**Upgrade Plan Web Tool**

**Shad (Shadman) Ahmed**

**Co-Op**

**Technical Implementation Management**

**Spring 2020**



Table of Contents

[1 Overview 3](#_Toc38885578)

[1.1 About this Tool 3](#_Toc38885579)

[1.2 Source Code 3](#_Toc38885580)

[2 Code Reference Documentation 4](#_Toc38885581)

[2.1 User Flow Diagram 4](#_Toc38885582)

[2.2 Database Connection Files and Code Dependencies 4](#_Toc38885583)

[2.2.1 Client-Side and Server-Side Dependencies 4](#_Toc38885584)

[2.2.2 Session Files 7](#_Toc38885585)

[2.2.3 Queries Used 7](#_Toc38885586)

[2.2.4 Code Comments 10](#_Toc38885587)

[2.3 Excel Files and Code Dependencies 10](#_Toc38885588)

[2.3.1 Client-Side and Server-Side Dependencies 10](#_Toc38885589)

[2.3.2 Session Files 12](#_Toc38885590)

[2.3.3 Code Comments 12](#_Toc38885591)

[3 Error Code Documentation 13](#_Toc38885592)

[3.1 Database Connection Scripts Error Code and Possible Solutions 13](#_Toc38885593)

[3.2 Excel Scripts Error Code and Possible Solutions 13](#_Toc38885594)

[4 Setup Apache Server Settings, Python Libraries and Windows Drivers 14](#_Toc38885595)

[4.1 Apache Server Settings 14](#_Toc38885596)

[4.2 Python Libraries and Windows Drivers 15](#_Toc38885597)

# 1 Overview

## 1.1 About this Tool

The purpose of this tool is to help you create your firmware upgrade plan for your customer using this automated tool. The tool runs on WAMP stack server that uses JavaScript and Python to take user input to make guidelines and tables for an Upgrade Plan.

The tool allows a database connection to Oracle and MSSQL to handle larger customer data and offers a quicker execution time. Using a database connection allows you to filter for certain Firmware, if you do not want all firmware types into account. All queries used to fill a report can be found later in this documentation.

The tool also allows Excel uploads (up to 1 GB) to handle reporting. For larger datasets, you can expect longer execution times due Read/Write and File IO times of Excel documents. For any errors in Oracle DB or MSSQL, excel uploads provides an alternative to creating reports.

## 1.2 Source Code

The source code for this project is hosted on GitHub. GitHub allows for collaboration, version control and hosting.

The GitHub source code allows you to clone and change anything you would like to this tool. The GitHub Repo includes all Python, HTML, PHP and Bash Scripts used. Please note that the Apache webserver config files are not included.

The GitHub source code can be found here.

*\*Note that some files on the Repo are removed/redacted for confidential information. This tool is for internal L+G usage.*

# 2 Code Reference Documentation

## 2.1 User Flow Diagram

A close up of a map

Description automatically generatedThe user flow diagram below shows the breakdown of the expected user usage of this web tool. The diagram also shows the breakdown of all the files are dependent upon certain webpages which all leads to the report page. More information regarding code and file dependencies can be found below.

Figure 1. User Flow Diagram of Upgrade Plan Tool

## 2.2 Database Connection Files and Code Dependencies

One of my main goals in making this tool was to keep everything as modular as possible. Keeping code that way allowed me to pinpoint errors and issues much more easily and apply bug fixes.

The tool is essentially broken into two large chunks: Database Connections and Excel Uploads. Below is information regarding the Database portion.

### 2.2.1 Client-Side and Server-Side Dependencies

Once the user navigates to the ‘Build a Plan’ page, the user is required to select either an Oracle or MSSQL database connection.

If a user picks Oracle, all fields in the form must be filled for a valid connection to occur. However, if a user chooses MSSQL only the database hostname (URL) and database name is required as authentication to L+G internal MSSQL servers is already set up in the back-end python files.

The user is then able to test the connection before proceeding to see if the information on the form is valid. The files can be seen here:

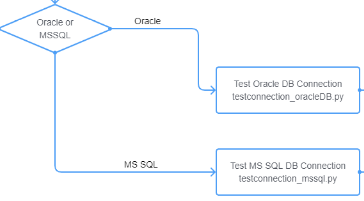


Figure 2. Oracle and MSSQL Authentication File Dependencies

Based on the fields selected, jQuery AJAX call is sent to the server which is handled by the **db\_handler.php** file. **All python files are not directly called from client-side**, instead a HTTP POST request to the server is sent with information on what is needed is called and only then **db\_handler.php** calls the appropriate python file with command line arguments to compute the request. This php files uses the Common Gateway Interface (CGI) to call python.exe on python scripts. When the ‘Test Connection’ button is selected and based on filed inputs, one of these python files will test connections to the appropriate database provided.

If authentication succeeded, then the user may click ‘Submit’, which will take them to **db\_inforform.php** which is page that allows the user to input information that is required for generating the plan. The information form page has a dependency for creating the ‘Target Firmware’ table. The files can be seen here:

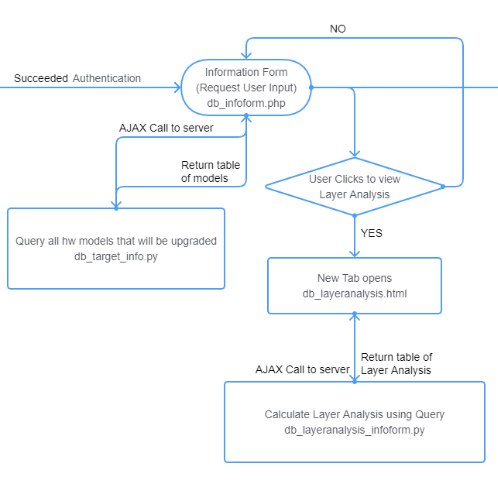


Figure 3. Oracle and MSSQL: Information Form Page Dependencies

The file **db\_target\_info.py** generates the table that allows the user to fill out the ‘Target Firmware’ they want the Customers hardware to be upgraded to. The information form page also allows the user to view the layer analysis for the customer so the user can make an informative decision on the number of hops and TTL for the Firmware Broadcast. Figure 3 shows that if the user clicks to view the Layer Analysis, a new tab will open to the **db\_layeranalysis\_infoform.html** which then (using jQuery AJAX) call the python file to compute and generate the Layer Analysis table.

Once all the data is entered as needed in the information form page, the user can click the ‘Generate Plan’ button which will load in the **db\_report.php** file. The Report Page takes in the data field filled out previously and calls numerous python files to populate the report. The files can be seen here:

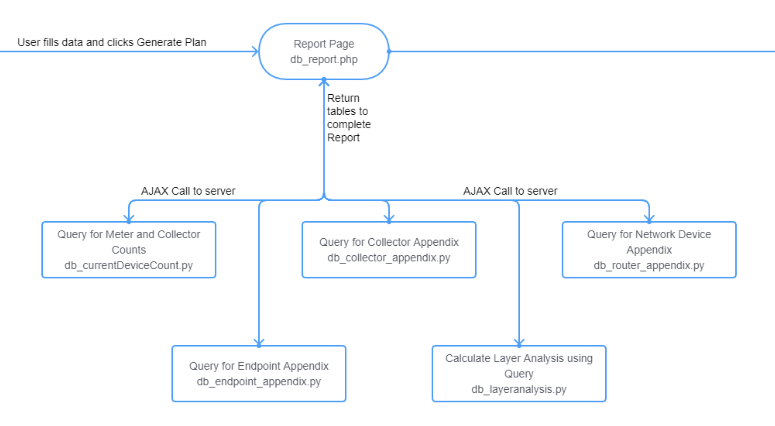


Figure 4. Oracle and MSSQL: Report Page Dependencies

The file **db\_currentDeviceCount.py** generates the table that include the Hardware counts of the all the Collectors and Meters that the customer has in the database. These **counts are filtered for models in Normal, Installed and Configure** statuses. The file **db\_layeranalysis.py** computes and generates the layer analysis table as mentioned above. The files **db\_collector\_appendix.py, db\_endpoint\_appendix.py and db\_router\_appendix.py** all generate the tables that are added to the appendix of the Upgrade Report. These files generate in-depth information on Hardware Model, Firmware Versions, DCW Versions and counts for all endpoints in the field. All the data **are filtered for models in Normal, Installed and Configure** statuses.

Once everything is loaded, the report can be easily copied using ‘Copy’ button found on the top of the page. This then can be pasted into the L+G template for Firmware Upgrades which is conveniently downloadable from this page. A simple paste and a refresh to the Table of Contents in docx file and Firmware Upgrade Plan should be ready!

### 2.2.2 Session Files

PHP allows one to create session id’s for users creating a report. In order to reduce execution time by not having to loop through large queries multiple times, some python scripts create session files that hold information used in later scripts. These files are stored in the **session\_files** folder in the website path. More information on the session files can be found in the comments of the python scripts.

### 2.2.3 Queries Used

**- Query For Layer Analysis:**

**INPUT:**

* User will provide what firmware versions to look for, based on that we query for endpoints only in that version.

**QUERY INFO:**

* Need the \*\*\*\* that has all the endpoints. -- can filter here against 'firmwareversion'
* Need \*\*\*\*table to find the layer. -- the column layer
* Need to Filter to Status Code for Normal, Discovered and Installed where codes are 8,14,13,28,31,33,34,50,72

**QUERY:**

SELECT \*\*\*\*, \*\*\*\*

FROM \*\*\*\*

\*\*\*\*ON \*\*\*\*= \*\*\*\*

WHERE \*\*\*\*LIKE '%FormInputHere%'

AND \*\*\*\*in ('8','14','13','28','31','33','34','50','72')

AND \*\*\*\*not in ('65560','65579','65569')

**- Query for Target Info:**

**INPUT:**

* User provided input, that is part of the SESSION global array in php.

**QUERY INFO:**

* Need the \*\*\*\* table that has \*\*\*\*
* Need the \*\*\*\* table that has hwmodelid which realtes to \*\*\*\* above
* Filter for fw verison and status code for Normal, Discovered and Installed where codes are 8,14,13,28,31,33,34,50,72

**QUERY:**

SELECT DISTINCT \*\*\*\*.name AS 'Hardware Model'

FROM \*\*\*\*

JOIN \*\*\*\* ON \*\*\*\*= \*\*\*\*

WHERE \*\*\*\* LIKE '%FormInputHere%'

AND \*\*\*\* in ('8','14','13','28','31','33','34','50','72')

AND \*\*\*\* not in ('65560','65579','65569')

**- Query For Network Devices:**

**INPUT:**

* User provided input, that is part of the SESSION global array in php.

**QUERY INFO:**

* Need \*\*\*\* table to get the name of the network device.
* Network devices have the \*\*\*\* of '65550','65558'
* Endpoints have the firmware while \*\*\*\* have the dcw
* Filter for status code of normal, installed and discovered

**QUERY:**

SELECT \*\*\*\*.name, \*\*\*\*, \*\*\*\*

FROM \*\*\*\*

JOIN \*\*\*\* ON \*\*\*\* = \*\*\*\*

JOIN \*\*\*\* ON \*\*\*\*= \*\*\*\*

WHERE \*\*\*\* in ('65550','65558')

AND \*\*\*\*in ('8','14','13','28','31','33','34','50','72')

**- Query For Layer Analysis and Endpoint Appendix:**

**INPUT:**

* User provided input, that is part of the SESSION global array in php

**QUERY INFO:**

* Need the layer, hardware model, firmware version, dcw version, meter firmware version
* Layer, dcw version can be found in the \*\*\*\*
* Firmware version can be found in \*\*\*\*
* Hardware model \*\*\*\*
* Meter firmware version can be found in the meter table, join on meterid

**QUERY:**

SELECT \*\*\*\* AS "Meter" \*\*\*\* AS "Hardware Model", \*\*\*\* AS "Firmware Version", \*\*\*\*, \*\*\*\*, \*\*\*\*

FROM \*\*\*\*

JOIN \*\*\*\* ON \*\*\*\*= \*\*\*\*

JOIN \*\*\*\* ON \*\*\*\*= \*\*\*\*

JOIN \*\*\*\* ON \*\*\*\*= \*\*\*\*

WHERE \*\*\*\*LIKE '%INSERT FORM INPUT HERE%'

AND \*\*\*\*in ('8','14','13','28','31','33','34','50','72')

AND \*\*\*\*not in ('65560','65579','65569')

**Query for meter count:**

**INPUT:**

* User provided input, that is part of the SESSION global array in php

**QUERY INFO:**

* Need the \*\*\*\* to return the hardware types
* And need the \*\*\*\* to get counts

**QUERY:**

SELECT distinct (\*\*\*\*.name) AS "model", count (\*\*\*\*.endpointid) AS "meter"

FROM \*\*\*\*

JOIN endpoints \*\*\*\*= \*\*\*\*

WHERE \*\*\*\*LIKE '%6.17%'

AND \*\*\*\* in ('8','14','13','28','31','33','34','50','72')

AND \*\*\*\* not in ('65560','65579','65569')

group by endpointModels.name

### 2.2.4 Code Comments

All comments in the code can be found in its respective files.

## 2.3 Excel Files and Code Dependencies

The tool is essentially broken into two large chunks: Database Connections and Excel Uploads. Below is information regarding the Excel portion.

### 2.3.1 Client-Side and Server-Side Dependencies

Once the user navigates to the ‘Build a Plan’ page, the user can upload three excel (xlsx) files: Endpoint Extract, Collector Extract and Network Extract. These files can be uploaded by downloading them from **Command Center > Reporting > Endpoint Data Export** and picking the appropriate options to download the three extracts.

Once the user uploads the files and clicks ‘Upload Data’, the page will be redirected to **infoform.php.** This PHP file contains error checking to ensure that the files uploaded are appropriate. The files can be seen here:

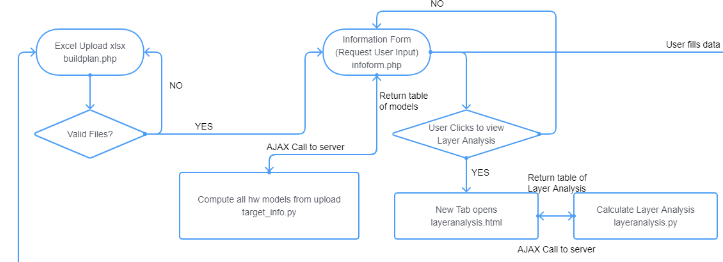


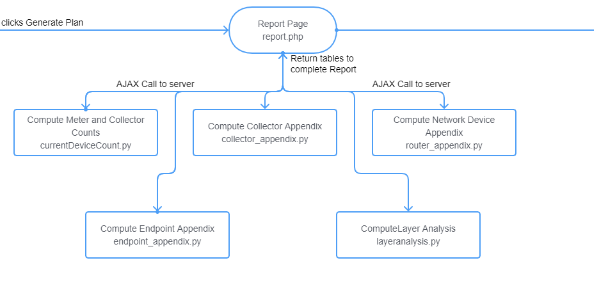
Figure 5. Excel Upload: Information Page Dependencies

Based on the fields selected, jQuery AJAX call is sent to the server which is handled by the **python\_handler.php** file. **All python files are not directly called from client-side**, instead a HTTP POST request to the server is sent with information on what is needed is called and only then **python\_handler.php** calls the appropriate python file with command line arguments to compute the request. This php files uses the Common Gateway Interface (CGI) to call python.exe on python scripts. When the ‘Test Connection’ button is selected and based on filed inputs, one of these python files will test connections to the appropriate database provided.

If file validation succeeded, then **nforform.php** page allows the user to input information that is required for generating the plan. The information form page has a dependency for creating the ‘Target Firmware’ table. The files can be seen in Figure 5.

The file **target\_info.py** generates the table that allows the user to fill out the ‘Target Firmware’ they want the Customers hardware to be upgraded to. The information form page also allows the user to view the layer analysis for the customer so the user can make an informative decision on the number of hops and TTL for the Firmware Broadcast. Figure 5 shows that if the user clicks to view the Layer Analysis, a new tab will open to the **layeranalysis.html** which then (using jQuery AJAX) call the python file to compute and generate the Layer Analysis table.

Once all the data is entered as needed in the information form page, the user can click the ‘Generate Plan’ button which will load in the **report.php** file. The Report Page takes in the data field filled out previously and calls numerous python files to populate the report. The files can be seen here:

Figure 6. Excel Upload: Report Page Dependencies

The file **currentDeviceCount.py** generates the table that include the Hardware counts of the all the Collectors and Meters that the customer has in the Excel File. These **counts are filtered for models in Normal, Installed and Configure** statuses. The file **layeranalysis.py** computes and generates the layer analysis table as mentioned above. The files **collector\_appendix.py, endpoint\_appendix.py and router\_appendix.py** all generate the tables that are added to the appendix of the Upgrade Report. These files generate in-depth information on Hardware Model, Firmware Versions, DCW Versions and counts for all endpoints in the field. All the data **are filtered for models in Normal, Installed and Configure** statuses from the Excel Files.

Once everything is loaded, the report can be easily copied using ‘Copy’ button found on the top of the page. This then can be pasted into the L+G template for Firmware Upgrades which is conveniently downloadable from this page. A simple paste and a refresh to the Table of Contents in docx file and Firmware Upgrade Plan should be ready!

### 2.3.2 Session Files

PHP allows one to create session id’s for users creating a report. In order to reduce execution time by not having to loop through large queries multiple times, some python scripts create session files that hold information used in later scripts. These files are stored in the **session\_files** folder in the website path. More information on the session files can be found in the comments of the python scripts.

### 2.3.3 Code Comments

All comments in the code can be found in its respective files.

# 3 Error Code Documentation

**For all other unexpected behavior and errors that are not listed below, check the Apache error log and the PHP error log.**

### 3.1 Database Connection Scripts Error Code and Possible Solutions

* ERROR CODE 101: (Common Ones Below)
  + **DPI-1047: Cannot locate a 64-bit Oracle Client library:** This error arises if the Oracle Drivers for windows are not downloaded and installed with the python library. More information on that is found is section 4 of this documentation. Another possible reason for this is windows [rights issue](https://stackoverflow.com/questions/17830276/run-windows-command-in-php). After starting WAMP, you may need to change the **Log On** rights to the wampapache64 service. To do this: Press the windows button > type services and open it > scroll to **wampapache64** > right click and open **Properties** > click the **Log On** tab > change to **this account and enter the AM\ username used to sign into this machine**. If migrated to a server, this issue may or may not arise and further investigation may be needed.
  + **ORA-12545: Connect failed because target host or object does not exist:** This could be a VPN issue. Ensure you are connected to the appropriate VPN to access the DB.
  + **All other errors under this code should return the exception** that has occurred like connection errors, DB not found etc. These can be solved by rechecking the entered information or making sure you are connected to the appropriate VPN requirement.
* ERROR CODE 102:
  + This error code pertains python execution error. The error code should return an exception to the issue that occurred. Errors could occur due to mismatch of session ids expiring. In this case it is recommended to open a new window and restarting making the plan.
* ­Other Errors that may arise:
  + Other errors that may arise on the server side should return an exception that will show where the issue has raised.

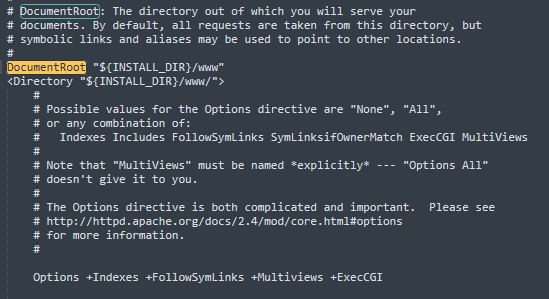
### 3.2 Excel Scripts Error Code and Possible Solutions

* ERROR CODE 201:
  + This error drives from a failure in the excel upload. The ISSET check failure could arise if the one of the files uploaded is not an Excel file (xlsx). A POST SET check failure could arise if you attempted to Upload without uploading all three documents. Try reuploading with them.
* ERROR CODE 202:
  + This error code pertains python execution error. The error code should return an exception to the issue that occurred. Errors could occur due to mismatch of session ids expiring. In this case it is recommended to open a new window and restarting making the plan.

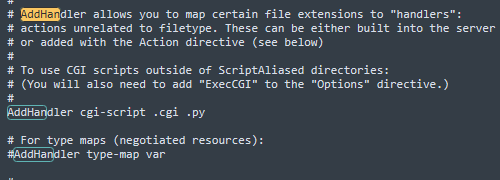
# 4 Setup Apache Server Settings, Python Libraries and Windows Drivers

### 4.1 Apache Server Settings

* Enabling CGI to run python files through PHP: Click on the WAMP icon > Hover over Apache > Click the httpd.conf file. Using the text editor, find the part of the document “DocumentRoot “${INSTALL\_DIR}/www …” > Below that there is a line starting with “Options +Indexes +FollowSymLins …”. Add to the end of that +ExecCGI:



Next using the text editor, find the part of the document “AddHandler”. There is a commented line starting with “AddHandler cgi-script .cgi”, add .py to the line and uncomment it:

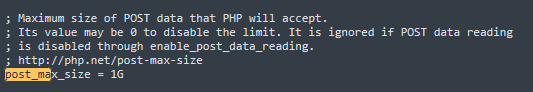


Save and exit the file.

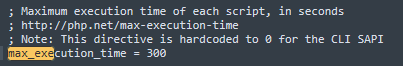
* Next setting that needs to be adjusted is the Upload Size limit. Click on the WAMP icon > Hover over PHP > Click php.ini file. Using the text editor, find the ‘upload\_max\_filesize’ and change the value to something larger. In my system I put 850M (megabytes):



Then search for ‘post\_max\_size’ and change the value to something larger. In my system I put 1G (Gig) which is enough for large excel files:



* Keeping the php.ini file open, the next setting that needs to be changed is the Execution time. Not increasing the execution time can result in php timeouts before python scripts are done running for larger excel data sets. Using the text editor, find the ‘max\_execution\_time’ and change the value to something larger. In my system I put 300 (seconds) which is enough:



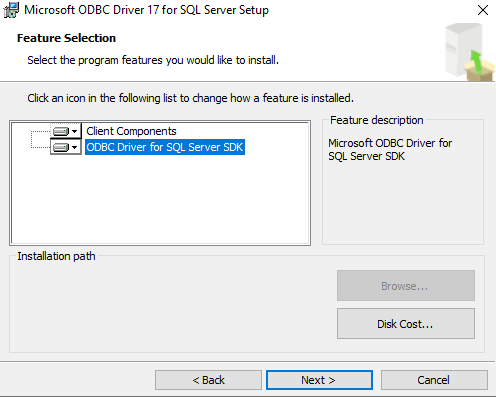
* The last setting that needs to be added is a windows scheduler for cleaning up uploaded and session files. The windows **batch file** is added in the repo that can be added to a windows scheduler to be run bi-weekly or monthly for cleanup. A tutorial on how to add a windows scheduler can be found [here](https://www.youtube.com/watch?v=JD865wnXGQ8).

### 4.2 Python Libraries and Windows Drivers

* To install Python, go to <https://www.python.org/downloads/>. The version that I used was 3.5.2. Make sure to **add to PATH** when prompted in the python installer to do so.
* Once Python is installed, open a windows command prompt and utilize pip package installer to download the necessary python libraries.
* Run the commands below:
  + ***pip install cx-Oracle***
  + ***pip install pyodbc***
  + ***pip install pandas***
  + ***pip install numpy***
  + ***pip install xlrd***
  + ***pip install xlwt***

In some cases, you may receive a ‘ssl certificate’ error when running these pip commands. In that case simply replace the command with this:

* ***pip install --trusted-host=pypi.org --trusted-host=files.pythonhosted.org --user PackageNameHere***
* Once all the python packages are downloaded, we need to download the Windows drivers for cx-Oracle and pyodbc libraries.
  + For cx-Oracle, we need to download instant-client for Windows and add it to our PATH. The link to instant client can be found [here](https://www.oracle.com/database/technologies/instant-client/winx64-64-downloads.html). Download the Basic Package. Once downloaded, unzip the file and place it in the Windows C folder (C:/instantclient\_##\_#). Then, press the windows button and type PATH and select ‘**Edit System Environment Variables’ > click Environment Variables > Select on Path and then click edit > click new and type the path of where you placed the instantclient (C:/instantclient\_##\_#) >** Click Ok for the rest of the windows. If there are issues with downloading Oracle, check this github.com [link](https://github.com/oracle/python-cx_Oracle/issues/362#issuecomment-578547430).
  + For pyodbc, we need the official windows drivers from Microsoft. Follow the installation guide provided by Microsoft [here](https://docs.microsoft.com/en-us/sql/connect/python/pyodbc/step-1-configure-development-environment-for-pyodbc-python-development?view=sql-server-ver15) (Skip to Step 3 and download the setup for drivers). Once you have the drivers, simply follow the steps, make sure when reaching here:



Select both client component and ODBC drivers. Finish the installation. Once the Drivers are installed, we need on more additional adapter. ODBC requires a Visual Studio adapter wheel which can be found [here](https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyodbc). The one I download was ‘pyodbc-4.0.28-cp35-cp35m-win\_amd64.whl’. Once downloaded, open a windows command prompt and type *pip install PathOfWhereWheelDownloaded\wheelNameHere*. It would look like this:

pip install \Users\ahmeds\Downloads\pyodbc‑4.0.27‑cp35‑cp35m‑win\_amd64.whl

The wheel version downloaded may or may not work for your system. Try downloading one from the list that may work and running the pip command on it.

Once this is properly installed, the setup for pyodbc should be complete.

* Download the code zip from Teams or utilize git to clone the repo. Once downloaded and unzip, place the ‘upgradeplantool’ folder in **C:/wamp64/www** (Or wherever the www directory is downloaded).
* Each python file has a shebang line for the path to where the python.exe is. This line is not needed in a Windows OS and can be removed without performance issues. This was added as more for reference.
* One file **that needs** to be changed for proper execution is the **db\_handler.php and python\_handler.php.** These files have a series of function that call command line python calls to be executed. You will notice that the path of the python.exe is mentioned first, then the file name and then the command line arguments. **Please update the python.exe path to the appropriate one in your system for proper performance and ensure the path for the python call mentioned is correct.**