**REST API for Restaurants:**

The code is committed to the GIT repository and below is the URL:

1. Clone the repository to the local from this url “https://github.com/stathari/Restaurant\_API.git”

It has 3 subfolders:

1. **Setup:** Database script file for setting up the tables
2. **Code:** Node.js project code
3. **Test Cases:** Captured test results in Unit Test document and end point URLs in .json file for postman client.

**Technologies used:** Node.js, MySQL.

**Initial setup for the web server:**

* Install Node.js, MySQL
* Run the Restaurants.sql script file on mysql prompt using source command.
* Edit the values for elements in the .env file
* As the application is built using Node.js the dependencies(modules)for the project can be installed using the "npm install" command from the root folder where the package.json
* Check whether node\_modules folder is created in the directory of the application
* Run "npm start" or "node ./server.js" to start the webserver
* Once the server is up and running open the url http://<>:port [ for checking the server is up and running ].

**Following are the service end points implemented:**

**Service url:** <http://(ip_address):(port)/>eg: http://localhost:5000/

|  |  |  |
| --- | --- | --- |
| Method | Description | Service Endpoint |
| GET | Get all the restaurants | /restaurants |
| GET | Get the restaurant with a restaurant\_ID | /restaurant/ID |
| POST | Adding a restaurant to the system | /restaurant/add |
| DELETE | Deleting a restaurant | /restaurant/delete/ID |
| GET | Get all menus in a restaurant | /menus/ID |
| POST | Add menu to the restaurant | /menus/add |
| DELETE | Delete menu | /menus/delete/RID&MENU |
| GET | Get menu items in a restaurant | /menuitems/RID |
| POST | Adding menu items | /menuitems/add |
| DELETE | Deleting a men item | /menuitems/delete |

**Testing:**

**Open the postman Rest client for running the test cases**

* Open the file “Restaurants.postman\_collection\_TestCases.json” which comprises of all the test cases with self-descriptive for testing.
* Unit test cases are captured in the UnitTestCases.xlsx document

**Scalable solution:**

The REST API services which are developed as part of this assignment work is able to server the restaurants that are fewer in number compared to the one that are in reality. [till the level of restaurants reaching few thousands. The MySQL database will lag in serving the requests within acceptable time.

As we know that it is preferable to have all the services respond within 1 sec time frame for an ideal REST API irrespective of the load.

So, to have the services up and running in scale and serving the future requirements, We can set the scalability following the below steps:

1. We can add caching to the menu and menu\_items which hold records in terms of Bigdata it will enable faster retrieval of data.
2. Having multiple servers on a load balancer masked and have this setup installed (eg. NGINX)
3. To have the services up and running we shall install fault tolerant service nodes.

I also feel that the following idea will also help us having the minimum load at any point on the ordering application within the restaurant:

* Implementing time wise activation flags for menu and menu items to reduce the number of records in business during a time

For example: During the day time we can keep only Day menu available for the services of Restaurant ordering system.

* I often feel that ordering food online is painful to see that the desired food recipe you ordered is unavailable for that day until the time of payment. This is the main reason why I have come up with this approach. [ Needs to be optimized in a way, as people sometimes needs to check all the food items a restaurant serves]