#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

import atexit

import datetime

import geopy.distance

from flask import Flask, request

import sqlite3

import json

app = Flask('\_\_main\_\_')

@app.route('/userparking/<id>')

def get\_parking\_for\_user(id):

query = 'select \* from parkings where userId = ' + str(id)

it = cursor.execute(query)

ans = it.fetchone()

if ans != None:

return "success"

return "no parking"

@app.route('/insertparking/',

methods=['POST'])

def insert\_parking():

#

myj = request.get\_json(force=True)

query = 'insert into parkings (' + ','.join(['%s' for i in myj.keys()]) % tuple(myj.keys())\

+ ') values (' + ','.join(['?' for i in myj.keys()]) + ')'

it = cursor.execute(query, tuple(myj.values()))

# sql injection check?

print myj

for i in it:

print i + ", "

db.commit()

return "Error in insertion"

@app.route('/parkings/page/<cntrLat>/<cntrLon>/<zoom>') #add /cntrLat/cntrLon/zoom?

def parking\_page(cntrLat, cntrLon, zoom):

# show the user profile for that user

query = '''SELECT \* FROM parkings'''

coords\_1 = (cntrLat, cntrLon)

avgsizeofscreeninDP = 160\*4 #160dp per inch

worldwidthinDP = 256.0 \* (2 \*\* float(zoom)) #according to meaning of zoom in googlemaps

worldwidthinKM = 40070.0 #circumference of earth

radius = (worldwidthinKM/worldwidthinDP)\*avgsizeofscreeninDP/2 #km in the screen (half the height)

if (radius > 50): radius = 50 #limit to up to 50km radius

cursor.execute(query)

ans = cursor.fetchall() #get data from the sql response

ans = [j for j in ans if (geopy.distance.vincenty(coords\_1, (j[1], j[2])).km <= radius)] #filter out far parkings

send = '[\n' + '\n,'.join([buildParkingJSON(j) for j in ans]) + '\n]' #json formatting

return send

@app.route('/request/<address>/<OPhour>/<ENDhour>')

def request\_parking(address, OPhour, ENDhour):

# handle a request from a client to order a parking for certain hours.

query = '''SELECT \* FROM messages where (((OPhour <= "'''+OPhour+'''") and (ENDhour > "'''+OPhour+'''")) or \

((OPhour > "'''+OPhour+'''") and (OPhour < "'''+ENDhour+'''"))) and \

type = "accept" and address = "'''+address+'''"'''

#check for previous requests of the parking spot that are still relevant

cursor.execute(query)

ans = cursor.fetchall()

for j in ans:

return "unavailable"

if ans == []:

if (OPhour == ENDhour):

return "available";

cursor.execute('''select \* from parkings where address="'''+address+'''"''')

id = cursor.fetchone()[0]

cursor.execute('''INSERT into messages values ('''+str(id)+''', "accept", "'''+address+'''","'''+OPhour+'''", "'''+ENDhour+'''")''')

cursor.execute('''INSERT into messages values ('''+str(id)+''', "unsent", "'''+address+'''","'''+OPhour+'''", "'''+ENDhour+'''")''')

db.commit()

print "check"

return "available"

return "bug"

@app.route('/messages')

def allMsgs():

cursor.execute("""SELECT \* FROM MESSAGES""")

return "<p></p>".join([",".join([str(i) for i in j]) for j in cursor.fetchall()])

@app.route('/getunsent/<id>')

def get\_unsent(id):

query = '''SELECT \* from messages where type = "unsent" and clientId = ''' + str(id)

cursor.execute(query)

ans = cursor.fetchall()

if ans != None:

send = '[\n' + '\n,'.join([buildMessageJson(j) for j in ans]) + '\n]'

# query = '''UPDATE messages set type = "sent" where type = "unsent" and clientId =''' + str(id)

cursor.execute(query)

else: send = "None"

print send

return send

def buildMessageJson(messageFromSQL):

return """{

"clientId":""" + str(messageFromSQL[0]) + """,

"type":\"""" + str(messageFromSQL[1]) + """\",

"address":\"""" + str(messageFromSQL[2]) + """\",

"OPhour":\"""" + str(messageFromSQL[3]) + """\",

"ENDhour":\"""" + str(messageFromSQL[4]) + """\"}"""

def buildParkingJSON(parkingFromSQL):

now = str(datetime.datetime.now())[:-7] #current time in the correct format

return """[{

"userId":"""+ str(parkingFromSQL[0]) +""",

"longitude":"""+ str(parkingFromSQL[1]) +""",

"latitude":"""+ str(parkingFromSQL[2]) +""",

"address":\""""+ parkingFromSQL[3] +"""\",

"hours":\""""+ parkingFromSQL[4] +"""\",

"costPerHour":"""+ str(parkingFromSQL[5]) +""",

"rating":"""+ str(parkingFromSQL[6]) +""",

"numberOfRaters":"""+ str(parkingFromSQL[7]) +""",

"size":\"""" + parkingFromSQL[8] + """\",

"description":\"""" + parkingFromSQL[9] + """\",

"isGate":"""+ ("true" if parkingFromSQL[10] else "false")+"""

},""" + ("1" if request\_parking(parkingFromSQL[3], now, now)=="available" else "0")+"""

]"""

def fill\_data\_base():

baseLat = 32.824685

baseLon = 35.234116

query = '''INSERT into parkings values (?,?,?,?,"07am-04pm",25,2.333,0,"medium","below the oak",1)'''

id = 2229034303789127

add = "Tsiyon St %s, Moreshet, Israel"

for i in range(1,4):

lowLat = baseLat - 0.01\*i

highLat = baseLat + 0.01\*i

cursor.execute(query, (id+i, lowLat, baseLon, add%str(100\*i+15)))

cursor.execute(query, (id+i, baseLat, baseLon-0.01\*i, add%str(100\*i+25)))

cursor.execute(query, (id+i, highLat, baseLon, add%str(100\*i+35)))

cursor.execute(query, (id+i, baseLat, baseLon+0.01\*i, add%str(100\*i+45)))

for j in range(1,4):

cursor.execute(query, (id+i, lowLat, baseLon-0.01\*j, add%str(100\*i-j)))

cursor.execute(query, (id+i, highLat, baseLon-0.01\*j, add%str(110\*i-j)))

cursor.execute(query, (id+i, lowLat, baseLon+0.01\*j, add%str(100\*i+j)))

cursor.execute(query, (id+i, highLat, baseLon+0.01\*j, add%str(110\*i+j)))

if \_\_name\_\_ == '\_\_main\_\_':

db = sqlite3.connect('example.db')

cursor = db.cursor()

atexit.register(db.close)

app.run(host="0.0.0.0")