SWE Project

RESTful APIs
Fall 2019
Team 7

Team 7 Roster

- Lateef
- Anthony
- Austin
- Christine
- David
- Andrew

Technology Overview

- XAMPP
- Windows OS
- PHP
- MySQL
- MariaDB
- myPHPadmin
- HTML/CSS/Bootstrap
- Postman
- Windows Task Scheduler
- SQLify.io   (third party tool/website)

Downloaded XAMPP

- https://www.apachefriends.org/index.html
- Downloaded XAMPP
- https://www.youtube.com/watch?time_continue=68&v=h6DEDm7C37A&feature=emb_title

Setting it up

- Download is simple
- When down, created shortcut of Control Panel and XAMPP Folder from C drive. Control panel was inside xampp folder in C drive.
- Sublime Text 3 editor would be the editor of choice.
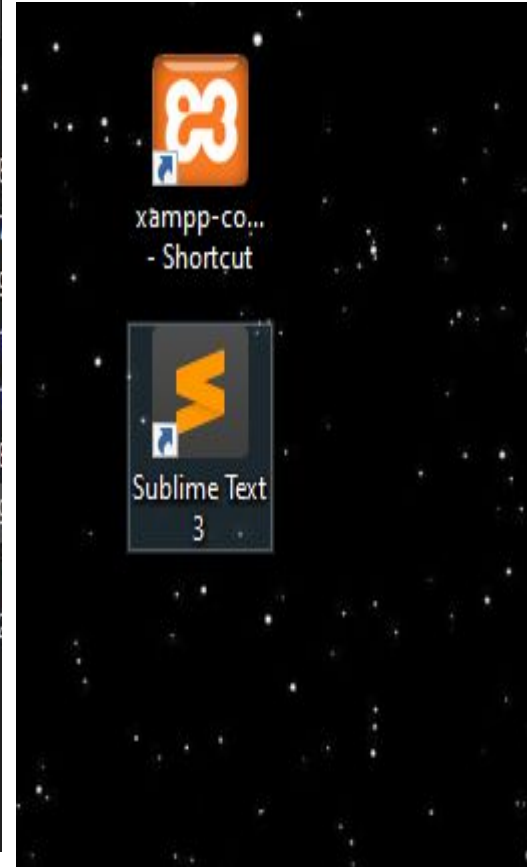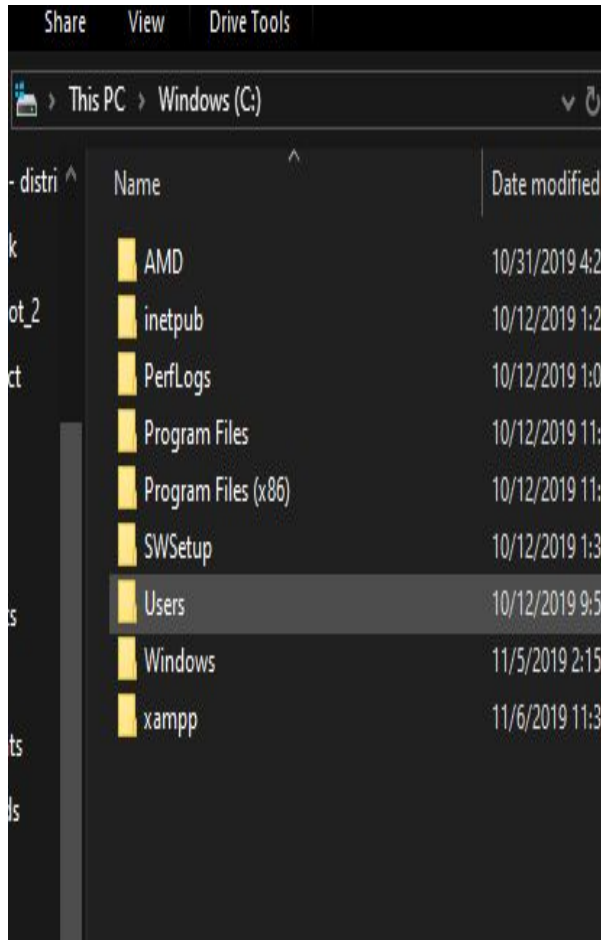
Languages Used

- Since we were using XAMPP, we had to try and abide by its main languages, MySQL and PHP.
- MariaDB is the standard of MySQL that was being used, thus documentation was read from their site.
- For front end was HTML, CSS, Bootstrap.

Working with XAMPP

- Allows us to turn our computer into server (localhost), and run server side operations.
- Same for database events.
- Open Control Panel ShortCut
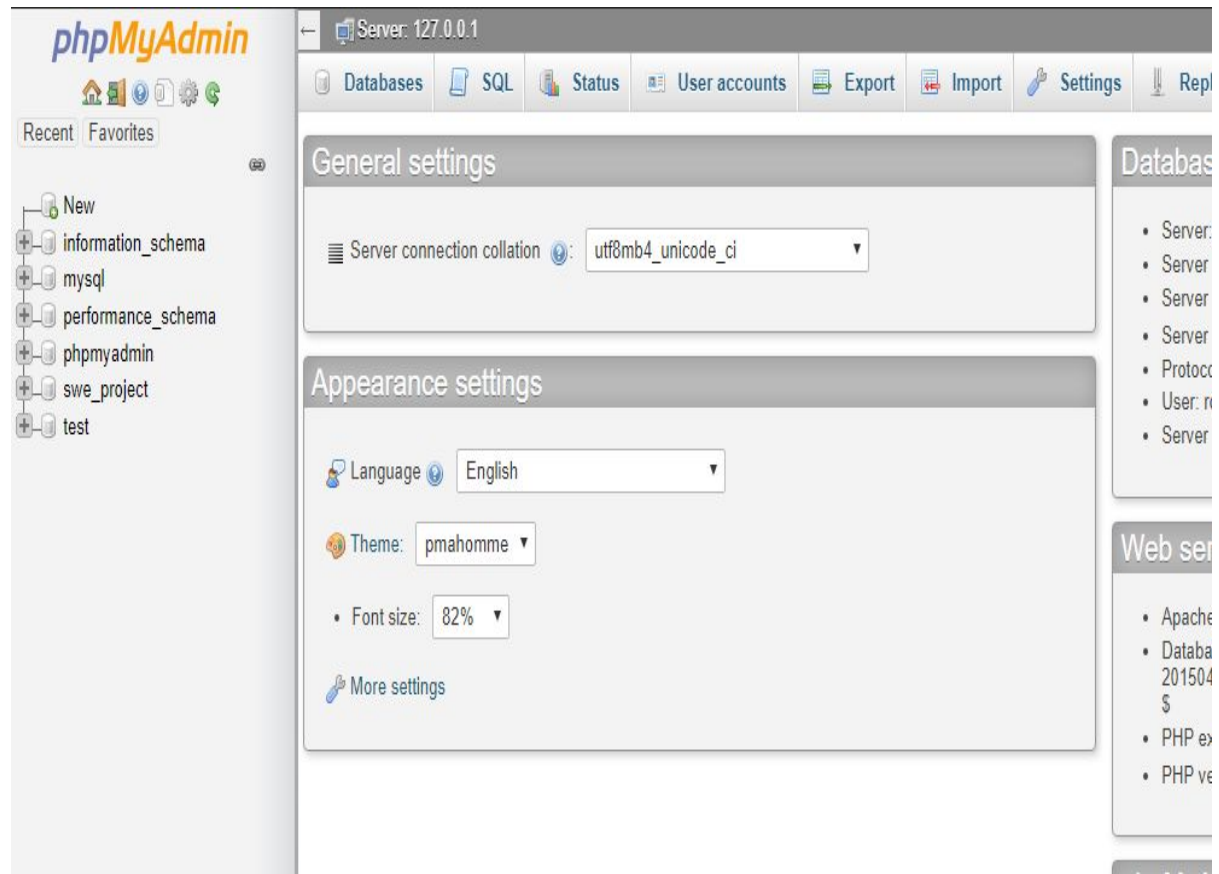- Click start on Apache and MySQL parts
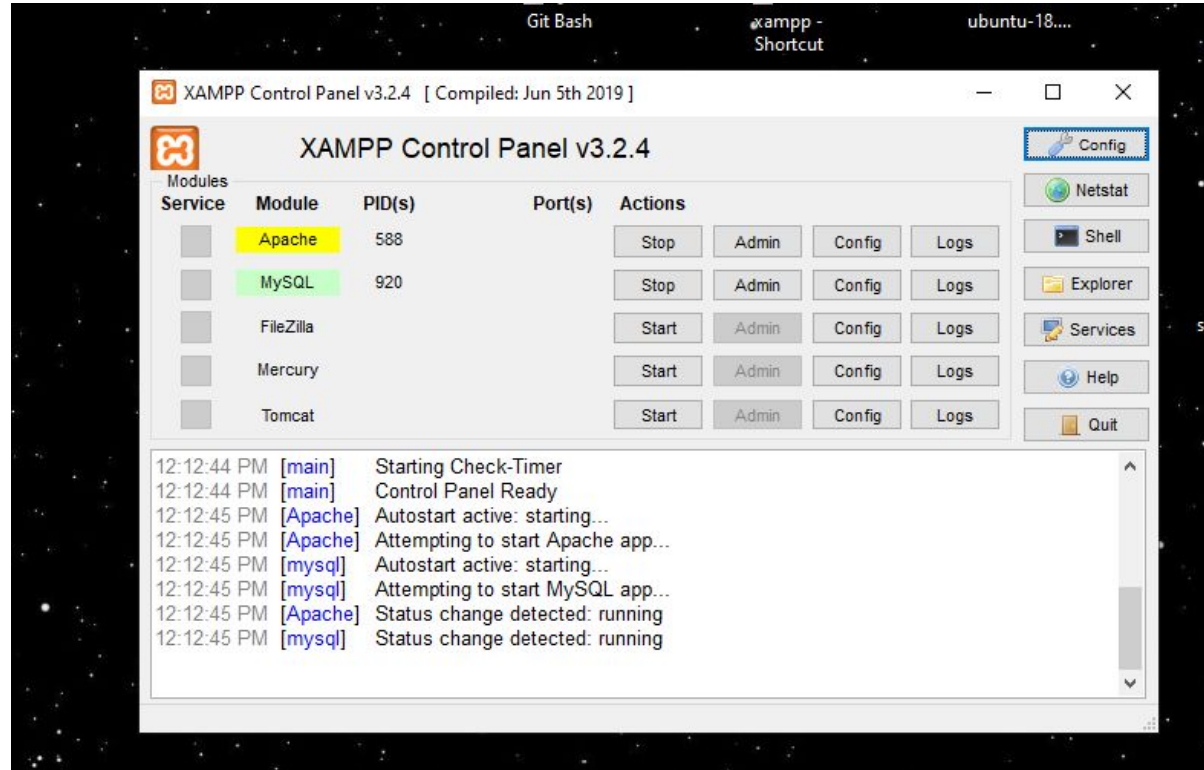
Open Web browser

- Our team used Google Chrome
- Type localhost in the url bar, and hit enter
- Should see screen similar to right (apache and mysql need to be on)

XAMPP Apache + MariaDB + PHP + Perl

Welcome to XAMPP for Windows 7.1.33

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.
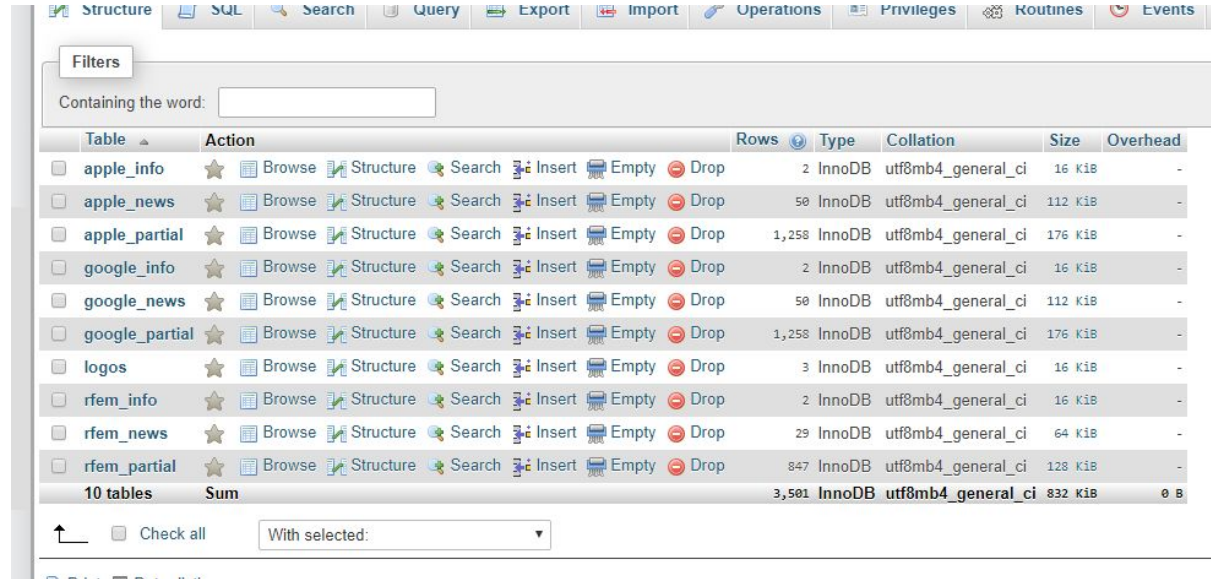
myPHPadmin

- Click on top right where it says myPHPadmin
- Once there, you should see screen similar to right
- This is a GUI that comes with XAMPP.
- Allows for table manipulation, database creation, setting events, managing permissions, etc.
- Simpler and helpful to go with MySQL shell,

myPHPadmin

- Relevant tables
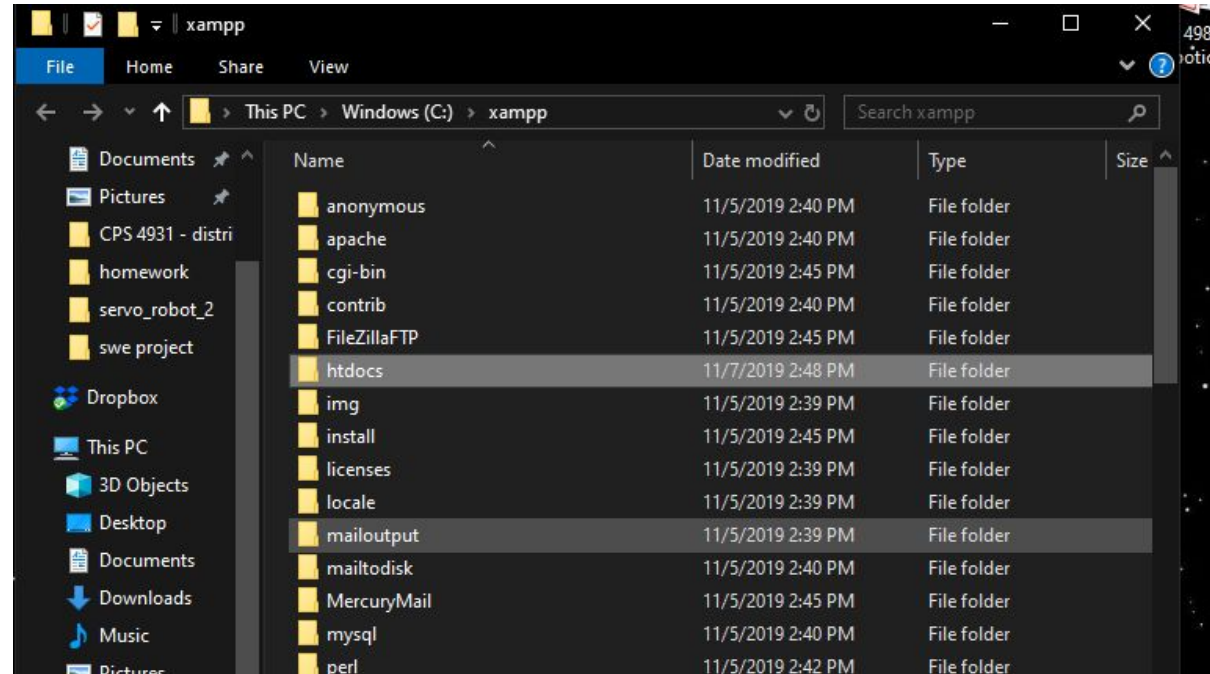- Database is MariaDB
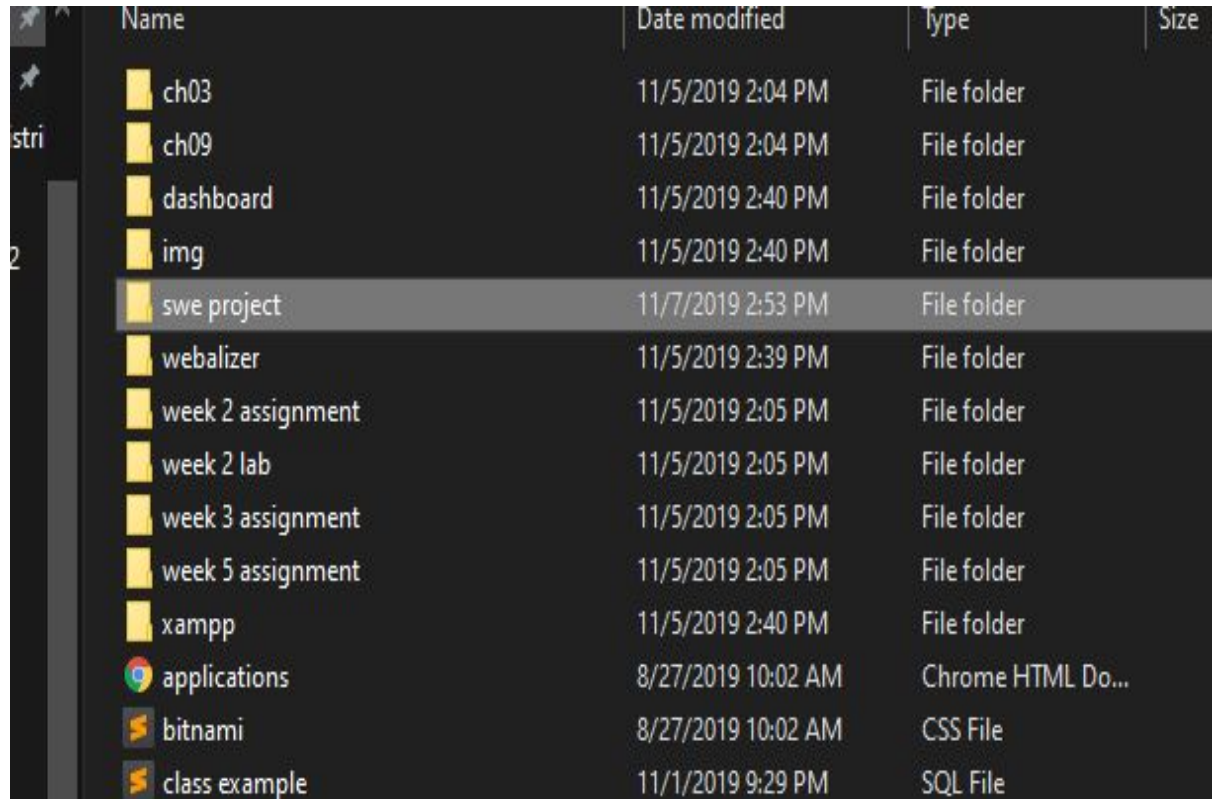- Comes installed with XAMPP

Coding

- To create and run files, open up XAMPP folder
- Go to htdocs
- Create relevant folder
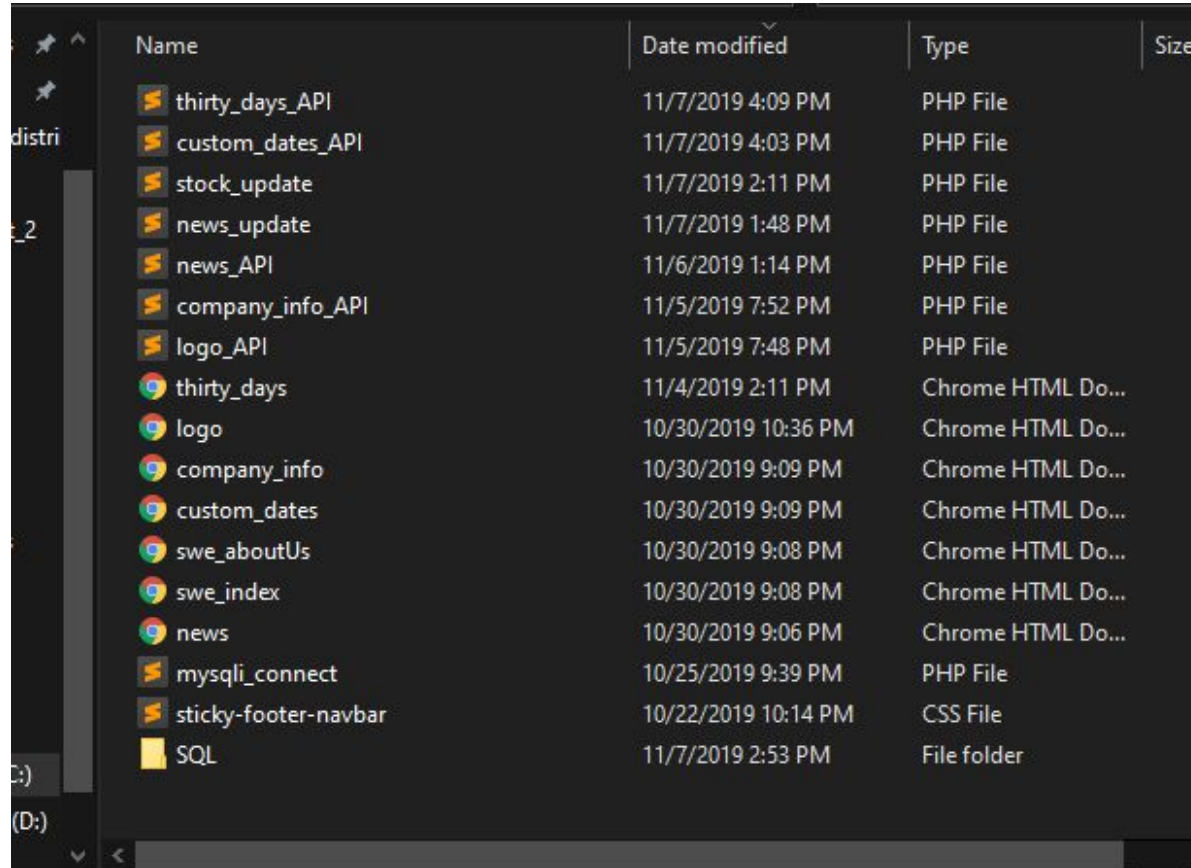- From there, place relevant source PHP, html, etc.

Coding

- To create and run files, open up XAMPP folder
- Go to htdocs
- Create relevant folder
- From there, place relevant source PHP, html, etc.
- Our project folder would be called 'swe project'

Name	Date modified	Type	Size
ch03	11/5/2019 2:04 PM	File folder	
ch09	11/5/2019 2:04 PM	File folder	
dashboard	11/5/2019 2:40 PM	File folder	
img	11/5/2019 2:40 PM	File folder	
swe project	11/7/2019 2:53 PM	File folder	
webalizer	11/5/2019 2:39 PM	File folder	
week 2 assignment	11/5/2019 2:05 PM	File folder	
week 2 lab	11/5/2019 2:05 PM	File folder	
week 3 assignment	11/5/2019 2:05 PM	File folder	
week 5 assignment	11/5/2019 2:05 PM	File folder	
xampp	11/5/2019 2:40 PM	File folder	
applications	8/27/2019 10:02 AM	Chrome HTML Do...	
bitnami	8/27/2019 10:02 AM	CSS File	
class example	11/1/2019 9:29 PM	SQL File	

Coding

- Once files are created, you are ready to test on browser.
- myPHPadmin will also help, with setting up database and table configuration, and just being an overall strong GUI
- Make sure apache and mySQL are still on control panel

- Type localhost and folder name. You don't need to type full path, and include htdocs.
- Hit enter
- You should see directory of files or just home index, depending on how files are titled.

File directory

- From here, you can see your files
- You can then click and start troubleshooting.

Index of /swe project

Name	Last modified	Size	Descripti
Parent Directory		-	
SQL/	2019-11-07 14:53	-	
company_info.html	2019-10-30 21:09	1.9K	
company_info_API.php	2019-11-05 19:52	1.0K	
custom_dates.html	2019-10-30 21:09	2.2K	
custom_dates_API.php	2019-11-07 16:03	5.9K	
logo.html	2019-10-30 22:36	1.9K	
logo_API.php	2019-11-05 19:48	1.1K	
mysqli_connect.php	2019-10-25 21:39	348	
news.html	2019-10-30 21:06	2.1K	
news_API.php	2019-11-06 13:14	1.3K	
news_update.php	2019-11-07 13:48	2.4K	
sticky-footer-navbar..>	2019-10-22 22:14	469	
stock_update.php	2019-11-07 14:11	2.2K	

After troubleshooting and testing

- We have to make sure the database is updated daily with appropriate values
- We, again, used PHP and mySQL to facilitate that.

```php
?>

<?php

    function fetch_json($url)
    {
        $curl = curl_init();
        curl_setopt_array($curl,
        [
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => $url
        ]);

        $resp = curl_exec($curl);
        curl_close($curl);

        if($resp == "You have exceeded your allotted message quota. Please enable pay-as-you-go to r
        {
            return "expired_token";
        }

        return json_decode($resp);
    }

    function build_url($base,$token,$symbol)
    {
        $replace_symbol = str_replace("xsymbolx",$symbol,$base);
        $replace_token = str_replace("xtokenx",$token,$replace_symbol);

        return $replace_token;
    }
?>
```

After troubleshooting and testing

- Ensure that daily IEX API calls are made
- Take the JSON from the calls, process it, and story it in our MariaDB

```php
?>
<?php
    function fetch_json($url)
    {
        $curl = curl_init();
        curl_setopt_array($curl,
        [
            CURLOPT_RETURNTRANSFER => 1,
            CURLOPT_URL => $url
        ]);

        $resp = curl_exec($curl);
        curl_close($curl);

        if($resp == "You have exceeded your allotted message quota. Please enable pay-as-you-go to r
        {
            return "expired_token";
        }

        return json_decode($resp);
    }

    function build_url($base,$token,$symbol)
    {
        $replace_symbol = str_replace("xsymbolx",$symbol,$base);
        $replace_token = str_replace("xtokenx",$token,$replace_symbol);

        return $replace_token;
    }
?>
```

Initial database setup

- We used Postman to visualize and perfect our API calls
- Once we got our proper calls, we downloaded the output into JSON files.
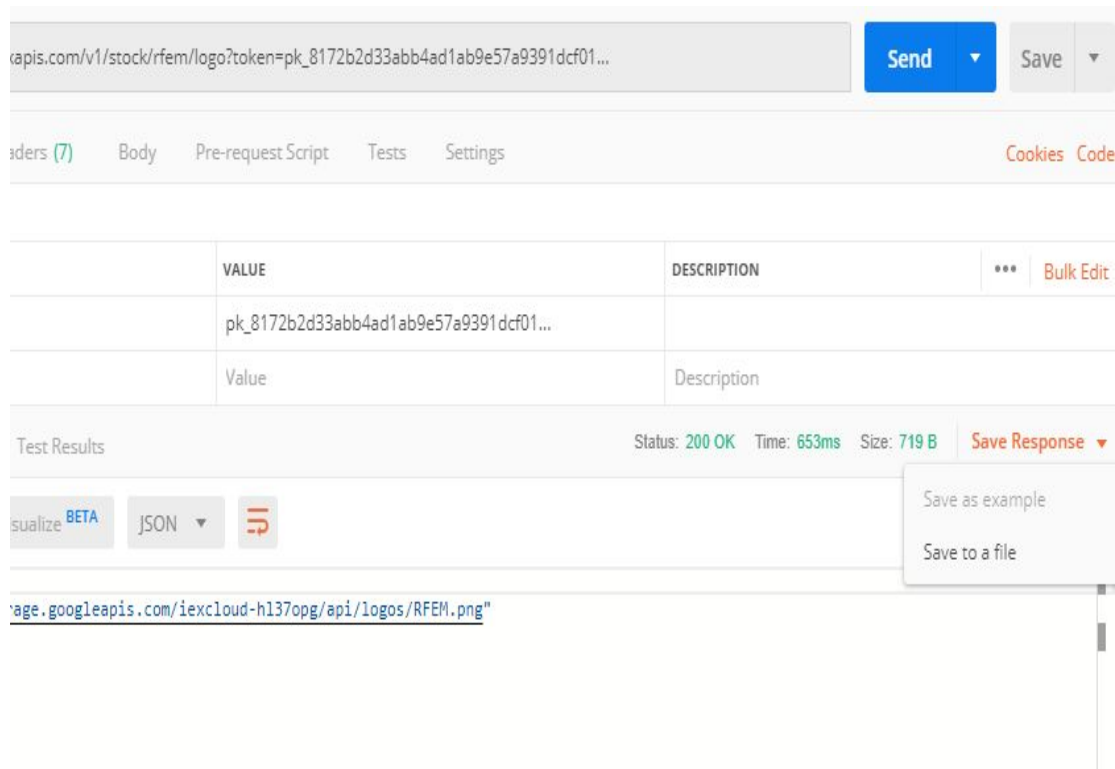
IEX API calls used

- We used these GET API calls from IEX to read and store our data into our databases
- https://iexcloud.io/docs/api/#historical-prices
- This was the going to be used for 5 year data pulls for Apple, Google, and RFEM. This contained stock info, such as open,close, etc.
- https://iexcloud.io/docs/api/#logo
- The above link shows and discusses the API call for getting company logos
- https://iexcloud.io/docs/api/#news
- This link, above, discusses getting news on respective companies.
- https://iexcloud.io/docs/api/#company
- The above link discusses using the IEX API to get basic info on a company.

Getting the JSON to SQL form

- Several ways were available to to convert our JSON to SQL.
- While our update script (for day to day updates) would have no issue making the conversion…..
- We ended up using a cool third party tool/website to assist with it.
- Once, the conversion was done, we took the SQL files, and made adjustments to them.
- Once the conversions were done, we imported them to our database using myPHPadmin.
- Database creation, as well as user privilege, was also handled on myPHPadmin.

The conversion tool

- https://sqlify.io/convert
- Strong tool
- Upload JSON
- Drop and add columns.
- Still needed to make some more adjustments in sublime text editor.

SQLIFY

Blog   Conver

Upload your CSV or JSON data

⬆ Upload a file   ☁ Input a URL   ▣ Paste raw

Click to select a file
or drop files here

We will automatically detect the format of your file

Select the result format

SQL   JSON   CSV
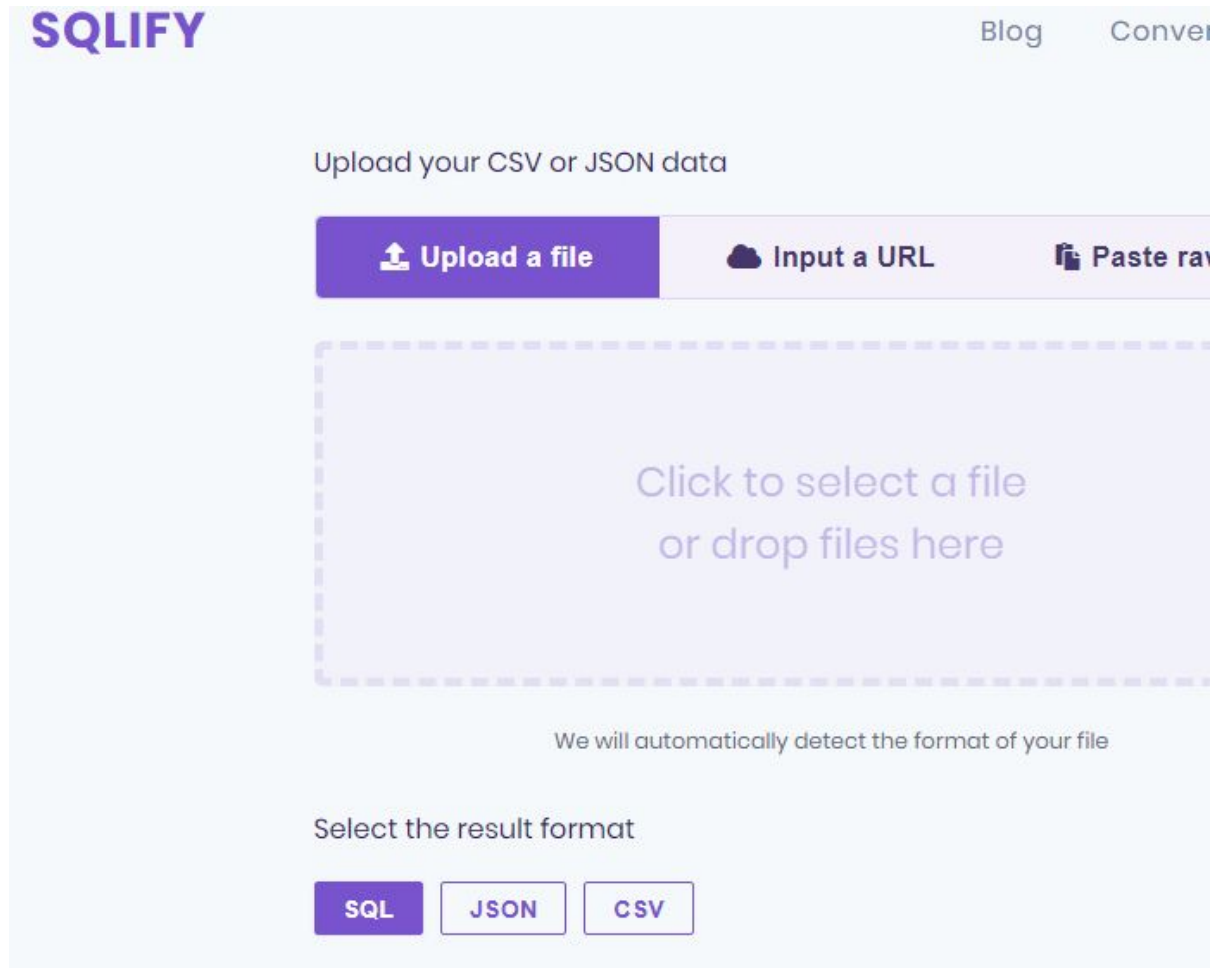
Importing the SQL

- Once done, we imported to myPHPadmin.
- The conversion tool was for initial use.
- Any updates would be done being read,converted, and inserted with PHP and MySQL.
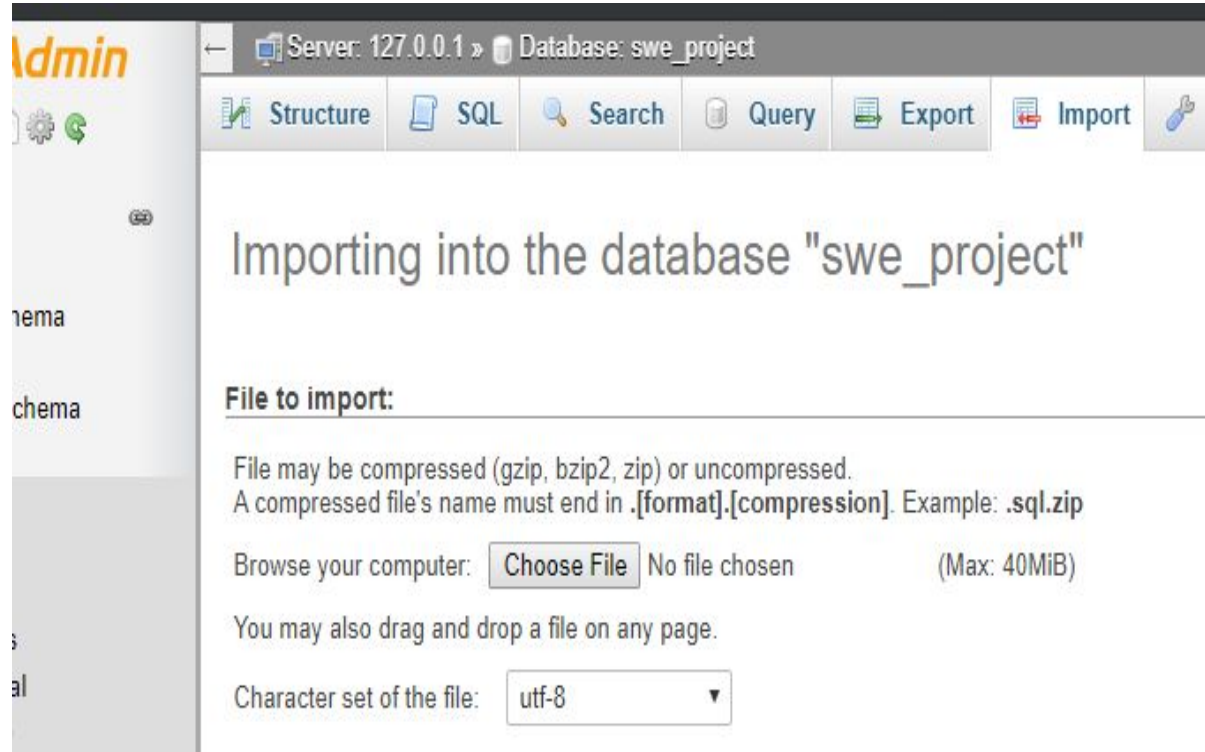
SQL files on GitHub

- The SQL files here, have 5 years of data for google and apple
- For RFEM, it goes back to June 15, 2016.
- The SQL files also feature multiple table creation.
- The files go up October 24, 2019. Because that is when I made the 5 year calls for each of the 3 companies.
- After October 24, 2019, The sql files were updated on MariaDB & myPHPadmin.
- This was done using two PHP files, acting as scripts.

Automation and daily updates

- We supported our daily update using Automation.
- We are developing in WIndows Environment, so we used Windows Task Scheduler.
- This would require a few things…
- Some useful links for all….
- https://www.youtube.com/watch?v=C4PdPqEOo6A
- https://code-examples.net/en/q/f81336
- https://www.computerhope.com/issues/ch001666.htm

First things first

- Any update done to the database required having our Apache and MySQL on.
- Thus when we automated our computer to turn on, when off, we had to include those in the start procedure.
- When turned on, we used our task scheduler to run our created batch file.
- The batch file would have php commands to files located in the htdocs folder.
- The php update scripts would be needed in that folder, in order for it to work.
- We had this all run daily, and automatically. If any issues, it would rerun until success
- Inside the PHP files, it had the IEX API urls and token. It would create a connection, read the JSON, convert it and send to the SQL database.
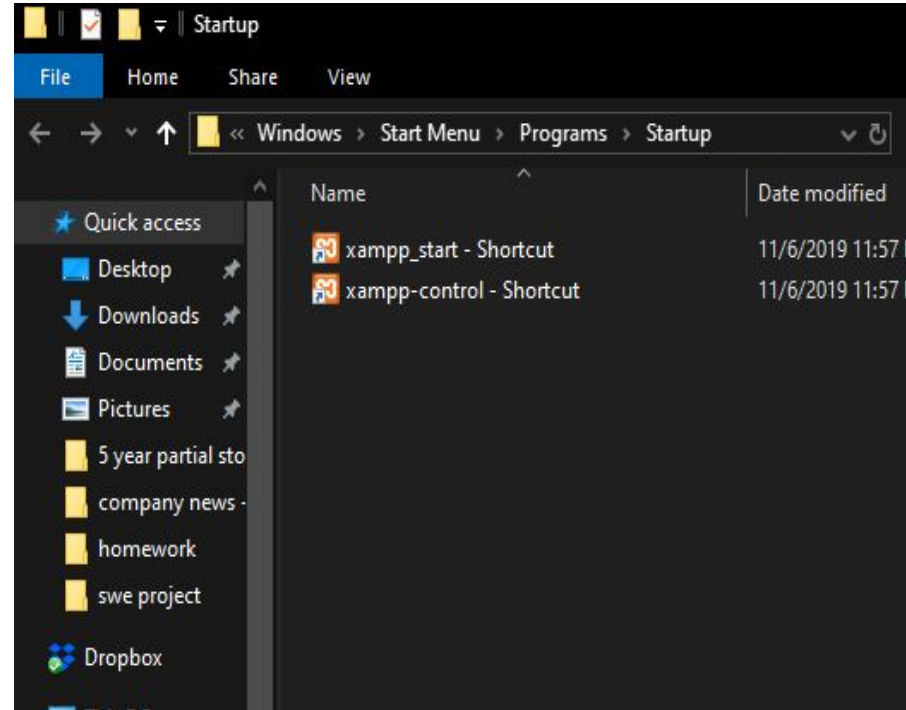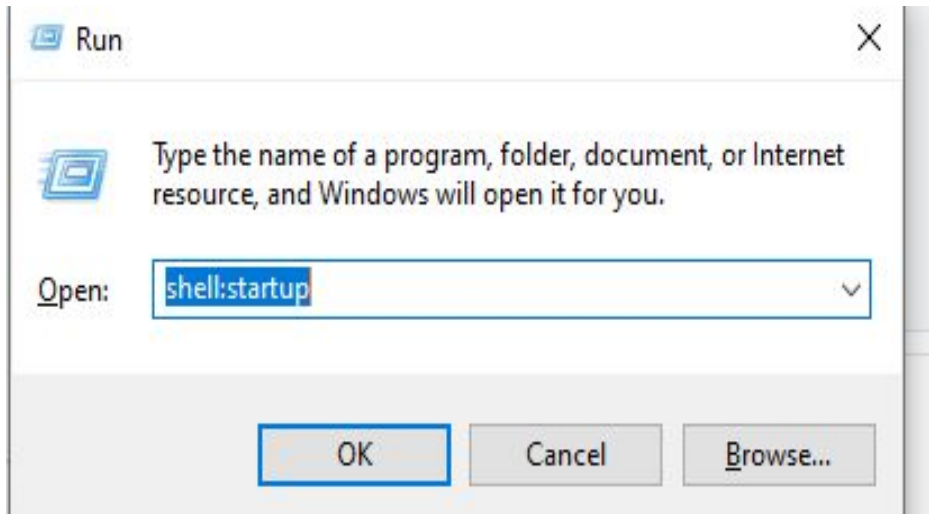
Getting XAMPP to start when the PC starts, so database
and server connection is alive.

daily_update - Notepad

File   Edit   Format   View   Help

```
C:\xampp\php\php.exe -f "C:\xampp\htdocs\swe project\stock_update.php"


C:\xampp\php\php.exe -f "C:\xampp\htdocs\swe project\news_update.php"
```

Simple batch file made to make calls to PHP compiler and PHP files. These files hold instructions to read and update.
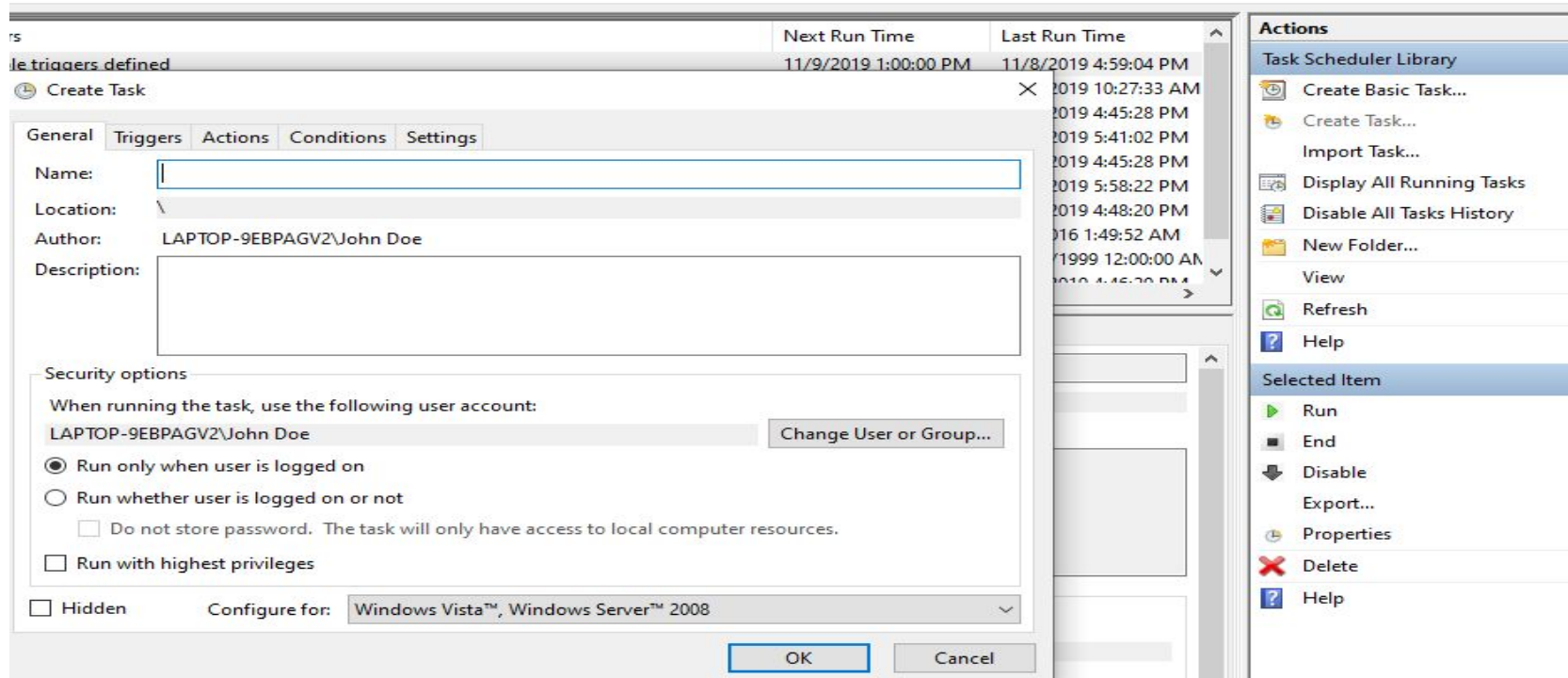
Windows Task Scheduler….where we set up the automated task of reading the batch daily, and starting up the PC, if necessary

Working with web and Postman

- Once we tested, and implemented our design, we had a working product.
- The product, as displayed in class, can work with postman, or work with our web front end.
- Our web front end had forms that connected to our PHP API.
- Our PHP API would read and calculate the SQL query and convert, finally to JSON output.

Info about how to test it on both Web and Postman

- Next few images will be self explanatory.
- It will show how we have true working APIs.
- We make our requests on the web, and get JSON from there.
- We can then take the same URL, and paste it into Postman, and get same result.
- We will be working with 'most recent 30 days API'. But the same principles apply to all our API php files, on our github.

Index of /swe project

Name	Last modified	Size	Description
Parent Directory		-	
SQL/	2019-11-08 23:16	-	
company_info.html	2019-10-30 21:09	1.9K	
company_info_API.php	2019-11-05 19:52	1.0K	
custom_dates.html	2019-10-30 21:09	2.2K	
custom_dates_API.php	2019-11-07 16:03	5.9K	
logo.html	2019-10-30 22:36	1.9K	
logo_API.php	2019-11-05 19:48	1.1K	
mysqli_connect.php	2019-10-25 21:39	348	

SWE Project   Home   30 Days   Custom Dates   Company Info   News   Logos   About

Company's most recent 30 trading days

Information, such as date,open, volume change percentage, etc are displayed.

Enter either: aapl (for Apple), googl (for Google), or rfem (for Riverfront)

company: aapl

submit your information

{ "row_ID": "1268", "date": "2019-11-07", "close": "258.74", "open": "259.43", "high": "260.35", "low": "258.11", "volume": "23735083", "change": 18.24", "volumeChange": "4768959", "volumeChangePercent": "25.14", "volumeChangeOverTime": "-6.41" }, { "row_ID": "1267", "date": "2019-11 257.49", "low": "255.37", "volume": "18966124", "change": "-0.28", "change_percent": "-0.11", "ChangeOverTime": "17.34", "volumeChange": "-100 volumeChangeOverTime": "-25.22" }, { "row_ID": "1266", "date": "2019-11-05", "close": "257.05", "open": "257.13", "high": "258.19", "low": "256. change_percent": "-0.11", "ChangeOverTime": "17.47", "volumeChange": "-5843525", "volumeChangePercent": "-22.63", "volumeChangeOverTime close": "257.33", "open": "257.5", "high": "257.85", "low": "255.38", "volume": "25817952", "change": "7.79", "change_percent": "3.12", "ChangeOv volumeChangePercent": "-31.66", "volumeChangeOverTime": "1.80" }, { "row_ID": "1264", "date": "2019-11-01", "close": "249.54", "open": "255.8 87781334", "change": "2.30", "change_percent": "0.93", "ChangeOverTime": "14.04", "volumeChange": "2990814", "volumeChangePercent": "8.60" 1263", "date": "2019-10-31", "close": "247.24", "open": "248.76", "high": "249.17", "low": "237.26", "volume": "34790520", "change": "2.48", "chan volumeChange": "3659998", "volumeChangePercent": "11.76", "volumeChangeOverTime": "37.18" }, { "row_ID": "1262", "date": "2019-10-30", "cl ow": "241.21", "volume": "31130522", "change": "-4.21", "change_percent": "-1.69", "ChangeOverTime": "11.85", "volumeChange": "-4579345", "v volumeChangeOverTime": "22.75" }, { "row_ID": "1261", "date": "2019-10-29", "close": "248.97", "open": "243.29", "high": "249.75", "low": "242. change_percent": "0.63", "ChangeOverTime": "13.78", "volumeChange": "11566626", "volumeChangePercent": "47.91", "volumeChangeOverTime": close": "247.42", "open": "249.05", "high": "249.25", "low": "246.72", "volume": "24143241", "change": "4.26", "change_percent": "1.75", "ChangeC volumeChangePercent": "31.43", "volumeChangeOverTime": "-4.80" }, { "row_ID": "1259", "date": "2019-10-25", "close": "243.16", "open": "246.5 18369296", "change": "-0.42", "change_percent": "-0.17", "ChangeOverTime": "11.12", "volumeChange": "453041", "volumeChangePercent": "2.53" 1258", "date": "2019-10-24", "close": "243.58", "open": "244.51", "high": "244.8", "low": "241.81", "volume": "17916255", "change": "0.40", "chang volumeChange": "-2016290", "volumeChangePercent": "-10.12", "volumeChangeOverTime": "-29.36" }, { "row_ID": "1257", "date": "2019-10-23", ow": "241.22", "volume": "19932545", "change": "3.22", "change_percent": "1.34", "ChangeOverTime": "11.13", "volumeChange": "-2751456", "vo volumeChangeOverTime": "-21.41" }, { "row_ID": "1256", "date": "2019-10-22", "close": "239.96", "open": "241.16", "high": "242.2", "low": "239.6

Pay attention to URL. We will copy and paste same URL into POSTMAN, and use GET option.
The output right now is JSON.

[{ "row_ID": "1268", "date": "2019-11-07", "close": "258.74", "open": "259.43", "high": "260.35", "low": "258.11", "volume": "23735083"
"18.24", "volumeChange": "4768959", "volumeChangePercent": "25.14", "volumeChangeOverTime": "-6.41" }, { "row_ID": "1267", "date'
"257.49", "low": "255.37", "volume": "18966124", "change": "-0.28", "change_percent": "-0.11", "ChangeOverTime": "17.34", "volumeCha
"volumeChangeOverTime": "-25.22" }, { "row_ID": "1266", "date": "2019-11-05", "close": "257.05", "open": "257.13", "high": "258.19", "1
"change_percent": "-0.11", "ChangeOverTime": "17.47", "volumeChange": "-5843525", "volumeChangePercent": "-22.63", "volumeChange

No Environment ▼ 👁 ⚙

Untitled Request

| GET ▼ | Enter request URL | **Send** ▼ | Save ▼ |

Params Authorization Headers Body Pre-request Script Tests Settings

Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	•••	Bulk Edit
Key	Value	Description		

Response

Pasting the URL generates the key fields

- Click send
- Wait
- Switch the option to JSON (drop down is near visualize beta button)
- And there you have it, a working RESTful API that supports the GET method.
- The output is in pretty JSON
- Any questions or comments, feel free to ask me in class or email
- --Andrew Nashid
- nashidan@kean.edu

No Environment ▼ ◉

Untitled Request

Comments (0

| GET ▼ | http://localhost/swe%20project/thirty_days_API.php?_company=aapl&submit=submit+your+information | **Send** ▼ | Save ▼ |

Params ●   Authorization   Headers (7)   Body   Pre-request Script   Tests   Settings   Cookies   Co

Query Params

	KEY	VALUE	DESCRIPTION	••• Bulk Ed
☑	_company	aapl		
☑	submit	submit+your+information		
	Key	Value	Description	

Body   Cookies   Headers (7)   Test Results

Status: 200 OK   Time: 278ms   Size: 12.38 KB   Save Response

Pretty   Raw   Preview   Visualize BETA   JSON ▼   ⇥

```
 1  [
 2      {
 3          "row_ID": "1268",
 4          "date": "2019-11-07",
 5          "close": "258.74",
 6          "open": "259.43",
 7          "high": "260.35",
 8          "low": "258.11",
 9          "volume": "23735083",
10          "change": "1.97",
```

Fin

Questions? Comments?