2019

CAB230 Assignment

SERVER SIDE REPORT

AUTHOR: ANYE ZHU (RENNY)

STUDENT NUMBER: N10322434

DATE OF SUBMISSION: 2ND JUNE

Table of Contents

1. Introduction

1.1. Authentication routes

Register: Register is implemented.
Login: Login is implemented.

1.2. Search and Helpers routes

Search: After authenticated with token, the search function could be used.
Helpers: Offences, areas, ages, genders and years could always be used.

1.3. Security

Middleware security: helmet is implemented.
HTTPS: HTTPS is used, which is much more secure than HTTP.
JWT: JWT is used for security.

1.4. Data connectivity

Use MySQL Workbench 8.0 CE to provide and save data.

1.5. Swagger docs

Swagger docs is implemented.

1.6. Middleware

Knex (https://knexjs.org/) is used, which is a well-established middleware.

2. Technical Description of the Application

2.1. Authentication routes

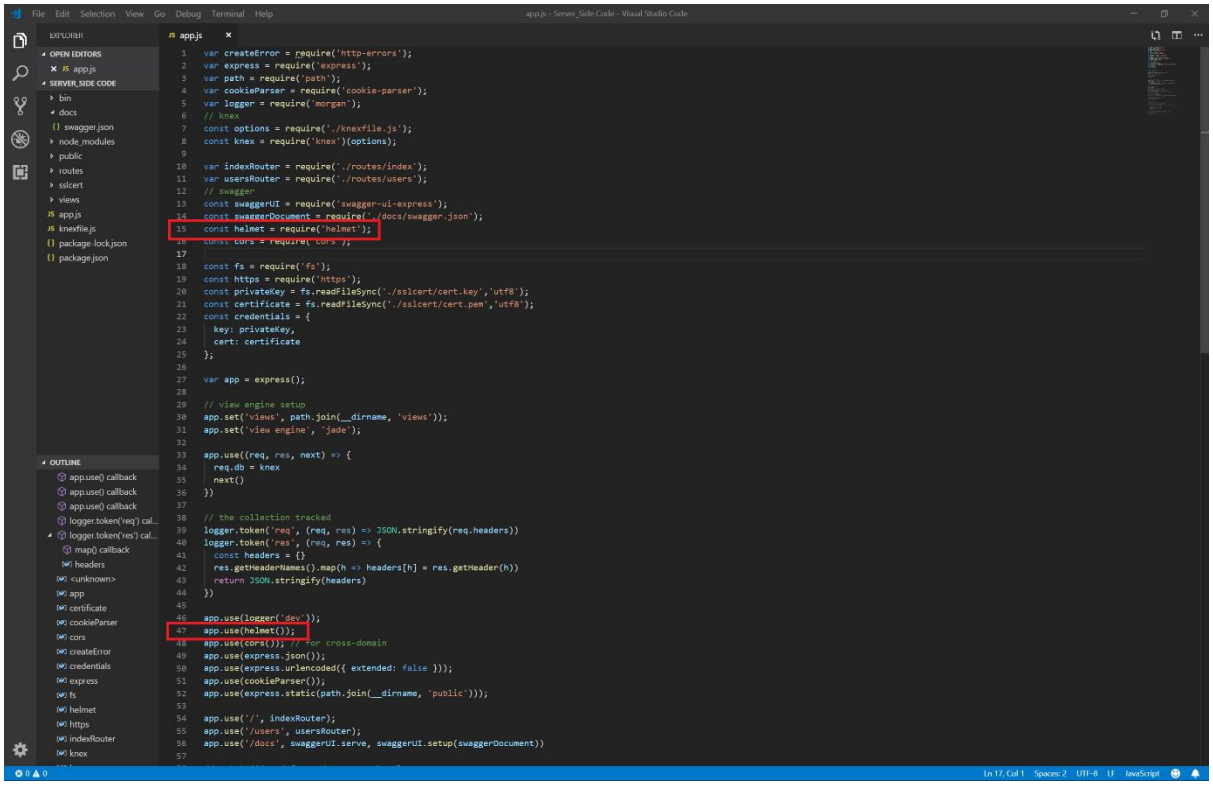
Register: The URL is https://localhost/register. When users register successfully, email and password will be saved into database.
Login: The URL is https://localhost/login. When users input the email and password, if the email and password have already existed in database, then users could get the token, and could use the following function, search.

2.2. Information routes

Search: The URL is https://localhost/search. User needs to login firstly and then gets the token. Then user can use search function in the client side. When the user enters some keywords and click the search button, the data would be selected from database. Next, the client-side responses the results.
Helpers: The URLs are https://localhost/offences, https://localhost/areas, https://localhost/ages, https://localhost/genders, https://localhost/years. These information routes are to select data from database and return the results.

2.3. Application Middleware

Knex and Helmet. *Actually, I already used 'helmet' before demoing on Friday, but I forgot to show the tutor the version with 'helmet', which made me lose the score for this part. I hope I can get the score of this part.*

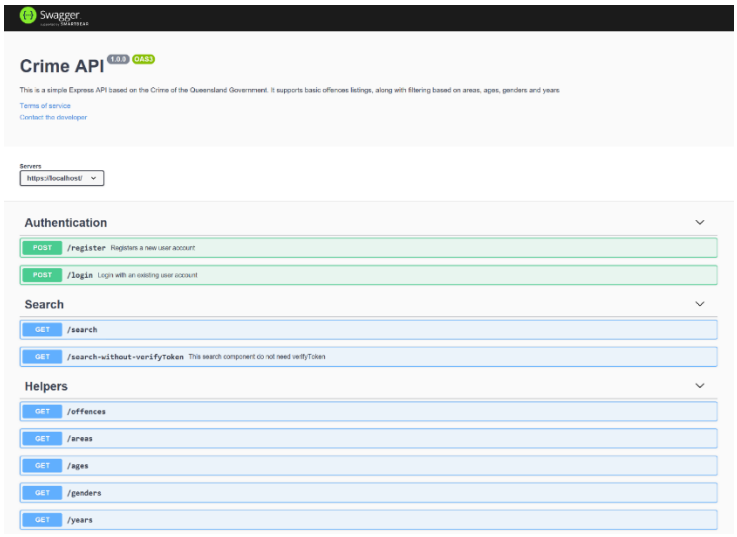


2.4. Data connectivity

Use MySQL Workbench 8.0 CE to provide and save data.

2.5. Swagger docs

The URL is https://localhost/docs/. The user can see the APIs in this website. This is swagger UI (https://swagger.io/tools/swagger-ui/), it is for visualizing and interacting with the API's resources.



3. Security

Middleware security: Knex, Helmet.

Https: HTTPS is used for secure communication. It is the standard security technology that establishes an encrypted connection between a web server and a browser.

JWT: JWT is used for security, because only the server should know the "secret" that is used to generate the JWT

4. Testing and limitations

Test/results example in the table

Action / Test/results	4.1. Register	4.2. Login	4.3. offences	4.4. Search	4.5. Search (Filters)
Test example	email: demo01-user@gmail.com password: userpwd123	email: demo01-user@gmail.com password: userpwd123	Click on the offences button	offence: Assault	offence: Assault area: Brisbane City Council age: Adult gender: Female year: 2001
Results - Client side	Register successfully	Login successfully	Show the offences table	Show the table with the search result	Show the table with the search result
Results - Server side	Email and password are created (201)	Login is OK (200)	Offence is OK (304)	Search is OK (200)	Search with these filters is OK (200)
Results - Database	Email and password are saved to database	Check the same data in the database as the entered email and password	Get offence data from database	Get the search result from database	Get the search results from database

4.1. Register

Client side:

Server side:


```
POST /register 201 15.364 ms - {"x-dns-prefetch-control":"off","x-frame-options":"SAMEORIGIN","strict-transport-security":"max-age=15552000; includeSubDomains","x-download-options":"noopen","x-content-type-options":"nosniff","x-xss-protection":"1; mode=block","access-control-allow-origin":"*","content-type":"application/json; charset=utf-8","content-length":"70","etag":"W/\"46-2kzoy+9xUSuF8K/71GGISxEDhFk\""}
```

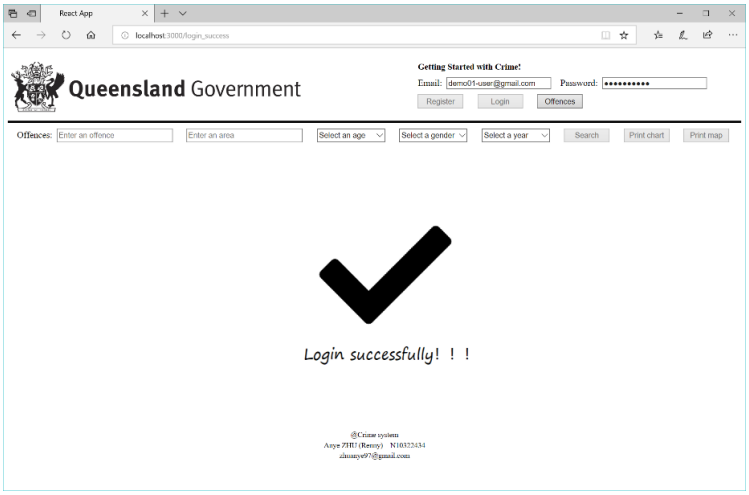

Database:

demo01-user@gmail.com	userpwd123	2019-06-01 01:21:07	2019-06-01 01:21:07

So, the /register part can run successfully on both the client side and the server side.

4.2. Login

Client side:



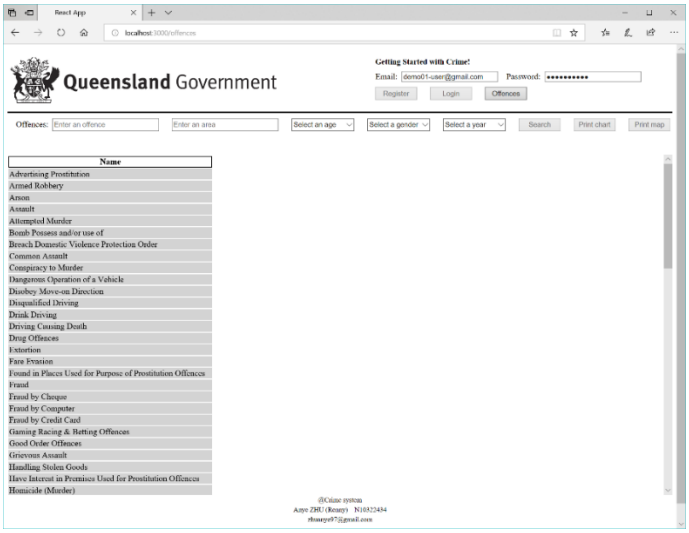
Server side:


```
POST /login 200 2.512 ms - {"x-dns-prefetch-control":"off","x-frame-options":"SAMEORIGIN","strict-transport-security":"max-age=15552000; includeSubDomains","x-download-options":"noopen","x-content-type-options":"nosniff","x-xss-protection":"1; mode=block","access-control-allow-origin":"*","content-type":"application/json; charset=utf-8","content-length":"215","etag":"W/\"d7-oD22aPgQxR3nIAEPO2jMrHc03kY\""}
```


As can be seen from the pictures above, the /login part can work successfully as well.

4.3. Offences

Client side:



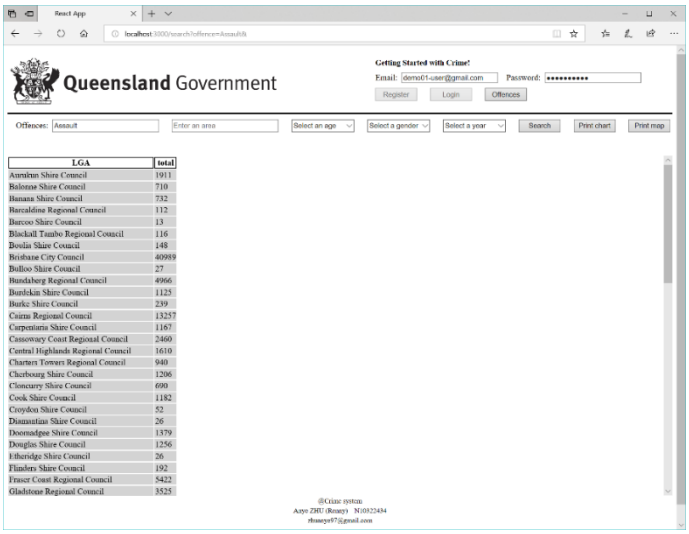
Server sides:


```
GET /offences 304 5.107 ms - {"x-dns-prefetch-control":"off","x-frame-options":"SAMEORIGIN","strict-transport-security":"max-age=15552000; includeSubDomains","x-download-options":"noopen","x-content-type-options":"nosniff","x-xss-protection":"1; mode=block","access-control-allow-origin":"*","etag":"W/\"960-O4zLdTMUyLvNb4Bvz7naOH7ubZs\""}
```


The code returned by the server is 304, which is different from the code that needs to be returned (returns 200). So, although this part work successfully, it is not perfect.

4.4. Search

Client side:



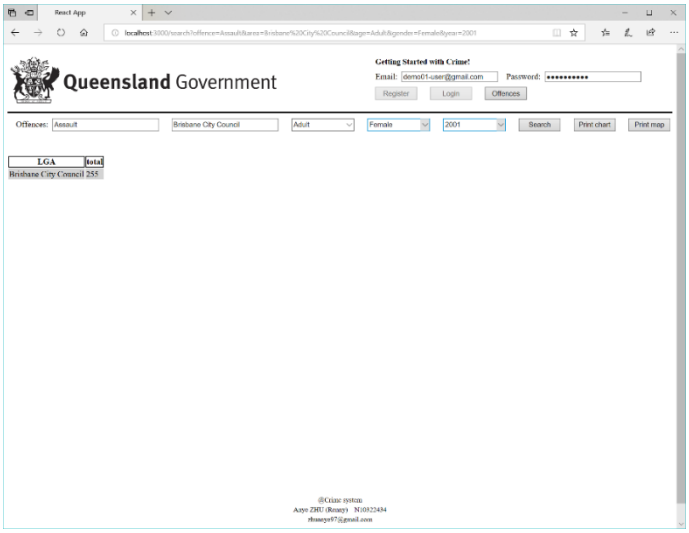
Server side:

```
GET /search?offence=Assault& 200 254.030 ms - {"x-dns-prefetch-control":"off","x-frame-options":"SAMEORIGIN","strict-transport-security":"max-age=15552000; includeSubDomains","x-download-options":"noopen","x-content-type-options":"nosniff","x-xss-protection":"1; mode=block","access-control-allow-origin":"*","content-type":"application/json; charset=utf-8","content-length":"3745","etag":"W/\"ea1-mQ95sH9m5WyJt5doJGom0jSGg5M\""}
```

The returned code in the search section meets the requirement (return 200). And this part could run successfully.

4.5. Search (Filters)

Client side:



Server side:

```
GET /search?offence=Assault&area=Brisbane%20City%20Council&age=Adult&gender=Female&year=2001 200 198.416 ms - {"x-dns-prefetch-control":"off","x-frame-options":"SAMEORIGIN","strict-transport-security":"max-age=15552000; includeSubDomains","x-download-options":"noopen","x-content-type-options":"nosniff","x-xss-protection":"1; mode=block","access-control-allow-origin":"*","content-type":"application/json; charset=utf-8","content-length":"56","etag":"W/\"38-ZGRrxZfeHJ8kyYEhx3PFWlwDtLE\""}
```

The returned code meets the requirement (return 200) when the search section runs with filters. This part can work successfully.

5. References

Swagger UI, 2019. Retrieved from https://swagger.io/tools/swagger-ui/

KNEX.JS, 2019. Retrieved from https://knexjs.org/

Helmet, 2019. Retrieved from https://helmetjs.github.io/

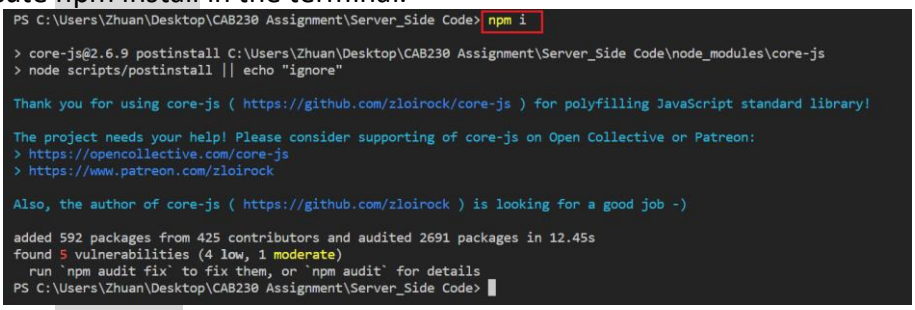
Jsonwebtoken, 2019. Retrieved from https://www.npmjs.com/package/jsonwebtoken
Download MySQL Installer, 2019. Retrieved from https://dev.mysql.com/downloads/windows/installer/8.0.html
What's the Difference between HTTP and HTTPS?, 2019. Retrieved from https://www.globalsign.com/en/blog/the-difference-between-http-and-https/
How safe is JWT?, 2019. Retrieved from https://stackoverflow.com/questions/35817325/how-safe-is-jwt

6. Appendix

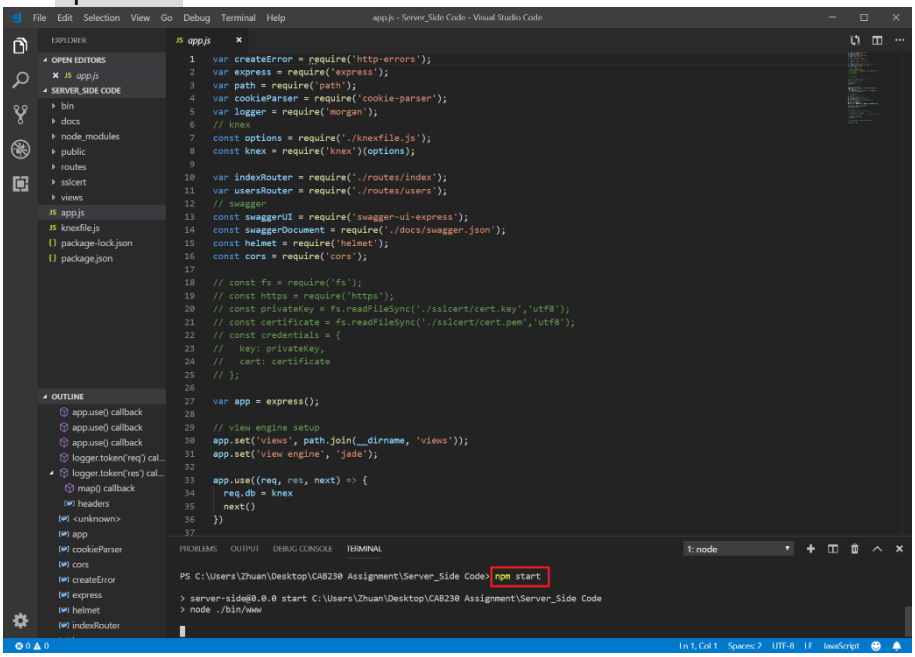
6.1. Installation guide

Step1: Open server file with VS Code.

Step2: Execute npm install in the terminal.



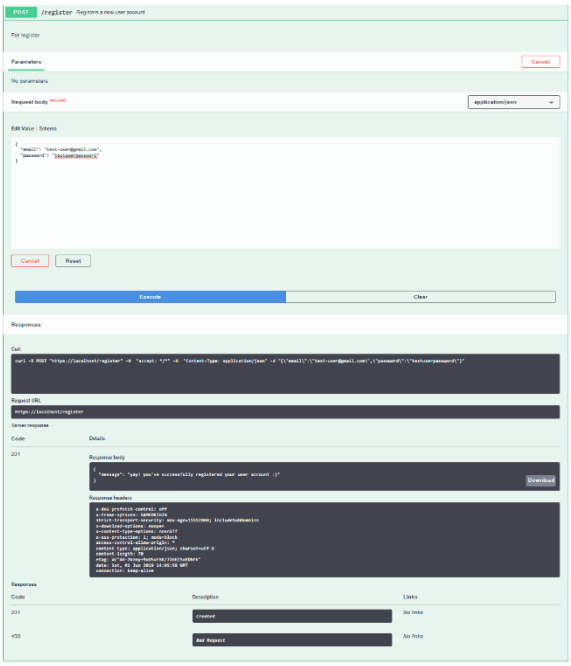
Step3: Execute npm start in the terminal to start the server.

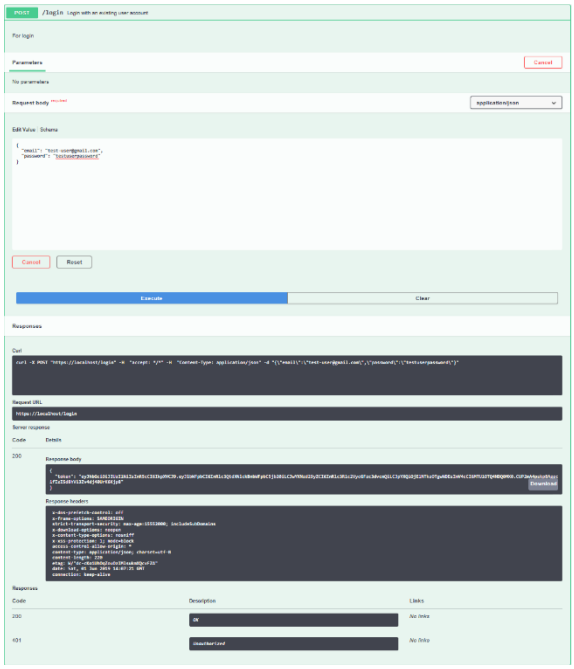


Then, the server starts up.

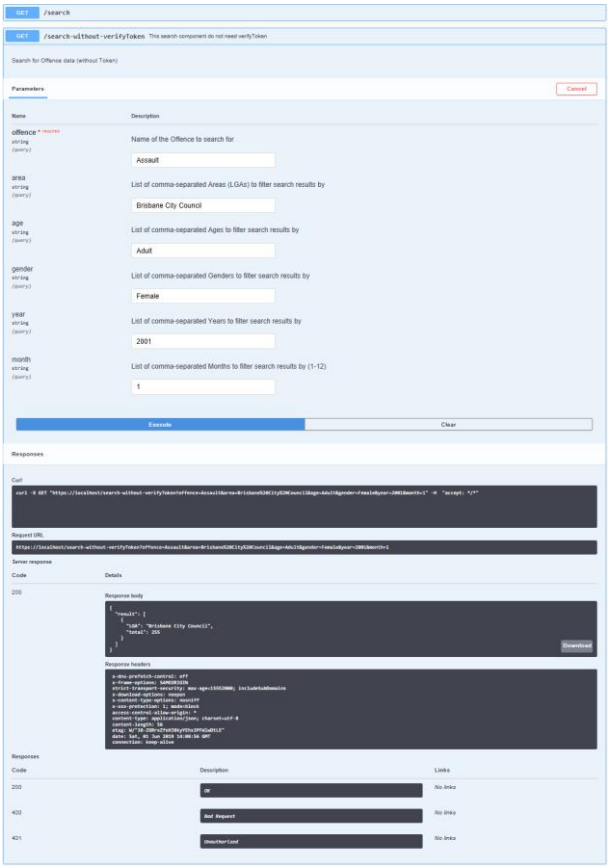
6.2. Swagger UI (https://localhost/docs/)

Register: https://localhost/register Login: https://localhost/login

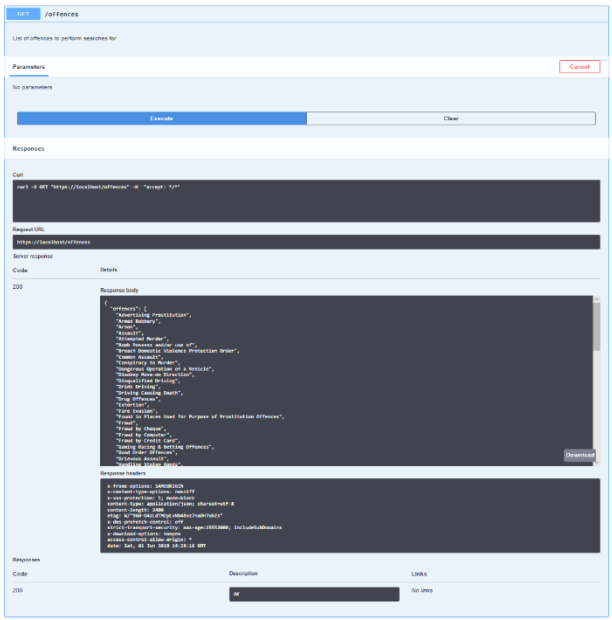




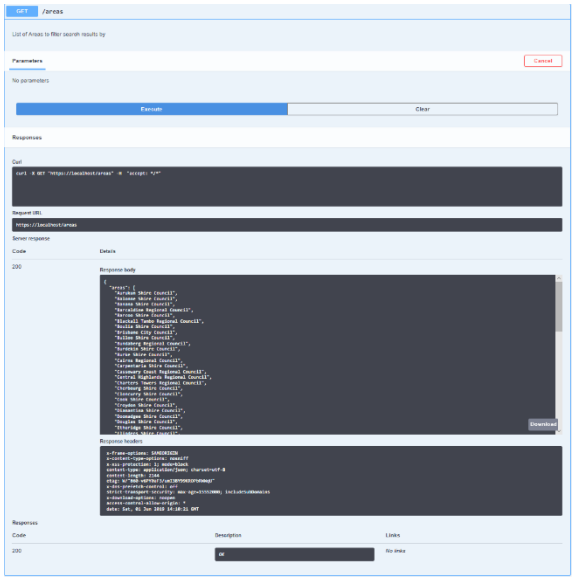
Search: https://localhost/search. This part needs the token, so I would test this function with the other route https://localhost/search-without-verifyToken



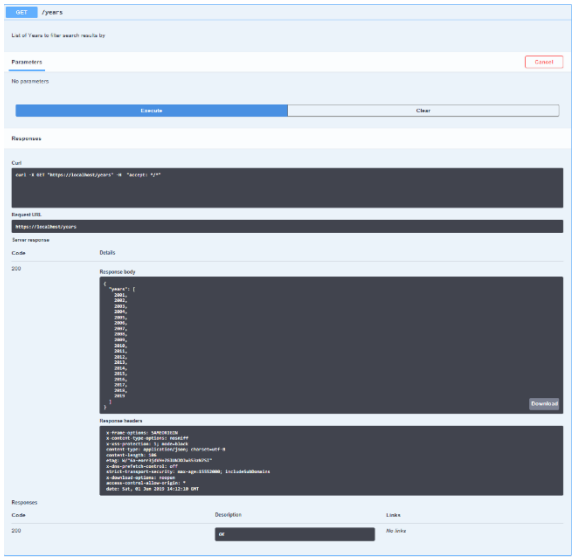
Offences: https://localhost/offences



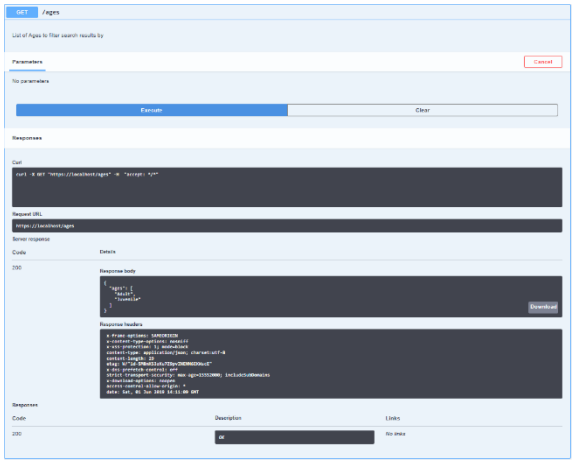
Areas: https://localhost/areas



Years: https://localhost/years



Ages: https://localhost/ages



Genders: https://localhost/genders