/\* This server, unlike our previous ones, uses the express framework \*/

var express = require('express');

var formidable = require('formidable'); // we upload images in forms

var queries = require("./queries");

// this is good for parsing forms and reading in the images

// make a new express server object

var app = express();

// Now we build a pipeline for processing incoming HTTP requests

// Case 1: static files

app.use(express.static('public')); // serve static files from public

// if this succeeds, exits, and rest of the pipeline does not get done

// Case 2: queries

// An example query URL is "138.68.25.50:???/query?img=hula"

app.get('/query', function(request, response) {

// console.log("query");

query = request.url.split("?")[1]; // get query string

if (query) {

queries.answer(query, response); // This is where you call on queries.js

} else {

sendCode(400, response, 'query not recognized');

}

});

var sqlite3 = require("sqlite3").verbose(); // use sqlite

var dbFile = "photos.db";

var db = new sqlite3.Database(dbFile); // new object, old DB

function errorCallback(err) {

if (err) {

console.log("error: ", err, "\n");

}

}

function dataCallback(err, tableData) {

if (err) {

console.log("error: ", err, "\n");

} else {

console.log("got: ", tableData, "\n");

}

}

//db.serialize (function () {

// Case 3: upload images

// Responds to any POST request

app.post('/', function(request, response) {

var form = new formidable.IncomingForm();

form.parse(request); // figures out what files are in form

var localFile = "";

// callback for when a file begins to be processed

form.on('fileBegin', function(name, file) {

// put it in /public

file.path = \_\_dirname + '/public/' + file.name;

localFile = file.name;

// console.log("uploading ", file.name, name);

});

// callback for when file is fully recieved

form.on('end', function() {

// console.log('success');

// sendCode(201, response, 'recieved file'); // respond to browser

db.serialize(function() {

db.run('INSERT OR REPLACE INTO PhotoLabels VALUES ("' + localFile + '" , "", 0)', errorCallback);

// console.log("HEREEEEE~~~~~~~!!!!");

// console.log('HEREEE: UPDATE photoLabels SET labels = "DYING,DEATH," WHERE fileName = "' + localFile + '" ', errorCallback);

// db.run('UPDATE photoLabels SET labels = "DYING,DEATH," WHERE fileName = "' + localFile + '" ', errorCallback);

//this will be replaced soon

// console.log('UPDATE photoLabels SET labels = "DYING,DEATH," WHERE fileName = "' + localFile + '" ', errorCallback);

});

//call to GOOGLE CLOUD API HERE?!?!

// PUT request to the GCV API, with a link to the photo and requesting "label Detection".

// console.log("HEREEEEE~~~~~~~~```");

var LIVE = true;

var request = require('request');

var XMLHttpRequest = require("xmlhttprequest").XMLHttpRequest;

// An object that gets stringified and sent to the API in the body of an HTTP request

var urlReq = "http://138.68.25.50:11751/" + localFile;

// console.log("urlReq = " + urlReq);

requestObject = {

"requests": [{

"image": {

"source": {

// "imageUri": "http://138.68.25.50:11751/hula.jpg"

"imageUri": urlReq

}

},

"features": [{

"type": "LABEL\_DETECTION"

}]

}]

}

// URL containing the API key

url = 'https://vision.googleapis.com/v1/images:annotate?key=AIzaSyDkOkU6y2LRL7NUJqKJgukzXqXtQ20aJKs';

if (LIVE) {

// console.log("live!!");

// The code that makes a request to the API

// Uses the Node request module, which packs up and sends off an XMLHttpRequest.

request({ // HTTP header stuff

url: url,

method: "POST",

headers: {

"content-type": "application/json"

},

// stringifies object and puts into HTTP request body as JSON

json: requestObject,

},

// callback function for API request

APIcallback

);

} else { // not live! return fake response

// call fake callback in 2 seconds

console.log("not live");

setTimeout(fakeAPIcallback, 2000);

}

function APIcallback(err, APIresponse, body) {

if ((err) || (APIresponse.statusCode != 200)) {

console.log("Got API error");

} else {

var APIresponseJSON = body.responses[0];

var APIresponseStr = JSON.stringify(APIresponseJSON);

var length = APIresponseJSON.labelAnnotations.length;

// console.log("length = " + length);

// console.log("I'M HERE = " + APIresponseStr);

var labelStr = "";

for (r = 0; r < length; r++) {

var description = APIresponseJSON.labelAnnotations[r].description;

strDes = JSON.stringify(description);

strDes2 = strDes.replace(/"/g, "");

// console.log("description = " + strDes2);

labelStr = labelStr + strDes2 + ','

}

console.log(labelStr);

db.run('UPDATE photoLabels SET labels = "' + labelStr + '" WHERE fileName = "' + localFile + '" ', errorCallback);

sendCode(201, response, labelStr); // respond to browser

//I think this should be 200??

}

}

// }

});

});

//});

// You know what this is, right?

app.listen(11751);

// sends off an HTTP response with the given status code and message

function sendCode(code, response, message) {

response.status(code);

response.send(message);

}