**Zappos Data Science – Software Engineering Intern Challenge**

**Github:** <https://github.com/Anuj2512/Restaurant-REST-API>

**Anuj Chaudhari**

[**chaudhari.anuj93@gmail.com**](mailto:chaudhari.anuj93@gmail.com)

# Restaurant-REST-API

|  |  |
| --- | --- |
| Technology | Version |
| npm | 5.5.5 |
| Node.js | 6.11.5 |
| MongoDB | 3.4.9 |
| Redis | 4.0.8 |

## How to Run the Project

1. Install NodeJs, MongoDB, Redis locally. [Tested with MacOSX 10.13.3]
2. Clone or download the repository.
3. Make sure mongodb server and redis database server are running locally.
4. Go to api folder. (package.json file is located here)
5. run following command, to install node modules locally

* npm install

1. run following command to start node server which exposes restaurant REST api

* npm start

1. run following command to run unit tests

* npm test

## Database Schema

Restaurants  
{  
 \_id : <restaurant\_id>,  
 name : <restaurant\_name>,  
 description : <description>,  
 cuisines : [<cuisines-1>, <cuisines-2>, <cuisines-3>],  
 contact\_no : <contact\_no>,  
 address : <address>,  
 menus : [  
 {   
 menu\_id : <menu\_id>,  
 menu\_type : <breakfast, lunch, dinner, drinks>,  
 menu\_items : [  
 {  
 item\_name : <item\_name>,  
 description : <item\_description>,  
 price : <price>,  
 image\_url : <image\_url>  
 }  
 ]  
 }  
  
 ]  
}

## REST API Endpoints

**BaseURL** : localhost:3000/zappos/api/v1

GET /restaurants

POST /restaurants

GET /restaurants/:RestaurantID

DELETE /restaurants/:RestaurantID

POST /restaurants/:RestaurantID/menus

GET /restaurants/:RestaurantID/menus

GET /restaurants/:RestaurantID/menus/:MenuID

DELETE /restaurants/:RestaurantID/menus/:MenuID

POST /restaurants/:RestaurantID/menus/:MenuID/items

GET /restaurants/:RestaurantID/menus/:MenuID/items

GET /restaurants/:RestaurantID/menus/:MenuID/items/:ItemID

DELETE /restaurants/:RestaurantID/menus/:MenuID/items/:ItemID

## REST API Requests

* Here is the sharable link for POSTMAN to use all of this request. Just Import Request from the given link to POSTMAN.   
    
  <https://www.getpostman.com/collections/0833962d50247e7f69a0>

1. **Adding new restaurant**  
   POST http://localhost:3000/zappos/api/v1/restaurants

POST http://localhost:3000/zappos/api/v1/restaurants

201 Created

{

"name" : "Starbucks",

"description": "Starbucks is considered the main representative of second wave coffee, initially distinguishing itself from other coffee-serving venues in the US by taste, quality, and customer experience while popularizing darkly roasted coffee",

"contact\_no": "(408) 275-9368",

"area": "San Jose",

"cuisines" : ["cafe"],

"menu": [{

"menu\_type" :   "Coffee",

"menu\_items":   [{

"item\_name": "Coffee Latte",

"Description": "coffee with the best flovour of latte",

"price" : 3

},

{

"item\_name": "Caffe Mocha",

"Description": "coffee with the best flovour of mocha",

"price" : 4

}]

},

{

"menu\_type" :   "Tea",

"menu\_items":   [{

"item\_name": "Classic Chai Tea Latte",

"Description": "Classic Chai Tea with the best flovour of latte",

"price" : 2

}]

}

]

}

1. **Get restaurant details**

GEThttp://localhost:3000/zappos/api/v1/restaurants/

GET http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose

200 OK

{

"description": "Starbucks is considered the main representative of second wave coffee, initially distinguishing itself from other coffee-serving venues in the US by taste, quality, and customer experience while popularizing darkly roasted coffee",

"contact\_no": "(408) 275-9368",

"area": "San Jose",

"cuisines": [

"cafe"

],

"menu": [

{

"menu\_items": [

{

"decription": "",

"item\_name": "Coffee Latte",

"price": 3

},

{

"decription": "",

"item\_name": "Caffe Mocha",

"price": 4

}

],

"menu\_type": "Coffee",

"menu\_id": "coffee"

} ,

{

"menu\_items": [

{

"decription": "",

"item\_name": "Classic Chai Tea Latte",

"price": 2

}

],

"menu\_type": "Tea",

"menu\_id": "tea"

}

],

"locality\_count": 0,

"timestamp": "Tue Feb 06 2018 01:45:33 GMT-0800 (PST)",

"id": "starbucks-san-jose",

"name": "Starbucks",

}

1. **Delete restaurant**

DELETE http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>

DELETE http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose

204 No Content

1. **Add Menu**

POST http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus

POST http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus

201 Created

{

    "menu\_type" :   "Hot Drinks",

    "menu\_items":   [{

                        "item\_name": "Coffee Latte",

                        "Description": "coffee with the best flovour of latte",

                        "price" : 3

                    },

                    {

                        "item\_name": "Caffe Mocha",

                        "Description": "coffee with the best flovour of mocha",

                        "price" : 4

                    }]

}

1. **Get Menu**

GET http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>

GET http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee

200 OK

{

"menu\_items": [

{

"decription": "",

"item\_name": "Coffee Latte",

"price": 3

},

{

"decription": "",

"item\_name": "Caffe Mocha",

"price": 4

}

],

"menu\_type": "Coffee",

"menu\_id": "coffee"

}

1. **Delete Menu**

DELETE http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>

DELETE http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee

204 No Content

1. **Add MenuItem**POST http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>/items

POST http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee/items

201 Created

{

"item\_name": "Hot Chocolate",

"Description": "hot milk drink with chocolate cyrup",

"price" : 3

}

1. **Get MenuItem**  
   GET http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>/items/<item-id>

GET http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee/items/ 5a7c16156d887752a071b3a2

200 OK

{

"decription": "",

"\_id": "5a7c16156d887752a071b3a2",

"item\_name": "Coffee Latte",

"price": 3

}

1. **Delete MenuItem**

DELETE http://localhost:3000/zappos/api/v1/restaurants/<restaurant-id>/menus/<menu-type>/items/<item-id>

DELETE http://localhost:3000/zappos/api/v1/restaurants/starbucks-san-jose/menus/coffee/items/ 5a7c16156d887752a071b3a2

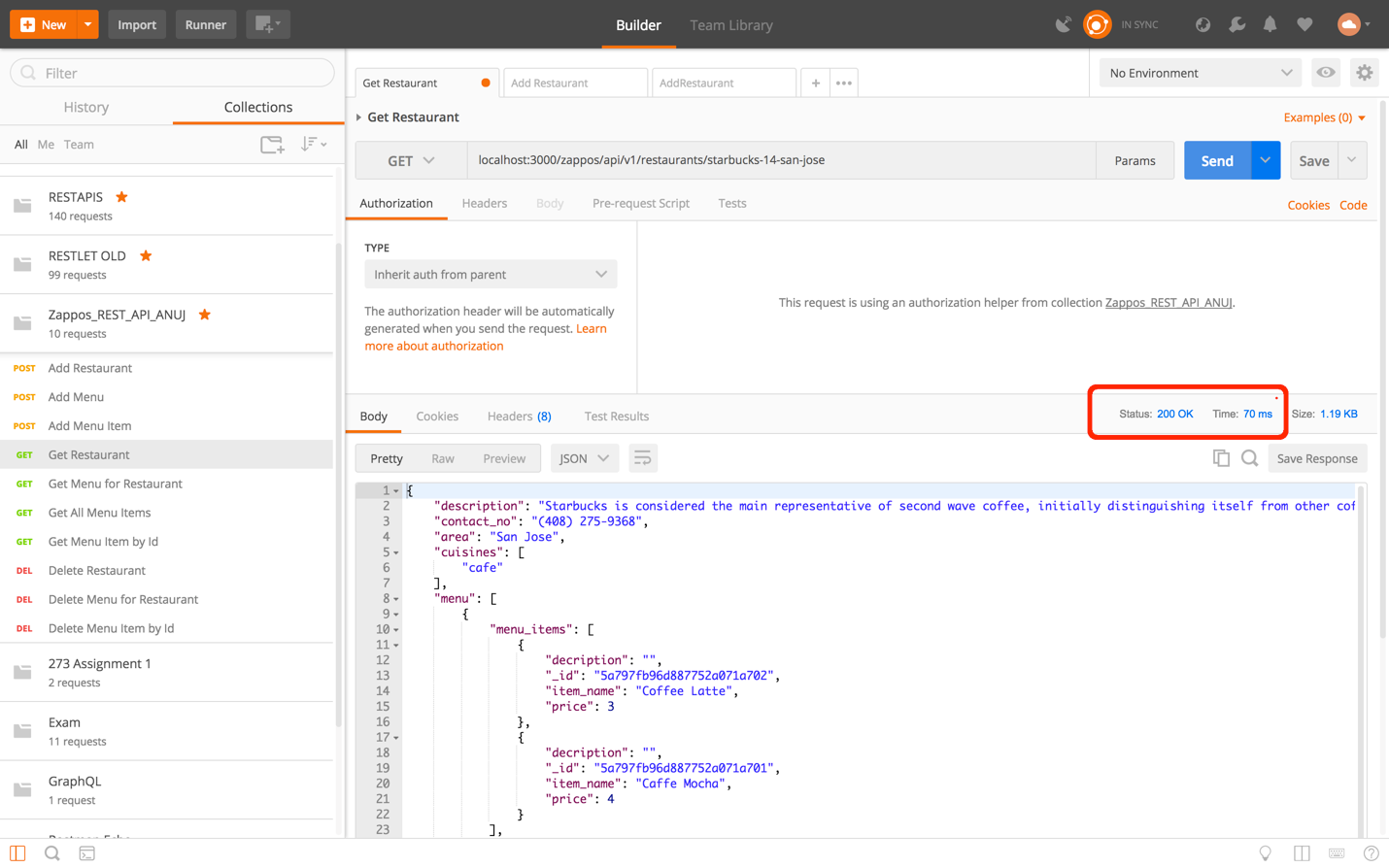
204 No Content

## 

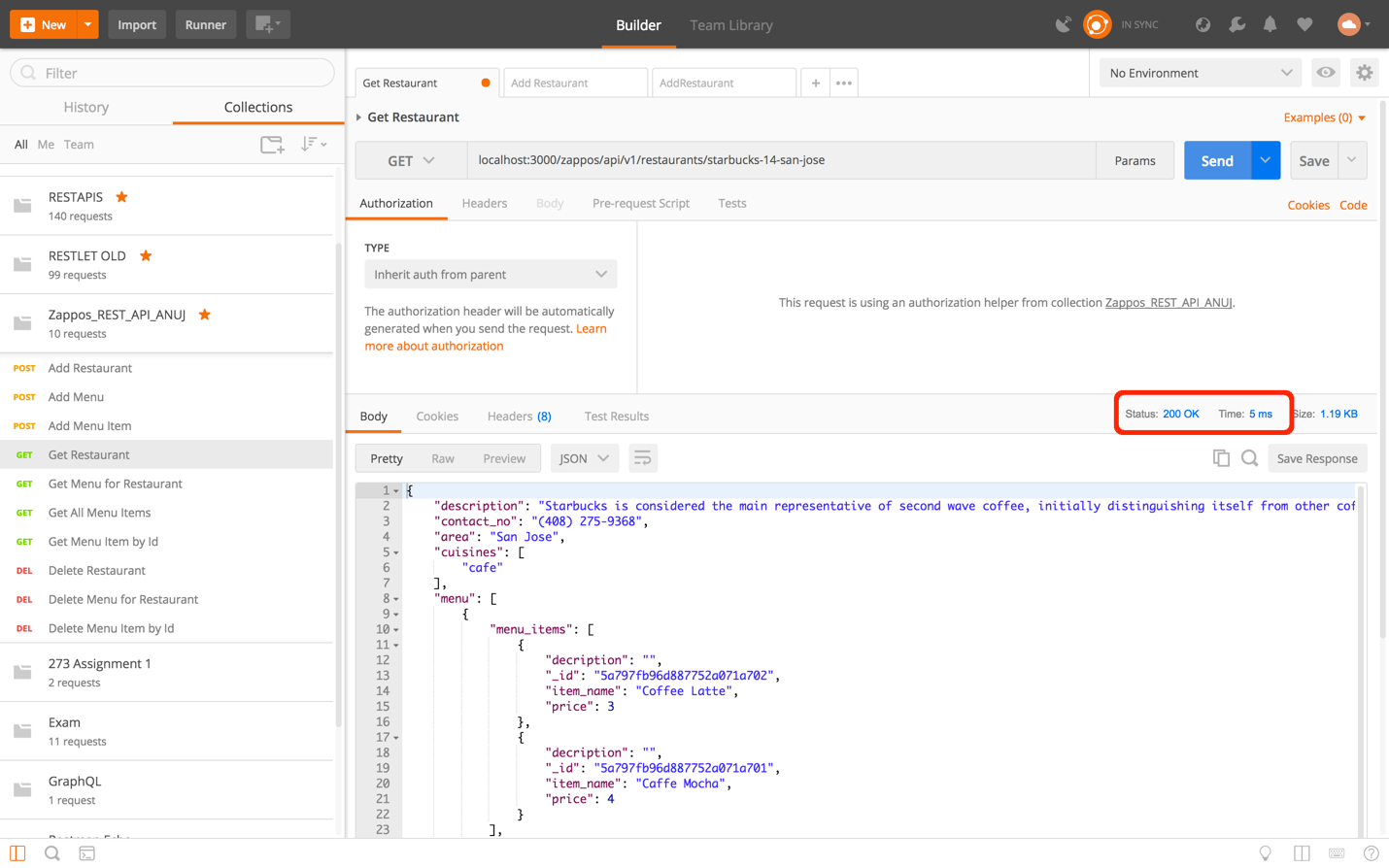
## Redis for Caching

* To compare the request performace of redis and direct mongodb access, I have added approximatly 30,000 dummy data to the mongodb.
* Import dummy data to mongodb
* cd db\_backup  
   mongorestore -d anuj\_restaurant\_api anuj\_restaurant\_api
* So, that when randomly sending a GET request for the first time, I can measure response time and sending the same request again, it will hit the redis cache, I can see the response time improvement in the request.
* I have attached some screenshots displaying the response time for both the scenario.
  1. MongoDB Access : **70ms**
  2. Redis Cache : **5ms**

## 



MongoDB hit

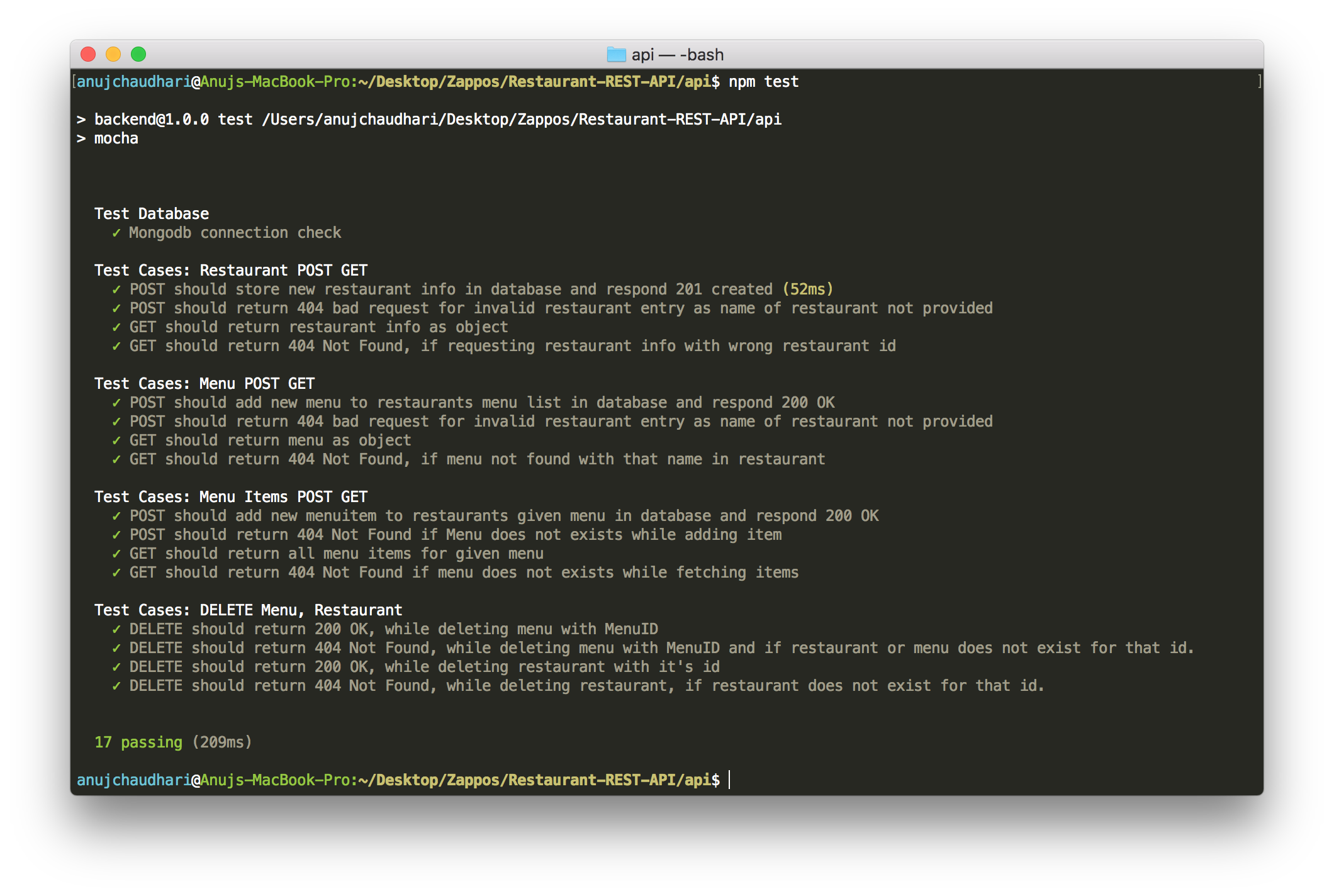


Redis Hit

## Unit Tests

Run following command to run unit tests

npm test



## Handling millions of Requests at once

For handling these many request at once we can use multiple servers running the same service and we can put the mongodb database on different machine and not where the node server is currently running.

For this we can use mongodb replicaset as well as load balancer on top of the node server.

If we use AWS services for handling scaling for this type of problem, Here is the server architecture diagram.

I have hosted this service on my AWS EC2 instance.

Here is the link for the API server. [http://ec2-54-215-224-120.us-west-1.compute.amazonaws.com:3000](http://ec2-54-215-224-120.us-west-1.compute.amazonaws.com:3000/)

