# Restaurant Guide Project Instructions

**Overview**

In this project you will build some data files and data structures in JSON to hold restaurant information including reviews and ratings data. Next you will write a server application in NodeJS and Express to read the data files and serve the data as RESTful Web services. You will also build a Web application that accesses the Web service data and displays restaurant information to the user. The server application will do double duty as a standard web server. It will be used to serve the html, image, JavaScript, and CSS files in addition to the RESTful services.

**Conventions**

Throughout this project the following typographic conventions will be used:

| Alias | Path |
| --- | --- |
| *<RESTAURANT>* | *c:\project\restaurants.com* |
| *<DATA >* | *c:\project\restaurants.com\data* |
| *<PUBLIC>* | *c:\project\restaurants.com\public* |
| *<CSS>* | *c:\project\restaurants.com\public\css* |
| *<IMAGES>* | *c:\project\restaurants.com\public\images* |
| *<SCRIPTS>* | *c:\project\restaurants.com\public\scripts* |

**Setup**

The project files for this lab have been included in a VM Ware virtual machine. The virtual machines is pre-configured with Windows 7, NodeJS, Express, WebStorm (and IDE) and Chrome. All of you work will be done in the virtual machine.

**Data Files - JSON**

In this step you will create the JSON files with the restaurant data. You have been provided with the restaurant data in plain text form. You will need to convert it to JSON and store it in four JSON files with a *.json* extension. The first file contains a list of cuisines (sushi, tapas, pho). The other files contain restaurant information. Each file contains the info for two restaurants of the same cuisine type. At the end of the section you will have created four files: cuisines.json, sushi.json, pho.json, and tapas.json.

1. Open the file <DATA>\cuisines.txt in a text editor.
2. Create a JSON array from the strings and store it in a file called cuisines.json.
3. Open the file <DATA>\sushi.txt in a text editor.
4. Notice that the structure of the data differs a bit between the files. The body of Fuki Sushi’s review consists of two lines (really pargraphs) while Musashi’s consist of only one.
5. Create a JSON array of objects from the restaurant information and store it in a file called sushi.json.
   1. Make sure that your data structure can handle review bodies with an arbitrary (non-zero) number of paragraphs.
   2. Each line in the text file should be represented by a single JSON value (this is important because we will use it later!)
6. Do the same for <DATA>\pho.txt and <DATA>\tapas.txt.

**Read Data into the NodeJS/Express Server**

This project uses NodeJS and Express to create a server. The basic setup to initialize Node and Express is done for you. The code is also included to bind IP 127.0.0.1 and port 80 and to serve static files from the <PUBLIC> directory. (In apache terminology, the <PUBLIC> directory is the document root.

The server code is in a single file <RESTAURANT>\server.js

The code already contains a function, getRestaurantData(), has been provided to read a JSON file and turn it into a JavaScript object. In your code you create an array of cuisine types. Then you will iterate through that array and open the JSON file for each cuisine type and read it into memory. You will store all the restaurant info in a single array. Since the restaurant info does not contain cuisine information, you will have to add it. (HINT: add cuisine when you create the array of restaurant information objects.)

1. Open the file <RESTAURANT>\ server.js in a text editor.
2. Look for:

// \*\* CREATE VARIABLES TO HOLD RESTAURANT AND CUISINE ARRAYS HERE \*\*

1. Create arrays to hold cuisines and restaurant info. Info for all restaurants regardless of cuisine type will be stored in a single array. (HINT: make sure your solution can handle an arbitrary number of cuisine types and restaurants.)
2. Look for:

// \*\* GET ALL RESTAURANTS FOR EACH CUISINE TYPE \*\*

1. Iterate through the cuisines and load all the restaurants of each cuisine type
2. Add a field to each restaurant to indicate the cuisine type and add the restaurant to the restaurant array
3. Add console.log() calls to your code verify that the data is read in correctly
4. Open a DOS window
5. Navigate to the <RESTAURANT> directory and lad dependencies by running:

npm install

1. directory and execute the following at the command line:

node server

1. You should see your log messages in the DOS window
2. Use CTRL-C to shut down the server

**RESTful Services**

The last part of the server is the RESTful services. There are three services the server needs:

GET/cuisines – return all cuisines

GET/restaurants – return all restaurants

GET/restaurants/{cuisine} – return all restaurants of a given cuisine type

The function that implements GET/cuisines is provided as an example. Partial implementations are provided for the others.

1. Open the file <RESTAURANT>\restaurant.js in a text editor.
2. Complete the GET/restaurants function to return all restaurants
3. Complete the GET/restaurants/{cuisine} function to return all restaurants of the specified cuisine type
4. Open a DOS window
5. Navigate to the <RESTAURANT>directory and execute the following at the command line:

node server

1. Use the following URLS in Chrome to test your RESTful services

http://localhost/cuisines

http://localhost/restaurants

http://localhost/restaurants/pho

http://localhost/restaurants/tapas

http://localhost/restaurants/sushi

1. Use CTRL-C to shut down the server

**Client App**

The Client App should display a radio button for each cuisine type. The first time it is displayed the first cuisine should be selected and the related restaurant information should be displayed. Clicking on a different cuisine type should switch the restaurant display. Clicking the plus/minus icons should show or hide the review portion of the info.

The app consists of the following files:

<PUBLIC>\index.html

<SCRIPTS>\main.js

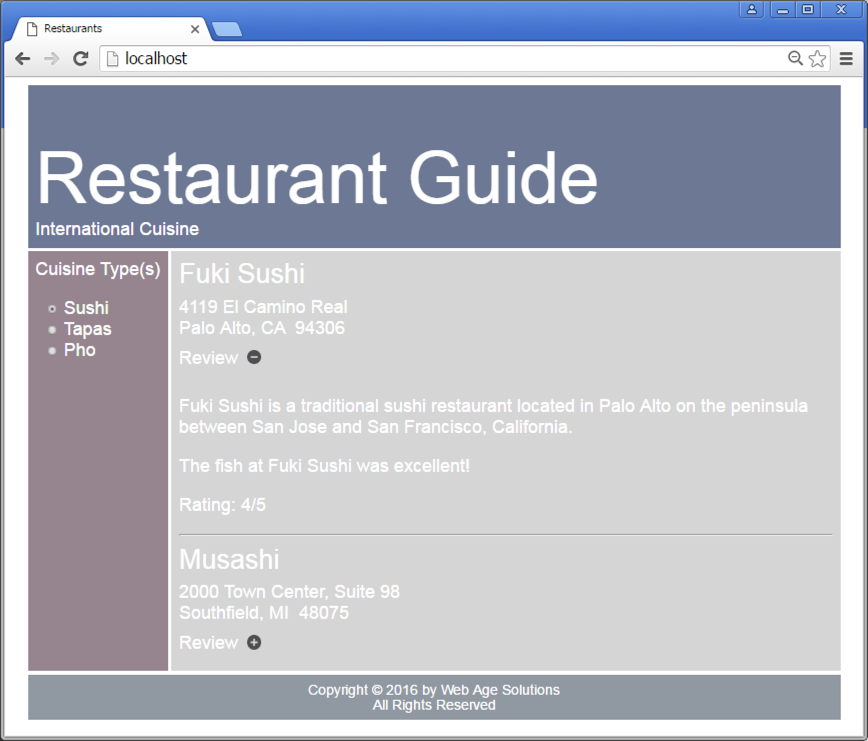
<CSS>\main.css

<IMAGES>\plus.png

<IMAGES>\minus.png

You should only have to edit main.js. The index.html file has HTML and CSS with data stubbed in. Your task will be to edit main.js to dynamically generate the HTML from the data received from the RESTful services.

This is what the final app should look like:



1. Open <SCRIPTS>\main.js in a text editor.
2. Complete the onload function by using the XMLHttpRequest object to call the /cuisines service. When the cuisines have been received:
   1. Call buildMenu()to populate the radio buttons that make up the menu
   2. Select the first radio button
   3. Call getRestaurants() and pass in the first cuisine
3. Complete the buildMenu() function.
   1. Iterate through the cusines and create a radio button for each
   2. Designate selectCuisine() as the onclick handler
4. Complete the getRestaurants() function.
   1. Use the XMLHttpRequest object to call the /restaurant service and get the restaurants with selected cuisine
   2. When the results come back add a boolean field to each restaurant object to indicate whether review is displayed
   3. Add each restaurant to the restaurants array
   4. Call showRestaurants() to display the restaurants
5. Complete the showRestaurants() function
   1. Display the restaurant name and address for all restaurants
   2. Create an element to wrap the review and initially hide the review
   3. Toggle the review display when the user clicks the plus (or minus) (HINT: use toggleReview() to show or hide the review)
6. Complete the toggleReview()function
7. Open a DOS window
8. Navigate to the <RESTAURANT>directory and execute the following at the command line:

node server

1. Open a browser and connect to http://localhost and test the app
2. Use CTRL-C to shut down the server

**EXTRA**

To test really test your app add another cuisine and associated restaurant information (BBQ?) Your application should be fully data driven and able to handle changes.