Server Team: Guangyue Cao, Cameron Cortez, Noele Anna Illien

**Describe how the server interacts with the client, including:**

**The arguments that are sent from the client and received by the server at the beginning of the request.**

Our goal is to query data from a certain area. The area is simply defined by the window size of the front web page. So the first request sent from the client and received by the server is to get coordinates in the format of “(lat1, lat2, lng1, lng2)”. And print out the result:

*"received coordinates: [" + lat1 + ", " + lat2 + "], [" + lng1 + ", " + lng2 + "]"*

In the meanwhile, sent back message to the menu panel on the client side:

*q.put("received coordinates: [" + lat1 + ", " + lat2 + "], [" + lng1 + ", " + lng2 + "]"+" Yellow Detective is querying the data....")*

Then the server will query data from the database within the latitude and longitude information and within a certain period of time. and print out the number of Checkins received in this area. And get these user id from these Checkins.

As the next step is to get the connected places, which is the other places these users have checked in.

While running this query the server will send back the real time message to client side:

*q.put("received" + str(len(places)) + "connected places from" + str(record.\_in))*

For each original places and connected places the server will store the information in separate lists in Json format.

For original places, we stored the latitude and longitude information for the Checkin points and the content texts posted along with the Checkins;

For connected places, we stored the tile, catalog (type of the place) and the latitude & longitude information.

What’s more, we want to connect the origin points and other related points by lines. In our case, the start point will be the original place and the end point will be the connected place. We build up the line information in the form of (x1,y1,x2,y2)

In the end we return to the client side these three sets of information by using the code:

*output["features"].append(originPlaces)*

*output["features1"].append(connectedPlaces)*

*output["lines"] = lines*

and send back the message to the menu panel:

*q.put("Done! Received " + str(numListings) + " Checkins, Yellow Detective is having a rest")*

**Describe how the server interacts with the database, including:**

**The query that is sent to the database.**

The query sent to the database to is select the Checkins from a given geo area during a certain period of time.

The area is defined by latitude and longitude value which is extracted from the window size of front web page.

The time is defined specifically between 2013-12-03 21:00:00 and 2013-12-04 04:00:00. (in the future, we want our time in this tool can be defined by the number input from the web page, so people can get the results of anytime they want)

*query = 'SELECT FROM Checkin WHERE latitude BETWEEN {} AND {} AND longitude BETWEEN {} AND {} AND time BETWEEN "2013-12-03 21:00:00" and "2013-12-04 04:00:00"'*

**How the results of the query are processed and formatted for sending back to the client?**

For querying the results, we mainly have 5 steps.

Step 1: Get Checkins from given area and time period.

Step 2: For each Checkin, get the user id.

Step 3: Skip repeating users.

Step 4: Find connected places for each Checkin.

Step 5: Store the information of original places/Checkins and connected places in Json format separately for sending back to the client side.

The Json format we established are as below:

*originPlaces = {"type":"Feature","properties":{},"geometry":{"type":"Point"}}*

*connectedPlaces = {"type":"Feature","properties":{},"geometry":{"type":"Point"}}*

*lines.append({'coordinates': [record.lat, record.lng, place.lat, place.lng]})*

this information are sent to the client side using these code:

*output["features"].append(originPlaces)*

*output["features1"].append(connectedPlaces)*

*output["lines"] = lines*

*return json.dumps(output)*