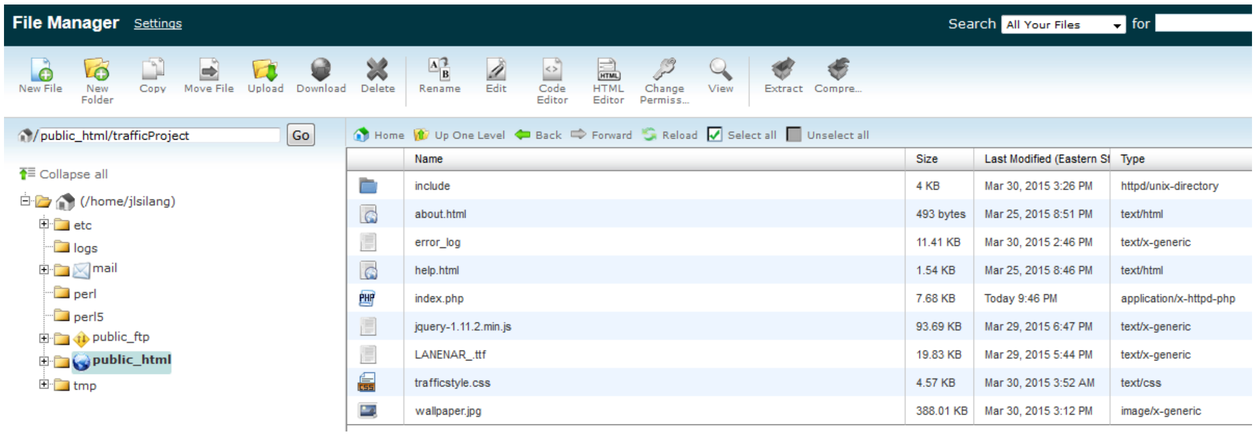
Technical Documentation
Traffic Monitoring Project
Group 13

Maxine Deines
Charu Jain
Mehul Salhotra
Akshay Sardana
Justin Silang
Jason Yang

File paths of the traffic monitoring service website. The picture below shows the path public_html/trafficProject

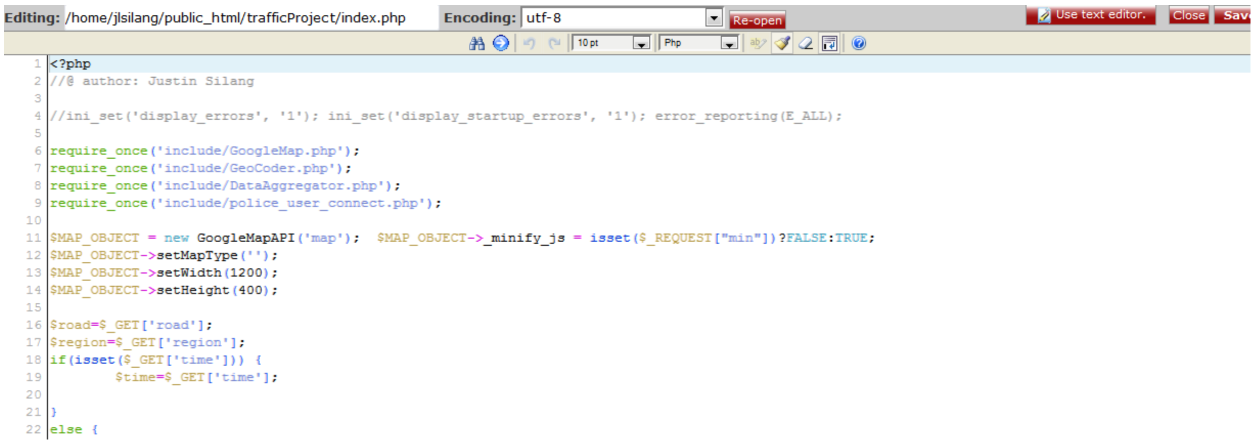


/public_html/trafficProject

(/home/jlsilang)
- etc
- logs
- mail
- perl
- perl5
- public_ftp
- public_html
- tmp

Name	Size	Last Modified (Eastern St	Type
include	4 KB	Mar 30, 2015 3:26 PM	httpd/unix-directory
about.html	493 bytes	Mar 25, 2015 8:51 PM	text/html
error_log	11.41 KB	Mar 30, 2015 2:46 PM	text/x-generic
help.html	1.54 KB	Mar 25, 2015 8:46 PM	text/html
index.php	7.68 KB	Today 9:46 PM	application/x-httpd-php
jquery-1.11.2.min.js	93.69 KB	Mar 29, 2015 6:47 PM	text/x-generic
LANENAR_.ttf	19.83 KB	Mar 29, 2015 5:44 PM	text/x-generic
trafficstyle.css	4.57 KB	Mar 30, 2015 3:52 AM	text/css
wallpaper.jpg	388.01 KB	Mar 30, 2015 3:12 PM	image/x-generic

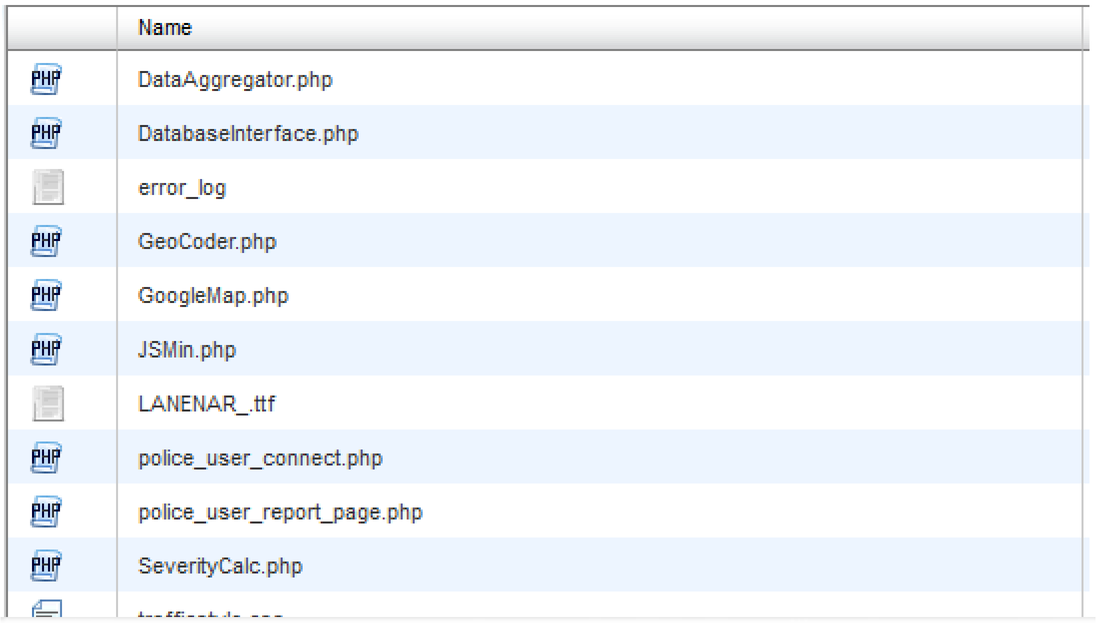
Preview of the index.php code which utilizes PHP along with our front end of our traffic service website.



Editing: /home/jlsilang/public_html/trafficProject/index.php
Encoding: utf-8
10 pt
Php

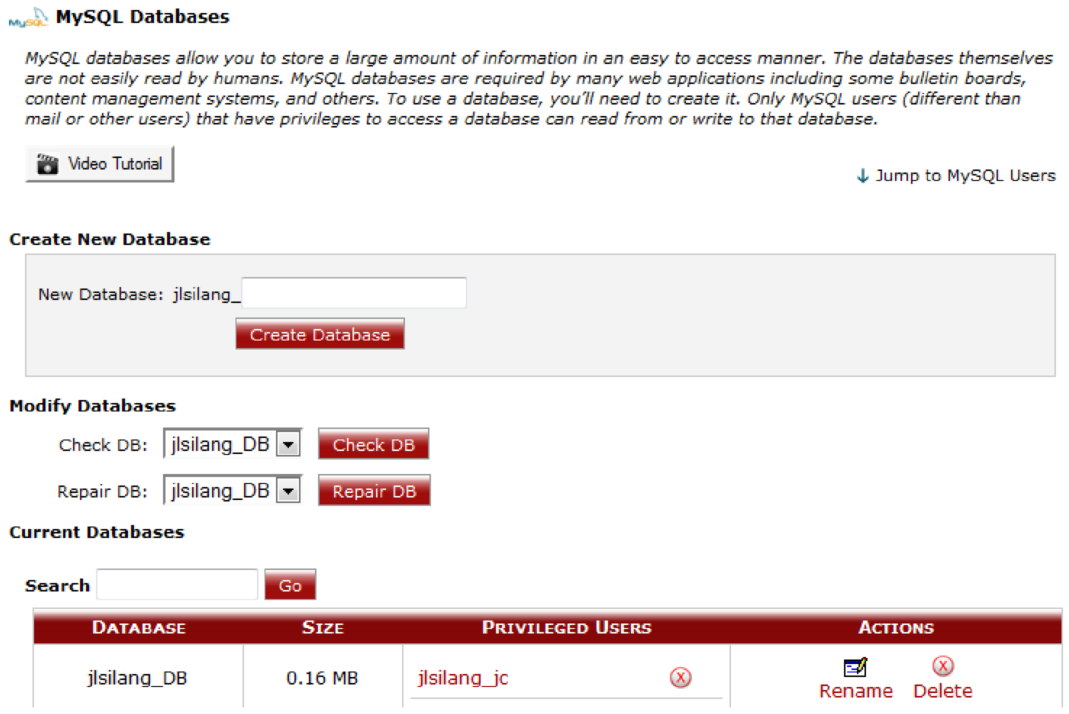
```php
<?php
//@ author: Justin Silang

//ini_set('display_errors', '1'); ini_set('display_startup_errors', '1'); error_reporting(E_ALL);

require_once('include/GoogleMap.php');
require_once('include/GeoCoder.php');
require_once('include/DataAggregator.php');
require_once('include/police_user_connect.php');

$MAP_OBJECT = new GoogleMapAPI('map');  $MAP_OBJECT->_minify_js = isset($_REQUEST["min"])?FALSE:TRUE;
$MAP_OBJECT->setMapType('');
$MAP_OBJECT->setWidth(1200);
$MAP_OBJECT->setHeight(400);

$road=$_GET['road'];
$region=$_GET['region'];
if(isset($_GET['time'])) {
        $time=$_GET['time'];

}
else {
```

This “include” folder includes our PHP code that handles incident reporting, etc.



Name
DataAggregator.php
DatabaseInterface.php
error_log
GeoCoder.php
GoogleMap.php
JSMin.php
LANENAR_.ttf
police_user_connect.php
police_user_report_page.php
SeverityCalc.php

This is the MySQL database used in the traffic website service.



MySQL Databases

MySQL databases allow you to store a large amount of information in an easy to access manner. The databases themselves are not easily read by humans. MySQL databases are required by many web applications including some bulletin boards, content management systems, and others. To use a database, you'll need to create it. Only MySQL users (different than mail or other users) that have privileges to access a database can read from or write to that database.

Video Tutorial

↓ Jump to MySQL Users

Create New Database

New Database: jlsilang_

Create Database

Modify Databases

Check DB: jlsilang_DB Check DB

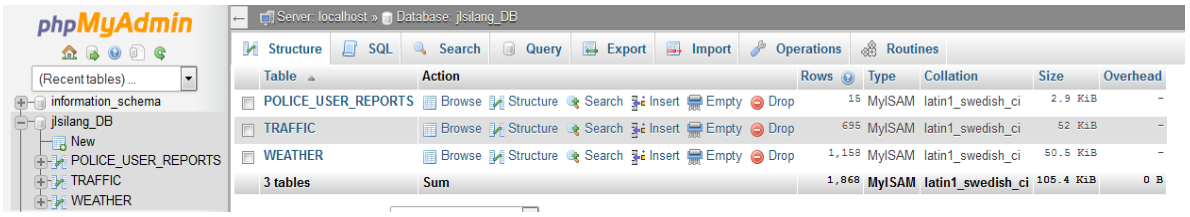
Repair DB: jlsilang_DB Repair DB

Current Databases

Search Go

Database	Size	Privileged Users	Actions
jlsilang_DB	0.16 MB	jlsilang_jc	Rename Delete

The tables of jlsilang_DB database.



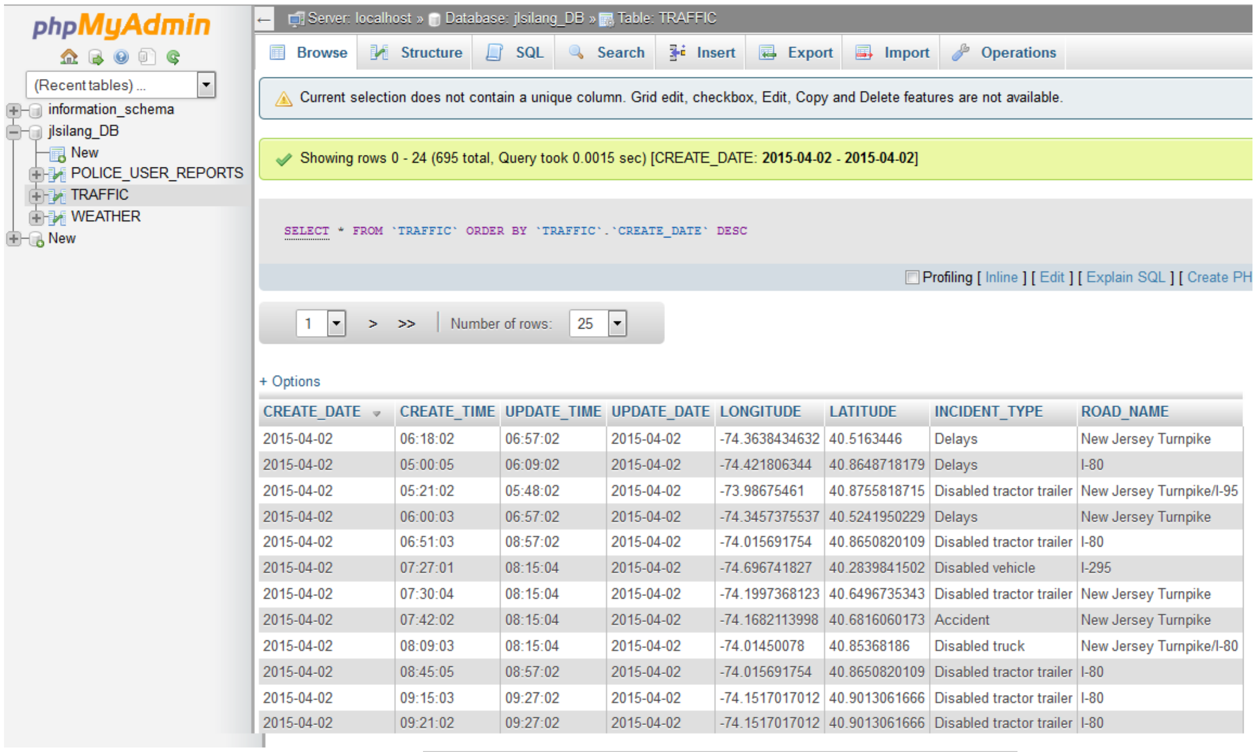
Server: localhost » Database: jlsilang_DB

Structure SQL Search Query Export Import Operations Routines

(Recent tables) ...
information_schema
jlsilang_DB
New
POLICE_USER_REPORTS
TRAFFIC
WEATHER

Table	Action	Rows	Type	Collation	Size	Overhead
POLICE_USER_REPORTS	Browse Structure Search Insert Empty Drop	15	MyISAM	latin1_swedish_ci	2.9 KiB	-
TRAFFIC	Browse Structure Search Insert Empty Drop	695	MyISAM	latin1_swedish_ci	52 KiB	-
WEATHER	Browse Structure Search Insert Empty Drop	1,158	MyISAM	latin1_swedish_ci	50.5 KiB	-
3 tables	Sum	1,868	MyISAM	latin1_swedish_ci	105.4 KiB	0 B

This is a preview of the TRAFFIC table entries. New data is stored here after extracting and parsing from various traffic data services.



Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 24 (695 total, Query took 0.0015 sec) [CREATE_DATE: **2015-04-02 - 2015-04-02**]

```
SELECT * FROM `TRAFFIC` ORDER BY `TRAFFIC`.`CREATE_DATE` DESC
```

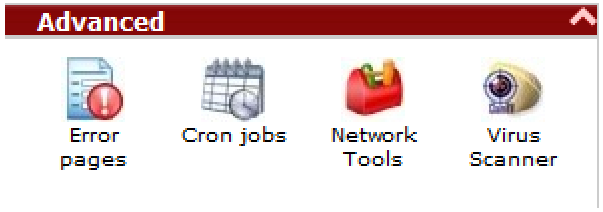
Profiling [Inline] [Edit] [Explain SQL] [Create PH

1 > >> Number of rows: 25

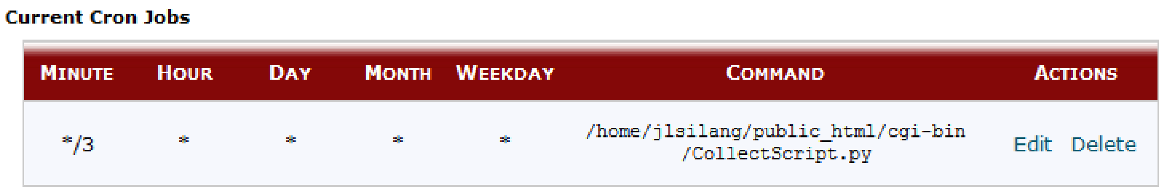
+ Options

CREATE_DATE	CREATE_TIME	UPDATE_TIME	UPDATE_DATE	LONGITUDE	LATITUDE	INCIDENT_TYPE	ROAD_NAME
2015-04-02	06:18:02	06:57:02	2015-04-02	-74.3638434632	40.5163446	Delays	New Jersey Turnpike
2015-04-02	05:00:05	06:09:02	2015-04-02	-74.421806344	40.8648718179	Delays	I-80
2015-04-02	05:21:02	05:48:02	2015-04-02	-73.98675461	40.8755818715	Disabled tractor trailer	New Jersey Turnpike/I-95
2015-04-02	06:00:03	06:57:02	2015-04-02	-74.3457375537	40.5241950229	Delays	New Jersey Turnpike
2015-04-02	06:51:03	08:57:02	2015-04-02	-74.015691754	40.8650820109	Disabled tractor trailer	I-80
2015-04-02	07:27:01	08:15:04	2015-04-02	-74.696741827	40.2839841502	Disabled vehicle	I-295
2015-04-02	07:30:04	08:15:04	2015-04-02	-74.1997368123	40.6496735343	Disabled tractor trailer	New Jersey Turnpike
2015-04-02	07:42:02	08:15:04	2015-04-02	-74.1682113998	40.6816060173	Accident	New Jersey Turnpike
2015-04-02	08:09:03	08:15:04	2015-04-02	-74.01450078	40.85368186	Disabled truck	New Jersey Turnpike/I-80
2015-04-02	08:45:05	08:57:02	2015-04-02	-74.015691754	40.8650820109	Disabled tractor trailer	I-80
2015-04-02	09:15:03	09:27:02	2015-04-02	-74.1517017012	40.9013061666	Disabled tractor trailer	I-80
2015-04-02	09:21:02	09:27:02	2015-04-02	-74.1517017012	40.9013061666	Disabled tractor trailer	I-80

The Cron jobs enables us to automatically run the collect data scripts automatically on our web server.



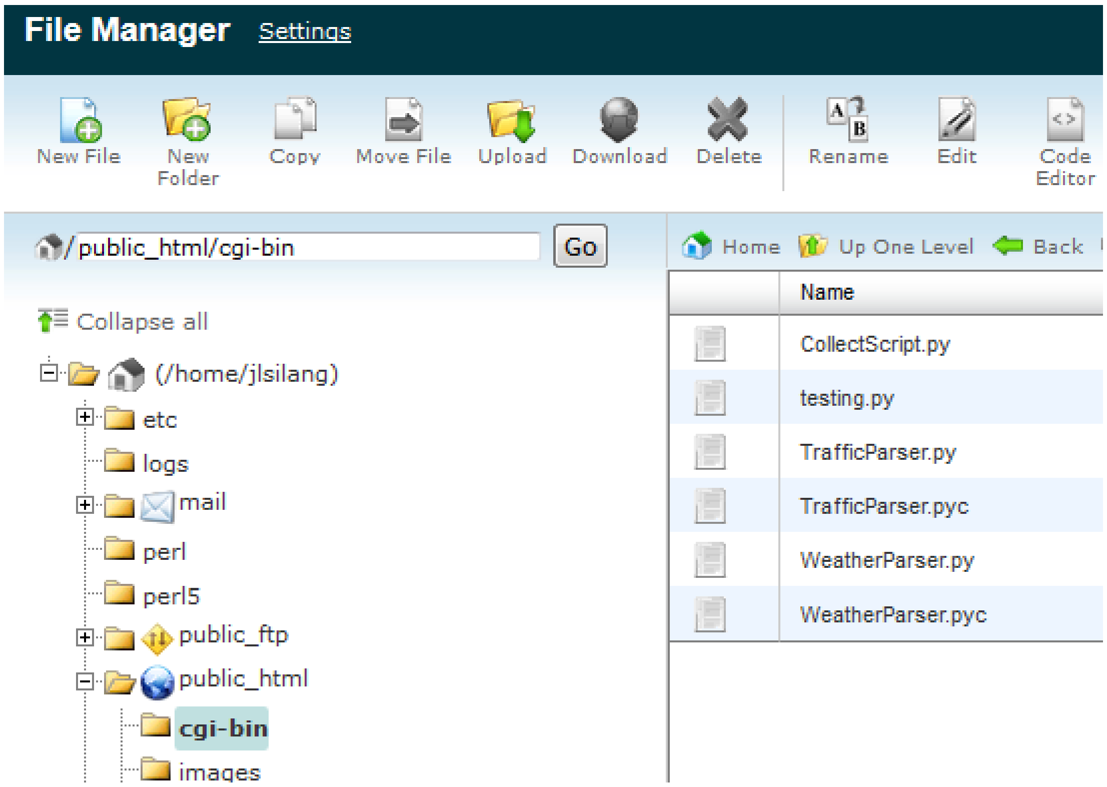
The CollectScript.py python file is run every three minutes in our Cron job.



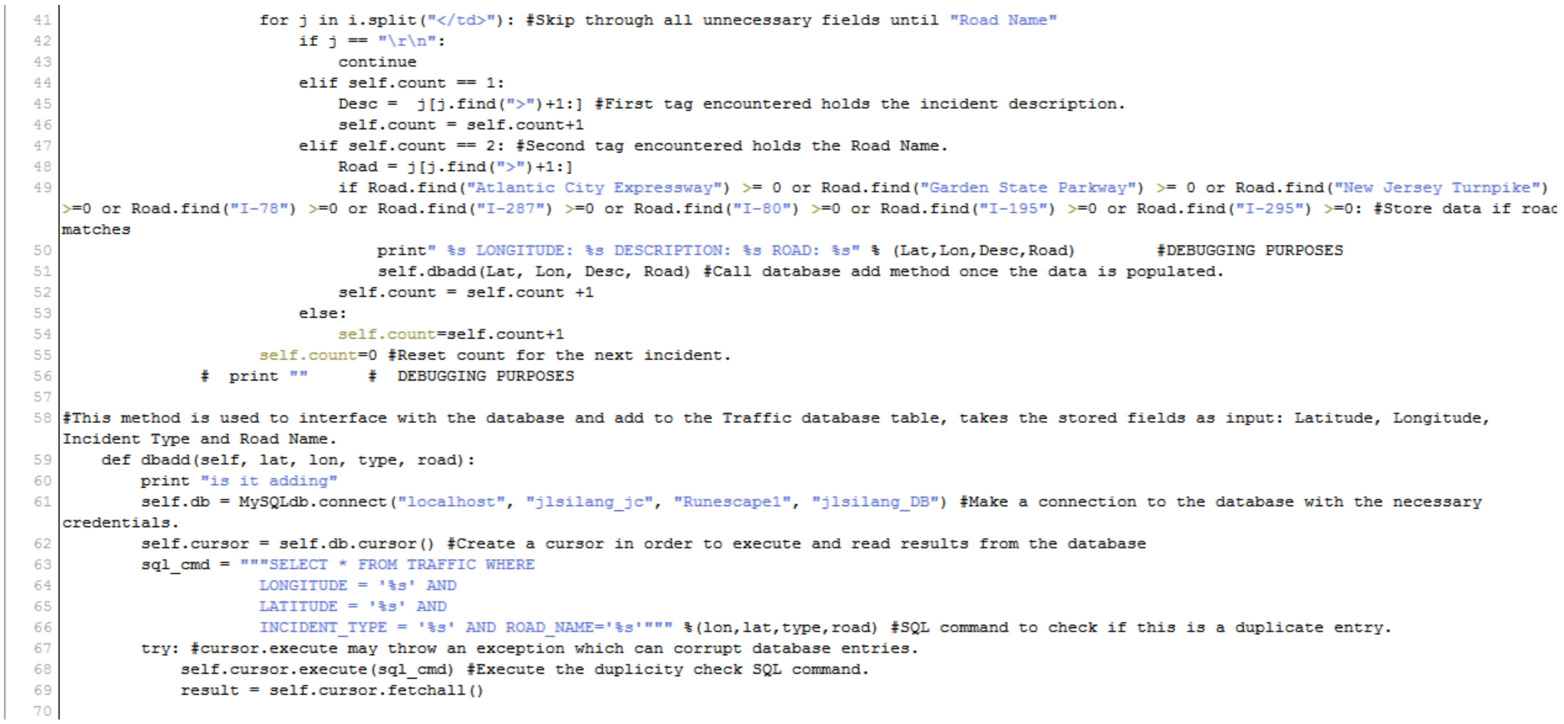
Current Cron Jobs

Minute	Hour	Day	Month	Weekday	Command	Actions
*/3	*	*	*	*	/home/jlsilang/public_html/cgi-bin /CollectScript.py	Edit Delete

The cgi-bin in public_html/cgi-bin contains all our python scripts that fetch new traffic data for our services.

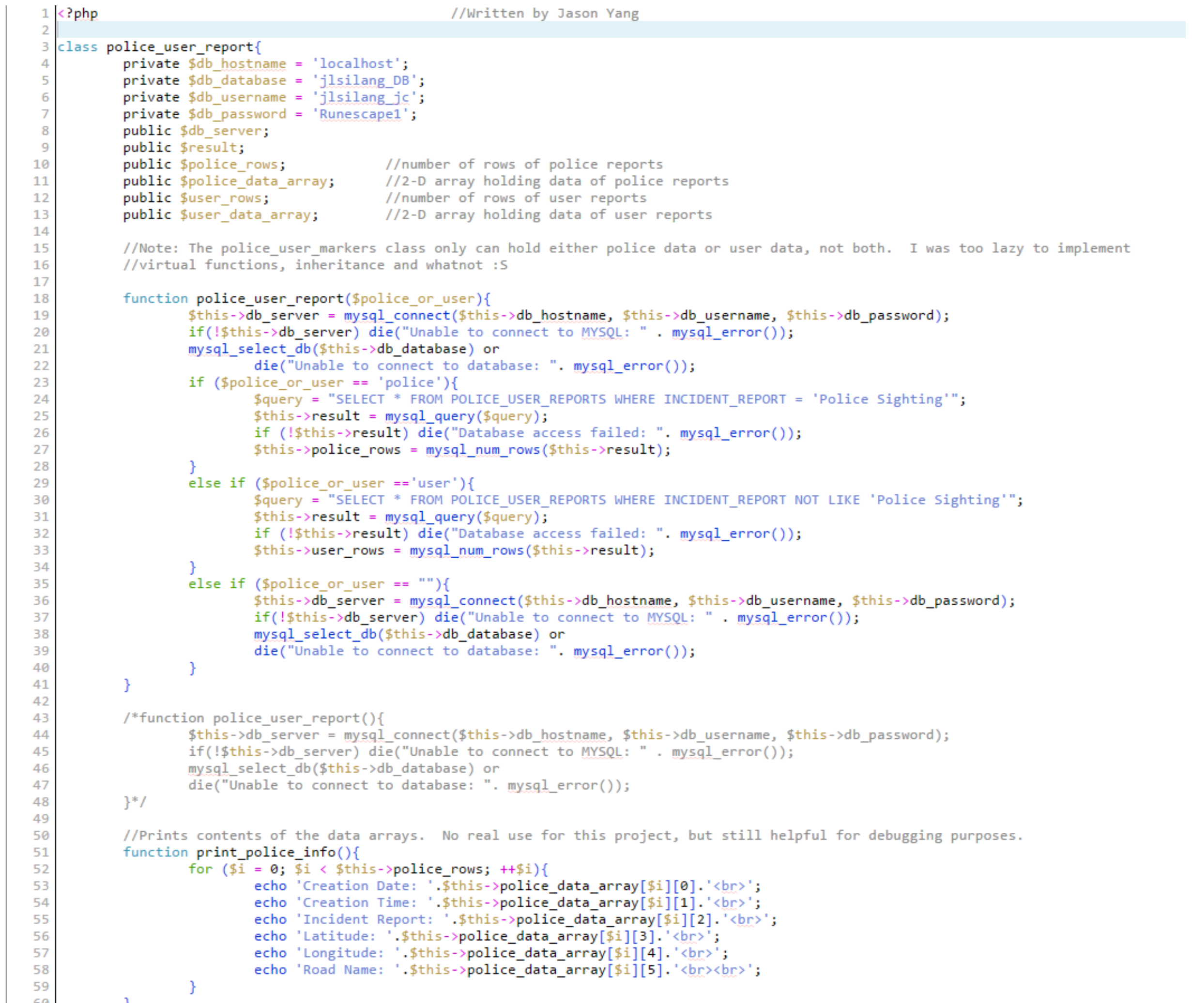


The image below is a preview of the trafficParser.py python script. If you look closely the parser searches for html elements, then if the specific road name is found then it will keep a count of it. If sufficient counts have reached, it adds it to the database with the appropriate arguments. Below you can see our database connection string and its respective credentials.



```
                    for j in i.split("</td>"): #Skip through all unnecessary fields until "Road Name"
                        if j == "\r\n":
                            continue
                        elif self.count == 1:
                            Desc =  j[j.find(">")+1:] #First tag encountered holds the incident description.
                            self.count = self.count+1
                        elif self.count == 2: #Second tag encountered holds the Road Name.
                            Road = j[j.find(">")+1:]
                            if Road.find("Atlantic City Expressway") >= 0 or Road.find("Garden State Parkway") >= 0 or Road.find("New Jersey Turnpike")
>=0 or Road.find("I-78") >=0 or Road.find("I-287") >=0 or Road.find("I-80") >=0 or Road.find("I-195") >=0 or Road.find("I-295") >=0: #Store data if roac
matches
                                print" %s LONGITUDE: %s DESCRIPTION: %s ROAD: %s" % (Lat,Lon,Desc,Road)        #DEBUGGING PURPOSES
                                self.dbadd(Lat, Lon, Desc, Road) #Call database add method once the data is populated.
                            self.count = self.count +1
                        else:
                            self.count=self.count+1
                    self.count=0 #Reset count for the next incident.
              #  print ""      #  DEBUGGING PURPOSES

#This method is used to interface with the database and add to the Traffic database table, takes the stored fields as input: Latitude, Longitude,
Incident Type and Road Name.
    def dbadd(self, lat, lon, type, road):
        print "is it adding"
        self.db = MySQLdb.connect("localhost", "jlsilang_jc", "Runescape1", "jlsilang_DB") #Make a connection to the database with the necessary
credentials.
        self.cursor = self.db.cursor() #Create a cursor in order to execute and read results from the database
        sql_cmd = """SELECT * FROM TRAFFIC WHERE
                    LONGITUDE = '%s' AND
                    LATITUDE = '%s' AND
                    INCIDENT_TYPE = '%s' AND ROAD_NAME='%s'""" %(lon,lat,type,road) #SQL command to check if this is a duplicate entry.
        try: #cursor.execute may throw an exception which can corrupt database entries.
            self.cursor.execute(sql_cmd) #Execute the duplicity check SQL command.
            result = self.cursor.fetchall()
```

Sample of police/user report interface:



```php
<?php                                           //Written by Jason Yang

class police_user_report{
        private $db_hostname = 'localhost';
        private $db_database = 'jlsilang_DB';
        private $db_username = 'jlsilang_jc';
        private $db_password = 'Runescape1';
        public $db_server;
        public $result;
        public $police_rows;            //number of rows of police reports
        public $police_data_array;      //2-D array holding data of police reports
        public $user_rows;              //number of rows of user reports
        public $user_data_array;        //2-D array holding data of user reports

        //Note: The police_user_markers class only can hold either police data or user data, not both.  I was too lazy to implement
        //virtual functions, inheritance and whatnot :S

        function police_user_report($police_or_user){
                $this->db_server = mysql_connect($this->db_hostname, $this->db_username, $this->db_password);
                if(!$this->db_server) die("Unable to connect to MYSQL: " . mysql_error());
                mysql_select_db($this->db_database) or
                        die("Unable to connect to database: ". mysql_error());
                if ($police_or_user == 'police'){
                        $query = "SELECT * FROM POLICE_USER_REPORTS WHERE INCIDENT_REPORT = 'Police Sighting'";
                        $this->result = mysql_query($query);
                        if (!$this->result) die("Database access failed: ". mysql_error());
                        $this->police_rows = mysql_num_rows($this->result);
                }
                else if ($police_or_user =='user'){
                        $query = "SELECT * FROM POLICE_USER_REPORTS WHERE INCIDENT_REPORT NOT LIKE 'Police Sighting'";
                        $this->result = mysql_query($query);
                        if (!$this->result) die("Database access failed: ". mysql_error());
                        $this->user_rows = mysql_num_rows($this->result);
                }
                else if ($police_or_user == ""){
                        $this->db_server = mysql_connect($this->db_hostname, $this->db_username, $this->db_password);
                        if(!$this->db_server) die("Unable to connect to MYSQL: " . mysql_error());
                        mysql_select_db($this->db_database) or
                        die("Unable to connect to database: ". mysql_error());
                }
        }

        /*function police_user_report(){
                $this->db_server = mysql_connect($this->db_hostname, $this->db_username, $this->db_password);
                if(!$this->db_server) die("Unable to connect to MYSQL: " . mysql_error());
                mysql_select_db($this->db_database) or
                die("Unable to connect to database: ". mysql_error());
        }*/

        //Prints contents of the data arrays.  No real use for this project, but still helpful for debugging purposes.
        function print_police_info(){
                for ($i = 0; $i < $this->police_rows; ++$i){
                        echo 'Creation Date: '.$this->police_data_array[$i][0].'<br>';
                        echo 'Creation Time: '.$this->police_data_array[$i][1].'<br>';
                        echo 'Incident Report: '.$this->police_data_array[$i][2].'<br>';
                        echo 'Latitude: '.$this->police_data_array[$i][3].'<br>';
                        echo 'Longitude: '.$this->police_data_array[$i][4].'<br>';
                        echo 'Road Name: '.$this->police_data_array[$i][5].'<br><br>';
                }
```

When the police_user_report class is created, it calls the constructor. Based on the constructor argument (either “police”, “user”, or null string), it will query the database for the requested values. Some functions of the interface include printing contents of data_array, storing a row into database, and querying database for either the police info or user info.