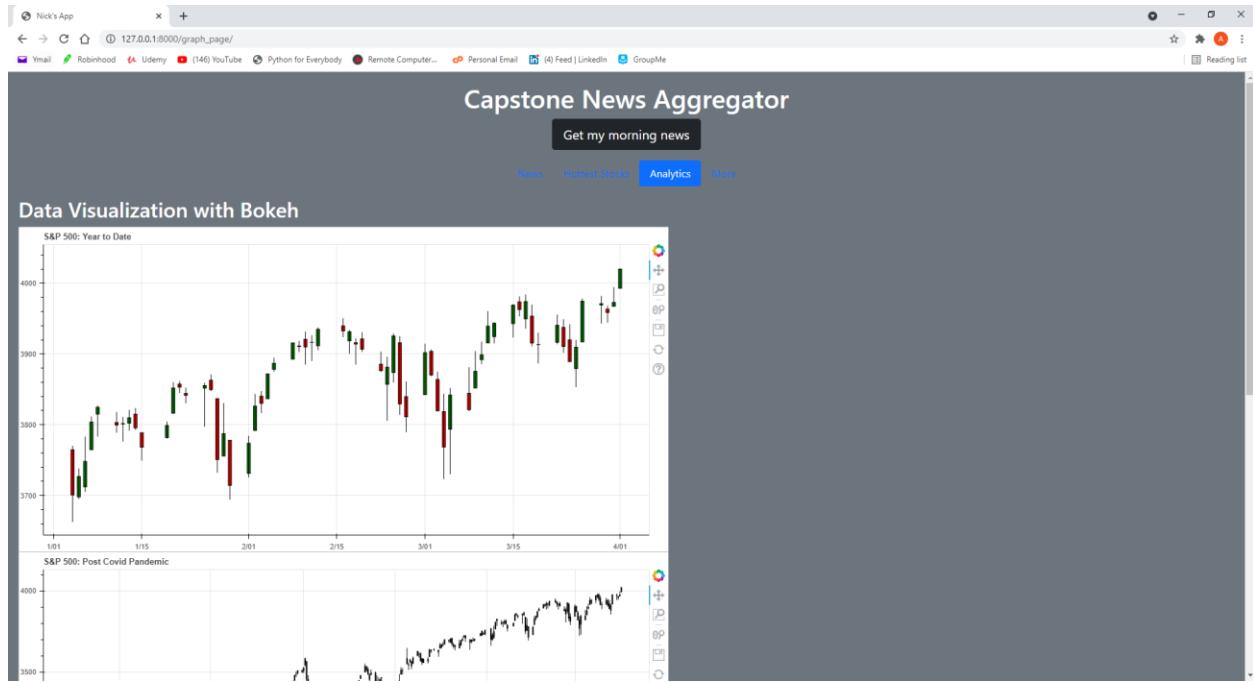


Figure4.2.3-11 Bokeh charts displayed in browser.



Figure4.2.3-11 Bokeh charts displayed in browser cont'd.

Since all the graphs were displayed correctly. The last thing I need to do is edit the other templates to configure the navbar, then center the new page elements (Bokeh plots).

5.3 Final Output

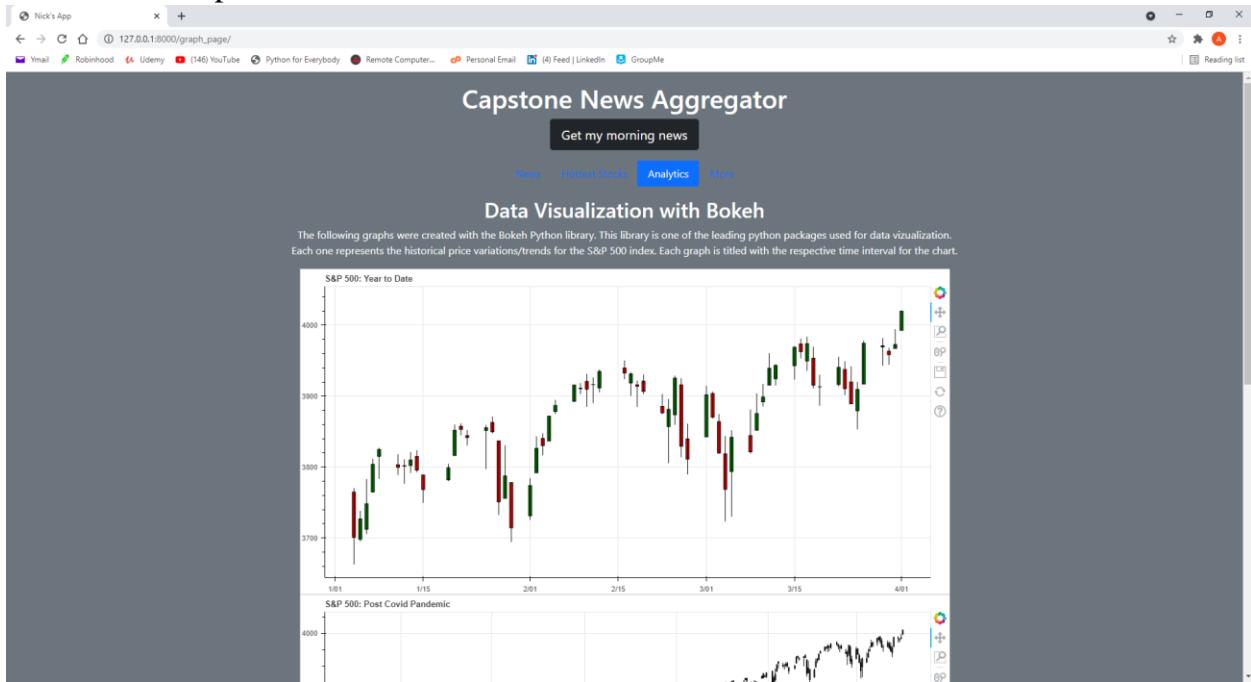


Figure4.3-1 Centered and fully formatted Bokeh candlestick charts.

5.4 Matplotlib & Pandas

Like Bokeh, Matplotlib is one of the leading Python libraries for data visualization. Therefore, majority of the coding and techniques used in this section will be very similar to those in creating a Bokeh plot. However, I will introduce a few new dependencies that will be needed to successfully collect and graph the data.

5.4.1 Context for the Analysis

I will be analyzing Verizon, AT&T, and T-Mobile by using various *price evaluation techniques* and analyzing graphical representations for certain trends.

5.4.2 Collecting the data

To begin, I will be using relatively the same technique to collect the data as I did for the S&P 500. The first set of charts will be based on 5 years of historical data. However, the “pandas_data reader” library will not be sufficient for finding all the financial values I will need (metrics excluding open, close, low, high, and volume). Therefore, I will be introducing the “yfinance” library, which is a powerful, convenient Python tool that gives access to a very wide range of financial data/statistics for any publicly traded security on finance.yahoo.com. However, this module will not be utilized until later in this section. For now, the following screenshots illustrate the same procedure/syntax I used in collecting the data for the Bokeh plots with “pandas_datareader”. Also, I created a new file for this script stored outside of the Django project directory for testing purposes.

The screenshot shows the Atom code editor interface. The left sidebar displays a project structure with several subfolders and files. The main editor area shows a Python script named `mplot.py` with the following code:

```

1  from pandas_datareader import data as web
2  import datetime
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  from matplotlib.widgets import CheckButtons
6  import yfinance as yf
7
8  start = datetime.datetime(2016,5,2)
9  end = datetime.datetime(2021,5,2)
10 vz = web.DataReader(name='VZ', data_source = 'yahoo', start=start, end=end)
11 att = web.DataReader(name='T', data_source = 'yahoo', start=start, end=end)
12 tmus = web.DataReader(name='TMUS', data_source = 'yahoo', start=start, end=end)
13 print(tmus)
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

```

The status bar at the bottom indicates the file is `junk\mplot.py` at line 71, with tabs for `CRLF`, `UTF-8`, `Python`, `GitHub`, and `Git`.

Figure4.4.2-1 Collecting the data for all 3 network carriers.

The screenshot shows a terminal window with the title "Command Prompt". The command `python mplot.py` is run, and the output is a pandas DataFrame for T-Mobile (TMUS) from May 2016 to May 2021. The output is as follows:

```

(virtual_env) F:\CapstoneFolder\junk>python mplot.py
   High      Low     Open     Close    Volume   Adj Close
Date
2016-05-02  40.099998  39.290001  39.360001  39.990002  3703600  39.990002
2016-05-03  39.880001  39.430000  39.830002  39.660000  3310500  39.660000
2016-05-04  39.910000  39.080002  39.589998  39.099998  3758400  39.099998
2016-05-05  39.669998  38.869999  39.099998  39.250000  3887800  39.250000
2016-05-06  39.619999  38.669998  39.000000  39.490002  2566000  39.490002
...
2021-04-09  130.529999  128.179993  130.529999  129.029999  2388100  129.029999
2021-04-12  131.000000  128.229996  128.490005  130.759995  2435800  130.759995
2021-04-13  131.199997  129.669998  130.300003  130.789993  4516400  130.789993
2021-04-14  130.789993  129.610001  130.679993  129.899994  2917300  129.899994
2021-04-15  131.779999  130.130005  131.100006  131.470001  3037007  131.470001
[1248 rows x 6 columns]
(virtual_env) F:\CapstoneFolder\junk>

```

Figure4.4.2-2 Output for printing the T-Mobile data frame.

Now that the data has been collected and stored in a structured data frame. I will first use the “closing prices” to create a line graph that illustrates each company’s price trend over the past five years.

5.4.3 Building the Charts

Inside the new “closing_prices” function, I used basic matplotlib methods to create a line graph showing all 3 companies closing price over a 5-year period.

```
13
14 def closing_prices():
15     vz['Close'].plot(label='Verizon')
16     att['Close'].plot(label='AT&T')
17     tmus['Close'].plot(label='T-Mobile')
18     plt.title('Closing Prices')
19     plt.legend()
20     plt.show()
21 closing_prices()
22
23
24
```

Figure4.4.3-1 Creating line graph for closing prices.

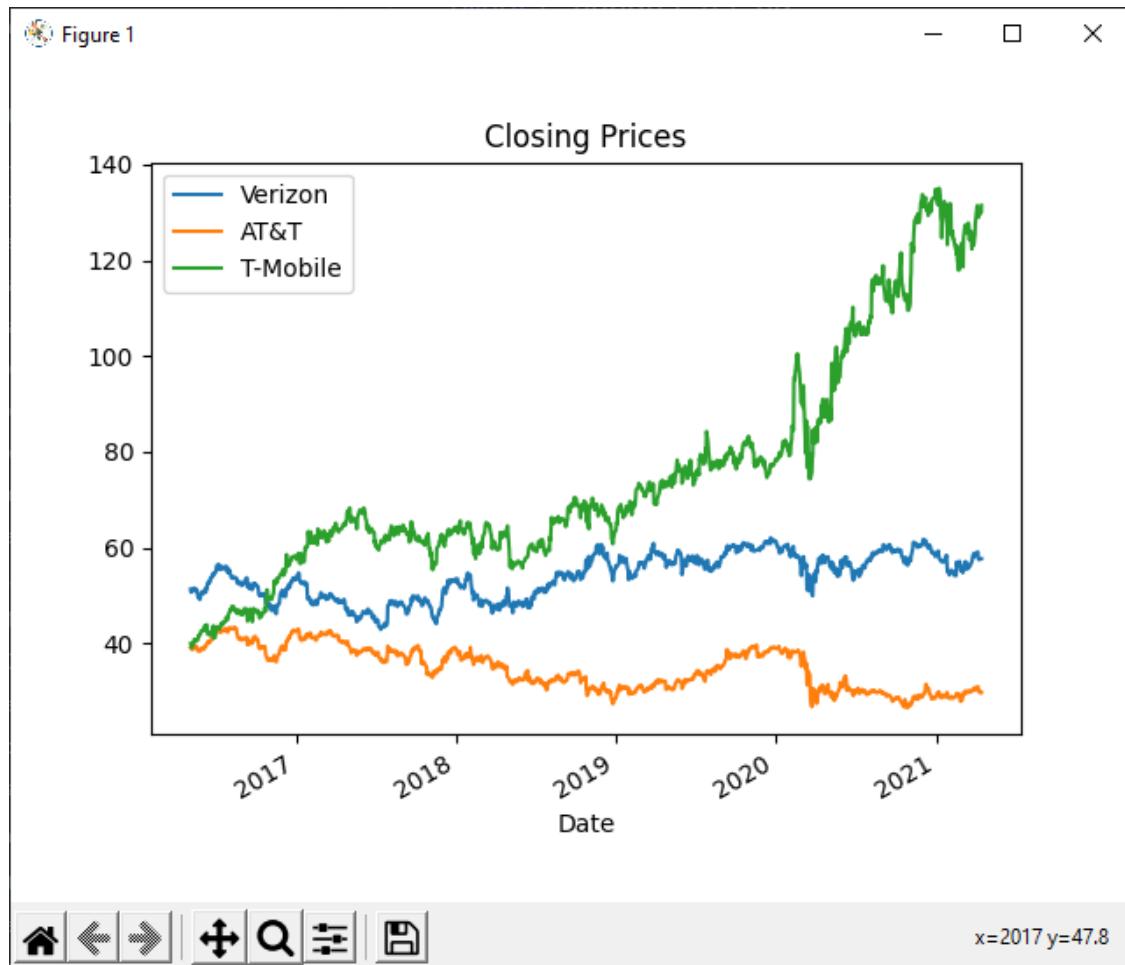
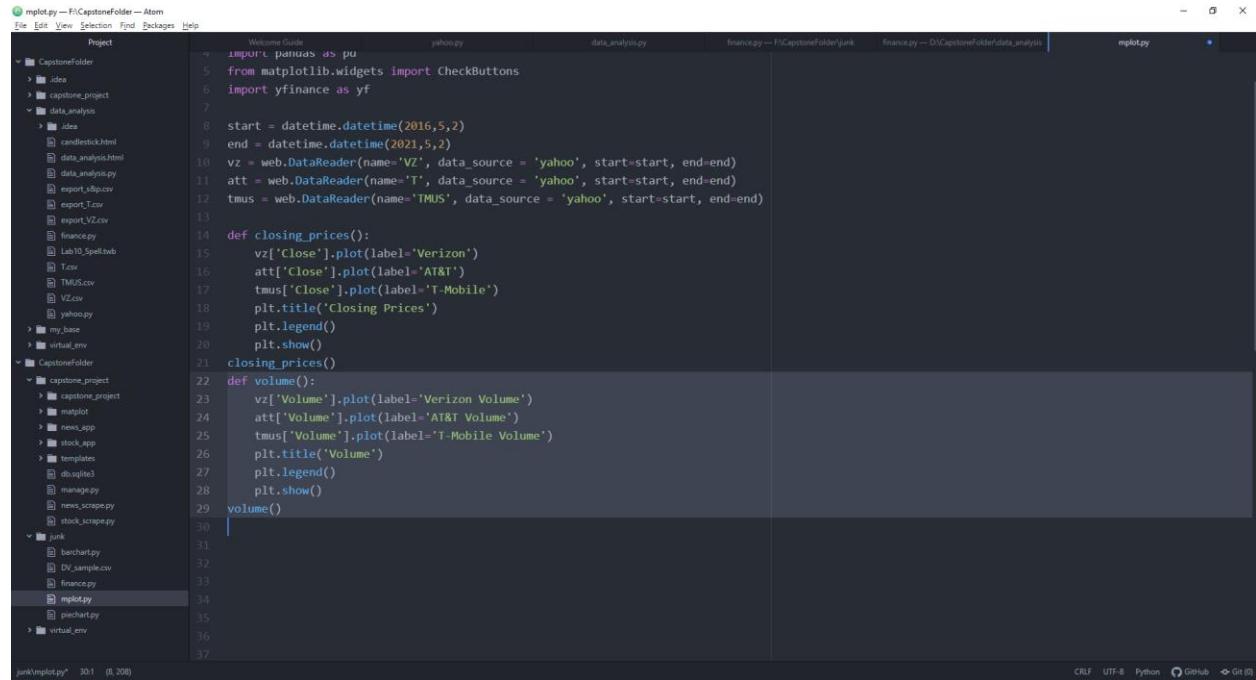


Figure4.4.3-2 Closing Prices Output

Next, I will do the same thing for the following graph; however, I will be using the stock's "volume" as the measure, rather than closing price. A stock's volume is a representation of the total amount of stocks/securities traded (bought/sold) over a period. In this case, we will be using the daily volume over the 5-year period.



```

mplot.py -- F:\CapstoneFolder -- Atom
File Edit View Selection Find Packages Help
Project Welcome Guide yahoo.py data_analysis.py finance.py -- F:\CapstoneFolder\data_analysis finance.py -- D:\CapstoneFolder\data_analysis mplot.py
- ⌂ X
4 import pandas as pd
5 from matplotlib.widgets import CheckButtons
6 import yfinance as yf
7
8 start = datetime.datetime(2016,5,2)
9 end = datetime.datetime(2021,5,2)
10 vz = web.DataReader(name='VZ', data_source = 'yahoo', start=start, end=end)
11 att = web.DataReader(name='T', data_source = 'yahoo', start=start, end=end)
12 tmus = web.DataReader(name='TMUS', data_source = 'yahoo', start=start, end=end)
13
14 def closing_prices():
15     vz['Close'].plot(label='Verizon')
16     att['Close'].plot(label='AT&T')
17     tmus['Close'].plot(label='T-Mobile')
18     plt.title('Closing Prices')
19     plt.legend()
20     plt.show()
21
22 def volume():
23     vz['Volume'].plot(label='Verizon Volume')
24     att['Volume'].plot(label='AT&T Volume')
25     tmus['Volume'].plot(label='T-Mobile Volume')
26     plt.title('Volume')
27     plt.legend()
28     plt.show()
29
30
31
32
33
34
35
36
37

```

File: mplot.py (8, 208) CRLF UTF-8 Python GitHub Git

Figure 4.4.3-3 Creating line graph for volume.

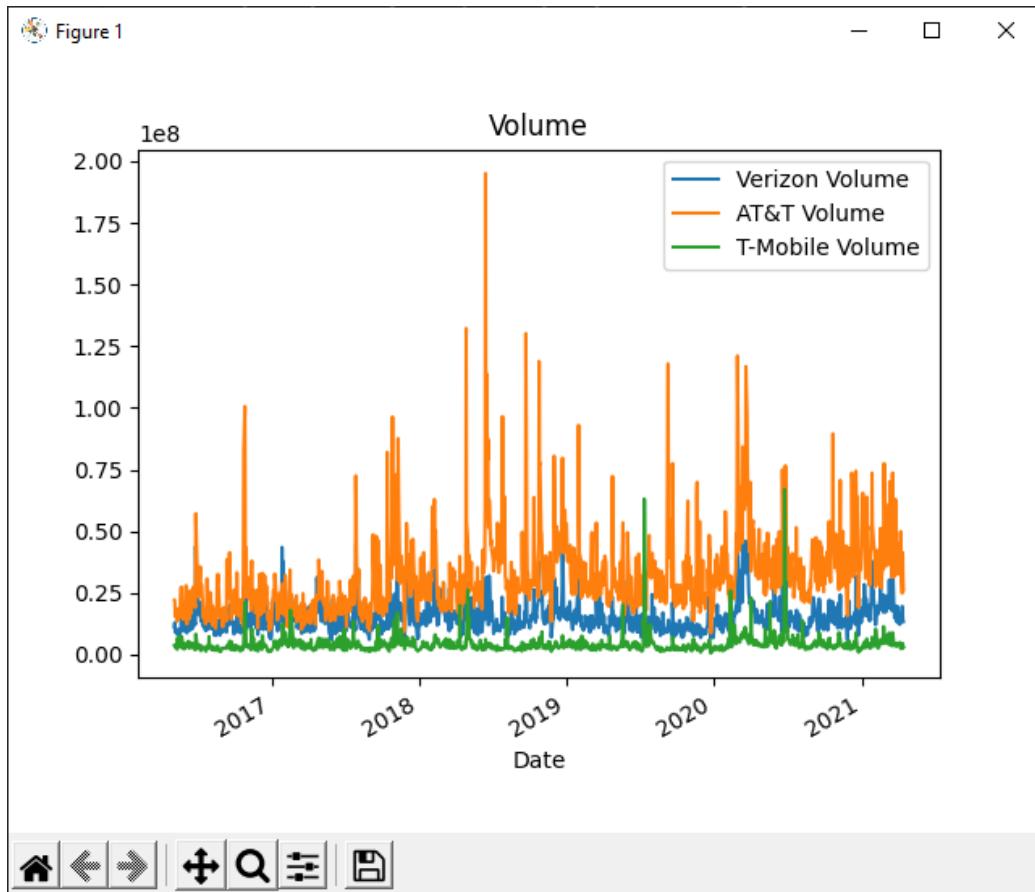
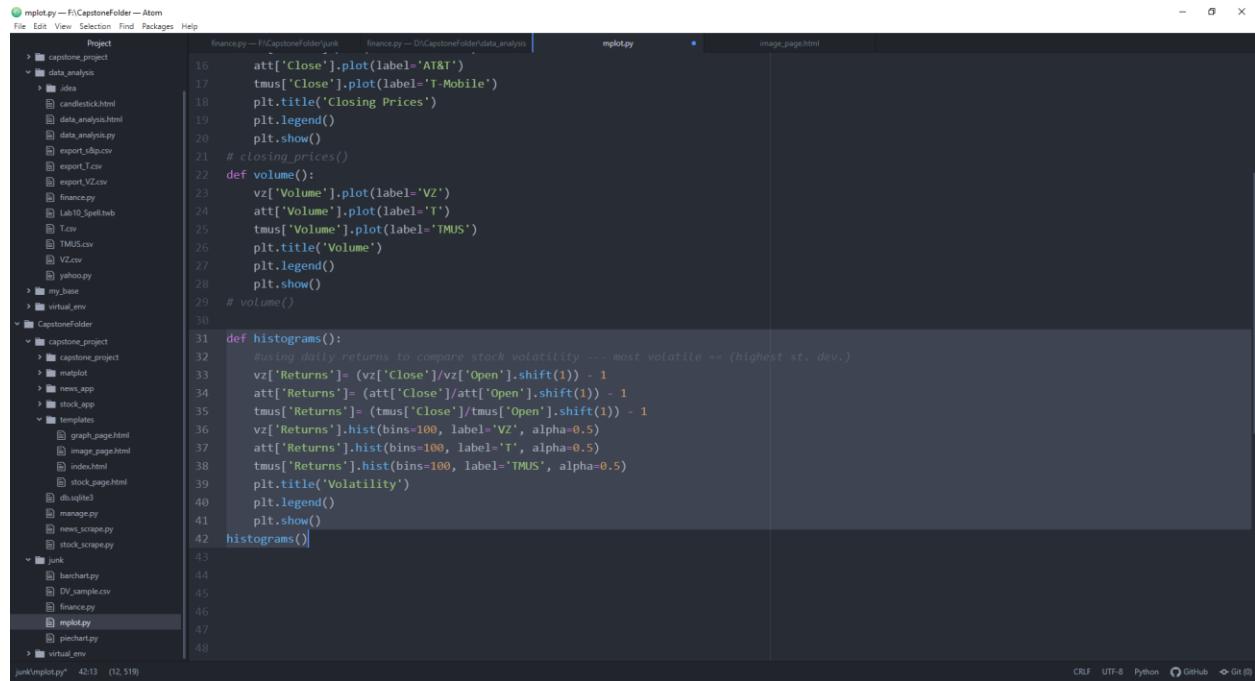


Figure 4.4.3-4 Volume Output

Next, I will create a histogram measuring “volatility”. Volatility is calculated by using the daily returns of each company. Then, the return values can be used to indicate a measure of variance or change in the company's stock price over time. This allows us to assess the amount of risk buying a stock may impose based on the likelihood of changing prices slightly or drastically.



```

mplplot.py -- F:\CapstoneFolder -- Atom
File Edit View Selection Find Packages Help
Project
  - capstone_project
    - data_analysis
      - candlestick.html
      - data_analysis.html
      - data_analysis.py
      - export_d4p.csv
      - export_t.csv
      - export_VZ.csv
      - finance.py
      - Lab10_Spell.twb
      - T.csv
      - TMUS.csv
      - VZ.csv
      - yahoo.py
    - my_base
    - virtual_env
  - CapstoneFolder
    - capstone_project
      - capstone_project
      - matplotlib
      - news_app
      - stock_app
      - templates
        - graph_page.html
        - image_page.html
        - index.html
        - stock_page.html
        - db.sqlite3
        - msnaps.py
        - news_scrape.py
        - stock_scrape.py
    - junk
      - barchart.py
      - DV_sample.csv
      - finance.py
      - mplplot.py
      - piechart.py
    - virtual_env
  - finance.py -- F:\CapstoneFolder\junk
  - finance.py -- D:\CapstoneFolder\data_analysis
mplplot.py
image_page.html
16      att['Close'].plot(label='AT&T')
17      tmus['Close'].plot(label='T-Mobile')
18      plt.title('Closing Prices')
19      plt.legend()
20      plt.show()
21      # closing_prices()
22      def volume():
23          vz[['Volume']].plot(label='VZ')
24          att[['Volume']].plot(label='T')
25          tmus[['Volume']].plot(label='TMUS')
26          plt.title('Volume')
27          plt.legend()
28          plt.show()
29      # volume()
30
31      def histograms():
32          #using daily returns to compare stock volatility --- most volatile == (highest st. dev.)
33          vz[['Returns']] = (vz['Close']/vz['Open'].shift(1)) - 1
34          att[['Returns']] = (att['Close']/att['Open'].shift(1)) - 1
35          tmus[['Returns']] = (tmus['Close']/tmus['Open'].shift(1)) - 1
36          vz[['Returns']].hist(bins=100, label='VZ', alpha=0.5)
37          att[['Returns']].hist(bins=100, label='T', alpha=0.5)
38          tmus[['Returns']].hist(bins=100, label='TMUS', alpha=0.5)
39          plt.title('Volatility')
40          plt.legend()
41          plt.show()
42      histograms()
43
44
45
46
47
48

```

Figure 4.4.3-5 Creating histogram for Volatility.

In the first three lines of code inside the “histograms” function displayed above, I am creating a new column in each data frame that calculates daily returns based off *the absolute value in the difference of the open/close prices* of that day. Next, I use basic matplotlib syntax to create a histogram representing the new columns.

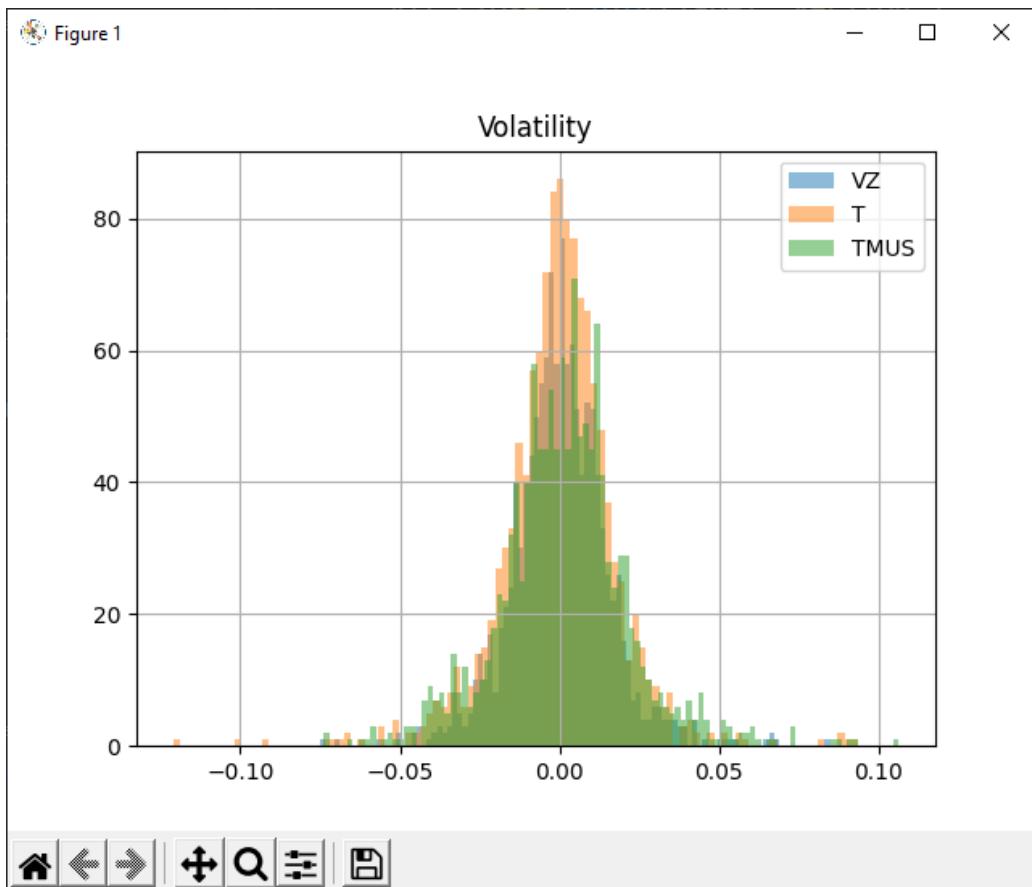


Figure 4.4.3-6 Volatility Output

The last graph I will create using the 5-year period sets of data, is another histogram that illustrates each company's market capitalization. Market capitalization refers to the total value of all a company's stock shares. It is calculated by multiplying the price of an individual stock by its total number of outstanding shares. The calculations for market capitalization are demonstrated in the first 3 lines of code inside the new "market_cap" function displayed below. Lastly, I used the same plotting technique to create this histogram as I used for the "volatility" graph.

```

 24     att['Volume'].plot(label='T')
 25     tmus['Volume'].plot(label='TMUS')
 26     plt.title('Volume')
 27     plt.legend()
 28     plt.show()
 29     volume()
 30
 31 def histograms():
 32     #using daily returns to compare stock volatility ... most volatile == (highest st. dev.)
 33     vz['Returns']= (vz['Close']/vz['Open'].shift(1)) - 1
 34     att['Returns']= (att['Close']/att['Open'].shift(1)) - 1
 35     tmus['Returns']= (tmus['Close']/tmus['Open'].shift(1)) - 1
 36     vz['Returns'].hist(bins=100, label='VZ', alpha=0.5)
 37     att['Returns'].hist(bins=100, label='T', alpha=0.5)
 38     tmus['Returns'].hist(bins=100, label='TMUS', alpha=0.5)
 39     plt.title('Volatility')
 40     plt.legend()
 41     plt.show()
 42     histograms()
 43
 44
 45 def market_cap():
 46     vz['Market Cap'] = vz['Close']* vz['Volume']
 47     att['Market Cap'] = att['Close']* att['Volume']
 48     tmus['Market Cap'] = tmus['Close']* tmus['Volume']
 49     vz['Market Cap'].hist(bins=100, label='VZ', alpha=0.5)
 50     att['Market Cap'].hist(bins=100, label='T', alpha=0.5)
 51     tmus['Market Cap'].hist(bins=100, label='TMUS', alpha=0.5)
 52     plt.title('Market Capitalization')
 53     plt.legend()
 54     plt.show()
 55     market_cap()

```

Figure 4.4.3-7 Creating histogram for market capitalization.

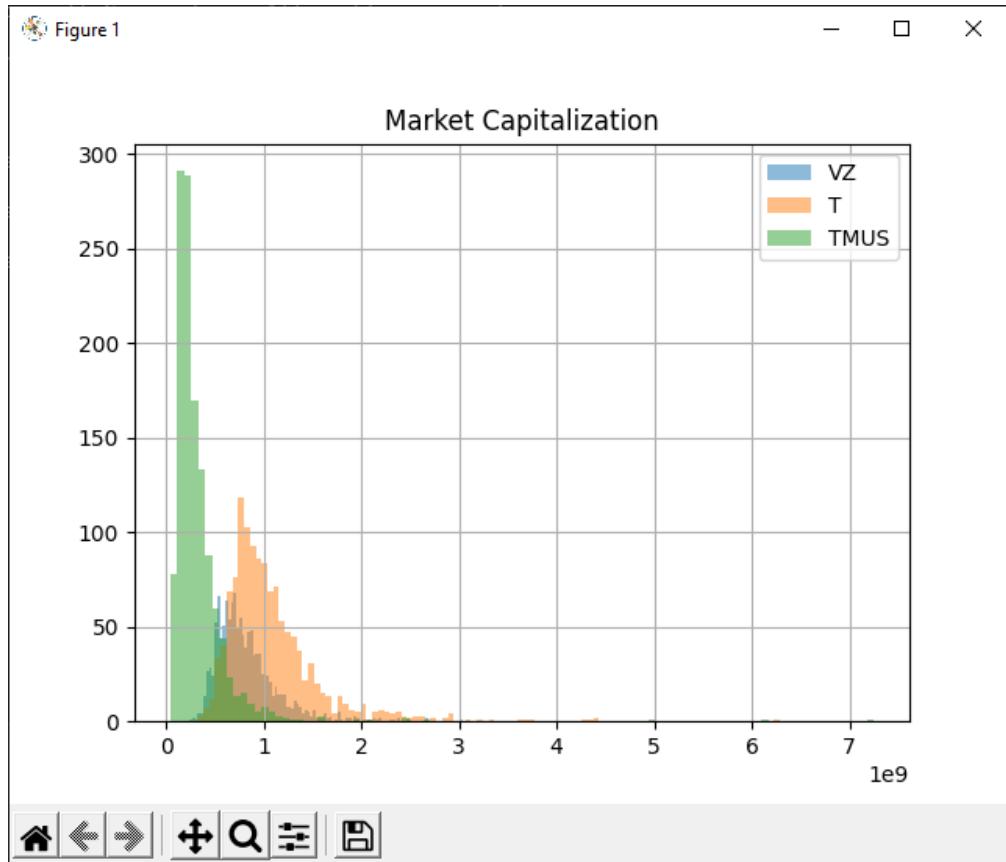


Figure4.4.3-8 Market Capitalization Output.

5.5 Matplotlib & Yfinance

Previously in this section and the last, I have been creating data visualizations with the assistance of the “pandas_datareader”. This worked flawlessly for the purpose I needed for, however, to take a step further in the analysis, I will need a new library that gives access to more complex financial data (besides open, close, high, volume, etc.). This library is known as, “Yfinance”. It is a popular open-source library to access the financial data available on Yahoo Finance.

5.5.1 Collecting the data

First, I will use the “info” method to show the general capabilities of this new library. When, the code is run, a large dictionary is returned containing a wide range of financial information for that company.

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import yfinance as yf
4
5 vz = yf.Ticker('VZ').info
6 print(vz)
7
8
```

Figure4.5.1-1 Testing the info method for Verizon's financial data.

Figure 4.5.1-2 info method output.

Next, I will use a for loop to obtain the “info” for all three companies and append the data to a new list.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import yfinance as yf
4
5 tickers = ['VZ', 'T', 'TMUS']
6 infos = []
7 for x in tickers:
8     infos.append(yf.Ticker(x).info)
9
10
```

Figure 4.5.1-3 Storing each company's financial information in a list.

Once this list has been created, I will define the values I want to obtain from the information in a new list. Although, we are not using the “pandas_datareader” sub-package in this section, we are using the “pandas” parent Python library to create data frames with the data collected using “Yfinance”. Once the data frame is created, I will set the index equal to the ticker symbols to make each row identifiable. The last line of code before the print statement, defines the columns of the data frame as the values in the “evals” list – the screenshot below illustrates this process and its output.

```

5  tickers = ['VZ', 'T', 'TMUS']
6  infos = []
7  for x in tickers:
8      infos.append(yf.Ticker(x).info)
9
10 evals = ['dividendYield', 'marketCap', 'beta', 'forwardPE']
11 df = pd.DataFrame(infos)
12 df.set_index('symbol')
13 x=x[x.columns[x.columns.isin(evals)]]
14 print(x)
15

```

Figure 4.5.1-4 Creating new columns to store evaluation values for each company.

```

C:\ Command Prompt

(virtual_env) F:\CapstoneFolder\junk>python finance.py
           beta  marketCap  forwardPE  dividendYield
symbol
VZ      0.463320  241279188992  11.272727      0.0431
T       0.737451  213796880384   9.418239      0.0699
TMUS    0.572531  165354536960  39.363903        NaN

(virtual_env) F:\CapstoneFolder\junk>_

```

Figure 4.5.1-5 Output for new data frame.

5.5.2 Building the charts

Now that the data is collected it is time to put Matplotlib to work. Since I will be representing/comparing numerical values rather than analyzing change over time, I will use simple bar charts to visualize the data. Therefore, most of the work was completed by just collecting the data and storing it in a structured format. First, I will be creating a bar chart for the *price-to-earnings ratio* of each company. To do this, I will just need to set the x-axis to show the tickers and the y-axis to measure the values. P/E ratio is one of the most widely used metrics for investors and analysts to determine stock valuation. In addition to showing whether a company's stock price is overvalued or undervalued, the P/E can reveal how a stock's valuation compares to its industry group or a benchmark like the S&P 500 index.

```

14
15 pe = plt.bar(x.index,x.forwardPE, color=['k','orange','cyan'])
16 pe = plt.title('Price-to-Earnings Ratio')
17 pe = plt.show()
18
19

```

Figure4.5.2-1 Creating bar chart for P/E ratio.

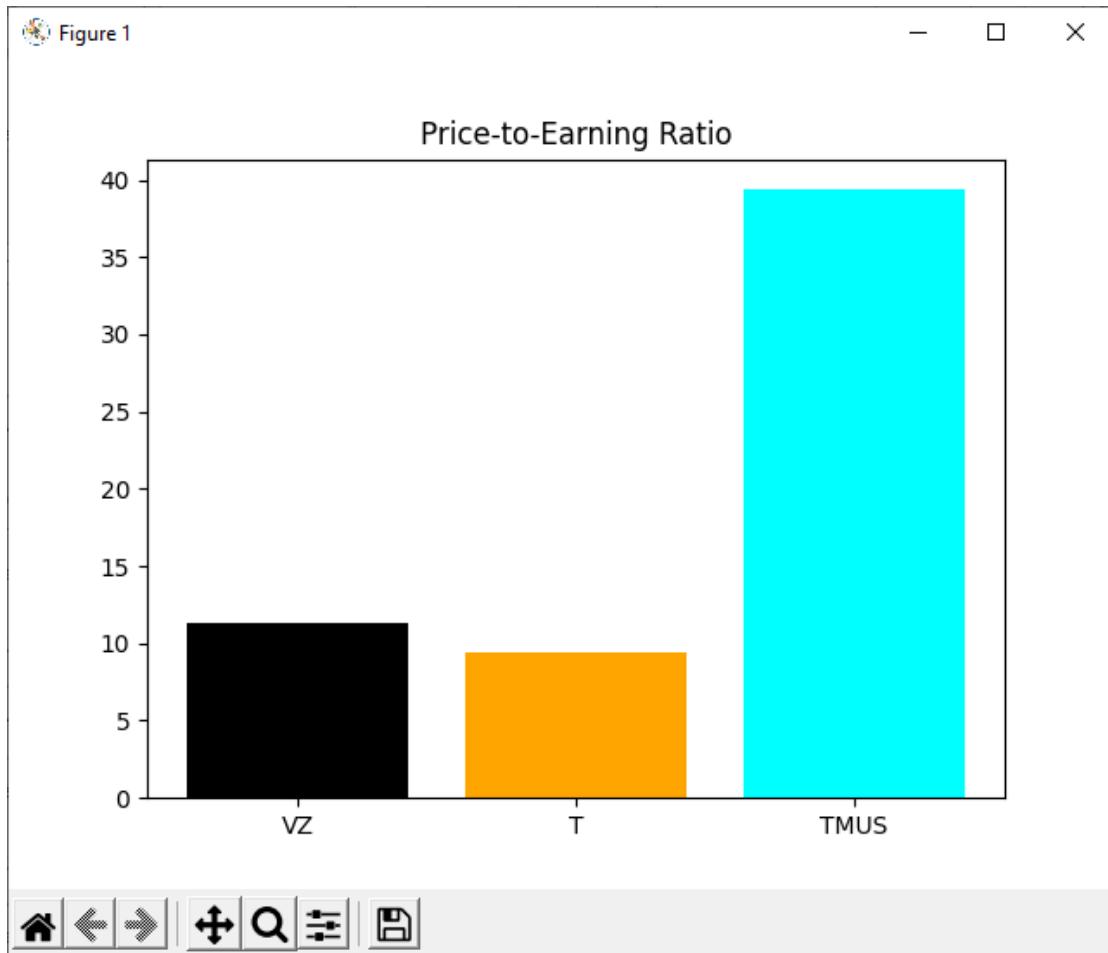


Figure4.5.2-2 P/E ratio Output.

Next, I will use the same process to create similar charts to represent each company's dividend yield, beta, and create another market capitalization chart using more accurate data.

```

18
19 beta = plt.bar(x.index,x.beta, color=('k','orange','cyan'))
20 beta = plt.title('Beta')
21 beta = plt.show()
22
23 mc = plt.bar(x.index,x.marketCap, color=('k','orange','cyan'))
24 mc = plt.title('Market Capitalization')
25 mc = plt.show()
26
27 divd = plt.bar(x.index,x.dividendYield, color=('k','orange','cyan'))
28 divd = plt.title('Dividend Yield')
29 divd = plt.show()

```

Figure4.5.2-3 Creating bar charts for Beta, Dividend Yield, & Market Capitalization

Beta is a measure of a stock's volatility in relation to the overall market (in this case the S&P 500, since all 3 companies are in this index). If a stock moves less than the market, the stock's beta is less than 1.0. High-beta stocks are supposed to be riskier but provide higher return potential; low-beta stocks pose less risk but also lower returns.

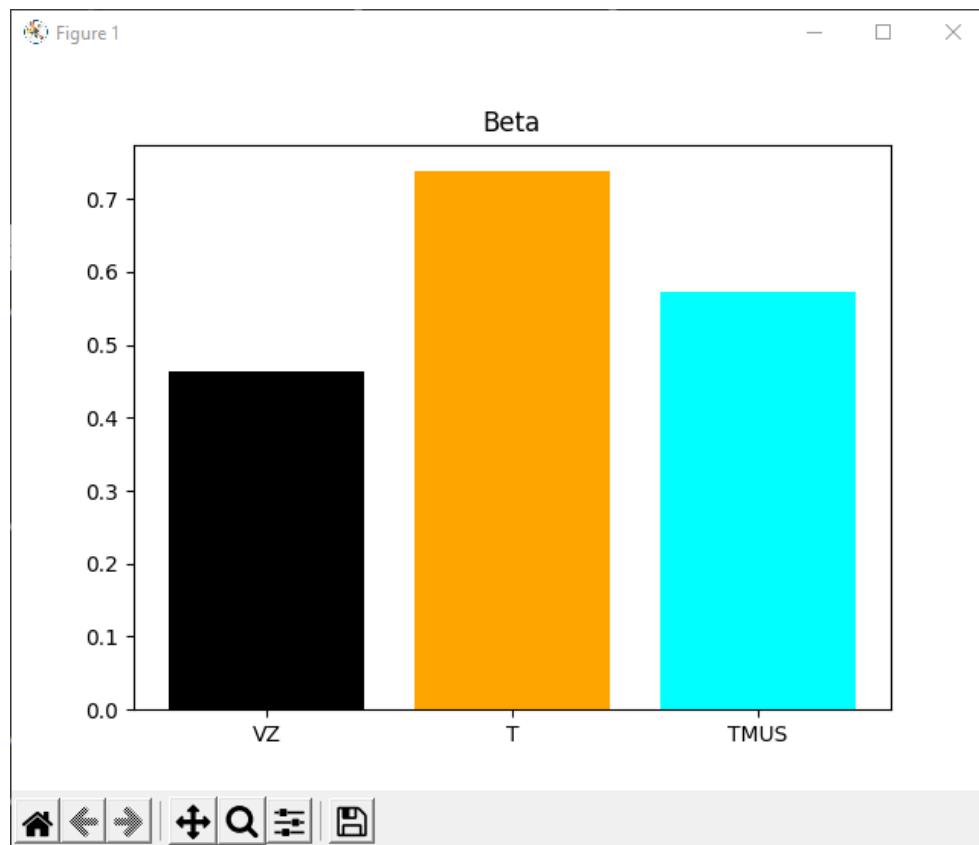


Figure4.5.2-4 Beta Output.

Market capitalization refers to the total value of all a company's shares of stock. It is calculated by multiplying the price of a stock by its total number of outstanding shares.

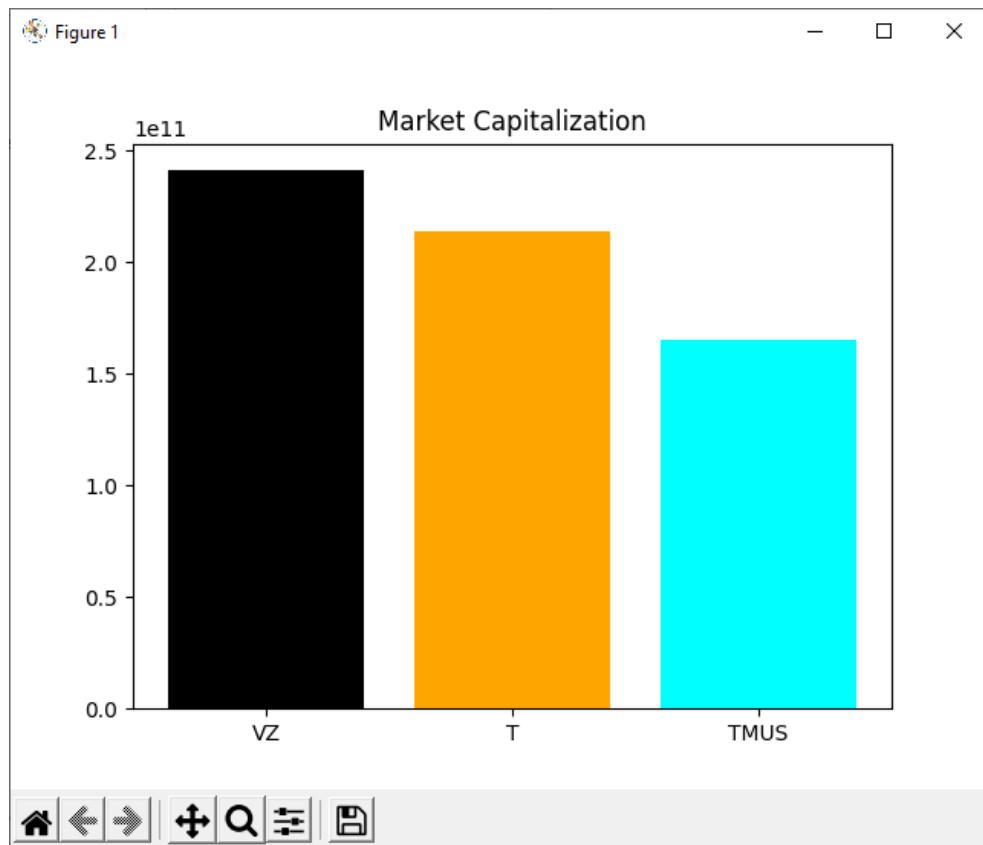


Figure 5.4.2-5 Market Capitalization Output.

A stock dividend is a dividend payment to shareholders that is made in shares rather than as cash.

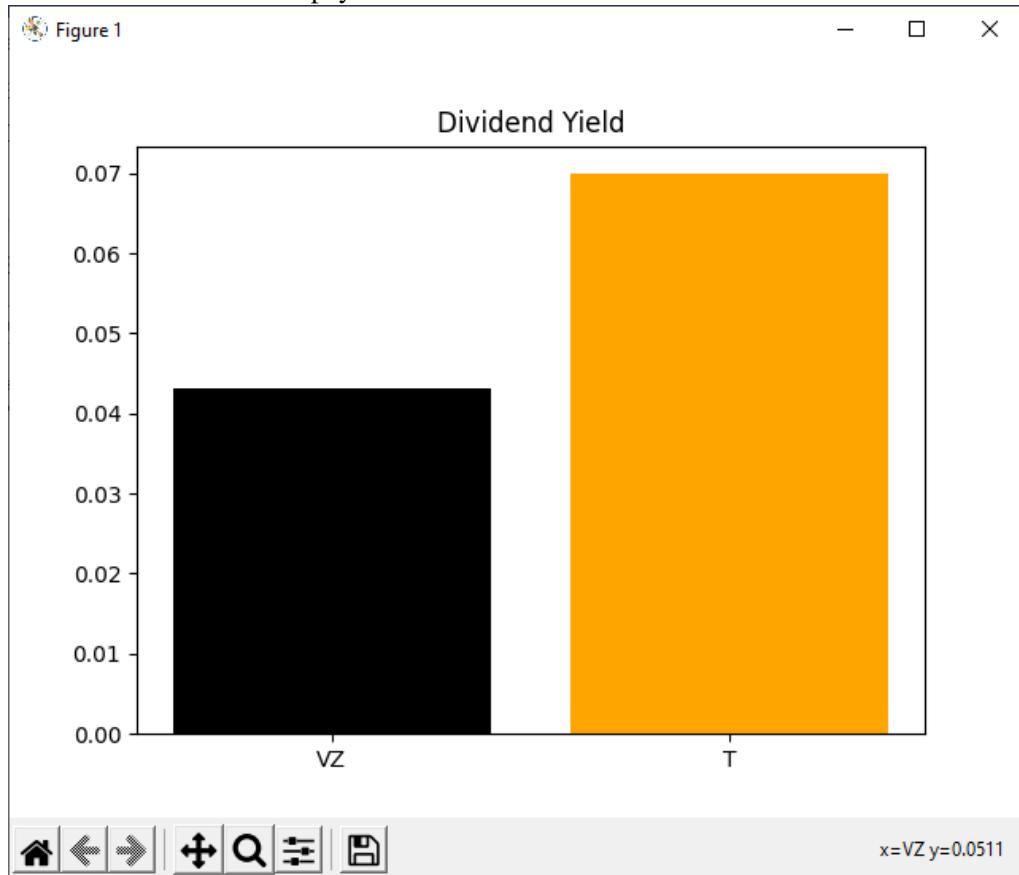


Figure 5.4.2-6 Dividend Yield

5.6 Integrating Matplotlib & Django

Since I am not currently aware of how to embed a matplotlib graph into an html template as I did with Bokeh, I will take time to research the topic and chose the best solution.

After an extensive amount of research, unfortunately I did not find a common, conventional way to embed multiple graphs into a single HTML template. However, since Matplotlib graphs have the capability of being saved as a png file, I will save each plot locally and configure my Django project to handle static files such as images.

5.6.1 Static file configuration

Like the process of configuring my Django project to point to the "template" directory for HTML files, I will also have to tell Django how to handle static files. This process begins by creating two new variables in the project "settings.py" file. This variable serves the same purpose as the "TEMPALATE_DIR" variable in that it tells Django where to look for static files in the main directory.

```

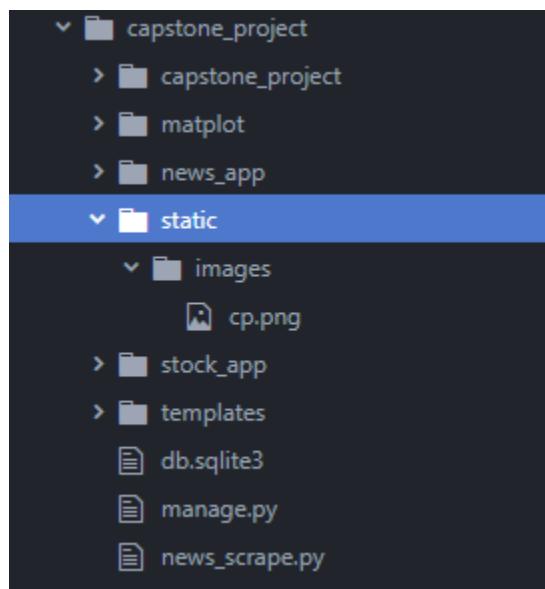
15  BASE_DIR = os.path.join(os.path.dirname(os.path.abspath(__file__)))
16  TEMPLATE_DIR = os.path.join(BASE_DIR, '../templates')
17  STATIC_DIR = os.path.join(BASE_DIR, '../static')

125
126  STATIC_URL = '/static/'
127
128  STATICFILES_DIRS = [
129      STATIC_DIR,
130  ]

```

Figure 4.6.1-1 Configuring Django settings for static files.

Next, I will go create a new directory called “static” in the same place I created the template directory (inside the django_Project folder) then, create another folder inside the static folder to hold images (called “images”).



After creating the new folders, I placed the “cp.png” image inside and will now try displaying the image in the browser by specifying the path in the address bar to my image folder.

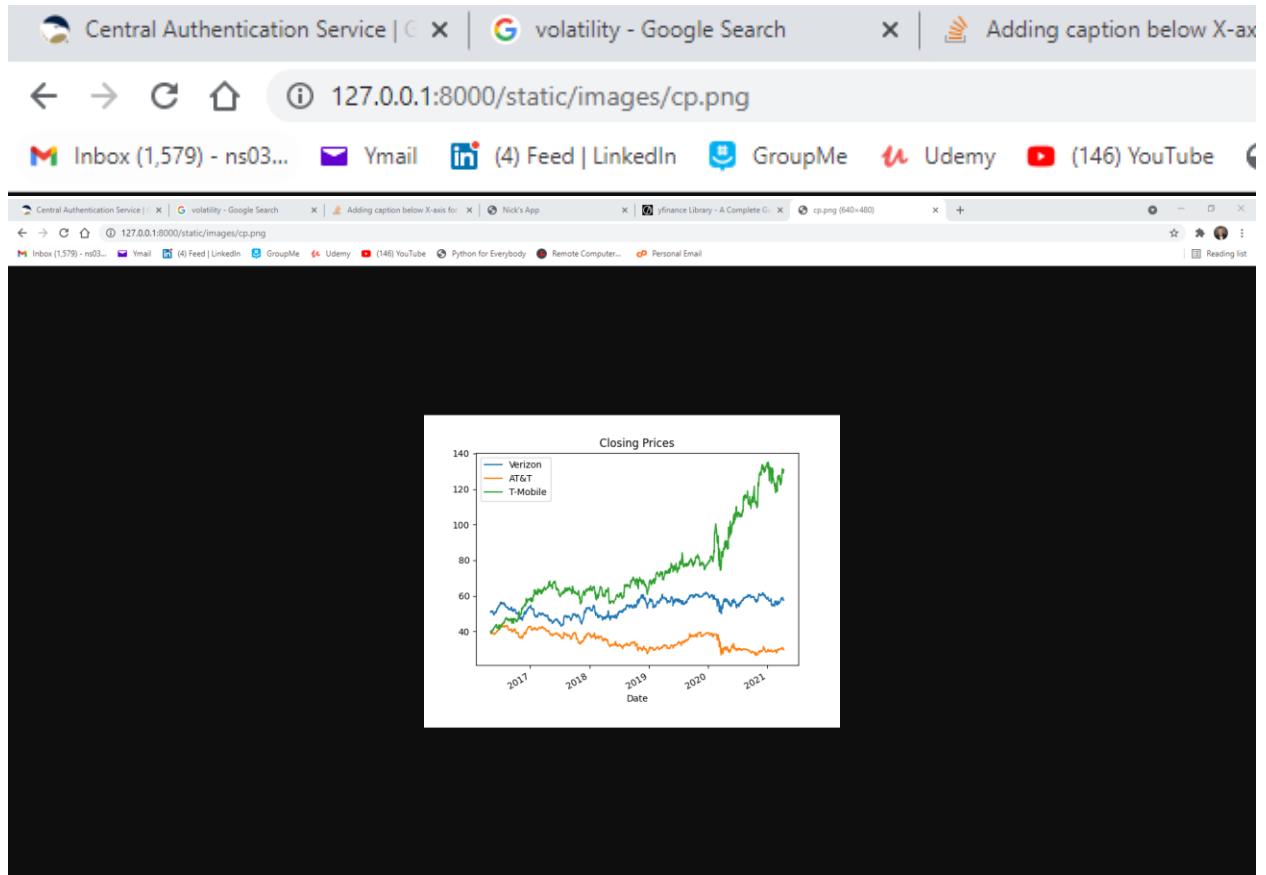


Figure4.6.1-2 Django successfully serving static image.

5.6.2 Creating the template

Now that my Django project knows how to handle static files, I will create a new template called “image_page.html” that will display all my Matplotlib graphs. To embed an image in a Django template, I will be using similar context processors as I have for all other templates except for I will use a special processor at the beginning of the HTML body to handle static files. The next picture encapsulates the HTML body containing the image tags for each chart previously created. I will also make sure each image has been added to my “static/images” directory.

```
1  <!DOCTYPE html>
2  {% load static %}
```

Figure4.6.2-1 Unique context processor for static files.

```
27     <br>
28     
29     
30     
31     
32     
33     
34     
35     
36     <br>
37 </body>
```

Figure4.6.2-2 Creating image element for each graph.

The last part of the process entangles a new function in the “matplotlib” application’s “views.py” file and configuring it in the app’s “urls.py” file as well. All we will need inside the function is a return statement that renders the new template and its appropriate context processors.

```
99
100    def mplot(request):
101        return render(request, 'image_page.html')
```

Figure4.6.2-3 Creating new function in the views to render new template.

```
1  from django.conf.urls import url
2  from matplotlib import views
3
4  urlpatterns = [
5      url(r'graph_page/', views.candlestick, name='candlestick'),
6      url(r'image_page/', views.mplot, name='mplot'),|
```

Figure4.6.2-4 Configuring application URLs to find my new function.

5.7 Final output



Figure 4.7-1 “image_page.html” final Output.

The last steps of this template only include front-end designing and adding paragraphs of text for contextual purposes.

6 Project Deployment

After taking the time to research a few well-known hosting services such as Heroku, azure, etc. I finally came across a free service called “Python Anywhere” ([link](#)). I will explain the service in more detail, why I chose it, and the process I went through to deploy my Django project. However, Python anywhere is not the only service I will need. Before I can use it, I will need to move my project to an online repository where it will then be maintained by a version control system. To do this, I will be using Github.

6.1 Setting up Github

The first step in setting up Github requires installing “Git”, which is version control system that helps keep track of changes in code. Github is an independent company/website that helps manage this control system, host files on their server, and excellent collaboration features. To install Git, I went through the installation process on their official website ([link](#)). Next, I will log into my free Github account and create a new repository to host my project.

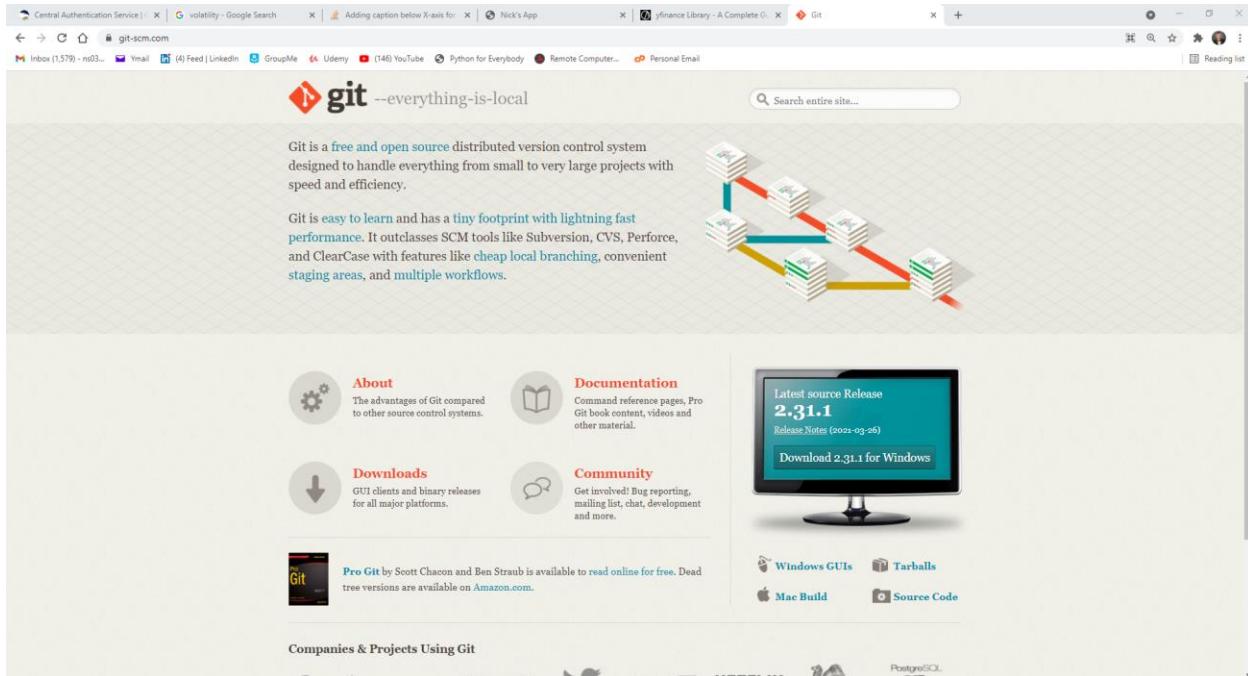


Figure 5.1-1 Downloading Git.

Completing the Git Setup Wizard

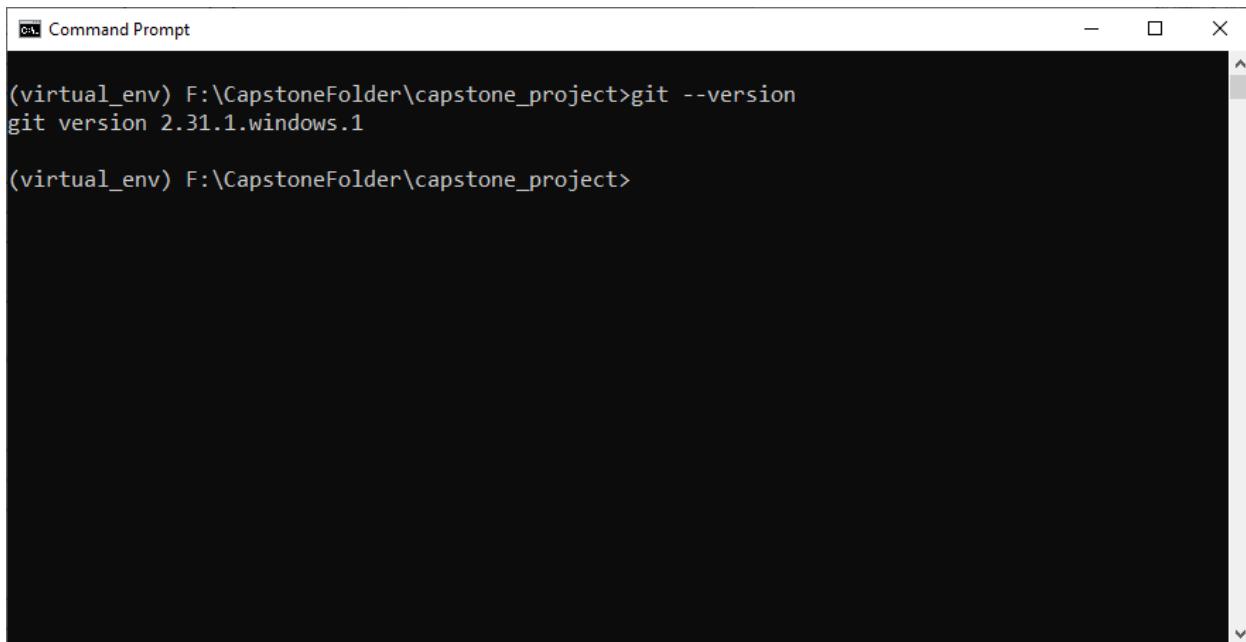
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.



Click Finish to exit Setup.

- Launch Git Bash
- View Release Notes

Figure 5-1.1 Installing Git

A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the following text:

```
(virtual_env) F:\CapstoneFolder\capstone_project>git --version
git version 2.31.1.windows.1
(virtual_env) F:\CapstoneFolder\capstone_project>
```

The window has a standard Windows title bar with minimize, maximize, and close buttons. A vertical scroll bar is visible on the right side of the window.

Figure 5-1.3 Git successfully installed

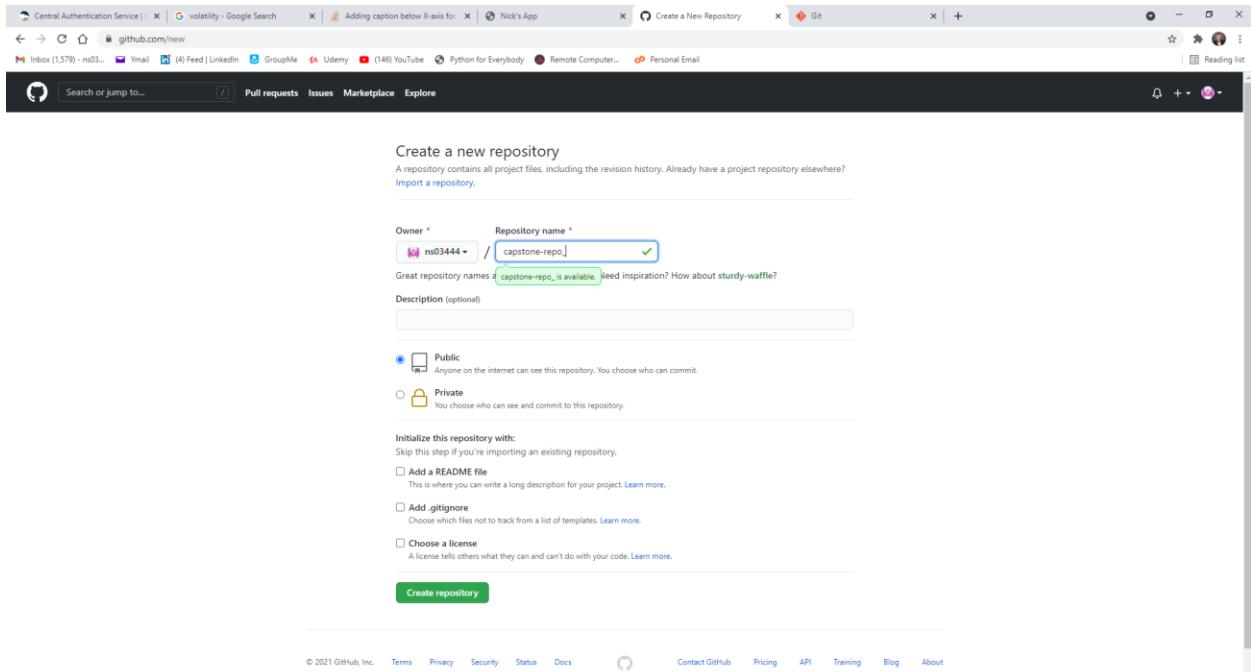


Figure5.1-4 Creating new Github repository.

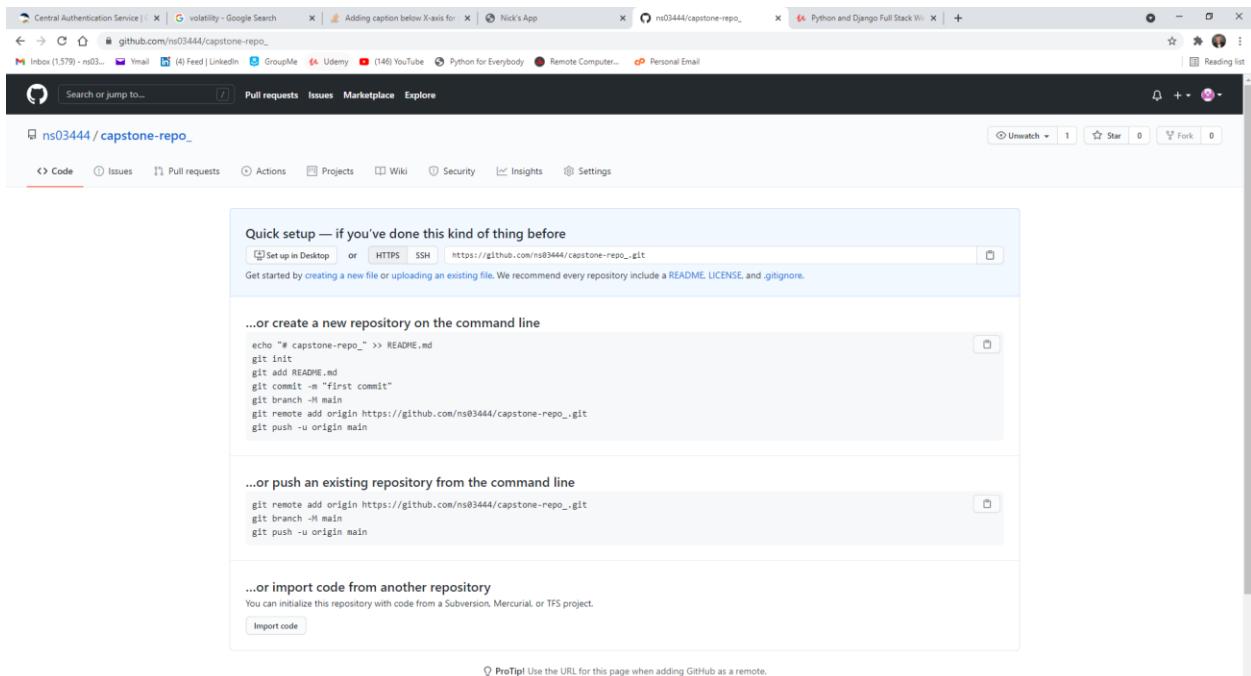


Figure5.1-5 Successfully created Github repository.

After creating my “capstone-repo”, GitHub prompted a new screen that displays a list of commands for uploading, pushing, etc. files to a repository via the command prompt. However, before I can run any of the commands, I will create a new folder, “my_base” inside of the top-level Django project folder to contain the actual git repository. Then, copy the actual Django project and paste it inside this new folder.

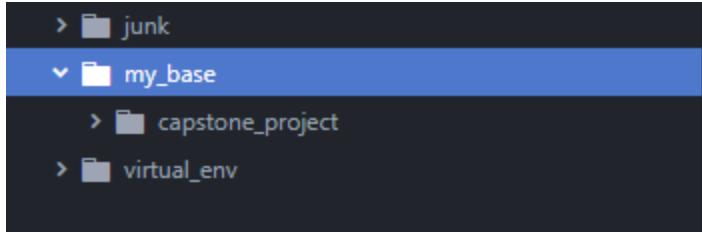
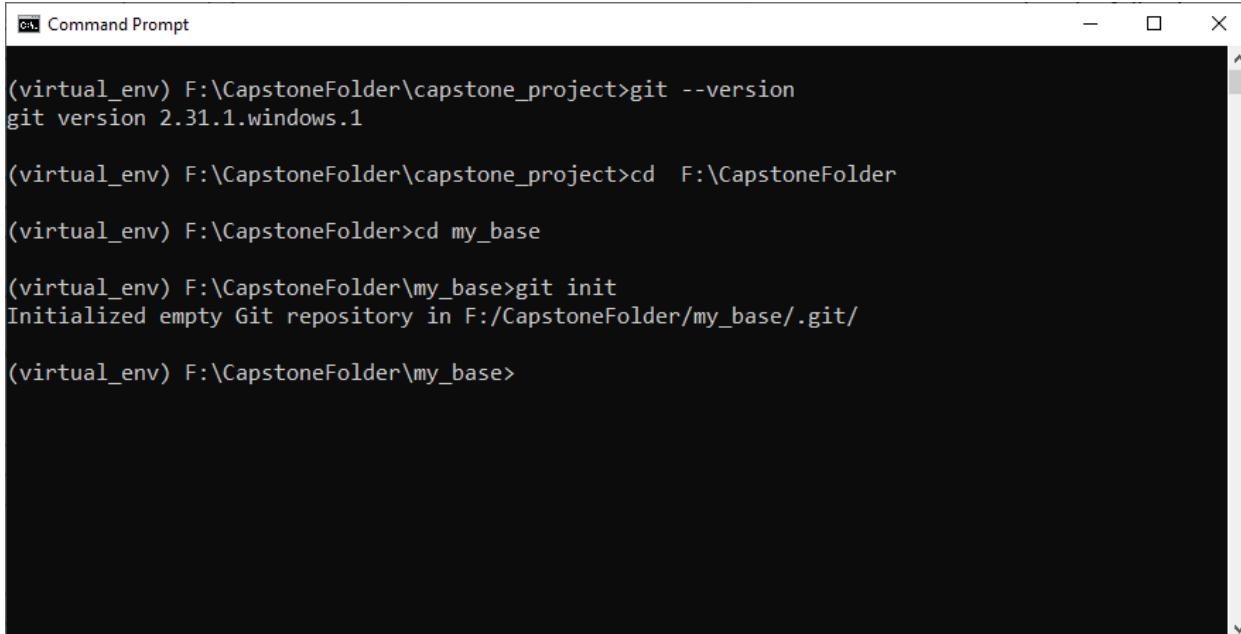
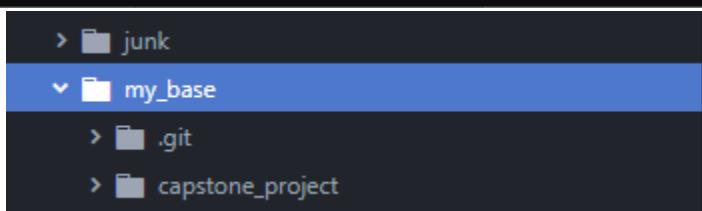
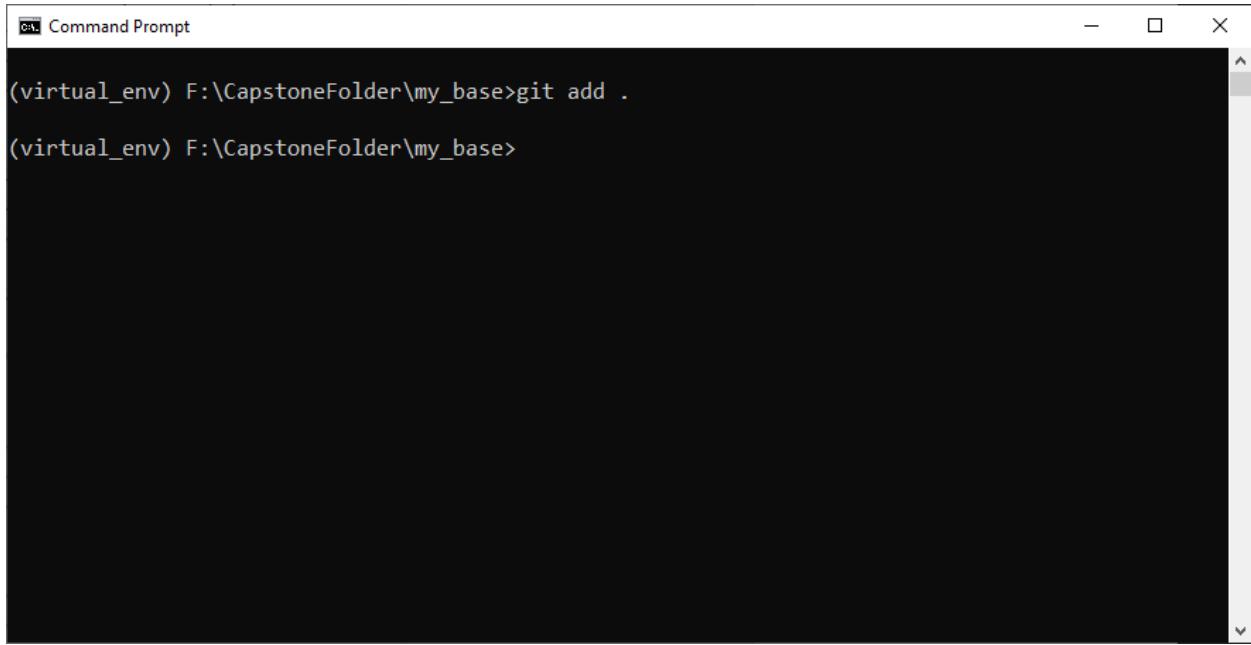




Figure 5.1-6 Git repository initialized.

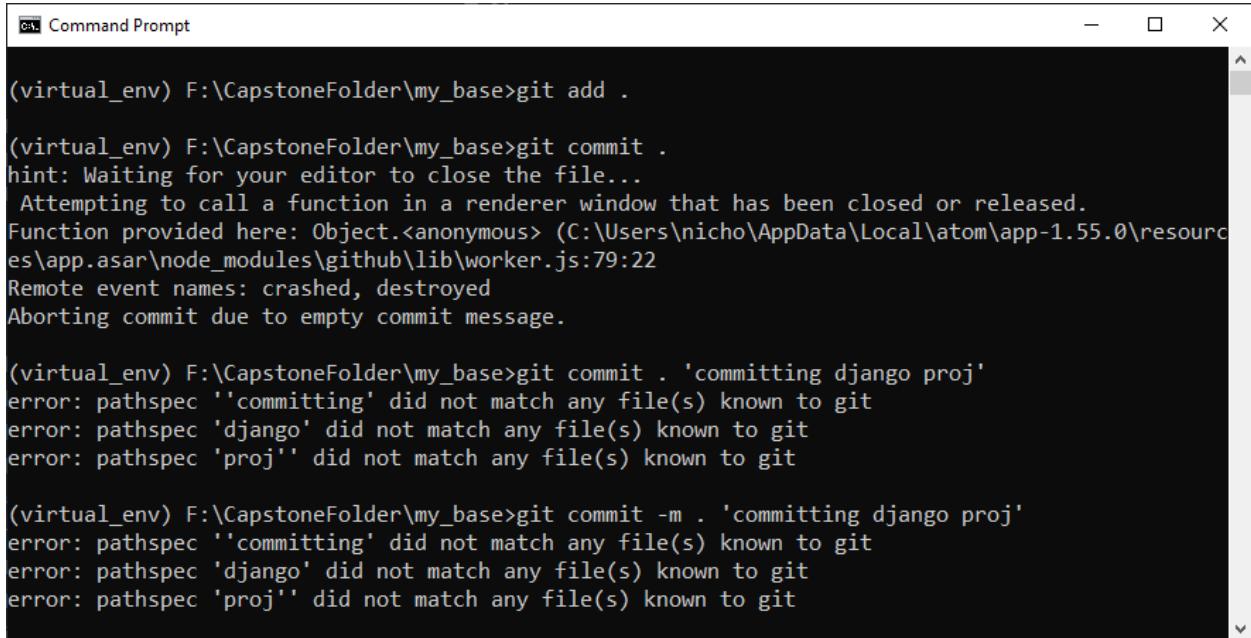
I will run the next command followed by “.” to add *all* of the files in the capstone_project folder to the git repository.



```
cmd Command Prompt
(virtual_env) F:\CapstoneFolder\my_base>git add .
(virtual_env) F:\CapstoneFolder\my_base>
```

Figure 5.1-7 Adding all project files to repository.

Now that everything has been added, it will need to be *committed* to Github.

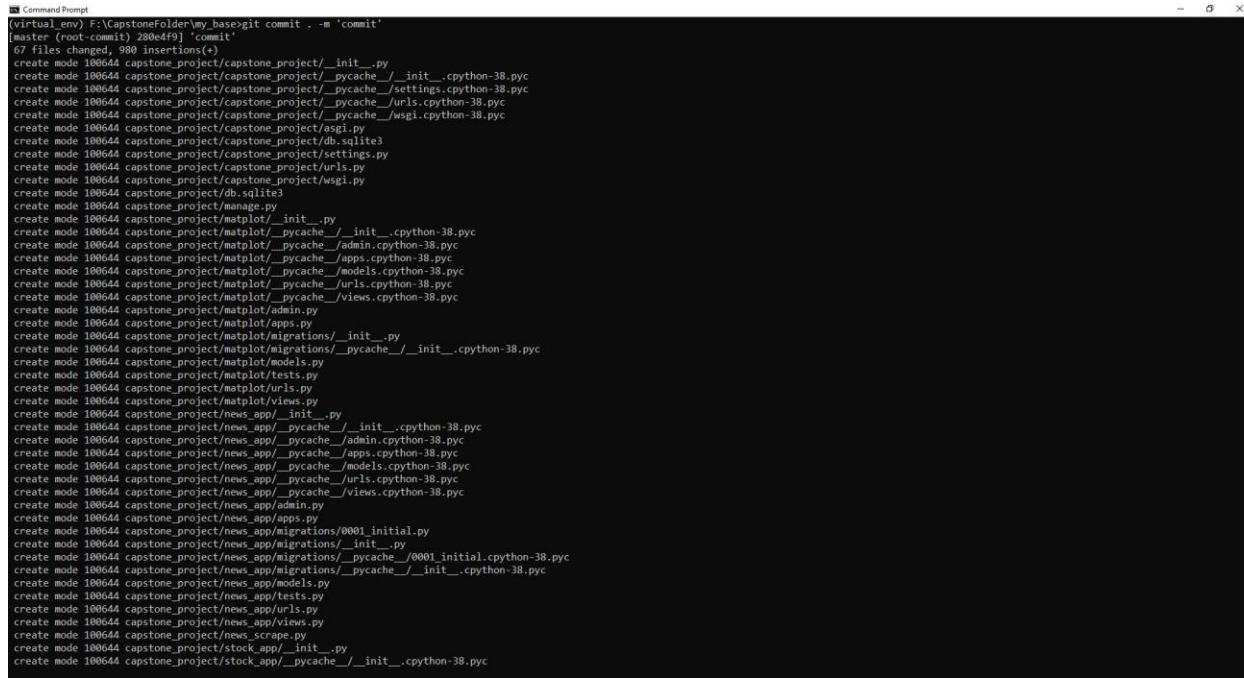


```
cmd Command Prompt
(virtual_env) F:\CapstoneFolder\my_base>git add .
(virtual_env) F:\CapstoneFolder\my_base>git commit .
hint: Waiting for your editor to close the file...
Attempting to call a function in a renderer window that has been closed or released.
Function provided here: Object.<anonymous> (C:\Users\nicho\AppData\Local\atom\app-1.55.0\resources\app.asar\node_modules\github\lib\worker.js:79:22
Remote event names: crashed, destroyed
Aborting commit due to empty commit message.

(virtual_env) F:\CapstoneFolder\my_base>git commit . 'committing django proj'
error: pathspec ''committing' did not match any file(s) known to git
error: pathspec 'django' did not match any file(s) known to git
error: pathspec 'proj'' did not match any file(s) known to git

(virtual_env) F:\CapstoneFolder\my_base>git commit -m . 'committing django proj'
error: pathspec ''committing' did not match any file(s) known to git
error: pathspec 'django' did not match any file(s) known to git
error: pathspec 'proj'' did not match any file(s) known to git
```

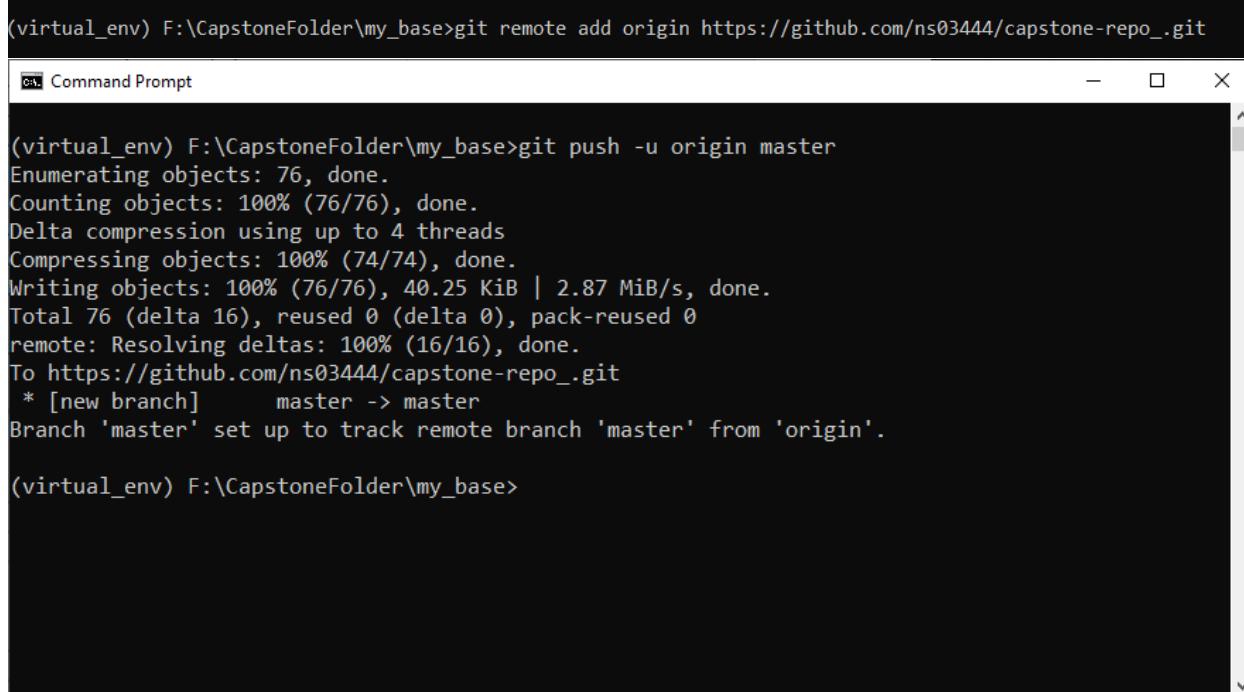
Figure 5.1-8 ERROR - Typo in command



```
(virtual_env) F:\CapstoneFolder\my_base>git commit . -m 'commit'
[main (root)] (67 files, 389 insertions(+), 0 deletions(-))
create mode 100644 capstone_project/capstone_project/__init__.py
create mode 100644 capstone_project/capstone_project/_pycache__init__.cpython-38.pyc
create mode 100644 capstone_project/capstone_project/_pycache__settings.cpython-38.pyc
create mode 100644 capstone_project/capstone_project/_pycache__urls.cpython-38.pyc
create mode 100644 capstone_project/capstone_project/_pycache__wsgi.cpython-38.pyc
create mode 100644 capstone_project/capstone_project/asgi.py
create mode 100644 capstone_project/capstone_project/db.sqlite3
create mode 100644 capstone_project/capstone_project/settings.py
create mode 100644 capstone_project/capstone_project/wsgi.py
create mode 100644 capstone_project/db.sqlite3
create mode 100644 capstone_project/project.py
create mode 100644 capstone_project/matplotlib/__init__.py
create mode 100644 capstone_project/matplotlib/_pycache__init__.cpython-38.pyc
create mode 100644 capstone_project/matplotlib/_pycache__admin.cpython-38.pyc
create mode 100644 capstone_project/matplotlib/_pycache__apps.cpython-38.pyc
create mode 100644 capstone_project/matplotlib/_pycache__models.cpython-38.pyc
create mode 100644 capstone_project/matplotlib/_pycache__urls.cpython-38.pyc
create mode 100644 capstone_project/matplotlib/_pycache__views.cpython-38.pyc
create mode 100644 capstone_project/matplotlib/admin.py
create mode 100644 capstone_project/matplotlib/apps.py
create mode 100644 capstone_project/matplotlib/migrations/__init__.py
create mode 100644 capstone_project/matplotlib/migrations/_pycache__init__.cpython-38.pyc
create mode 100644 capstone_project/matplotlib/models.py
create mode 100644 capstone_project/matplotlib/settings.py
create mode 100644 capstone_project/matplotlib/urls.py
create mode 100644 capstone_project/matplotlib/views.py
create mode 100644 capstone_project/news_app/__init__.py
create mode 100644 capstone_project/news_app/_pycache__init__.cpython-38.pyc
create mode 100644 capstone_project/news_app/_pycache__admin.cpython-38.pyc
create mode 100644 capstone_project/news_app/_pycache__apps.cpython-38.pyc
create mode 100644 capstone_project/news_app/_pycache__models.cpython-38.pyc
create mode 100644 capstone_project/news_app/_pycache__urls.cpython-38.pyc
create mode 100644 capstone_project/news_app/_pycache__views.cpython-38.pyc
create mode 100644 capstone_project/news_app/admin.py
create mode 100644 capstone_project/news_app/apps.py
create mode 100644 capstone_project/news_app/migrations/0001_initial.py
create mode 100644 capstone_project/news_app/migrations/_pycache__0001_initial.cpython-38.pyc
create mode 100644 capstone_project/news_app/migrations/_pycache__init__.cpython-38.pyc
create mode 100644 capstone_project/news_app/models.py
create mode 100644 capstone_project/news_app/tests.py
create mode 100644 capstone_project/news_app/urls.py
create mode 100644 capstone_project/news_app/views.py
create mode 100644 capstone_project/news_scrape.py
create mode 100644 capstone_project/stock_app/__init__.py
create mode 100644 capstone_project/stock_app/_pycache__init__.cpython-38.pyc
```

Figure 5.1-9 Typo fixed; files successfully committed.

Now I will be pushing the *git* repository to my actual *Github* repository (stored on their website) by including the URL to the repository in the next command. Then, use the “push” command to make the physical changes on Github.



```
(virtual_env) F:\CapstoneFolder\my_base>git remote add origin https://github.com/ns03444/capstone-repo_.git
(virtual_env) F:\CapstoneFolder\my_base>git push -u origin master
Enumerating objects: 76, done.
Counting objects: 100% (76/76), done.
Delta compression using up to 4 threads
Compressing objects: 100% (74/74), done.
Writing objects: 100% (76/76), 40.25 KiB | 2.87 MiB/s, done.
Total 76 (delta 16), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (16/16), done.
To https://github.com/ns03444/capstone-repo_.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

(virtual_env) F:\CapstoneFolder\my_base>
```

Figure 5.1-10 Files completely pushed to Github Repository.

Now, when I go back to Github's website to view my repository, my Django project and all of the files included can be viewed on their website.

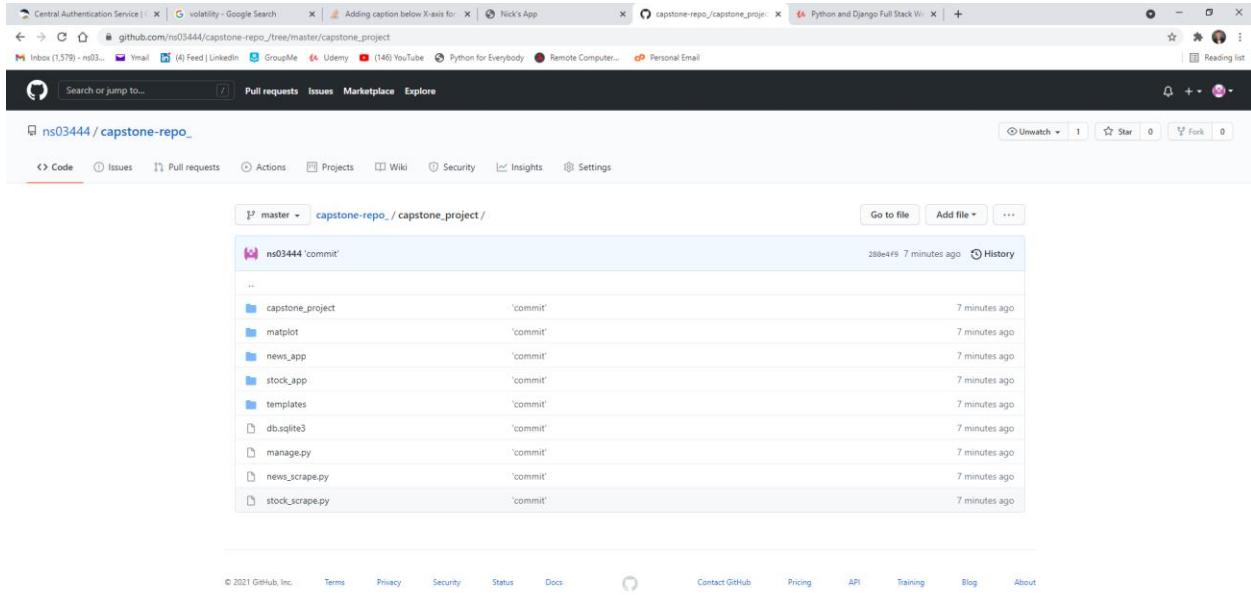


Figure 5.1-11 Django project moved to Github Repository successfully.

6.2 Github to Python Anywhere

I will begin by creating a free account with Python Anywhere, which is an *online, integrated development environment and web hosting service*. Then, I will open a new bash console which functions like the command prompt in that it allows interaction with bash files, the Python interpreter, etc. and, then create a new virtual environment (similar to “venv”, used locally) to host my project. After the environment has been created, I will install Django and all other dependencies that were installed to the local virtual environment.

Host, run, and code Python in the cloud!

Get started for free. Our basic plan gives you access to machines with a [full Python environment](#) already installed. You can develop and host your website or any other code directly from your browser without having to install software or manage your own server.

Need more power? Upgraded plans start at \$5/month.

[Start running Python online in less than a minute!](#)

[Watch our one-minute video](#)

Not convinced? [Read what our users are saying!](#)

Start hosting quickly
Just write your application. No need to configure or maintain a web server — everything is set up and ready to go.
[More](#)

Develop anywhere
Take your development environment with you! If you have a browser and an Internet connection, you've got everything you need.
[More](#)

Teach and learn
PythonAnywhere is a fully-fledged Python environment, ready to go, for students and teachers — concentrate on teaching, not on installation hassles.

Amazing support
Need help with PythonAnywhere? If you get in touch, you can talk directly with the development team. Help for developers, from developers.
[More](#)

Figure 5.2-1 Python anywhere homepage.

Welcome, ns03444

CPU Usage: 10% used ~ 208.06s of 2,000s. Resets in 2 hours, 22 minutes. [More info](#)

File storage: 38% full ~ 392.8 MB of your 1.0 GB quota. [More info](#)

[Upgrade Account](#)

Recent Consoles
You have no recent consoles.
[View all](#)

Recent Files
You have no recently edited files.
[+ Open another file](#) [Browse files](#)

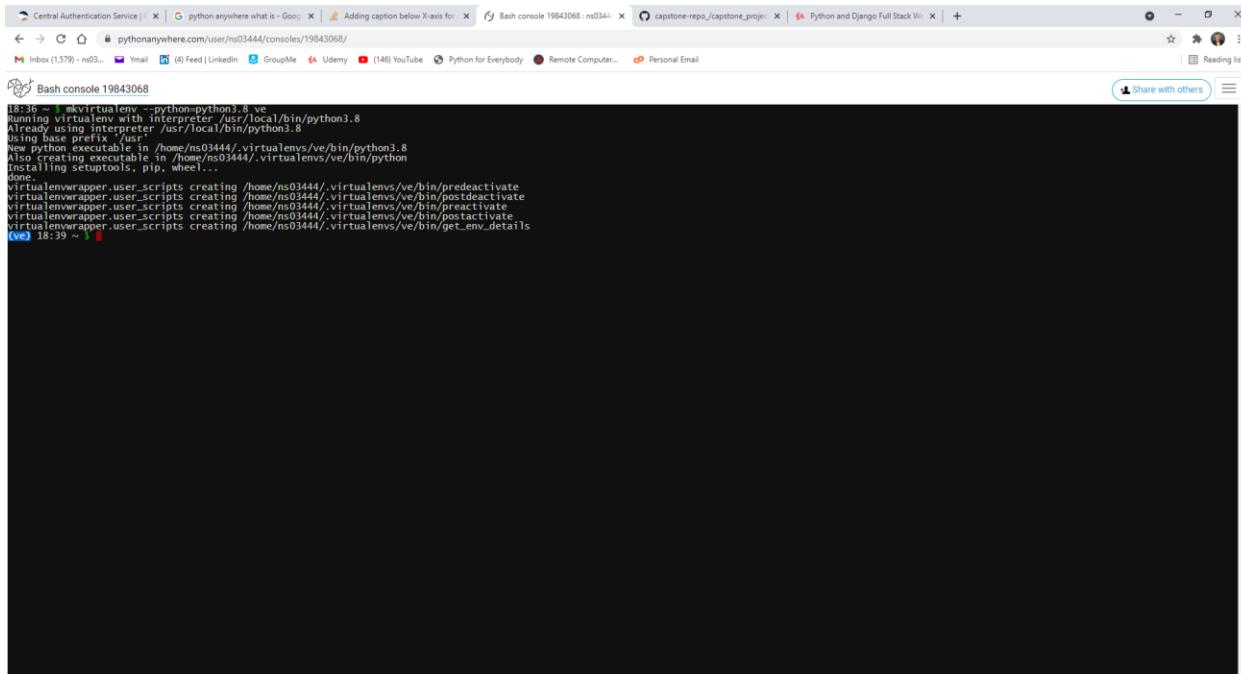
Recent Notebooks
You have no recent notebooks.
[+ Add new \(Python 3.6\)](#) [Browse all files](#)

All Web apps
[Open Web tab](#)

New console:
[\\$ Bash](#) [>>> Python](#) [More...](#)

Copyright © 2011-2021 PythonAnywhere LLP — [Terms](#) — [Privacy & Cookies](#)

Figure 5.2-2 Account dashboard.

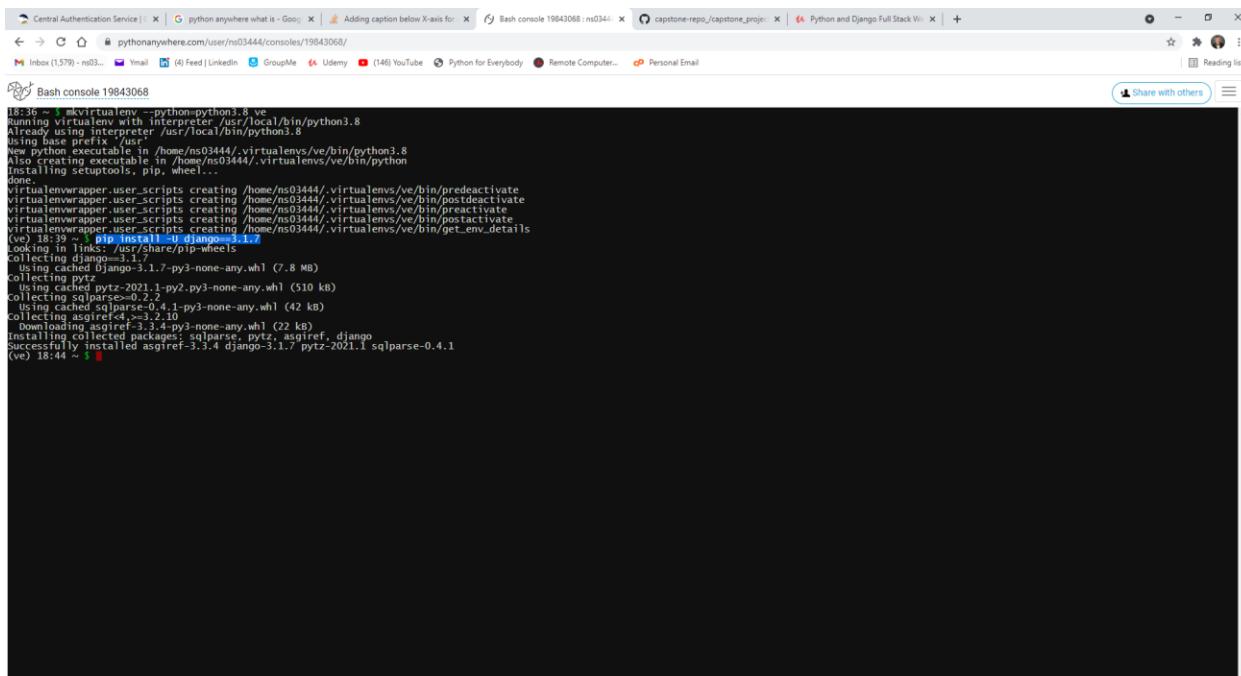


```

18:36 ~ $ mkvirtualenv --python=python3.8 ve
Running virtualenv with interpreter /usr/local/bin/python3.8
Already using interpreter /usr/local/bin/python3.8
Using base prefix '/usr'
New python executable in /home/ns03444/.virtualenvs/ve/bin/python3.8
Also creating executable in /home/ns03444/.virtualenvs/ve/bin/python
Installing setuptools, pip, wheel...
done.
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/predeactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/postdeactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/preactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/postactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/get_env_details
(v) 18:39 ~ $ 

```

Figure 5.2-3 Creating a new virtual environment.



```

18:36 ~ $ mkvirtualenv --python=python3.8 ve
Running virtualenv with interpreter /usr/local/bin/python3.8
Already using interpreter /usr/local/bin/python3.8
Using base prefix '/usr'
New python executable in /home/ns03444/.virtualenvs/ve/bin/python3.8
Also creating executable in /home/ns03444/.virtualenvs/ve/bin/python
Installing setuptools, pip, wheel...
done.
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/predeactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/postdeactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/preactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/postactivate
virtualenvwrapper.user_scripts creating /home/ns03444/.virtualenvs/ve/bin/get_env_details
(v) 18:39 ~ $ pip install -U django==3.1.7
Looking in links: /usr/share/pip-wheels
Collecting django==3.1.7
  Using cached Django-3.1.7-py3-none-any.whl (7.8 MB)
Collecting pytz
  Using cached pytz-2021.1-py3.py3-none-any.whl (510 kB)
Collecting sqlparse==0.2.2
  Using cached sqlparse==0.4.1-py3-none-any.whl (42 kB)
Collecting asgiref<4.0.0,>=3.3.4
  Using cached asgiref-3.3.4-py3-none-any.whl (22 kB)
Installing collected packages: sqlparse, pytz, asgiref, django
Successfully installed asgiref-3.3.4 django-3.1.7 pytz-2021.1 sqlparse-0.4.1
(v) 18:44 ~ $ 

```

Figure 5.2-4 Installing Django & other dependencies.

6.2.1 Connecting to the Repository

Now that Django and all other dependencies have been installed, it is time to get an actual copy of our git repository onto the hosting service. To do this, all I will need is the “clone” command and the address to my repository on my Github account.

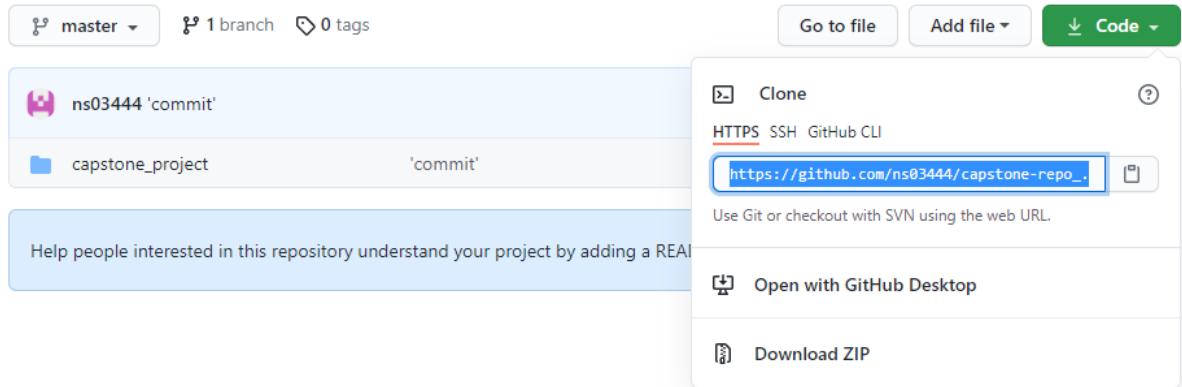


Figure 5.2.1-1 Copying address to Github repository.

```
(ve) 18:52 ~ $ git clone https://github.com/ns03444/capstone-repo_.git
Cloning into 'capstone-repo_...'...
remote: Enumerating objects: 76, done.
remote: Counting objects: 100% (76/76), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 76 (delta 16), reused 76 (delta 16), pack-reused 0
Unpacking objects: 100% (76/76), done.
Checking connectivity... done.
(ve) 18:53 ~ $
```

Figure 5.2.1-2 Cloning Github repository.

```
(ve) 19:03 ~/capstone-repo_ (master)$ ls
capstone_project
(ve) 19:03 ~/capstone-repo_ (master)$ cd capstone_project
(ve) 19:03 ~/capstone-repo_/capstone_project (master)$ ls
capstone_project db.sqlite3 manage.py matplotlib news_app news_scrape.py stock_app stock_scrape.py templates
(ve) 19:03 ~/capstone-repo_/capstone_project (master)$
```

Figure 5.2.1-3 Repository successfully cloned (project files listed).

6.2.2 Creating the web app

Now that my repository has been cloned to my virtual environment on Python Anywhere, I will return to the dashboard and use the web tab to create a *new web application*.

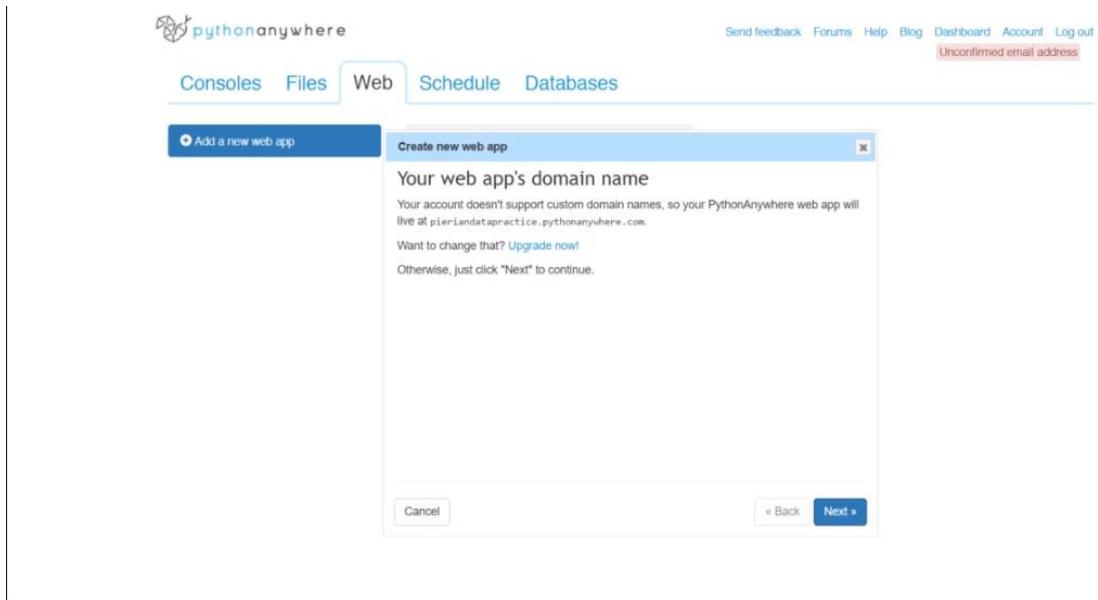


Figure 5.2.2-1 Python Anywhere Web tab.

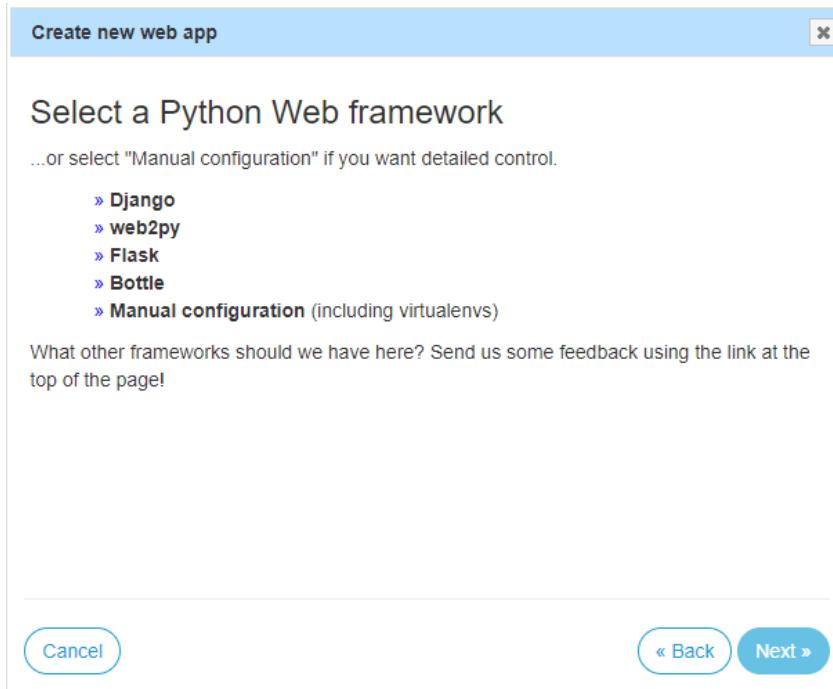


Figure 5.2.2-2 Creating a new web app with Python Anywhere.



Figure 5.2.2-3 Creating a new web app with Python Anywhere – cont'd.

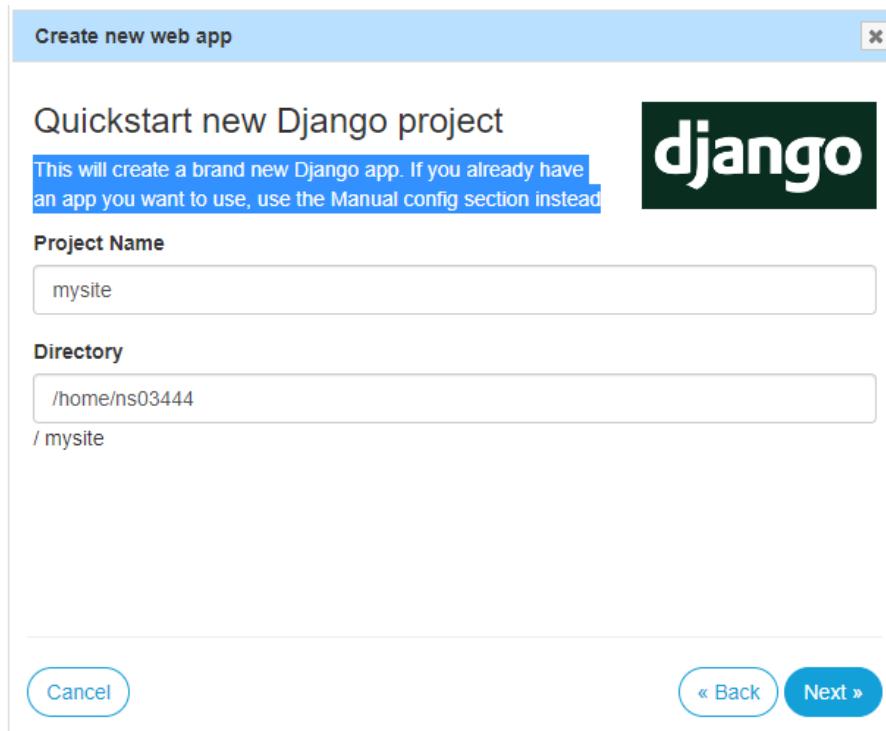


Figure 5.2.2-4 Creating a new web app with Python Anywhere – cont'd.

Since we already have an existing Django project, I will be making the application configurations manually.

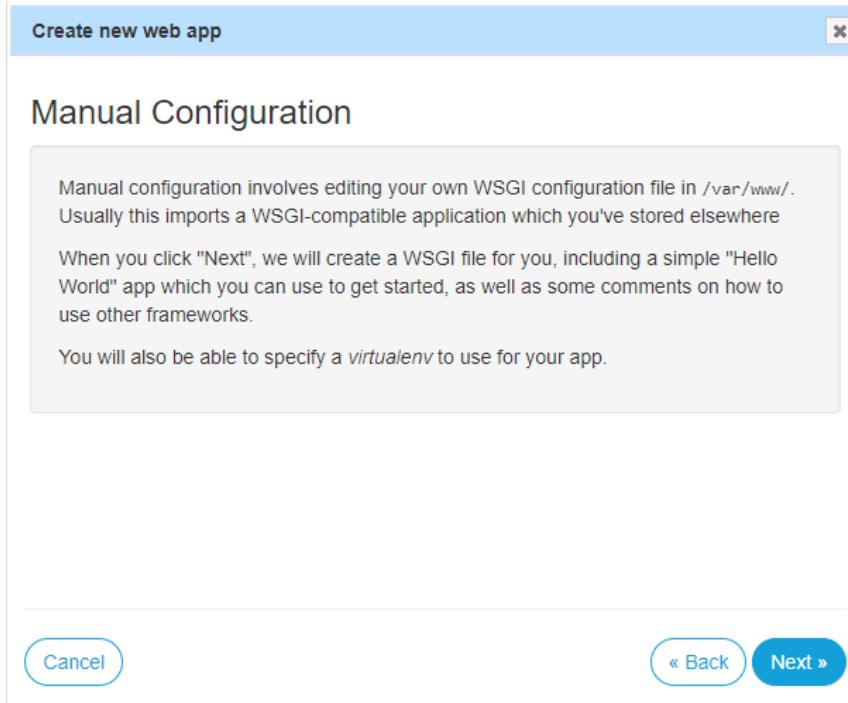


Figure 5.2.2-5 Using manual configuration for existing Django projects.

The image above explains the process for manually configuring the file. By default, all free accounts are assigned *one free domain name that starts with the account's username, followed by ".pythonanywhere.com"*. My new domain "ns03444.pythonanywhere.com" was created after clicking next in the dialogue box shown above.

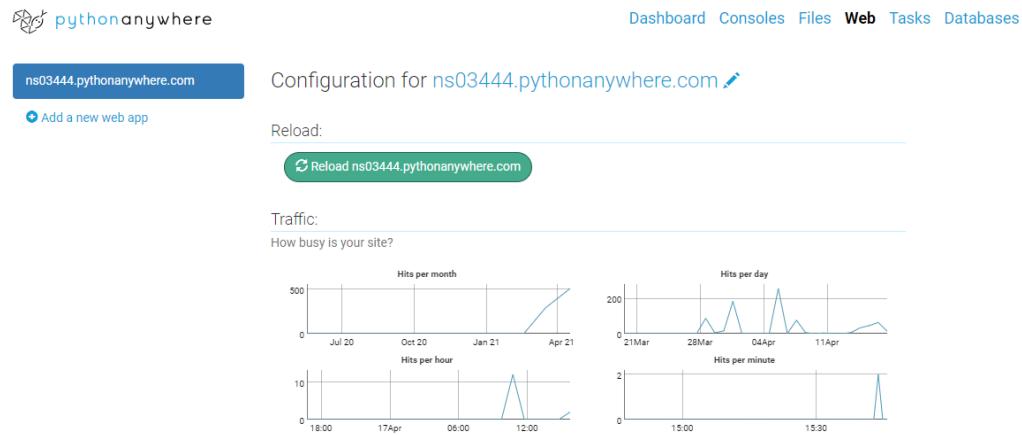


Figure 5.2.2-6 Python Anywhere web app created.

The image above displays that my new web app has been created and now I will continue following the manual configuration instructions by following the hyperlink to my domain.

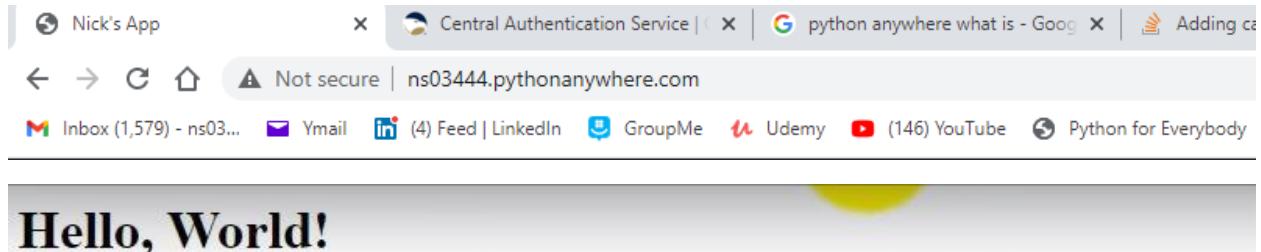


Figure5.2.2-7 Webpage returned when following link to my domain.

By following the “[web app setup](#)” page hyperlink displayed above, I am returned back to the web tab in Python Anywhere. Here, I will specify directory routes and configure the “wsgi.py” file for deployment. First, I will define the path to my virtual environment.

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to [Reload your web app](#) to activate it; NB - will do nothing if the virtualenv does not exist.

[/home/ns03444/.virtualenvs/my_ve](#)

[Start a console in this virtualenv](#)

Figure5.2.2-8 Specifying which virtual environment to use for web app.

Next, I will configure the file path to my *source code*, which is simply the name of my Django project (capstone_project).

Code:

What your site is running.

Source code: [/home/ns03444/capstone-repo/capstone_project](#) [Go to directory](#)

Working directory: [/home/ns03444/](#) [Go to directory](#)

WSGI configuration file: [/var/www/ns03444_pythonanywhere_com_wsgi.py](#)

Python version: 3.8 [Edit](#)

Figure5.2.2-9 Identifying source code for web app.

Next, I will configure the file path to my static files, this way the images embedded in the “image_page.html” will be displayed.

Static files:

Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete	
/static/admin	/home/ns03444/virtualenvs/my_ve/lib/python3.8/site-packages/django/contrib/admin/static/admin		
/static/	/home/ns03444/capstone-repo/capstone_project/static		
<i>Enter URL</i>	<i>Enter path</i>		

Figure 5.2.2-10 Mapping to static files.

Last, I will be opening the web app's "wsgi.py" file to make some edits/additions. Luckily, Python Anywhere provides a text editor as well as a python shell to run code, right from the browser.

WSGI configuration file: [/var/www/ns03444_pythonanywhere_com_wsgi.py](http://var/www/ns03444_pythonanywhere_com_wsgi.py)

```
40+ # To use your own django app use code like this:
41 import os
42 import sys
43 #
44 ## assuming your django settings file is at '/home/ns03444/mysite/mysite/settings.py'
45 ## and your manage.py is is at '/home/ns03444/mysite/manage.py'
46 path = '/home/ns03444/capstone-repo/capstone_project'
47+ if path not in sys.path:
48     sys.path.append(path)
49 os.chdir(path)
50 os.environ.setdefault("DJANGO_SETTINGS_MODULE", "capstone_project.settings")
51 import django
52 django.setup()
53 #os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'
54 #
55+ ## then:
56 from django.core.wsgi import get_wsgi_application
57 application = get_wsgi_application()
58
```

Figure 5.2.2-11 Configuring "wsgi.py" file for Django deployment.

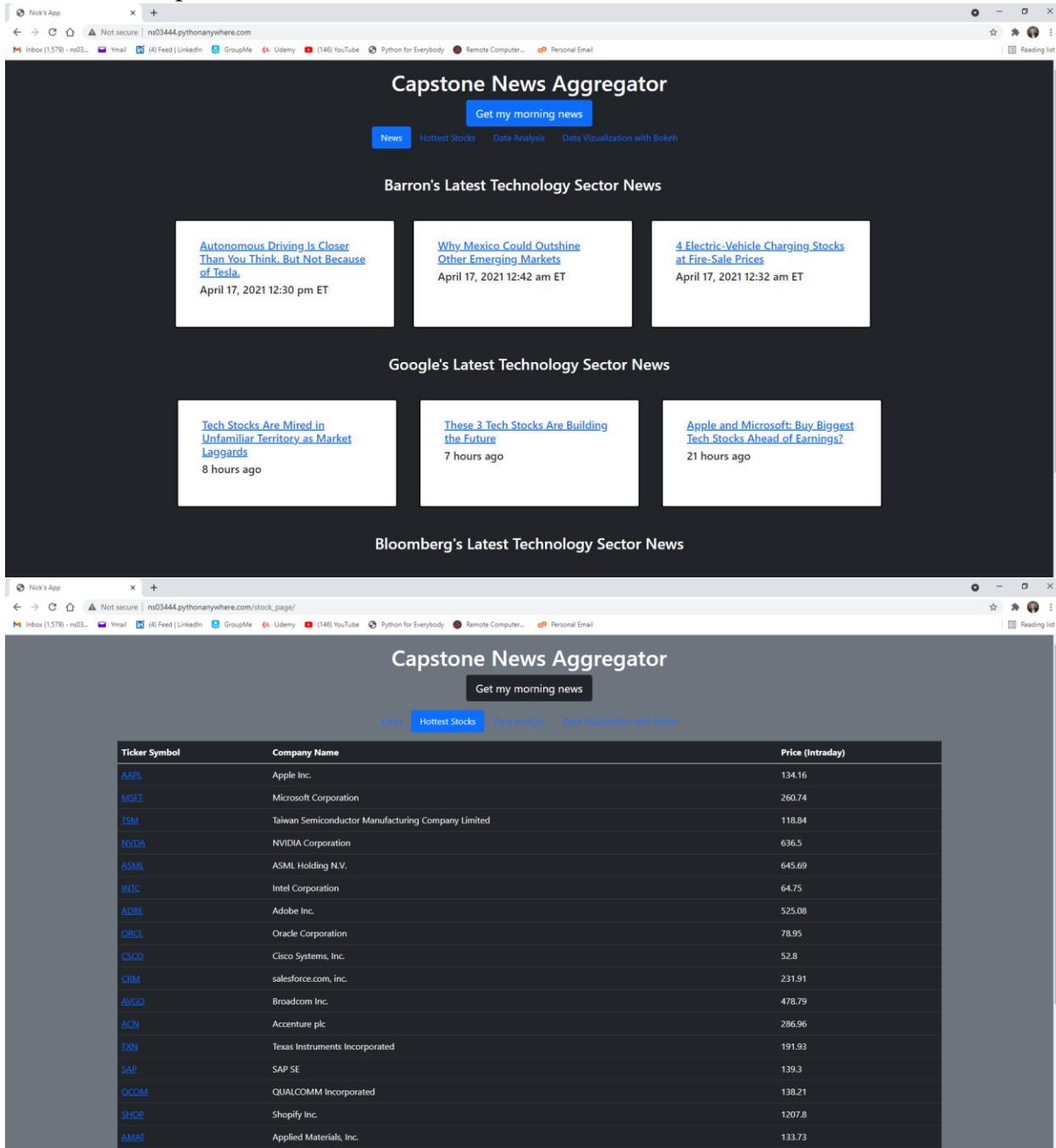
The code shown above represents the changes made to the "wsgi.py" file. Like the code used in some local files, this will allow Python Anywhere to serve the Django application when the server is requested, as well as, allow hosts to the website. The last thing that must be done is adding my new domain name to "ALLOWED HOSTS" in the project settings file.

```
26 DEBUG =False
27
28 ALLOWED_HOSTS = ['ns03444.pythonanywhere.com']
29
```

Figure 5.2.2-12 Configuring project settings for new host name.

After making the additions in the project settings; setting debug mode to false since I am no longer in the development phase of my application; and reloading the web application, my website is now live!

6.3 Final output



The screenshot displays the final output of the Capstone News Aggregator application. The interface is a news aggregator with a dark theme. At the top, there is a navigation bar with links to 'News', 'Hottest Stocks', 'Data Analysis', and 'Data Visualization with Bokeh'. A 'Get my morning news' button is also present.

The main content area is divided into three sections:

- Barron's Latest Technology Sector News:** Contains three news cards:
 - [Autonomous Driving Is Closer Than You Think, But Not Because of Tesla.](#) (April 17, 2021 12:30 pm ET)
 - [Why Mexico Could Outshine Other Emerging Markets](#) (April 17, 2021 12:42 am ET)
 - [4 Electric-Vehicle Charging Stocks at Fire-Sale Prices](#) (April 17, 2021 12:32 am ET)
- Google's Latest Technology Sector News:** Contains three news cards:
 - [Tech Stocks Are Mired in Unfamiliar Territory as Market Laggards](#) (8 hours ago)
 - [These 3 Tech Stocks Are Building the Future](#) (7 hours ago)
 - [Apple and Microsoft: Buy Biggest Tech Stocks Ahead of Earnings?](#) (21 hours ago)
- Bloomberg's Latest Technology Sector News:** Contains three news cards:
 - [Tech Stocks Are Mired in Unfamiliar Territory as Market Laggards](#) (8 hours ago)
 - [These 3 Tech Stocks Are Building the Future](#) (7 hours ago)
 - [Apple and Microsoft: Buy Biggest Tech Stocks Ahead of Earnings?](#) (21 hours ago)

Below these sections is a table of stock prices:

Ticker Symbol	Company Name	Price (intraday)
AAPL	Apple Inc.	134.16
MSFT	Microsoft Corporation	260.74
TSM	Taiwan Semiconductor Manufacturing Company Limited	118.84
NVDA	NVIDIA Corporation	636.5
ASML	ASML Holding N.V.	645.69
INTC	Intel Corporation	64.75
ADBE	Adobe Inc.	525.08
ORCL	Oracle Corporation	78.95
CSCO	Cisco Systems, Inc.	52.8
CRM	salesforce.com, inc.	231.91
AVGO	Broadcom Inc.	478.79
ACN	Accenture plc	286.96
TXN	Texas Instruments Incorporated	191.93
SAP	SAP SE	139.3
QCOM	QUALCOMM Incorporated	138.21
SHOP	Shopify Inc.	1207.8
AMAT	Applied Materials, Inc.	133.73

Figure 5.3-1 Final Project Output.

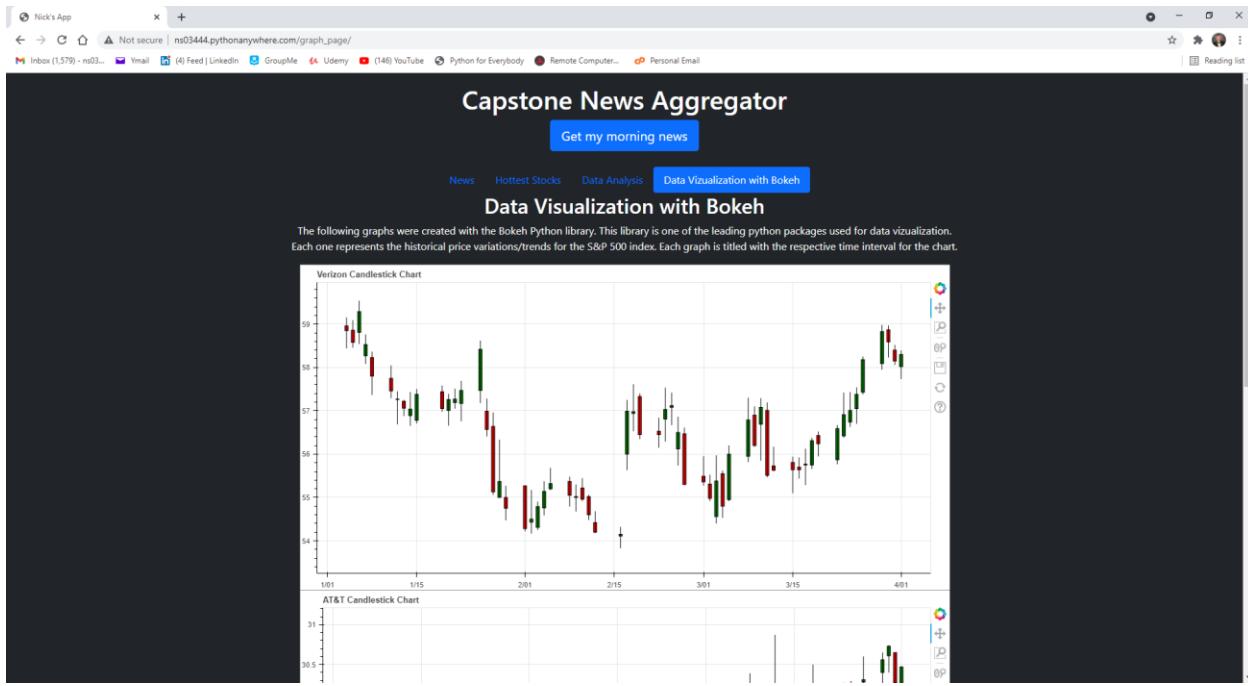
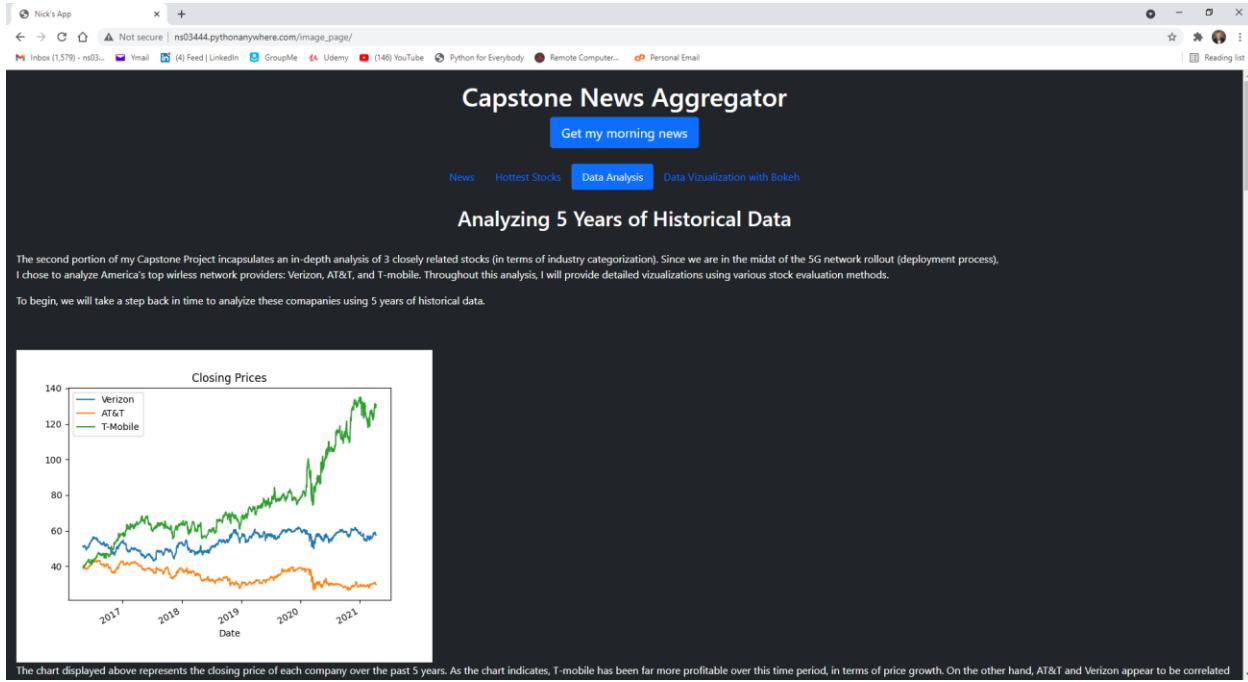


Figure 5.3-1 Final Project Output.

6.4 Scheduling Tasks

One of the final, yet most important tasks I have yet to accomplish is, creating a timer/scheduler to dynamically updated the content on the news and stock table pages. Fortunately, the hosting service I chose to use, “Python Anywhere” has task scheduling capabilities. However, at this point, the “news_scrape.py” and “stock_scrape.py” scripts are only being used to populate the database tables when they are running. Therefore, I will need to add a new function in both files to delete the old objects before repopulating them. Then, I will schedule both python scripts to run as often as possible using Python Anywhere.

6.4.1 Modifying the tasks

I began the process by logging into my python anywhere account and navigating the file tab. There, I can open and edit the news/stock scraping files. I will add a function to the top of both files that deletes all objects in the tables, then use the same functions I previously used to repopulate the database.

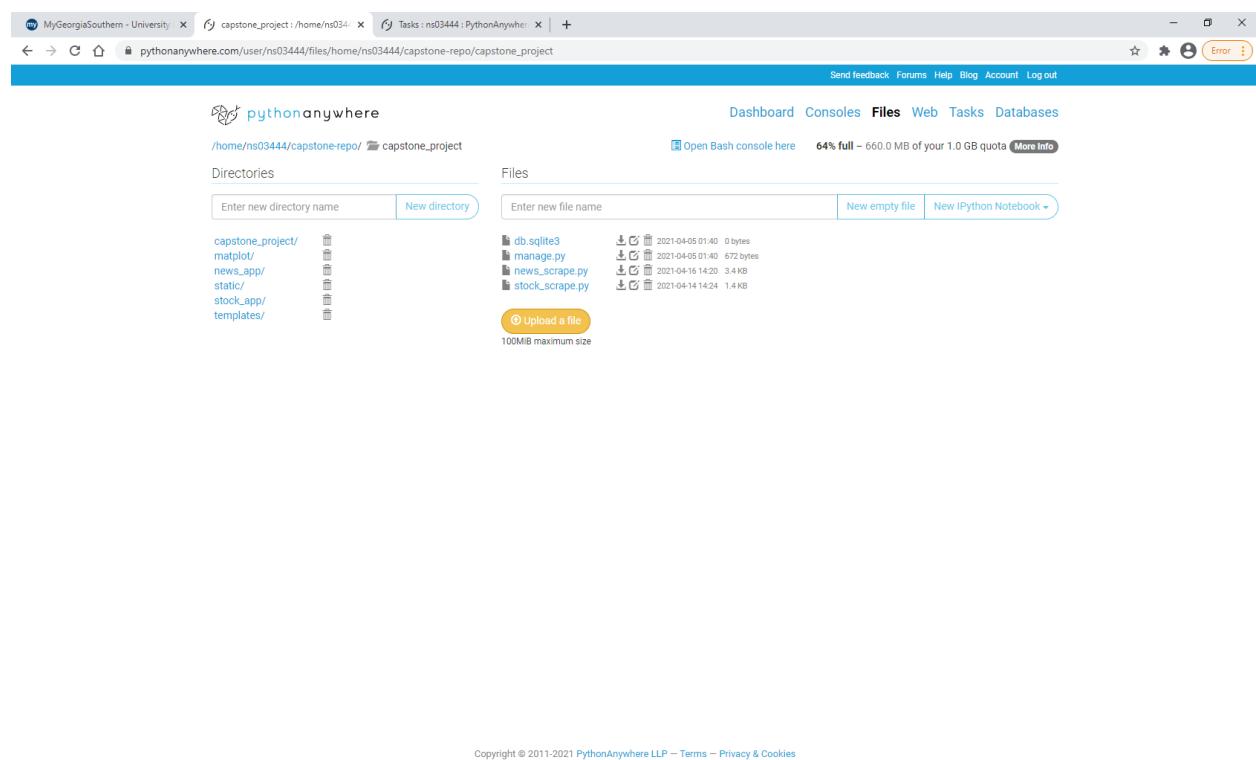


Figure 5.4.1-1 Python Anywhere file tab – My project directory.

```
9
10 def news_delete():
11     deleter1 = Google.objects.all().delete()
12     deleter2 = Bloomberg.objects.all().delete()
13     deleter3 = Barron.objects.all().delete()
14     print('deleted.')
15 news_delete()
16
17 def news_scrape():
18
```

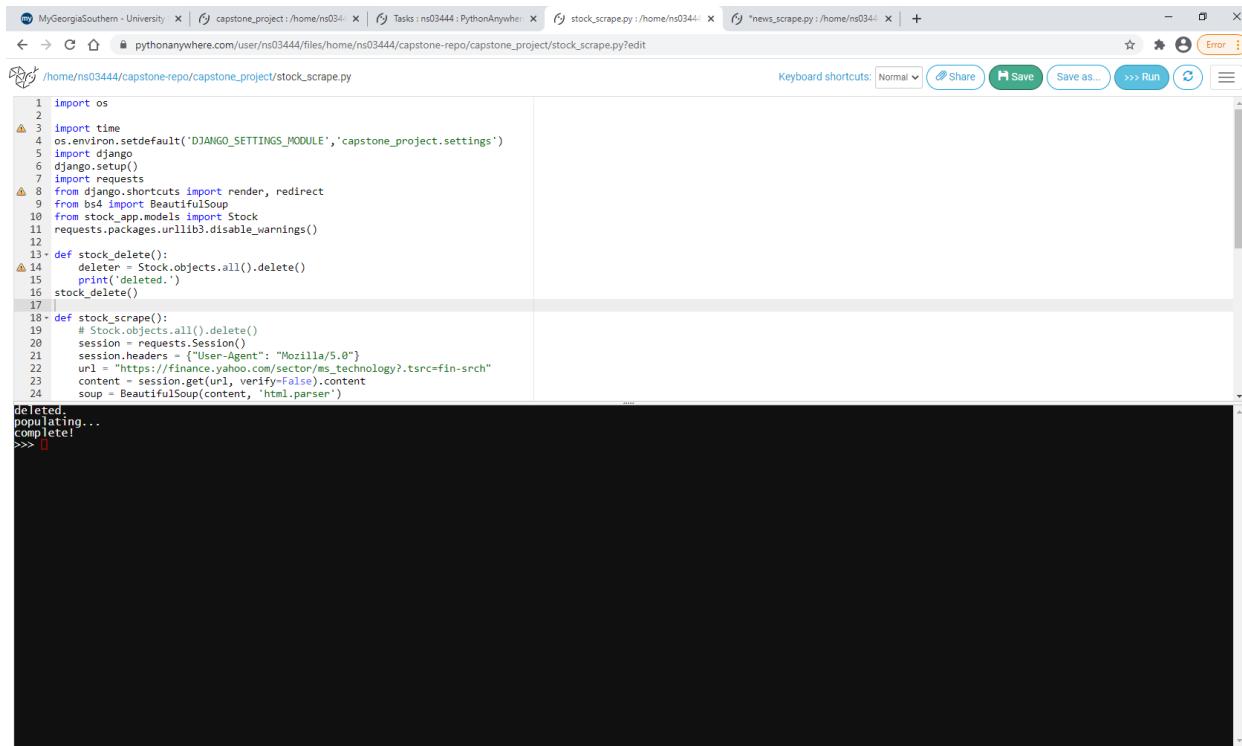
Figure 5.4.1-2 Creating function to delete all news article items from each table.

```

12
13 def stock_delete():
14     deleter = Stock.objects.all().delete()
15     print('deleted.')
16 stock_delete()
17
18 def stock_scrape():

```

Figure 5.4.1-3 Creating function to delete all stock items from each table.



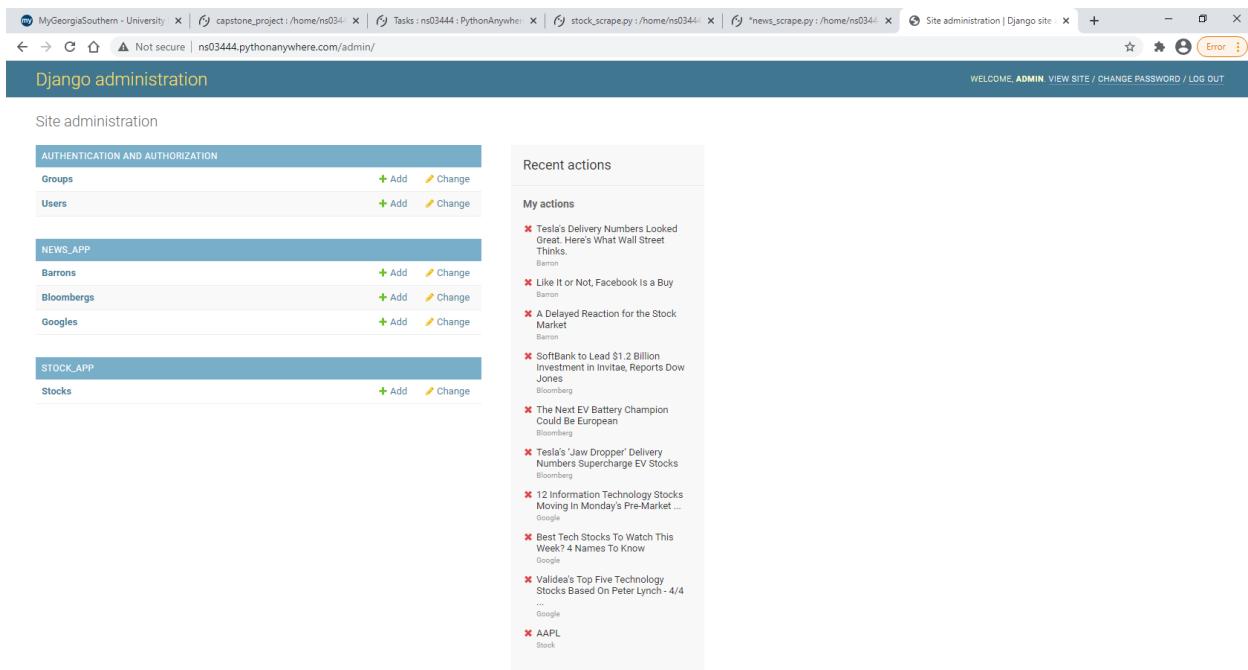
```

1 import os
2
3 import time
4 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'capstone_project.settings')
5 import django
6 django.setup()
7 import requests
8 from django.shortcuts import render, redirect
9 from bs4 import BeautifulSoup
10 from stock_app.models import Stock
11 requests.packages.urllib3.disable_warnings()
12
13 def stock_delete():
14     deleter = Stock.objects.all().delete()
15     print('deleted.')
16 stock_delete()
17
18 def stock_scrape():
19     # Stock.objects.all().delete()
20     session = requests.Session()
21     session.headers = {"User-Agent": "Mozilla/5.0"}
22     url = "https://finance.yahoo.com/sector/ms_technology?.tsrc=fin-srch"
23     content = session.get(url, verify=False).content
24     soup = BeautifulSoup(content, 'html.parser')

```

deleted
populating...
complete!

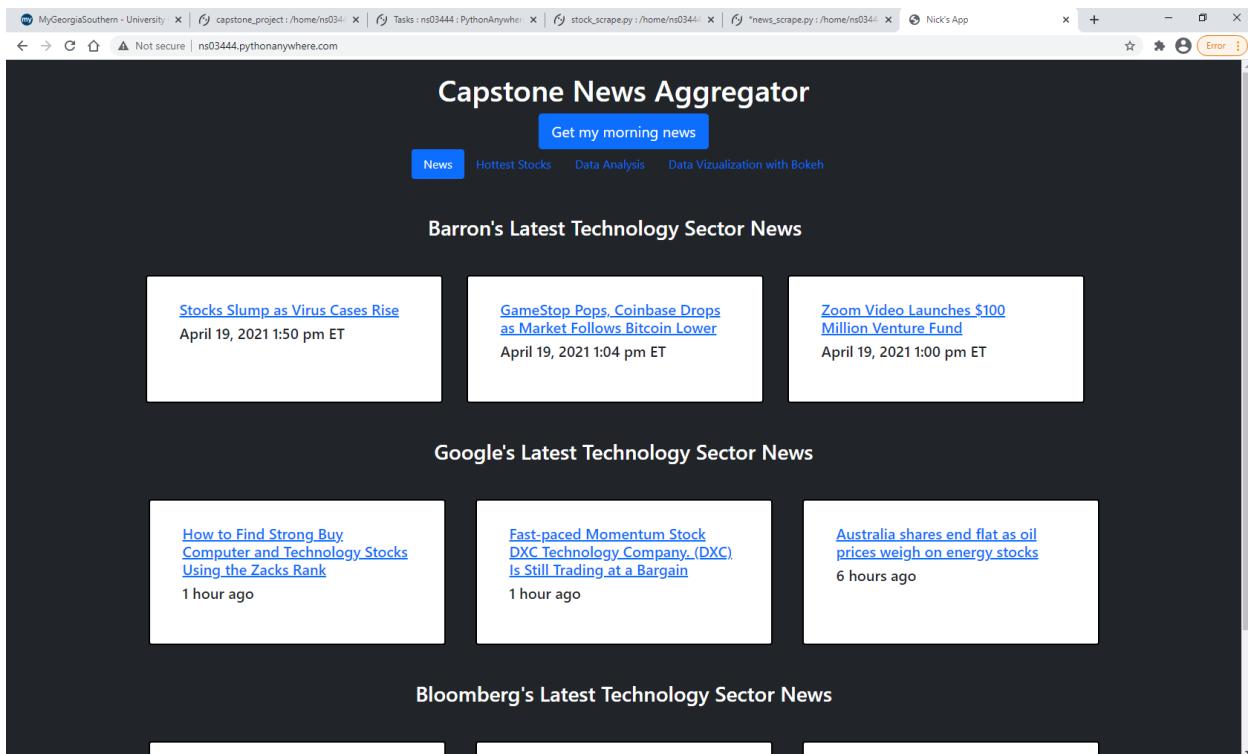
Figure 5.4.1-4 Running news & stock scrape files with delete functions added.



The screenshot shows the Django administration interface. On the left, there is a sidebar with 'AUTHENTICATION AND AUTHORIZATION' sections for 'Groups' and 'Users'. Below these are sections for 'NEWS_APP' (Barons, Bloomberg, Googles) and 'STOCK_APP' (Stocks). On the right, there are two panels: 'Recent actions' and 'My actions'. The 'Recent actions' panel lists several news items that have been removed, each with a red 'X' icon and a link to the news source. The 'My actions' panel shows a list of items that the user has recently deleted.

Category	Item	Source
Recent actions	Tesla's Delivery Numbers Looked Great. Here's What Wall Street Thinks.	Baron
Recent actions	Like It or Not, Facebook Is a Buy	Baron
Recent actions	A Delayed Reaction for the Stock Market	Baron
Recent actions	Scotiabank to Lead \$1.2 Billion Investment in Invatae, Reports Dow Jones	Bloomberg
Recent actions	The Next EV Battery Champion Could Be European	Bloomberg
Recent actions	Tesla's 'Jaw Dropper' Delivery Numbers Supercharge EV Stocks	Bloomberg
Recent actions	12 Information Technology Stocks Moving in Monday's Pre-Market ...	Google
Recent actions	Best Tech Stocks To Watch This Week? 4 Names To Know	Google
Recent actions	Validea's Top Five Technology Stocks Based On Peter Lynch - 4/4	Google
Recent actions	AAPL	Stock
My actions	Tesla's Delivery Numbers Looked Great. Here's What Wall Street Thinks.	Baron
My actions	Like It or Not, Facebook Is a Buy	Baron
My actions	A Delayed Reaction for the Stock Market	Baron
My actions	Scotiabank to Lead \$1.2 Billion Investment in Invatae, Reports Dow Jones	Bloomberg
My actions	The Next EV Battery Champion Could Be European	Bloomberg
My actions	Tesla's 'Jaw Dropper' Delivery Numbers Supercharge EV Stocks	Bloomberg
My actions	12 Information Technology Stocks Moving in Monday's Pre-Market ...	Google
My actions	Best Tech Stocks To Watch This Week? 4 Names To Know	Google
My actions	Validea's Top Five Technology Stocks Based On Peter Lynch - 4/4	Google
My actions	AAPL	Stock

Figure 5.4.1-5 Django admin panel indicating that items have recently been removed.



The screenshot shows the Capstone News Aggregator website. The header features a 'Get my morning news' button and navigation links for 'News', 'Hottest Stocks', 'Data Analysis', and 'Data Visualization with Bokeh'. The main content is divided into three sections: 'Barron's Latest Technology Sector News', 'Google's Latest Technology Sector News', and 'Bloomberg's Latest Technology Sector News'. Each section contains three news cards with titles, publication dates, and timestamps.

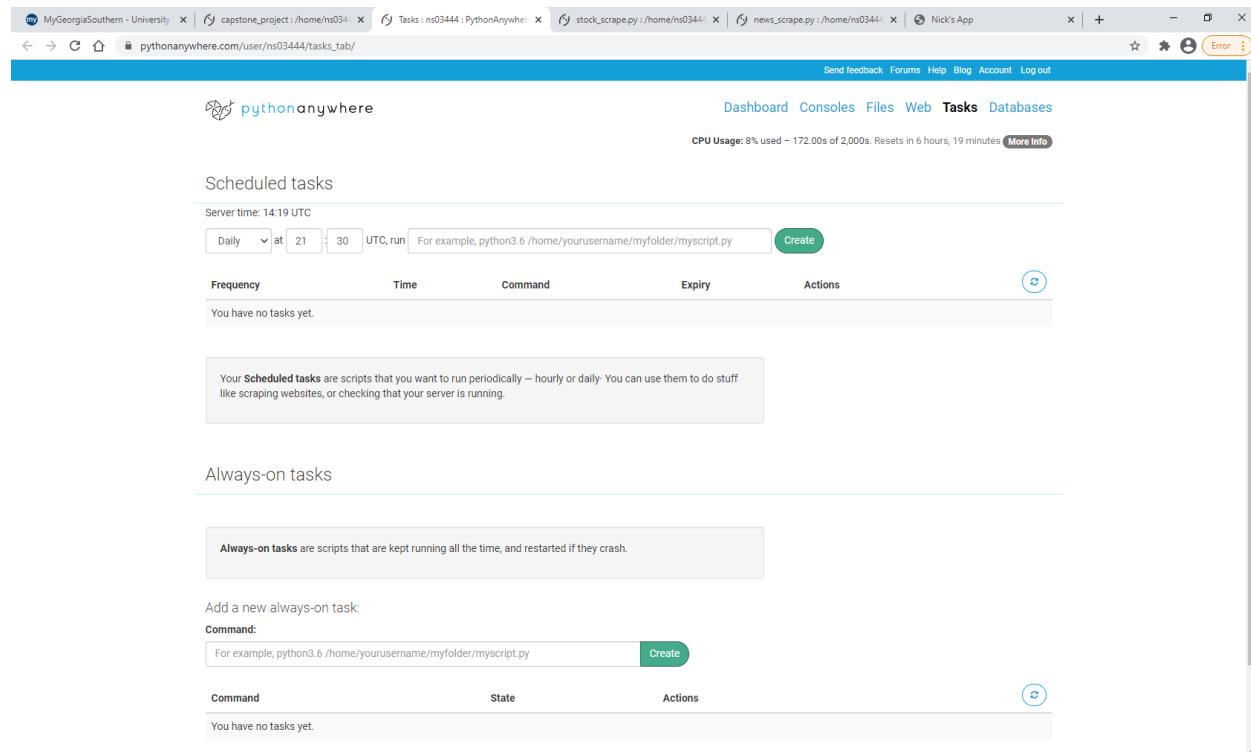
Section	News Title	Published	Timestamp
Barron's Latest Technology Sector News	Stocks Slump as Virus Cases Rise	April 19, 2021	1:50 pm ET
Barron's Latest Technology Sector News	GameStop Pops, Coinbase Drops as Market Follows Bitcoin Lower	April 19, 2021	1:04 pm ET
Barron's Latest Technology Sector News	Zoom Video Launches \$100 Million Venture Fund	April 19, 2021	1:00 pm ET
Google's Latest Technology Sector News	How to Find Strong Buy Computer and Technology Stocks Using the Zacks Rank	1 hour ago	
Google's Latest Technology Sector News	Fast-paced Momentum Stock DXP Technology Company, (DXC) Is Still Trading at a Bargain	1 hour ago	
Google's Latest Technology Sector News	Australia shares end flat as oil prices weigh on energy stocks	6 hours ago	
Bloomberg's Latest Technology Sector News			

Figure 5.4.1-6 News content updated.

After both scripts ran, I checked the webpage content to make sure they worked correctly and found that it did! Now I will just need to schedule them to run.

6.4.2 Scheduling the tasks

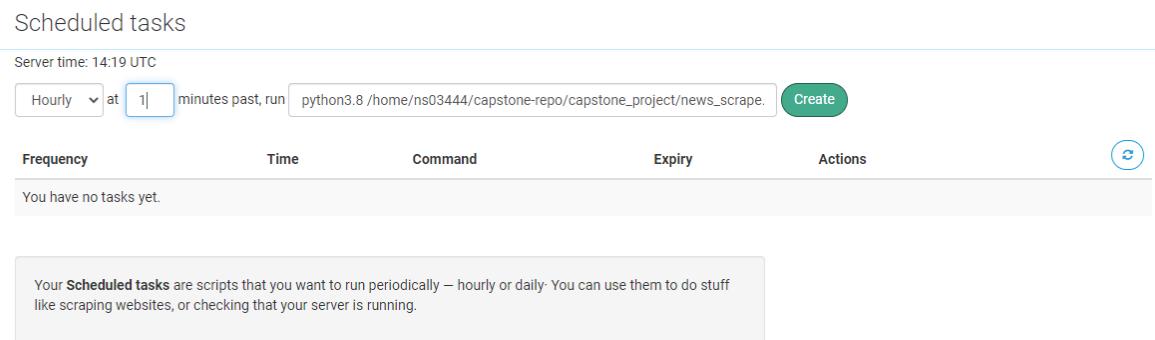
Python Anywhere offers “scheduled” tasks, as well as, “always-on” tasks, however, with the free account, I am only able to schedule one task to run once a day. Since I did not feel that this would be sufficient for updating the content as often as I wanted, I upgraded my subscription to the cheapest paid account type for \$5.00 USD/month.



The screenshot shows the Python Anywhere task tab interface. At the top, there are several tabs: MyGeorgiaSouthern - University, capstone_project : /home/ns0344..., Tasks : ns03444 : PythonAnywhere, stock_scrape.py : /home/ns0344..., news_scrape.py : /home/ns0344..., and Nick's App. Below the tabs, the main content area has a header with the Python Anywhere logo and navigation links: Dashboard, Consoles, Files, Web, Tasks (which is selected and highlighted in blue), and Databases. A CPU Usage status bar indicates 8% used - 172.00s of 2,000s. Resets in 6 hours, 19 minutes, with a 'More Info' link. The main content area is titled 'Scheduled tasks' and shows a table for scheduling tasks. The table has columns: Frequency, Time, Command, Expiry, and Actions. A 'Create' button is located at the top right of the table. Below the table, a message box states: 'Your **Scheduled tasks** are scripts that you want to run periodically – hourly or daily. You can use them to do stuff like scraping websites, or checking that your server is running.' The 'Always-on tasks' section is also visible, showing a table for always-on tasks with a 'Create' button and a message box stating: 'Always-on tasks are scripts that are kept running all the time, and restarted if they crash.' The 'Scheduled tasks' table currently shows 'You have no tasks yet.'

Figure 5.4.2-1 Python Anywhere task tab.

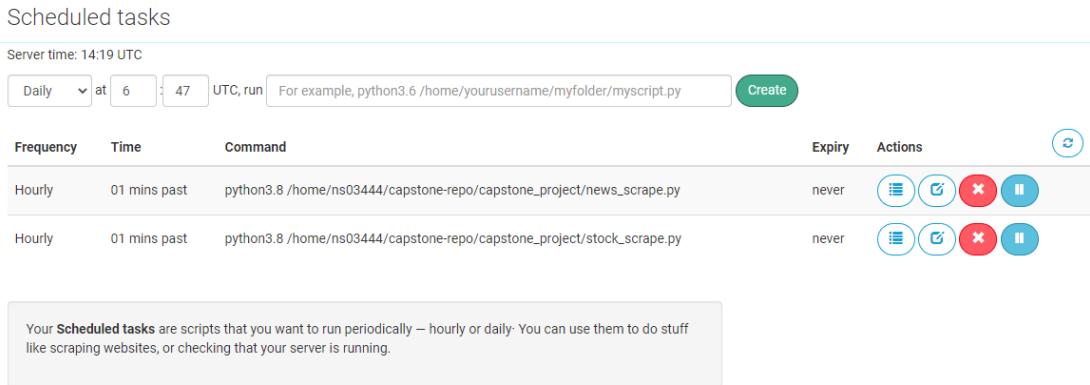
Now I can schedule the tasks to run on an hourly basis throughout the day.



The screenshot shows the Python Anywhere task tab interface, specifically the 'Scheduled tasks' section. The table for scheduling tasks has a new entry: 'Hourly' at '1' minutes past, run 'python3.8 /home/ns03444/capstone-repo/capstone_project/news_scrape.py'. The 'Create' button is visible at the top right of the table. Below the table, a message box states: 'Your **Scheduled tasks** are scripts that you want to run periodically – hourly or daily. You can use them to do stuff like scraping websites, or checking that your server is running.' The 'Always-on tasks' section is also visible, showing a table for always-on tasks with a 'Create' button and a message box stating: 'Always-on tasks are scripts that are kept running all the time, and restarted if they crash.' The 'Scheduled tasks' table currently shows 'You have no tasks yet.'

Figure 5.4.2-2 Scheduling “news_scrape.py” to run hourly.

By clicking the create button in the image above, I will create a scheduled task for running the “news_scrape.py” script, one minute past every new hour (ex. 12:01pm). Next, I will do the same for the “stock_scrape.py” file and it will be time test my final product!



The screenshot shows a 'Scheduled tasks' interface. At the top, it displays 'Server time: 14:19 UTC'. Below this is a search bar with 'Daily' selected, 'at 6 : 47 UTC, run For example, python3.6 /home/yourusername/myfolder/myscript.py' and a 'Create' button. The main area lists two tasks:

Frequency	Time	Command	Expiry	Actions
Hourly	01 mins past	python3.8 /home/ns03444/capstone-repo/capstone_project/news_scrape.py	never	
Hourly	01 mins past	python3.8 /home/ns03444/capstone-repo/capstone_project/stock_scrape.py	never	

At the bottom, a note states: 'Your **Scheduled tasks** are scripts that you want to run periodically – hourly or daily. You can use them to do stuff like scraping websites, or checking that your server is running.'

Figure 5.4.2-2 Scheduling both tasks to run hourly.

6.4.3 Testing the Scheduler

As previously noted, the tasks are scheduled to run once every hour, I will now take screenshots of the webpages containing content before and after the scheduled tasks run at the beginning of the hour. Notice that some article titles stay the same, while the time changes. This is because the website's I am scraping the information from have not released any new articles, however, the amount of time the article has been posted updates. The prices on the stock performance table are also updated.

***One special note, I ran into time zone complications with the “Barron’s” website, therefore, all times are displayed for their articles is in UTC time (four hours ahead).*

Capstone News Aggregator

Get my morning news

News Hottest Stocks Data Analysis Data Visualization with Python

Ticker Symbol	Company Name	Price (Intraday)
AAPL	Apple Inc.	133.73
MSFT	Microsoft Corporation	260.43
TSM	Taiwan Semiconductor Manufacturing Company Limited	116.26
NVDA	NVIDIA Corporation	609.5
ASML	ASML Holding N.V.	651.59
INTC	Intel Corporation	62.76
ADBE	Adobe Inc.	510.35
ORCL	Oracle Corporation	74.76
CSCO	Cisco Systems, Inc.	51.89
CRM	salesforce.com, inc.	232.95
ACN	Accenture plc	292.85
AVGO	Broadcom Inc.	456.73
TXN	Texas Instruments Incorporated	188.11
SAP	SAP SE	145.04
QCOM	QUALCOMM Incorporated	135.17
SHOP	Shopify Inc.	1127.05
IBM	International Business Machines Corporation	142.26
AMAT	Applied Materials, Inc.	133.26
INTU	Intuit Inc.	409.15
SQ	Square, Inc.	248.49

Figure5.4.3-1 Stock table content at 12:59pm 4/22/2021

Capstone News Aggregator

Get my morning news

News Hottest Stocks Data Analysis Data Visualization with Python

Ticker Symbol	Company Name	Price (Intraday)
AAPL	Apple Inc.	134.02
MSFT	Microsoft Corporation	260.83
TSM	Taiwan Semiconductor Manufacturing Company Limited	116.13
NVDA	NVIDIA Corporation	610.53
ASML	ASML Holding N.V.	654.08
INTC	Intel Corporation	63.4
ADBE	Adobe Inc.	511.73
ORCL	Oracle Corporation	75.17
CSCO	Cisco Systems, Inc.	52.03
CRM	salesforce.com, inc.	233.14
ACN	Accenture plc	292.33
AVGO	Broadcom Inc.	460.54
TXN	Texas Instruments Incorporated	188.36
SAP	SAP SE	144.77
QCOM	QUALCOMM Incorporated	135.46
SHOP	Shopify Inc.	1129.66
IBM	International Business Machines Corporation	142.57
AMAT	Applied Materials, Inc.	133.96
SQ	Square, Inc.	255.08
INTU	Intuit Inc.	410.33

Figure5.4.3-2 Stock table content at 1:02 4/22/2021- (***Prices have been updated)

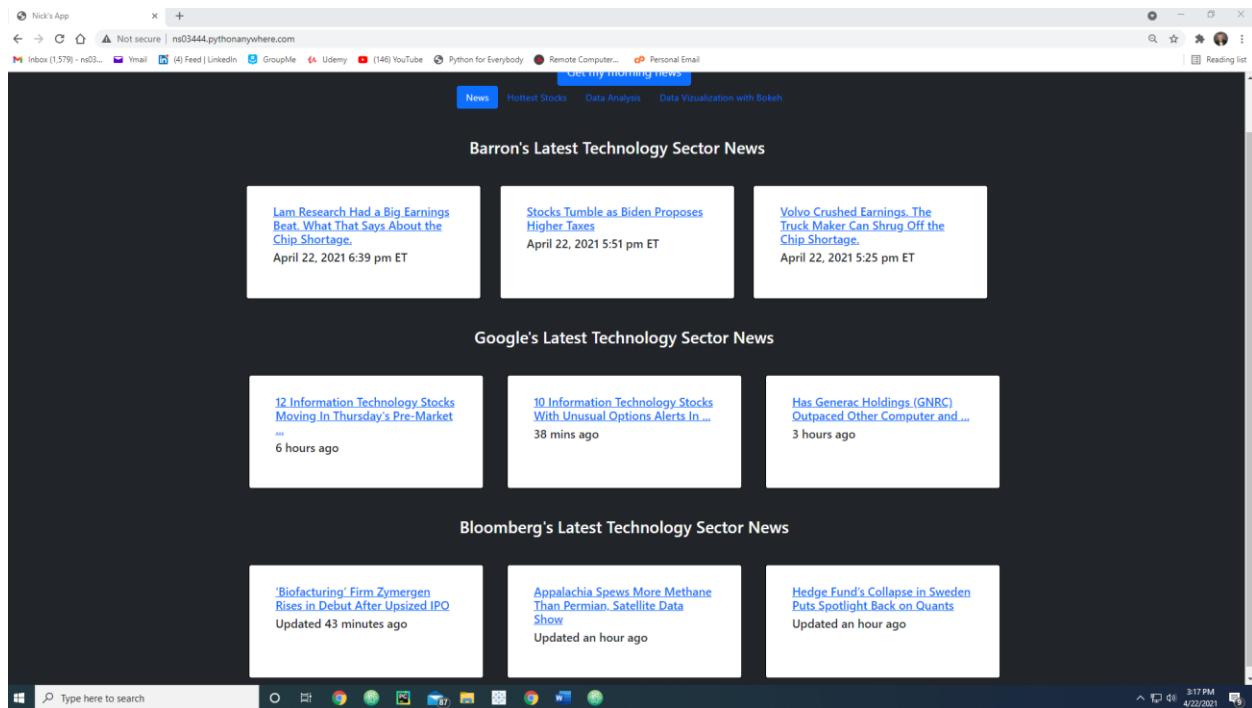


Figure5.4.3-3 News content at 3:17 4/22/2021

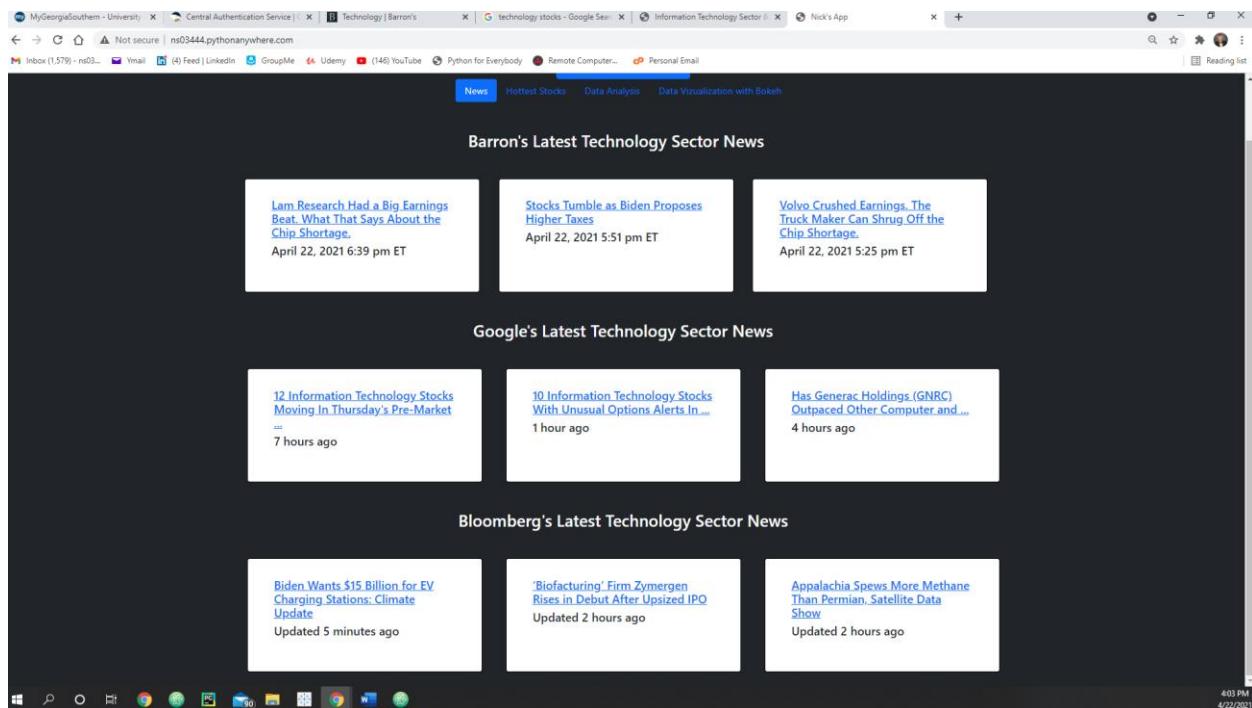


Figure5.4.3-4 News content at 4:03 4/22/2021- (***Content has been updated)

MyGeorgiaSouthern - University | Central Authentication Service | Technology | Barron's | Technology stocks - Google Search | Information Technology Sector | Nick's App

← → ⌛ bloomberg.com/markets/sectors/information-technology | ↗ Technology | Sign In | Subscribe | Reading list

Inbox (1,579) - ns0... | Ymail | (4) Feed | LinkedIn | GroupMe | Udemy | (148) YouTube | Python for Everybody | Remote Computer... | Personal Email

☰ Menu | Q Search | Bloomberg | More Sector News >

Latest Sector News

Updated 3 minutes ago [Biden Wants \\$15 Billion for EV Charging Stations: Climate Update](#)

President Joe Biden's climate summit has drawn 40 heads of state including China's Xi Jinping, Vladimir Putin of Russia and Boris Johnson from the U.K. The two-day virtual event ends Friday and will include corporate executives, union heads, Pope Francis and Bill Gates.

Updated 2 hours ago ['Biofacturing' Firm Zymergen Rises in Debut After Upsized IPO](#)

Zymergen Inc., which uses biological processes to manufacture chemicals, rose as much as 33% after expanding its initial public offering to raise \$500 million.

Updated 2 hours ago [Appalachia Spews More Methane Than Permian, Satellite Data Show](#)

The Appalachian Basin spanning Alabama to Maine spewed more methane last year than the oil- and gas-heavy Permian Basin of Texas and New Mexico -- making the region the biggest emitter of the greenhouse gas in the nation.

Updated 6 minutes ago [The Dirty Little Secrets of Electric Cars](#)

Adam Jonas of Morgan Stanley talks about the role electric cars can play in cutting greenhouse gases. He also talks about how making EV batteries is a "dirty" process. (Source: Bloomberg)

Bloomberg Green BIDEN'S CLEAN ENERGY, ELECTRIC VEHICLE BUYING PLAN DELAYED TO JULY

Taboola Feed

Get retirement plan support from people who get small business

4:03 PM 4/22/2021

Bloomberg's Latest Technology Sector News

[Biden Wants \\$15 Billion for EV Charging Stations: Climate Update](#) Updated 5 minutes ago

['Biofacturing' Firm Zymergen Rises in Debut After Upsized IPO](#) Updated 2 hours ago

[Appalachia Spews More Methane Than Permian, Satellite Data Show](#) Updated 2 hours ago

MyGeorgiaSouthern - University | Central Authentication Service | Technology | Barron's | Google Search | Information Technology Sector | Nick's App

Inbox (1,579) - ns03... Ymail (4) Feed | LinkedIn (46) YouTube Python for Everybody Remote Computer... Personal Email

Search News & Quotes

BARRON'S Topics Magazine Data Advisor Penta

Subscriber Benefits | Sign In

Lam Research Had a Big Earnings Beat. What That Says About the Chip Shortage.

By Max A. Chasney

April 22, 2021 2:09 pm ET

2 min

[Go to article >](#)



Stocks Tumble as Biden Proposes Higher Taxes

By Steve Goldstein, Jacob Sommersheide, and Ben Levisohn

April 22, 2021 1:51 pm ET

According to a report, the Biden administration will propose nearly doubling the capital-gains tax rate for wealthy Americans to 39.6%.

2 min

[Go to article >](#)



Volvo Crushed Earnings. The Truck Maker Can Shrug Off the Chip Shortage.

By Callum Koivisto

April 22, 2021 1:25 pm ET

Volvo stock jumped early on Thursday, as the Swedish truck maker crushed expectations in the first quarter amid surging demand.

4:02 PM

4/22/2021

Barron's Latest Technology Sector News

[Lam Research Had a Big Earnings Beat. What That Says About the Chip Shortage.](#)

April 22, 2021 6:39 pm ET

[Stocks Tumble as Biden Proposes Higher Taxes](#)

April 22, 2021 5:51 pm ET

[Volvo Crushed Earnings. The Truck Maker Can Shrug Off the Chip Shortage.](#)

April 22, 2021 5:25 pm ET

MyGeorgiaSouthern - University | Central Authentication Service | Technology | Barron's | Google - technology stocks | Information Technology Sector | Nick's App

← → ⌛ 🔍 google.com/search?q=technology+stocks&tbs=nvs&source=lnrt&tbs=qdr:cds&sa=X&ved=0ahUKEwjs8XwpJvAhURLs0HRyCxEQpwUIKQ&biw=1920&bih=937&dpr=1

Inbox (1,579) - ns03... Ymail (4) Feed | LinkedIn GroupMe Udemy (146) YouTube Python for Everybody Remote Computer... Personal Email

Reading list

technology stocks

Q All News Shopping Books Images More Settings Tools

Past 24 hours • Sorted by relevance • Clear

Benzinga
[12 Information Technology Stocks Moving In Thursday's Pre ...](#)
The market value of their outstanding shares is at \$14.7 million. Mechanical Technology (NASDAQ: MKTY) stock fell 3.28% to \$8.56. The company's market cap ...
7 hours ago

Benzinga
[10 Information Technology Stocks With Unusual Options ...](#)
10 Information Technology Stocks With Unusual Options Alerts In Today's Session. by Benzinga Insights 5 min read. in 23 minutes ...
1 hour ago

Nasdaq
[Has Generac Holdings \(GNRC\) Outpaced Other Computer ...](#)
Investors with an interest in Computer and Technology stocks should continue to track GNRC. The stock will be looking to continue its solid performance. Want the ...
4 hours ago

Nasdaq
[TSX Ends On Firm Note As Healthcare, Technology Stocks Rally](#)
Healthcare, information technology and materials shares were the prominent gainers. A few stocks from consumer discretionary, energy and financial sectors too ...
22 hours ago

4:02 PM 4/22/2021

Google's Latest Technology Sector News

[12 Information Technology Stocks Moving In Thursday's Pre-Market](#)
...
7 hours ago

[10 Information Technology Stocks With Unusual Options Alerts In ...](#)
1 hour ago

[Has Generac Holdings \(GNRC\) Outpaced Other Computer and ...](#)
4 hours ago

7 Reflection

Developing this application has provided me an indescribable learning experience that could not have been taught any other way. In my opinion, the project was a huge success for several reasons. One of the most important reasons being that all milestones were met, and the final product is functioning exactly as described in the project proposal. Moving forward, I will use the information I learned from creating this project throughout my professional career.