



CAB230 WEB COMPUTING REPORT

Client and Server Side

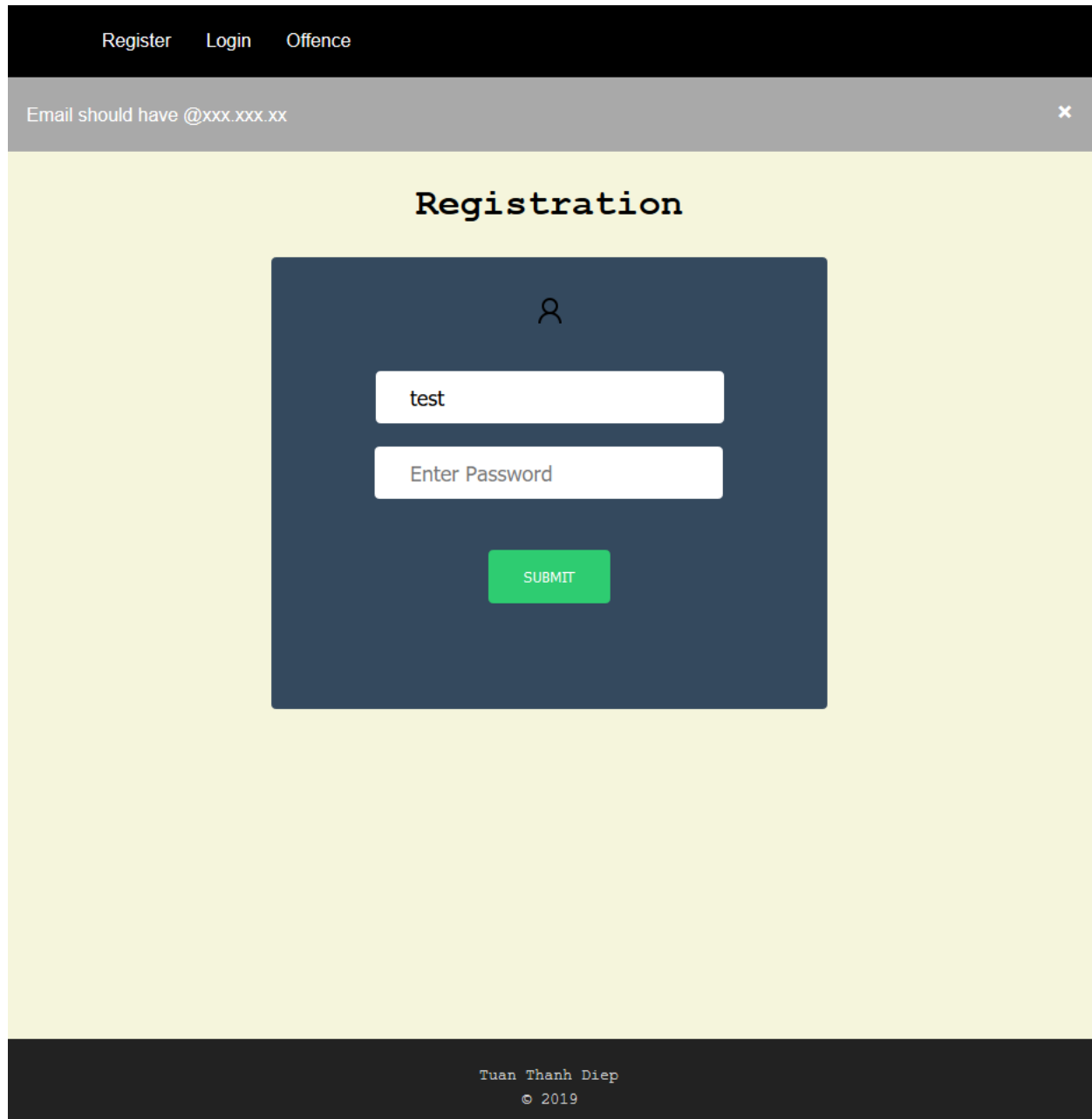
Tuan Thanh Diep
Student ID: n9607234

Table of Contents

1. Client Side	2
2. Server Side	8

1. Client Side

1.1 Registration Page



The screenshot displays a web application interface. At the top, a black navigation bar contains the links "Register", "Login", and "Offence". Below this, a grey banner displays an error message: "Email should have @xxx.xxx.xx" with a close button (X) on the right. The main content area has a light yellow background and is titled "Registration" in a bold, black, monospace font. Centered on this page is a dark blue registration form. The form includes a user icon at the top, a text input field containing the word "test", another text input field with the placeholder text "Enter Password", and a green "SUBMIT" button at the bottom. The footer of the page is a black bar containing the text "Tuan Thanh Diep" and "© 2019".

Figure 1 Registration Page

When user register an existing account or with a non-email username, an alert appear as the figure display above. However, with a successful registration, the page will redirect to the login page where your details are automatically filled.

1.2 Login Page

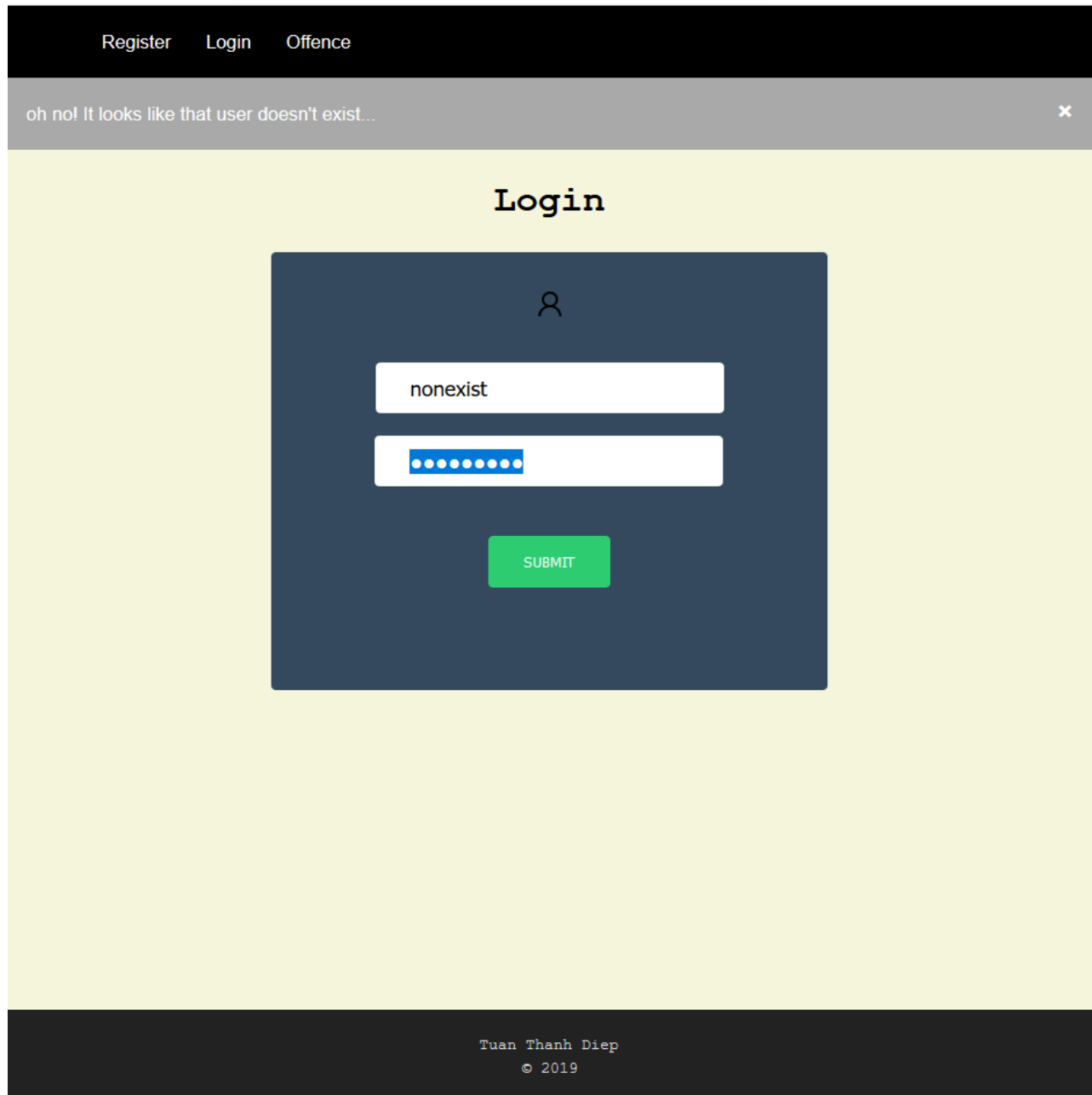


Figure 2 Login Page

Similarly, the login site will have the similar alert for invalid authentication such as entering a non-existing account, wrong password or email. When the user has successfully log in, a JWT token is received and the user will be redirected to the Offence page where a list of offences is displayed in a table as presented in **Figure 3**. The user is no longer able to access to both register and login page until they have successfully log out.

1.3 Offence Page

Offence	Search	Logout
Offence		
Offences		
Advertising Prostitution		
Armed Robbery		
Arson		
Assault		
Attempted Murder		
Bomb Possess and/or use of		
Breach Domestic Violence Protection Order		
Common Assault		
Conspiracy to Murder		
Dangerous Operation of a Vehicle		
Disobey Move-on Direction		
Disqualified Driving		
Drink Driving		
Driving Causing Death		
Drug Offences		
Extortion		
Fare Evasion		
Tuan Thanh Diep © 2019		

Figure 3 Offence Page

The Offence page is accessible for unauthenticated and authenticated users.

1.4 Search Page

The screenshot shows a web application interface for a search page. At the top, there is a black navigation bar with the text "Offence" and "Search" in white, and a green "Logout" button on the right. Below the navigation bar, the main content area has a light yellow background. The word "Search" is centered at the top of this area. On the left side, there are four input fields labeled "Area", "Age", "Gender", and "Year". To the right of these fields is a large white search bar. Below the search bar is a green "Search" button. Below the input fields is a green "Apply Filters" button. At the bottom of the page, there is a black footer bar with the text "Tuan Thanh Diep" and "© 2019" in white.

Figure 4 Search Page

The Search page presented in **Figure 4**, requires access token after log in authentication in order to have the accessibility. When an offence is entered in the search bar and submitted, a bar chart displaying the total of cases against the locations of that particular offence. Additionally, a world map will be displayed with markers generated by the latitude and long from the retrieved API data from the search. An example from a search result is in **Figure 5**.

1.5 Invalid Access Attempts

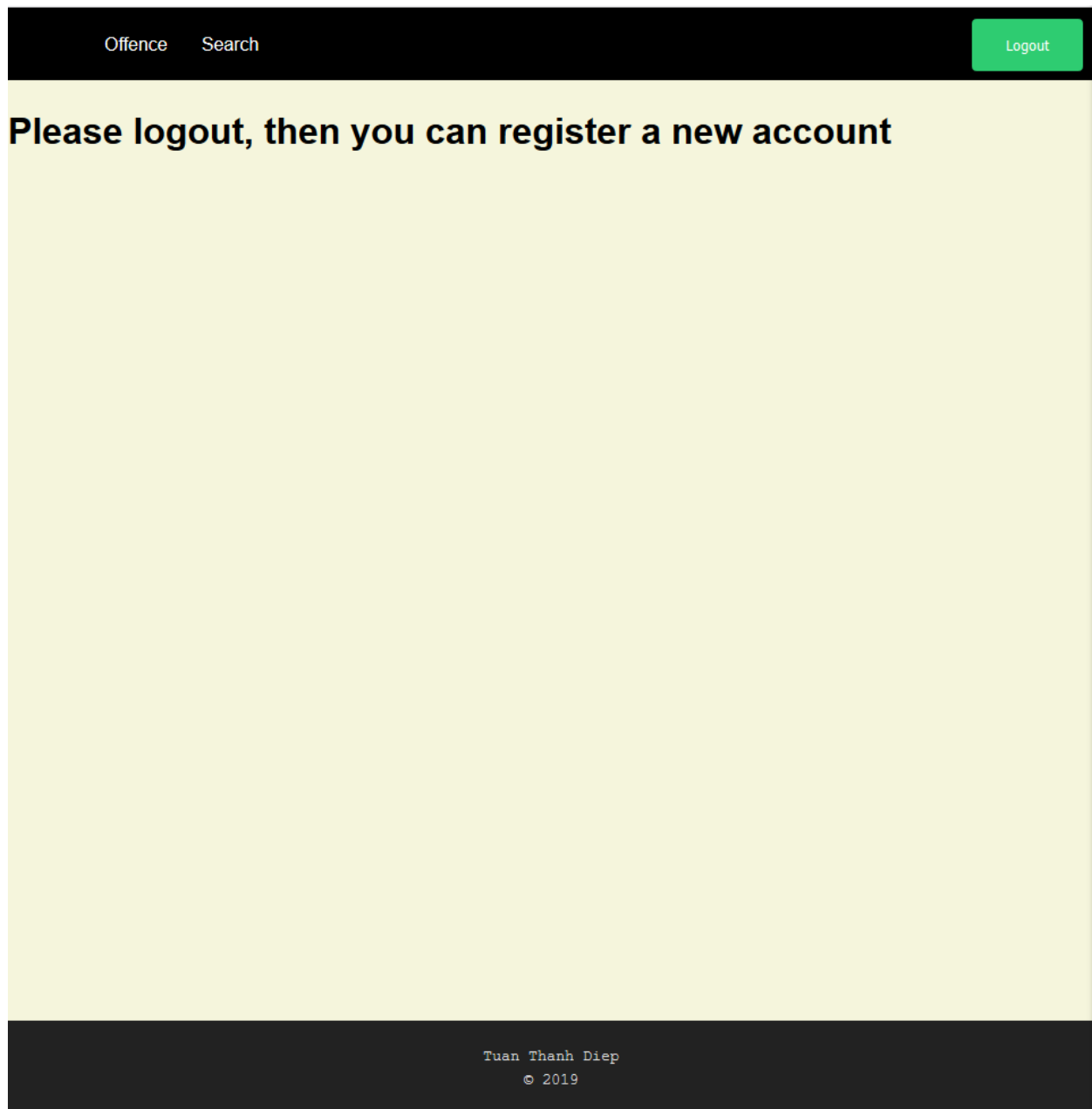


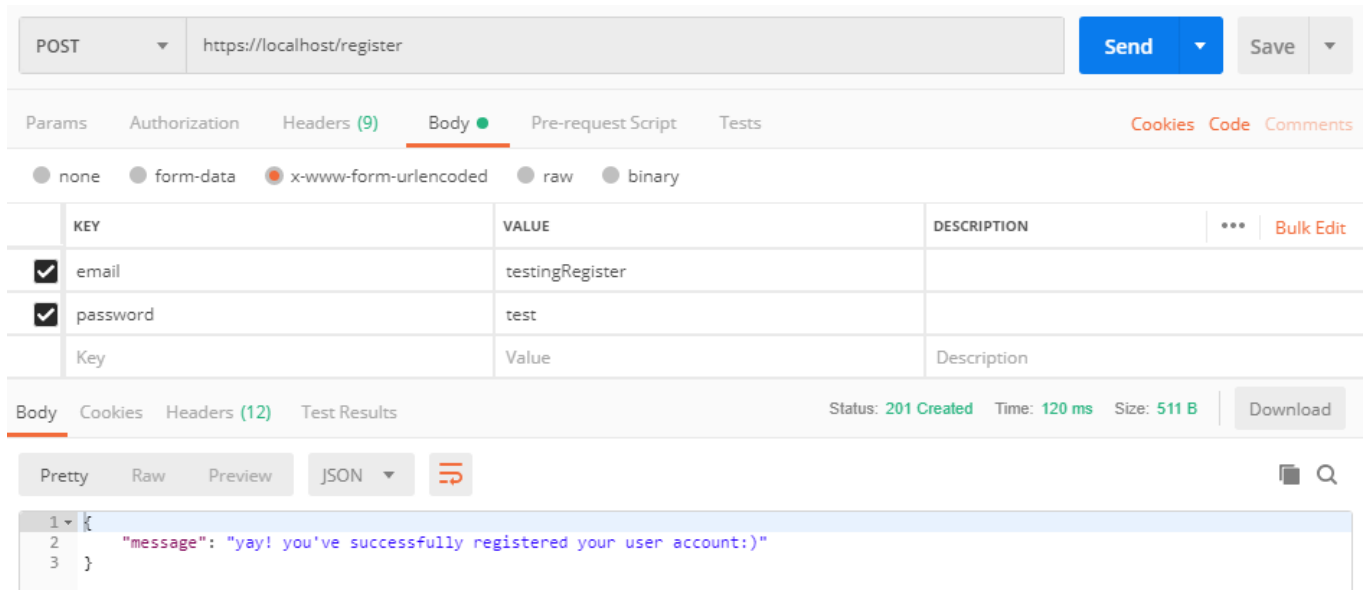
Figure 6 An attempt of accessing Login or Register when logged in

As obvious as it is, user can enter Login and Registration page route into the browser. In **Figure 6**, the user has logged in and attempted to access the register and login while are still logged in. Similarly, when accessing the search page when not logged in, a similar block page appears like above.

2. Server Side

2.1 Routes

2.1.1 /Register



The screenshot shows a Postman interface for a POST request to `https://localhost/register`. The request body is set to `x-www-form-urlencoded` and contains the following data:

KEY	VALUE	DESCRIPTION
email	testingRegister	
password	test	

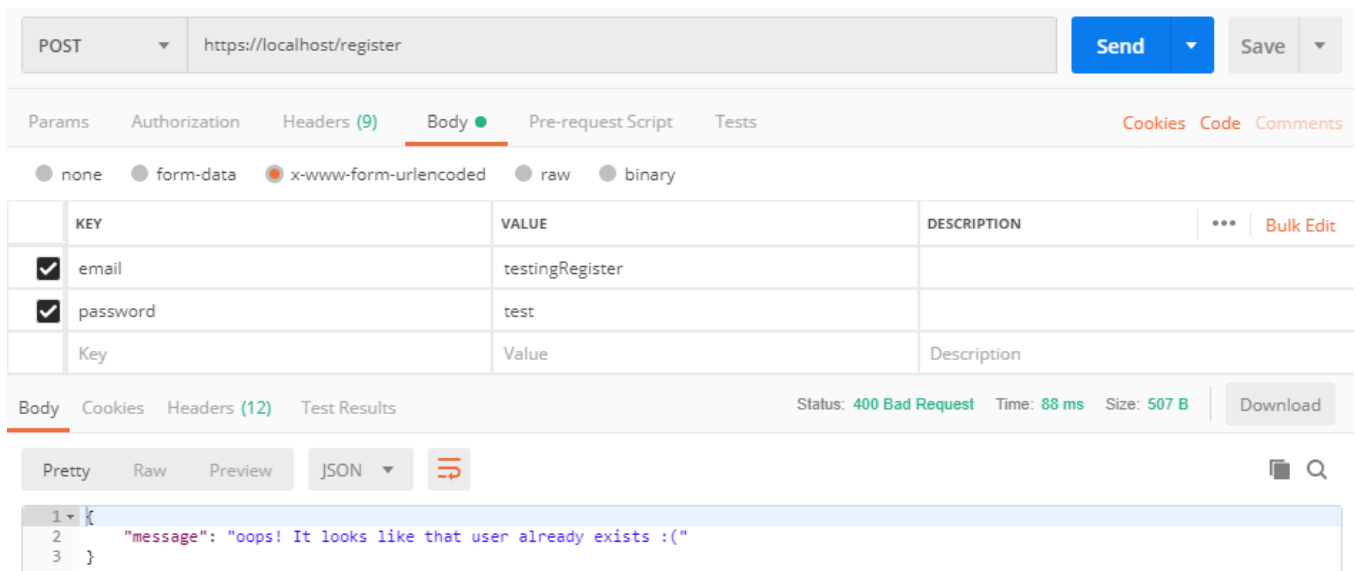
The response is displayed in JSON format:

```
{  "message": "yay! you've successfully registered your user account:"}
```

Response status: 201 Created, Time: 120 ms, Size: 511 B.

Figure 7 A POST request on Register route

In **Figure 7**, a POST request was made to the register route with a new account details embedded the body using Postman application. Hitting the same request with the same account details, the database will respond with an alert message like below



The screenshot shows a Postman interface for a POST request to `https://localhost/register`. The request body is set to `x-www-form-urlencoded` and contains the following data:

KEY	VALUE	DESCRIPTION
email	testingRegister	
password	test	

The response is displayed in JSON format:

```
{  "message": "oops! It looks like that user already exists :("}
```

Response status: 400 Bad Request, Time: 88 ms, Size: 507 B.

Figure 8 Database response for creating an existing account

2.1.2 /Login

POST https://localhost/login

Send Save

Params Authorization Headers (9) **Body** Pre-request Script Tests Cookies Code Comments

none form-data **x-www-form-urlencoded** raw binary

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> email	testingRegister			
<input checked="" type="checkbox"/> password	test			
Key	Value	Description		

Body Cookies Headers (12) Test Results Status: 200 OK Time: 103 ms Size: 702 B Download

Pretty Raw Preview JSON

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
   .eyJ1c2VyRGV0YWlscyI6eyJ1bmFpbCI6InRlc3RpbmdSZWdpc3R1ciIsIn8hcn3N3b3JkIjoidGVzdCJ9LCJpYXQiOiJlNTkyODI3MzUsImV4cCI6MTU1OTI4Mjg
   yMX0._ANat3Nt-dn5ukh5vxj8uAD-bxxp6QVDpyr8YKcyh-Q",
3   "token_type": "Bearer",
4   "expires_in": "86400"
5 }
```

Figure 9 Login with the new account

“token”, “token_access” and “expires_in” are return on a 200 response and the token has a time limit for 24 hours.

POST https://localhost/login

Send Save

Params Authorization Headers (9) **Body** Pre-request Script Tests Cookies Code Comments

none form-data **x-www-form-urlencoded** raw binary

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> email	testingRegister99			
<input checked="" type="checkbox"/> password	test			
Key	Value	Description		

Body Cookies Headers (12) Test Results Status: 401 Unauthorized Time: 27 ms Size: 489 B Download

Pretty Raw Preview JSON

```
1 {
2   "message": "invalid login - bad password"
3 }
```

Figure 10 Status 401 Unauthorized

Logging in with incorrect account details, user will be returned with a JSON message and a 401-error response.

2.1.3 /Offences

GET Send Save

Params Authorization Headers (9) **Body** Pre-request Script Tests Cookies Code Comments

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	email	testingRegister			
<input checked="" type="checkbox"/>	password	test			
	Key	Value	Description		

Body Cookies Headers (12) Test Results Status: 200 OK Time: 35 ms Size: 2.77 KB Download

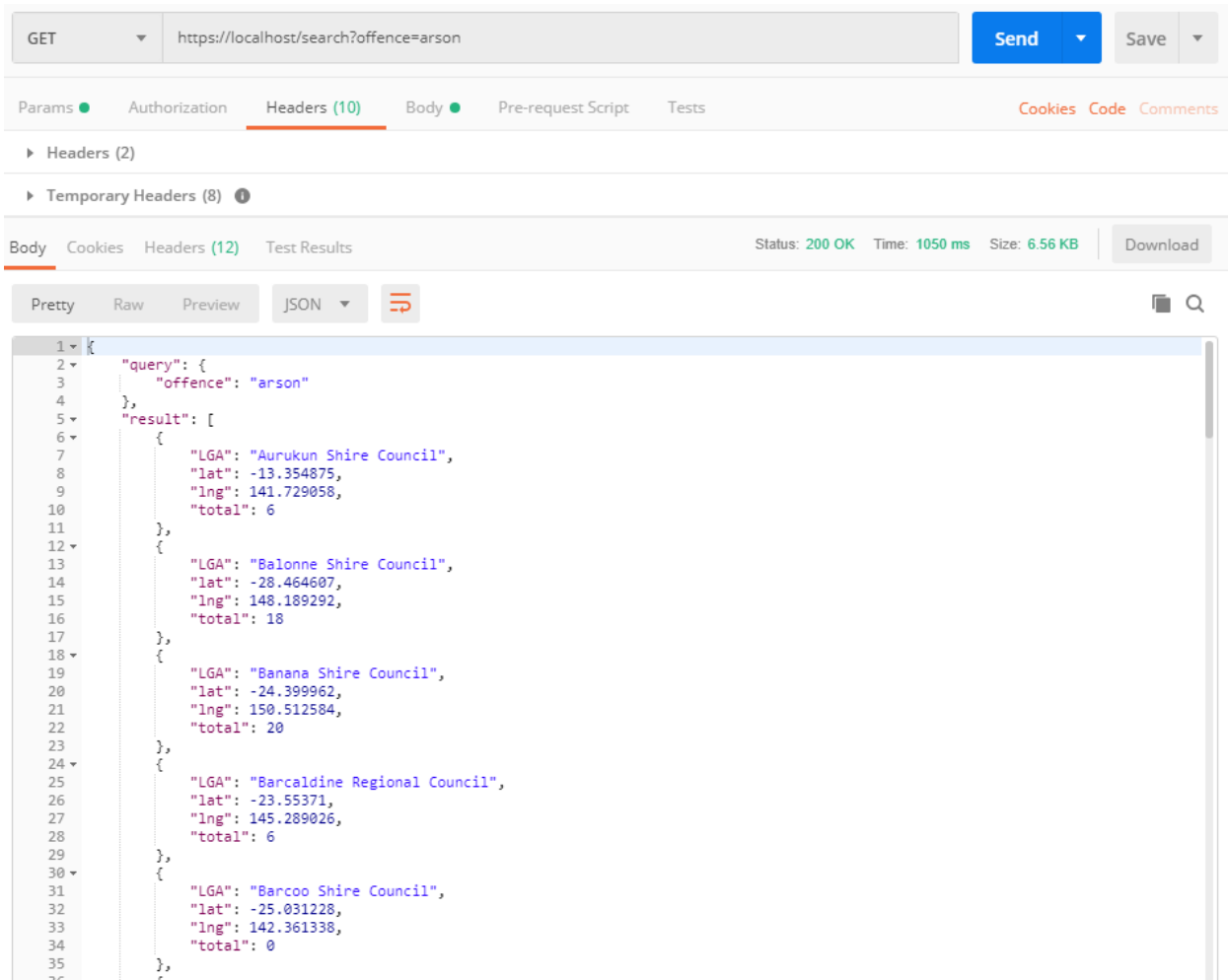
Pretty Raw Preview **JSON** ↺ 🔍

```
1 {
2   "offences": [
3     "Advertising Prostitution",
4     "Armed Robbery",
5     "Arson",
6     "Assault",
7     "Attempted Murder",
8     "Bomb Possess and/or use of",
9     "Breach Domestic Violence Protection Order",
10    "Common Assault",
11    "Conspiracy to Murder",
12    "Dangerous Operation of a Vehicle",
13    "Disobey Move-on Direction",
```

Figure 11 GET request at Offences route

By hitting the offences route, the response is a JSON with an array of all the offences in the database.

2.1.4 /Search



The screenshot shows a web browser interface for a REST client. The URL bar displays `https://localhost/search?offence=arson`. The `Send` button is highlighted. Below the URL bar, the `Headers (10)` tab is selected. The `Body` tab is also visible. The response status is `200 OK`, with a time of `1050 ms` and a size of `6.56 KB`. The response body is displayed in the `JSON` tab, showing a JSON object with a `query` property and a `result` array. The `result` array contains five objects, each representing a Local Government Area (LGA) with its name, latitude, longitude, and a `total` value.

```
1 {
2   "query": {
3     "offence": "arson"
4   },
5   "result": [
6     {
7       "LGA": "Aurukun Shire Council",
8       "lat": -13.354875,
9       "lng": 141.729058,
10      "total": 6
11    },
12    {
13      "LGA": "Balonne Shire Council",
14      "lat": -28.464607,
15      "lng": 148.189292,
16      "total": 18
17    },
18    {
19      "LGA": "Banana Shire Council",
20      "lat": -24.399962,
21      "lng": 150.512584,
22      "total": 20
23    },
24    {
25      "LGA": "Barcaldine Regional Council",
26      "lat": -23.55371,
27      "lng": 145.289026,
28      "total": 6
29    },
30    {
31      "LGA": "Barcoo Shire Council",
32      "lat": -25.031228,
33      "lng": 142.361338,
34      "total": 0
35    },
36  ]
37 }
```

Figure 12 A result of the protected route Search without filters

By using knex and some middleware, the search route is protected route which only allow authorized users to have access. Moreover, adding more parameters such as “&area=”, “&age=”, “&gender=” and “&year=” will filter out the results from **Figure 12**.

GET https://localhost/search?offence=arson&age=adult&year=2001 Send Save

Params Authorization Headers (10) Body Pre-request Script Tests Cookies Code Comments

▶ Headers (2)

▶ Temporary Headers (8)

Body Cookies Headers (12) Test Results Status: 200 OK Time: 215 ms Size: 6.53 KB Download

Pretty Raw Preview JSON

```
1 {
2   "query": {
3     "offence": "arson",
4     "age": "adult",
5     "year": "2001"
6   },
7   "result": [
8     {
9       "LGA": "Aurukun Shire Council",
10      "lat": -13.354875,
11      "lng": 141.729058,
12      "total": 0
13    },
14    {
15      "LGA": "Balonne Shire Council",
16      "lat": -28.464607,
17      "lng": 148.189292,
18      "total": 0
19    },
20    {
21      "LGA": "Banana Shire Council",
22      "lat": -24.399962,
23      "lng": 150.512584,
24      "total": 2
25    },
26    {
27      "LGA": "Barcaldine Regional Council",
28      "lat": -23.55371,
29      "lng": 145.289026,
30      "total": 0
31    },
32    {
33      "LGA": "Barcoo Shire Council",
34      "lat": -25.031228,
35      "lng": 142.361338,
36      "total": 0
37    },
38    {
39      "LGA": "Blackall Tambo Regional Council",
40      "lat": -24.42259,
```

Figure 13 Search request with filters

2.2 Swagger Docs

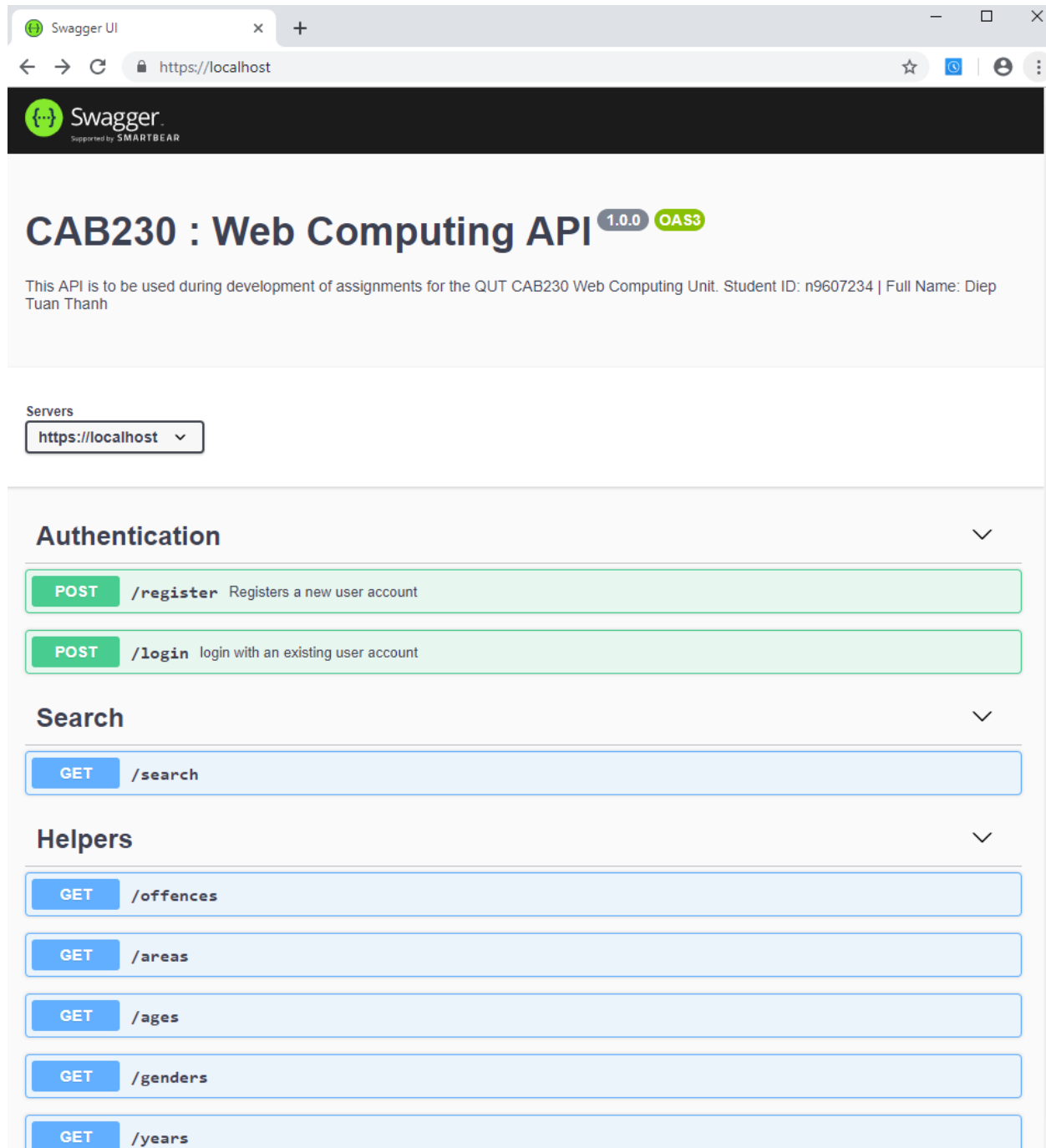


Figure 14 Swagger docs for the serverside

The Swagger document for this server side is implemented and heavily based on the Swagger document on <https://cab230.hackhouse.sh/>. All the results from both POST and GET request are identical to the example above.

2.3 Logging

For server logging, the format is as follow:

```
// Logging
var loggerFormat = '[:date[web]] ":method :url" :status :response-time';
app.use(morgan(loggerFormat));
```

Figure 15 Logging Format for the server