Introduction

I've successfully implemented the API using Nodejs with express, it is currently live at http://mohamey.me/api. The specific requests that can be made will be listed in the next section. There is also an in-memory database running behind the api, 'lokijs'. This database stores student objects which look like:

```
Student = {
        'id': 10,
        'forename': 'John',
        'surname' : 'Doe',
        'DOB': '01/01/1970',
        'course': 'Computer Science',
        'year': 2
}
```

I've also successfully completed the extra credit assignment using graphite to visualise the graph, statsd to accumulate the data and carbon-cache to aggregate data. Details on how to view this graph will be detailed in the extra credit section.

All Requests to the API should be sent to http://mohamey.me/api

API Requests

The code used to create the API can be found in the code submission, with it all being in main.js.

Quickstart:

Get Requests

All Get requests have two mandatory url parameters: **version** and **get**. **version** is in order to future proof the API, and currently the only version supported is 1.0. **get** is to specify whether you want to get a list of all students or just one, identified by their ID. As such there are only two options for **Get**, 'single' or 'all'. If requesting a single student's details, you must then provide the student's ID.

Example Get Single

http://mohamey.me/api?version=1.0&get=single&format=json&id=2

Example Get All

http://mohamey.me/api?version=1.0&get=all

Notice the parameter **format** is not necessary and will default to JSON. It is also case insensitive. At the moment, json is the only response format supported.

Post Requests

To create a new student, simply send a post request with the predefined student object as the data in JSON format to http://mohamey.me/api. No fields can be left out or the server will return an error message. The server will not allow a student to be created with an ID already belonging to another student.

Put Requests

Put requests are properly handled by the API. The format of the data to be sent with a PUT request is the same as with the POST request, where the student ID will be used to find the student whose details will be updated.

Currently the server responds to PUT requests properly, however an error occurs when trying to update the database. The API as a result sends the appropriate error message.

Delete Requests

In order to delete a student from the database, a JSON object of the form:

```
{
        'id': '23'
}
```

Should be sent to http://mohamey.me/api, where id references the ID number of the student to be deleted.

As with the PUT requests, while the API handles delete requests properly, deleting the student from the database throws an error. As a result, the appropriate error message is returned.

Response Times

There is no code submission for this part since I've deployed graphana on my server instead.

The graphana interface for showing response times can be accessed at http://mohamey.me:8081. However, since there's no one actually querying the API I have provided 4 scripts that can be run at the same time to generate live data to view on

Graphana. Below are some steps to generate and view the response times of the API. Please note the scripts must be executed using python3.

Step 1:

Navigate to the root directory of the challenge submission and using pip3 install requirements.txt. This is done by calling:

pip3 install -r requirements.txt

Step 2:

Navigate to the client directory and run all 4 python scripts in parallel. They output data received from the API so I recommend running them in separate terminal windows. Please note that Python3 is required to run these scripts. Leave them running until you're finished.

Step 3:

Open a browser and open http://mohamey.me:8081. This should open the Graphana interface. On the left you'll see a directory like structure. The response times are kept in 4 separate folders:

- Metrics/stats/timers/api/get
- Metrics/stats/timers/api/post
- Metrics/stats/timers/api/put
- Metrics/stats/timers/api/delete

Each of these folders have the menu option "mean". Select this option in all 4 directories and it will show the mean response times for the API for each of these request types in different colors.

Step 4:

By default, Graphana shows data from the previous 24 hours. To change this, click the clock symbol on top of the graph and change the time range to the last 5 minutes, or any range you'd prefer. Finally, click the auto-refresh button on the bottom of the graph to have the graph constantly update the volume.

Response times from the server are measured in Milliseconds