**Part 1.**

Create a Node.js application using Express that obtains weather data for “Omaha,NE”, “New York,NY”, “Singapore,Singapore”, “Jakarta,Indonesia” from a 3rd party API. Display the weather result in a table or list and omit any locations that fail to return data.

Store the weather data in a model and populate it using Express template engine (ejs/jade). Use a responsive framework like Bootstrap or SemanticUI. Log to file all parameters passed to server.

Use appropriate variable names & comments & file structures. Creativity is encouraged.

**Details :**The [Weather Underground](https://www.wunderground.com/) requires developer registration to use their [free API](https://www.wunderground.com/weather/api/). Upon registration you will receive a key which is required for your request.

**Resources :  
A. Node**<https://nodejs.org/en/> ( use Node version 4 and above )

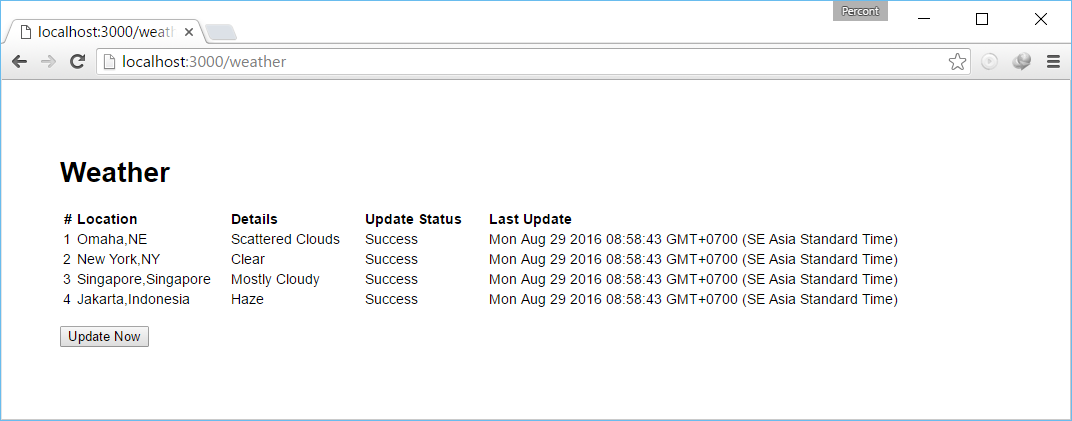
**B. Express**<http://expressjs.com/>

**C. IDE (Optional)**Use [Atom](https://atom.io/), [Webstorm](https://www.jetbrains.com/webstorm/) (30 days trial), or any text editor of your choice.

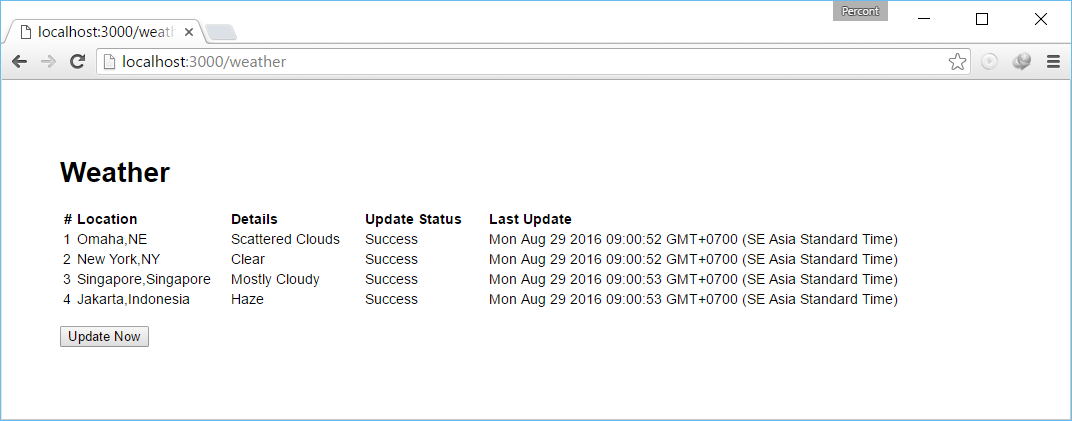
Submit the result by uploading it to your Github account. Please do not attach your answer because it will be removed by our email policy.

Answer:

1. Resources that was used: NodeJS, ExpressJS, SemanticUI, MongoDB, Notepad++.
2. Web Screenshot
3. Home page (localhost:3000) will be redirect to weather page (localhost:3000/weather)



1. If “Update Now” button was pressed, we will try to retrieve data for 4 cities (Omaha, New York, Singapore, Jakarta) from “api.wunderground.com”, and will be redirect into the same page.



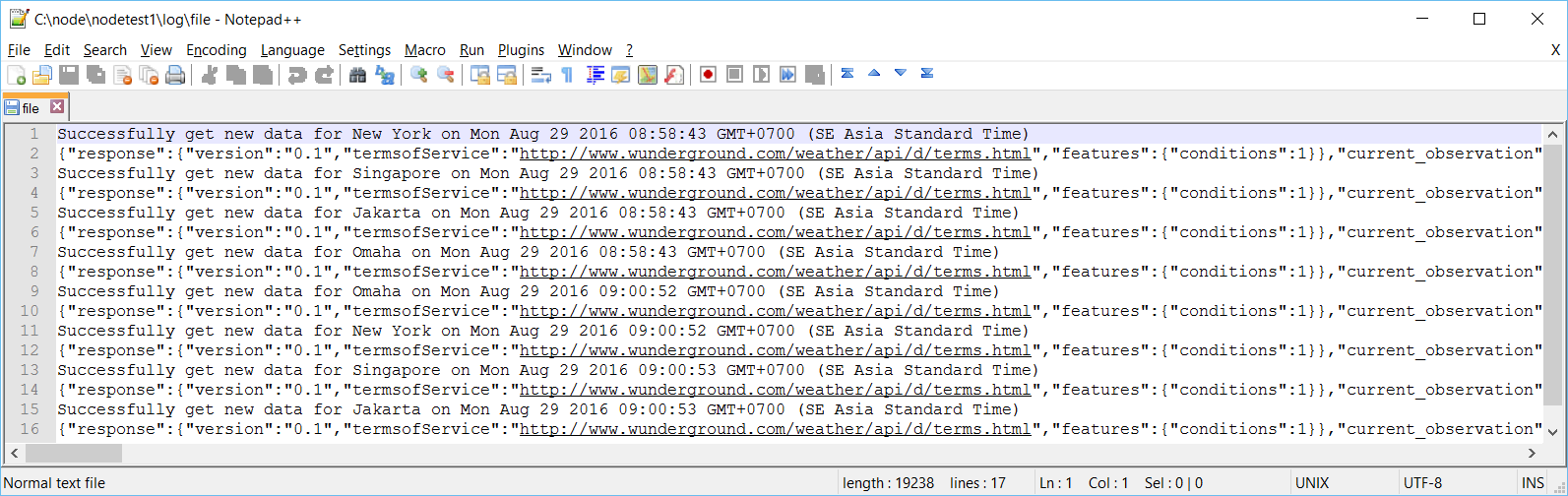
1. Database & File Screenshot

We store 4 data record on collection:weathercollection and db:nodetest1.



Each time we retrieve new data from Weather Underground API, it will update data, lastUpdate and updateStatus datafield.

Log file for other parameters will be stored on “log/file” file



1. Routes/Index.js

|  |
| --- |
| var express = require('express');  var http = require('http');  var router = express.Router();  var fs = require('fs');  var cityArray = ["Omaha,NE","NY/New\_York","SG/Singapore","ID/Jakarta"];  var key = '69098eef7b6a263c';  /\* Set http option \*/  var options = {  host: 'api.wunderground.com',  path: ''  };  /\* GET Home page. \*/  router.get('/', function(req, res, next) {  //redirect to weather page  res.redirect("weather");  });  /\* GET Weatherlist page. \*/  router.get('/weather', function(req, res) {  // Set our internal DB variable  var db = req.db;  // Set our collection  var collection = db.get('weathercollection');  collection.find({},{},function(e,docs){  res.render('weather', {  "weather" : docs  });  });  });  /\* GET Update page. \*/  router.get('/updateweather', function(req, res) {  // Set our internal DB variable  var db = req.db;  // Set our collection  var collection = db.get('weathercollection');    var jsonObject;  var jsonWeather;  var i;  //loop for each city  for (i = 0; i < cityArray.length; i++) {  //set correct url path  options.path = '/api/' + key + '/conditions/q/' + cityArray[i] + '.json'  var call = http.request(options, function(response){  if(response.statusCode == 200) {  var str = '';  //retrieve chunk  response.on('data', function (chunk) {  str += chunk;  });  //get the whole data  response.on('end', function () {  //parse into json object  jsonObject = JSON.parse(str);  //update correct data collection  collection.update({"key":jsonObject.current\_observation.display\_location.city}, {$set:{updateStatus:"Success",data:jsonObject.current\_observation.weather,lastUpdate:new Date()}}, function (err, doc) {  //if write failed  if (err) {  //if it failed, return error  res.send("There was a problem adding the information to the database.");  console.log('There was a problem adding the information to the database.');  }  //if write success  else {  //write a log on console  console.log('Success update ' + jsonObject.current\_observation.display\_location.city);  }  })  //log all parameter passed to server  fs.appendFile("log/file", "Successfully get new data for " + jsonObject.current\_observation.display\_location.city + " on " + new Date() + "\n" + JSON.stringify(jsonObject) + "\n", function(err) {  if(err) {  return console.log(err);  }  //write a log on console  console.log("Log file was appended!");  });  });  }  else {  //if it failed to retrieve data, update status  collection.update({"key":cityArray[i]}, {$set:{updateStatus:"Failed"}});  }  });  //error handler  call.on('error', (e) => {  console.log("Error");  });  //end http request  call.end();  }  delete require.cache['/weather'];    //redirect to home page  res.redirect('back');  });  module.exports = router; |

Views/weather.jade

|  |
| --- |
| extends layout  block content  h1(class="ui header").  Weather  table(border='0')  thead  tr  th(width="10") #  th(width="150", align="left") Location  th(width="130", align="left") Details  th(width="120", align="left") Update Status  th(width="500", align="left") Last Update  tbody  each item, i in weather  tr  td=i+1  td=item.location  td=item.data  td=item.updateStatus  td=item.lastUpdate  br  button(class="ui button", type="button", onclick="window.open('updateweather', '\_self');") Update Now |

App.js

|  |
| --- |
| var express = require('express');  var path = require('path');  var favicon = require('serve-favicon');  var logger = require('morgan');  var cookieParser = require('cookie-parser');  var bodyParser = require('body-parser');  // New Code  var mongo = require('mongodb');  var monk = require('monk');  var db = monk('localhost:27017/nodetest1');  var reload = require('reload')  var routes = require('./routes/index');  var users = require('./routes/users');  var app = express();  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'jade');  // uncomment after placing your favicon in /public  //app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));  app.use(logger('dev'));  app.use(bodyParser.json());  app.use(bodyParser.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, '/public')));  app.use(express.static(path.join(\_\_dirname, '/semantic/dist')));  // Make our db accessible to our router  app.use(function(req,res,next){  req.db = db;  next();  });  app.use('/', routes);  app.use('/users', users);  // catch 404 and forward to error handler  app.use(function(req, res, next) {  var err = new Error('Not Found');  err.status = 404;  next(err);  });  // error handlers  // development error handler  // will print stacktrace  if (app.get('env') === 'development') {  app.use(function(err, req, res, next) {  res.status(err.status || 500);  res.render('error', {  message: err.message,  error: err  });  });  }  // production error handler  // no stacktraces leaked to user  app.use(function(err, req, res, next) {  res.status(err.status || 500);  res.render('error', {  message: err.message,  error: {}  });  });  module.exports = app; |