Explanation of my approach to the Backend Hiring Test for Turing Technologies.

The Task:

Build a Node JS app to get all the new contributors for a given repository name from github using the Github API.

What is a new contributor?

Anyone who has not made a contribution to the repository before.

Assumptions:

The year/month provided to the API will determine new contributors in that range. Example for if the year is given as 2022, all the unique contributors of that year will be returned. If a user has made a contribution in 2021 and not in 2022, they won't be a part of the result set and similarly if a user has made one contribution in 2021 and also 2022. They will also be counted once for the time period of 2022, as we aren't taking into account 2021 in this scenario.

For Example :

1. Let's assume a repository is named "node-api", owned by "master".
2. User1 made a commit to a repository named "node-api" in 2021.
3. User1 made a commit to a repository named "node-api" in 2022.
4. User2 made a commit to a repository named "node-api" in 2022.

If we use a get request "master/node-api/2022", we will get 2 unique contributors as User 1 and User 2 for 2022.

Similarly let's see another example:

5. User1 made a commit to a repository named "node-api" in 2022.
6. User1 made a commit to a repository named "node-api" in 2022.

If we use a get request "master/node-api/2022", we will get 1 unique contributor i.e. User 1.

The Project Structure:

I kept the structure of the project as efficient as possible, by that I mean that I broke down the code in modules.

API Calls:

This code is situated in the folder named “apiCall”. It contains the middleware that I wrote to fetch the data using HTTP requests from Github using the Github API. After getting the repository names and commit history. This data is placed into the database (MongoDB), this is done at every get request to make sure the data in the DB is up-to-date.

UTILS:

This folder contains the helper code for the API calling as well as the custom error handler that I’m using.

Models:

This folder contains two models according to my design. Note: I did not use all the data that the api returned instead only the data that was needed was saved.

Controllers:

This folder contains my controller for the app. This controller contains functions that are defined for the app and when the appropriate route is triggered, the appropriate controller function is called. Which then fetches the date from the database, after that certain operations are performed on the fetched data to make a response object. To get unique contributors, I have used Hashmap with a count so If the count is greater than one, I ignored that value as a contribution from that particular user is already in the hashmap. Moreover, the date of contribution was also taken into account to make the result set.

Middlewares:

This folder just contains a simple middleware that we need to use to set the header for the API calls to Github. This is as per documentation.

Routes:

This contains my route file, which defines three routes. The first route just takes the owner name and the result set of this get request will contain all the repositories belonging to that user. The second one takes the name of the owner, the name of the repository and the year, the result set of this particular get request will contain the unique contribution of that year. The third route

takes the name of the owner, the name of the repository, the year and the month the result set of this particular get request will contain the unique contribution of that year.

Entry Point:

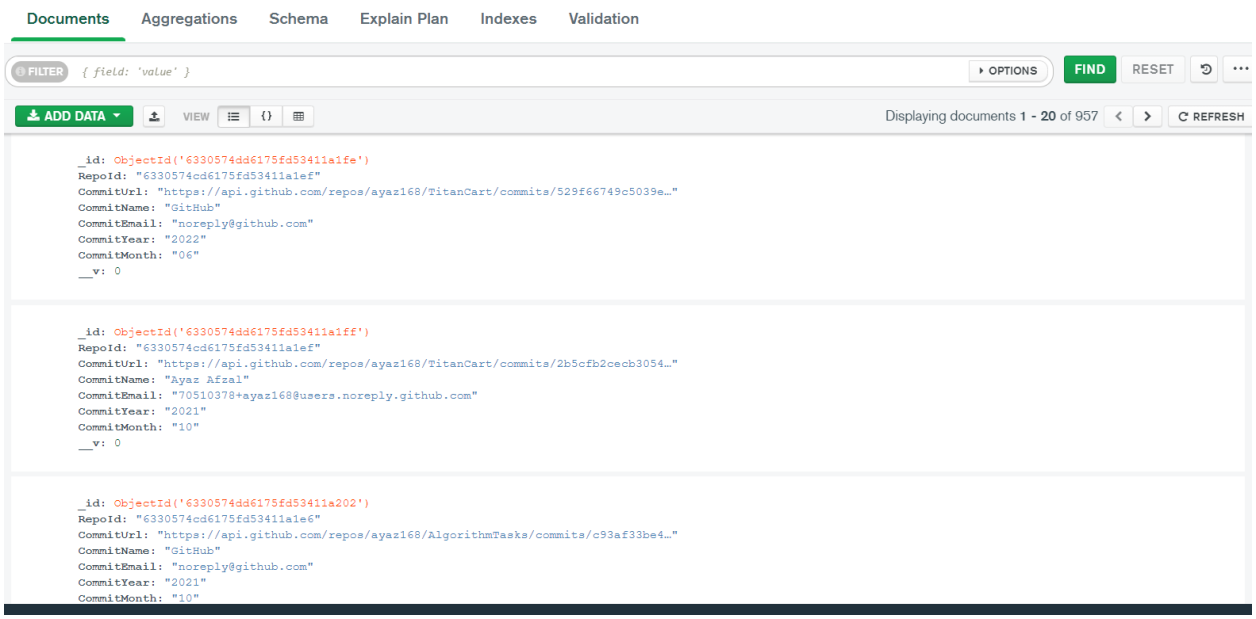
The entry point of this app is “app.js”. In this I have defined my app, connected it to a local database and defined the middlewares my app is using. These middlewares are “cors”, “makeGitHeader” and “limiter”, the “makeGitHeader” is used to set the header attributes needed to make api calls to Github and the “limiter” is used to limit the number of request that can be made, it is used as a protection measure and as of right now the max number of requests allowed are 20 request per 10 minute.

Improvements:

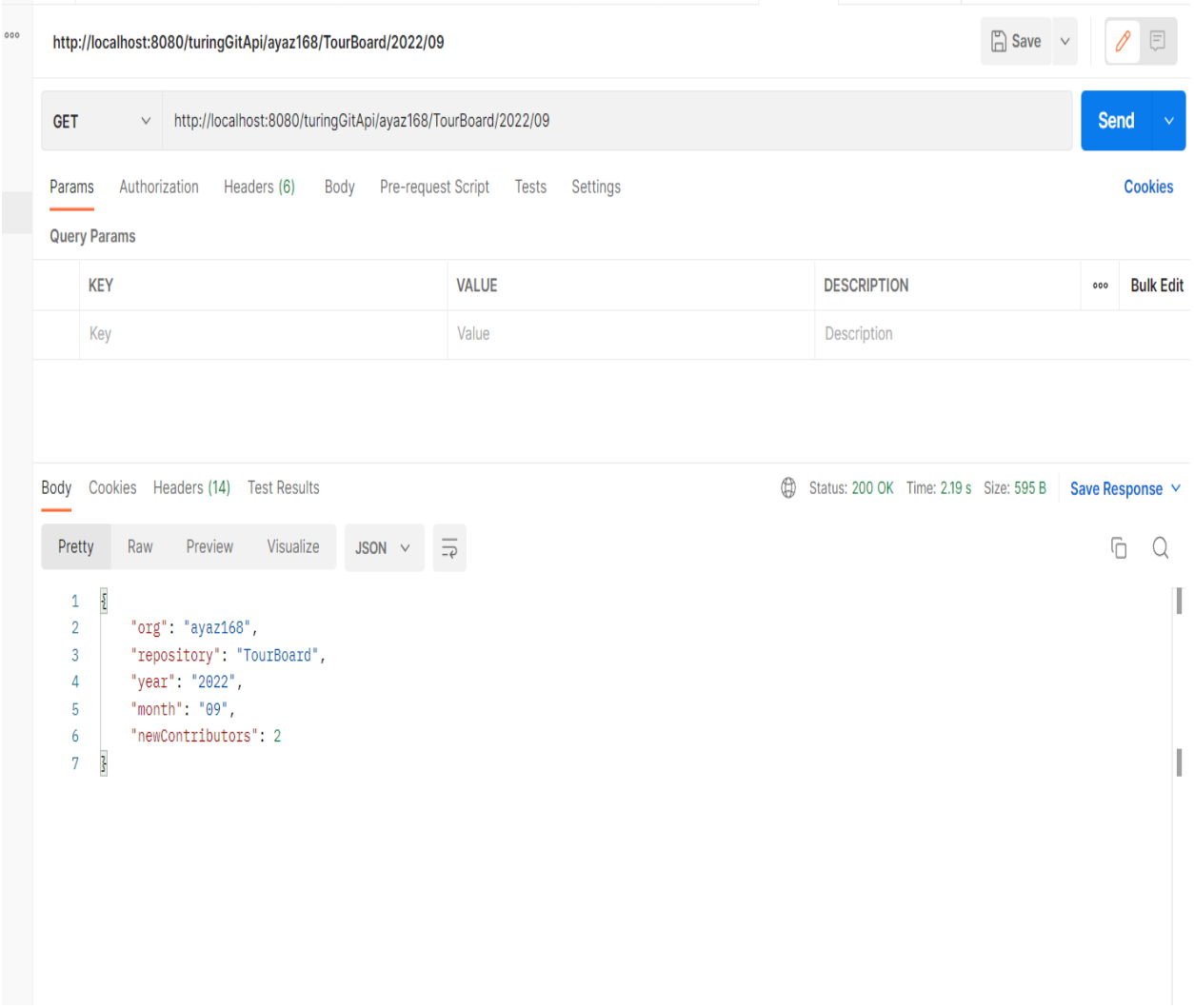
There’s room for improvement in how the data is stored in the database. Moreover, more bug testing needs to be done.

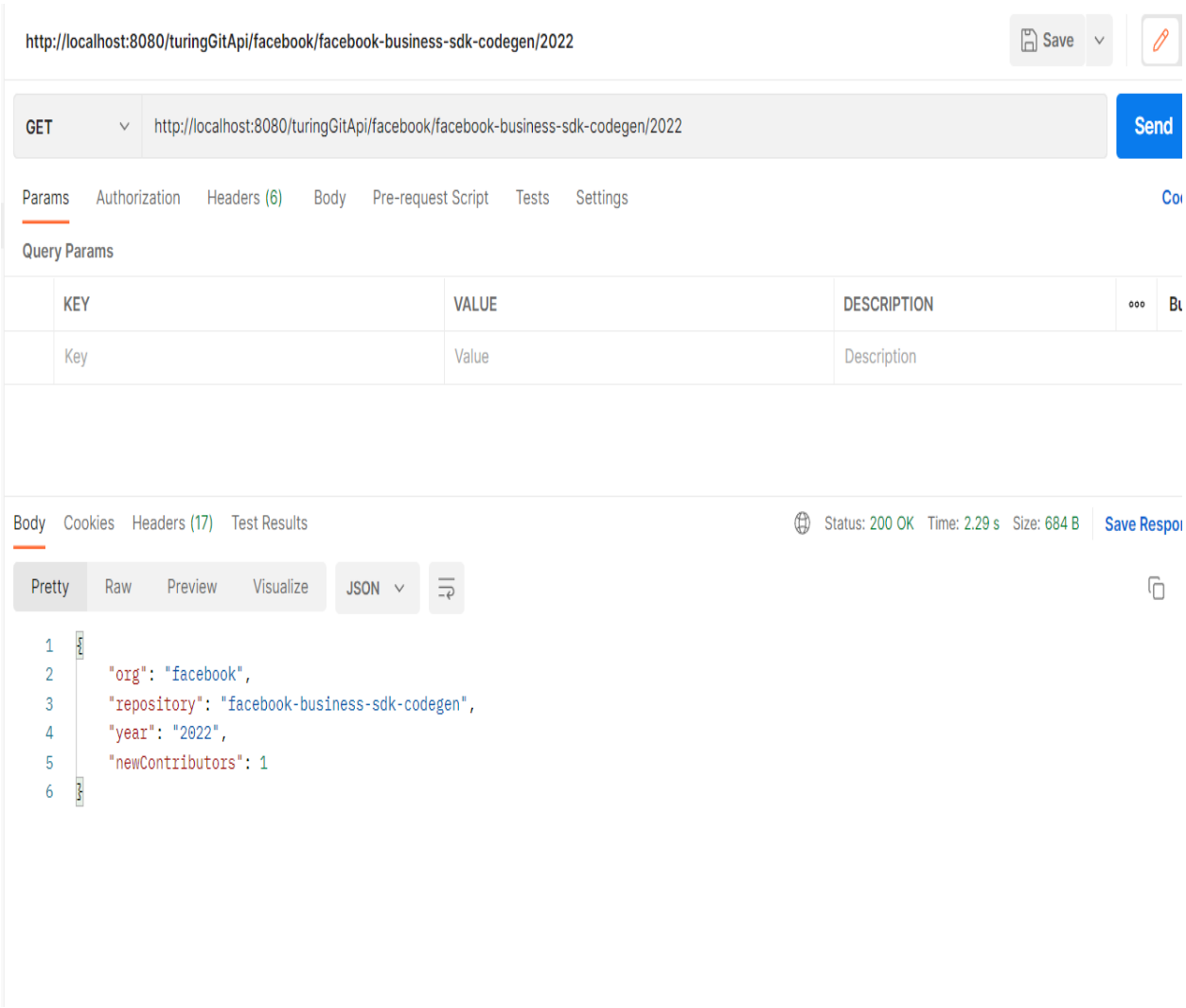
Result Screenshots:

Database:



Postman Requests:





Conclusion:

The app is working and the result set is accurate from the testing I have done but I feel like more testing needs to be done. Furthermore, I tried to break down the app in modules to ensure scalability. The code is well commented and I followed the “camelCase” style of variable names. To ensure maintainability I used a constant naming convention throughout the app, with comments explaining everything that is happening. I tried to group up similar functionality and employed the use of middlewares to prevent redundancy.

Dependencies:

- CORS
- dotenv
- express
- express-rate-limit
- mongoose
- nodemon

To Run:

- Make an “.env” file in the root directory.
    - Add to the file MY_TOKEN:<Your GITHUB TOKEN>
    - Add to the file PORT:<Your Preferred Port Number>
    - Add to the file DB_CON=<Your Local MONGODB URL/<DB NAME>>
- Run “npm install”
- Run “npm start”

Name: Ayaz Afzal
Email:ayazafzal168@gmail.com
Phone:+923481512600