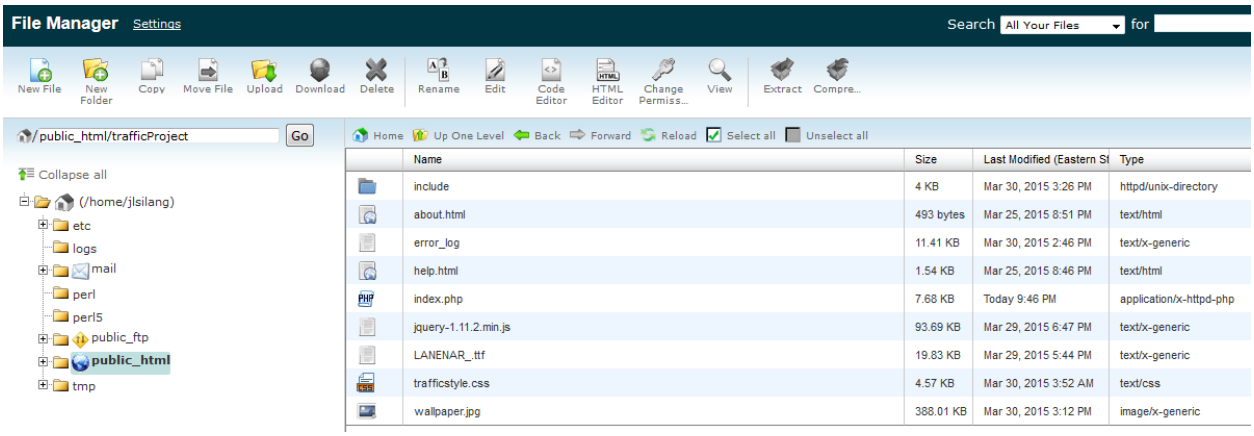
Technical Documentation
Traffic Monitoring Project
Group 13

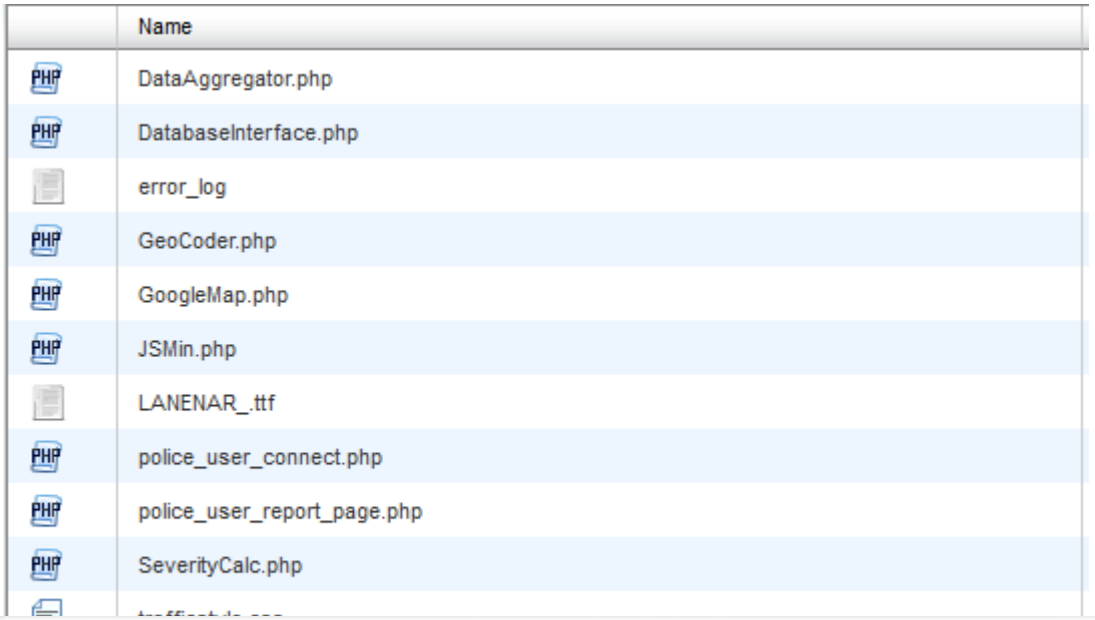
Maxine Deines
Charu Jain
Mehul Salhotra
Akshay Sardana
Justin Silang
Jason Yang

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

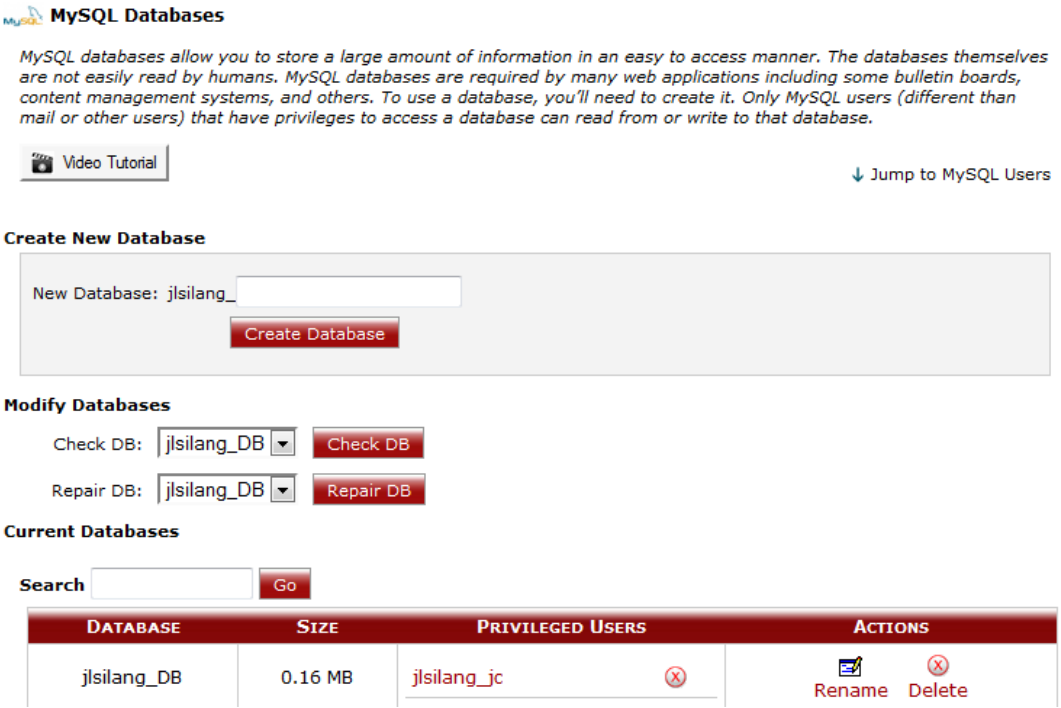


Preview of the index.php code which utilizes PHP along with our front end of our traffic service website.

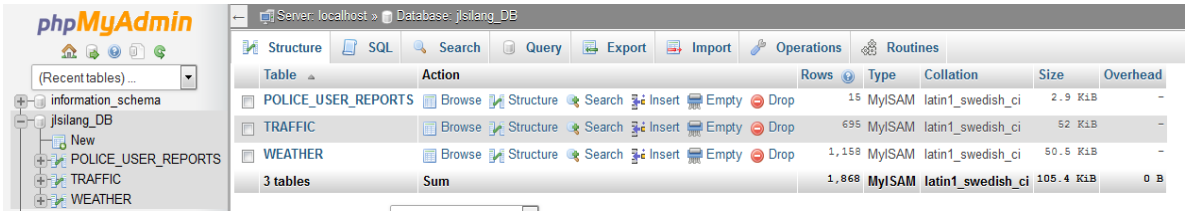
This “include” folder includes our PHP code that handles incident reporting, etc.



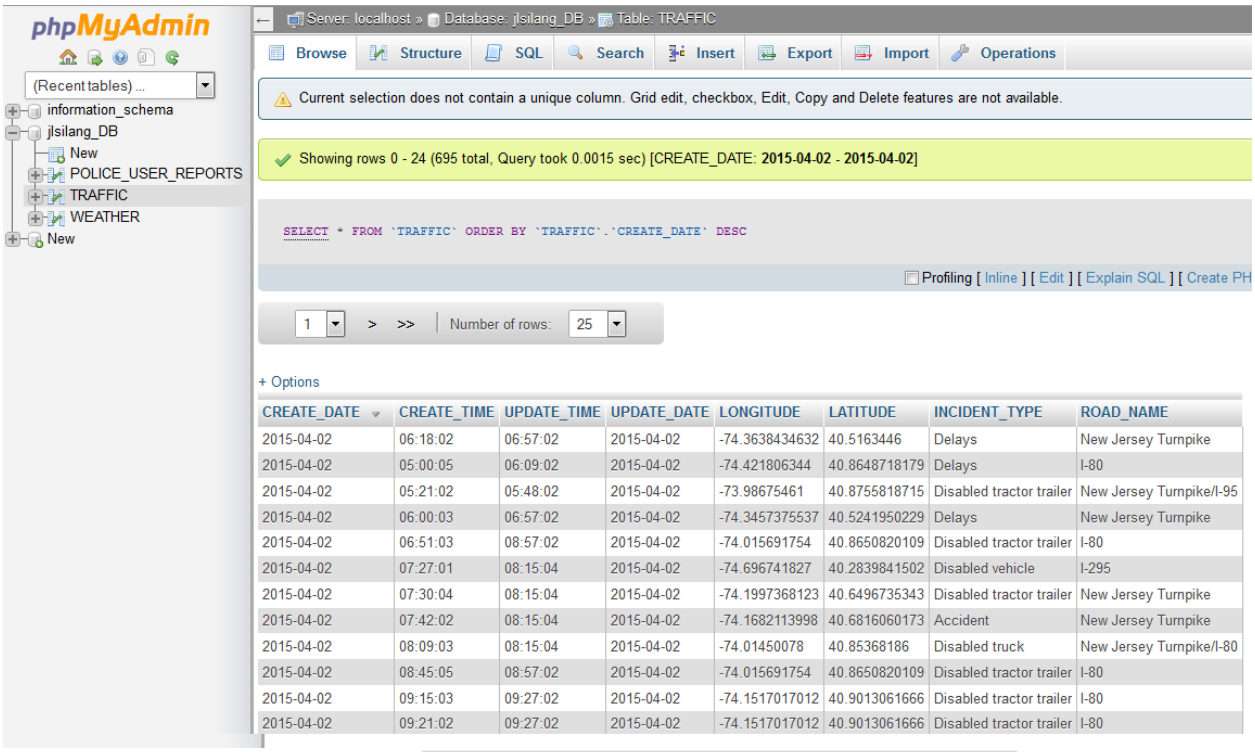
This is the MySQL database used in the traffic website service.



The tables of jlsilang_DB database.

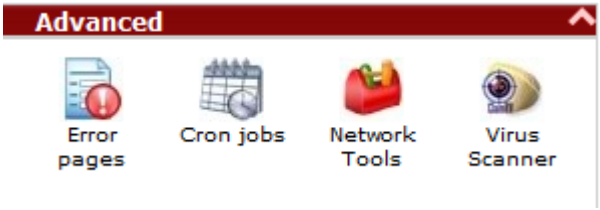


This is a preview of the TRAFFIC table entries. New data is stored here after extracting and parsing from various traffic data services.



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.

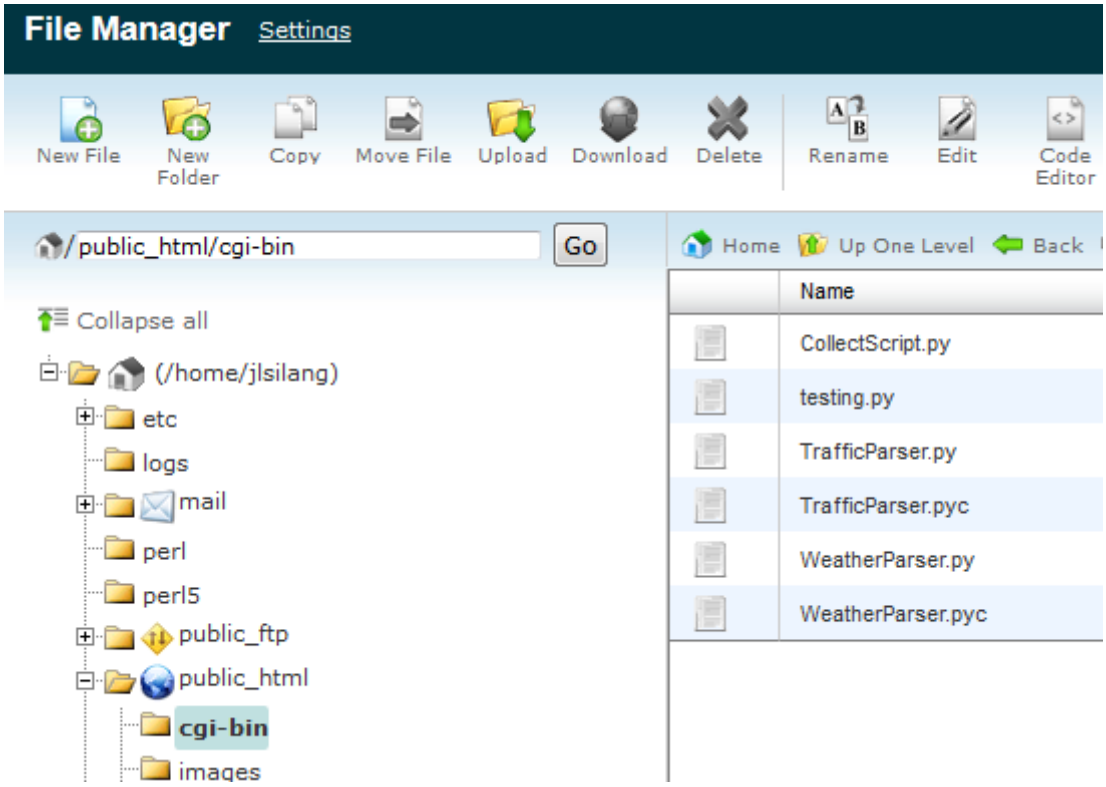


The CollectScript.py python file is run every three minutes in our Cron job.

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.

```
41                    for j in i.split("</td>"): #Skip through all unnecessary fields until "Road Name"
42                        if j == "\r\n":
43                            continue
44                        elif self.count == 1:
45                            Desc =  j[j.find(">")+1:] #First tag encountered holds the incident description.
46                            self.count = self.count+1
47                        elif self.count == 2: #Second tag encountered holds the Road Name.
48                            Road = j[j.find(">")+1:]
49                            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
50                                print" %s LONGITUDE: %s DESCRIPTION: %s ROAD: %s" % (Lat,Lon,Desc,Road)        #DEBUGGING PURPOSES
51                                self.dbadd(Lat, Lon, Desc, Road) #Call database add method once the data is populated.
52                            self.count = self.count +1
53                        else:
54                            self.count=self.count+1
55                    self.count=0 #Reset count for the next incident.
56                #  print ""      #  DEBUGGING PURPOSES
57
58 #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.
59     def dbadd(self, lat, lon, type, road):
60         print "is it adding"
61         self.db = MySQLdb.connect("localhost", "jlsilang_jc", "Runescape1", "jlsilang_DB") #Make a connection to the database with the necessary
credentials.
62         self.cursor = self.db.cursor() #Create a cursor in order to execute and read results from the database
63         sql_cmd = """SELECT * FROM TRAFFIC WHERE
64                     LONGITUDE = '%s' AND
65                     LATITUDE = '%s' AND
66                     INCIDENT_TYPE = '%s' AND ROAD_NAME='%s'""" %(lon,lat,type,road) #SQL command to check if this is a duplicate entry.
67         try: #cursor.execute may throw an exception which can corrupt database entries.
68             self.cursor.execute(sql_cmd) #Execute the duplicity check SQL command.
69             result = self.cursor.fetchall()
70
```