

OTOT Task B

Name: Erin May Gunawan

Matric No.: A0200765L

GitHub Link: [OTOT Task B](#)

Task B1

1. To run the API locally, clone this GitHub repo into your device

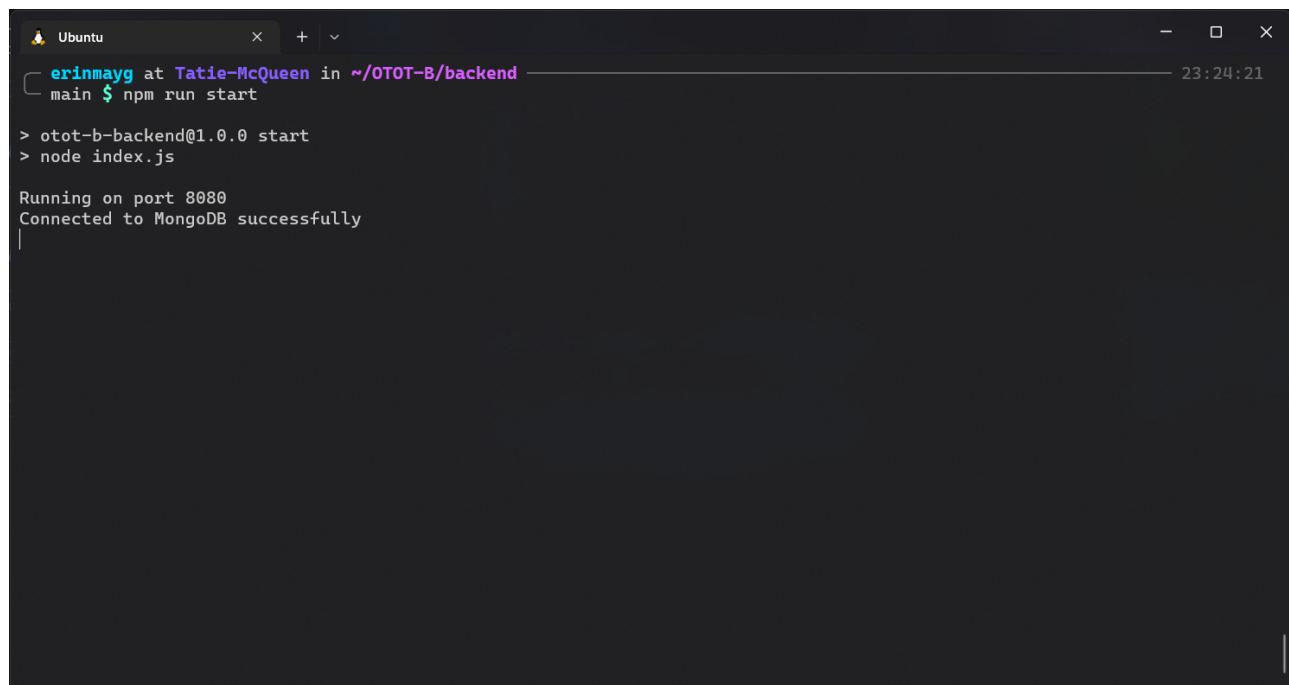
```
git clone https://github.com/erinmayg/OTOT-B
```

Then from the project root, navigate to the `backend` directory to run the backend.

```
# Navigate to backend
cd backend

# Run the backend
npm run start
```

It should produce the following output:



A screenshot of a terminal window titled "Ubuntu". The window shows the command `npm run start` being run in the directory `~/OTOT-B/backend`. The output indicates that the application is running on port 8080 and successfully connected to MongoDB.

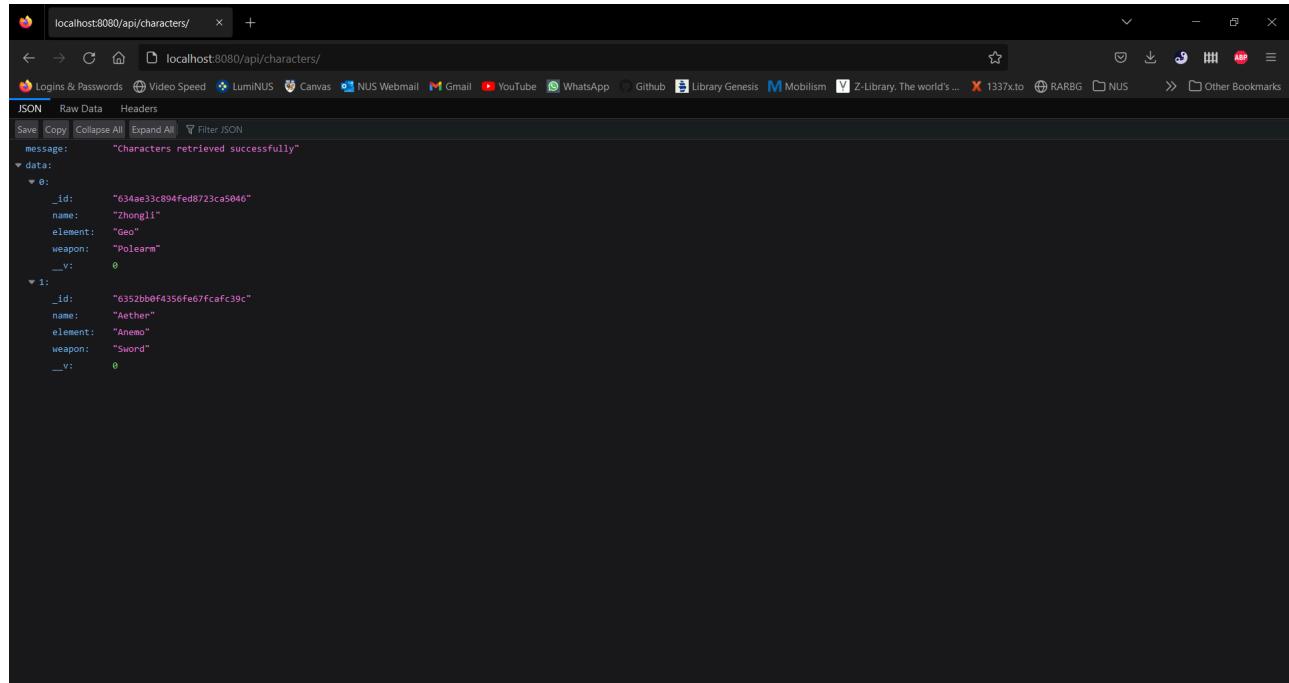
```
Ubuntu
erinmayg at Tatie-McQueen in ~/OTOT-B/backend
main $ npm run start
> otot-b-backend@1.0.0 start
> node index.js

Running on port 8080
Connected to MongoDB successfully
```

By default, the port is set to `8080`. You can change this by modifying `PORT` variable in `backend/.env` file.

You can check whether the backend is running correctly by opening
<https://localhost:8080/api/characters>

It should display the following page.



```

{
  "message": "Characters retrieved successfully",
  "data": [
    {
      "_id": "634ae33c894fed8723ca5046",
      "name": "Zhongli",
      "element": "Geo",
      "weapon": "Polearm",
      "__v": 0
    },
    {
      "_id": "6352bb0f4356fe67fcacf39c",
      "name": "Aether",
      "element": "Anemo",
      "weapon": "Sword",
      "__v": 0
    }
  ]
}

```

2. To test whether the API is working correctly, you can open the [Postman Collection](#)

however you may need to change the `base_url` to <http://localhost:8080/api/characters>
 since it is set to the [deployed backend link](#) by default

or you can run test it out manually with the following APIs:

API Requests

GET: <http://localhost:8080/api/characters>

Retrieves all characters from the DB

Status	Status	Body
SUCCESS	200 OK	{message: 'Characters retrieved successfully', data: CHARACTER[] }

POST: <http://localhost:8080/api/characters>

Creates a new character

Request body example:

```
{
  name: Lumine,
  element: Anemo,
```

```

        weapon: Sword,
    }

```

All fields are required, request will fail if there's any missing field.

Status	Status	Body
SUCCESS	200 OK	{message: 'New character created', data: CREATED_CHARACTER}
FAILED	400 Bad Request	{message: 'Missing MISSING_FIELDS'} or {message: 'Character with same name already exists'}

GET: <http://localhost:8080/api/characters/:characterName>

Retrieves the character with the given name from the DB, and displays its info.

An example of an existing character in the DB is [Zhongli](#), so you may replace `:characterName` in the URL with this.

Status	Status	Body
SUCCESS	200 OK	{message: 'Character retrieved', data: CHARACTER}
FAILED	404 Not Found	{message: 'Character not found'}

PUT: <http://localhost:8080/api/characters/:characterName>

Updates an existing character from the DB.

If you did the [POST](#) request, you may use [Lumine](#) as the `:characterName`.

Request param example:

```

{
  name: Aether
  // element: Dendro
  // weapon: Polearm
}

```

In this [PUT](#) request, all fields are optional.

Status	Status	Body
SUCCESS	200 OK	{message: 'Character updated!', data: UPDATED_CHARACTER}
FAILED	400 Bad Request	{message: 'Character not found'}

DELETE: http://localhost:8080/api/characters/:characterName

Deletes an existing character from the DB

If you did the **POST** request, and the **PUT** request, you may use **Aether** as the **:characterName**.

It returns a successful message and the deleted character upon successful request.

Status	Status	Body
SUCCESS	200 OK	{message: 'Character deleted', data: DELETED_CHARACTER }
FAILED	400 Bad Request	{message: 'Character not found'}

Task B2.1

1. Tests can be run locally via the command

```
npm run test
```

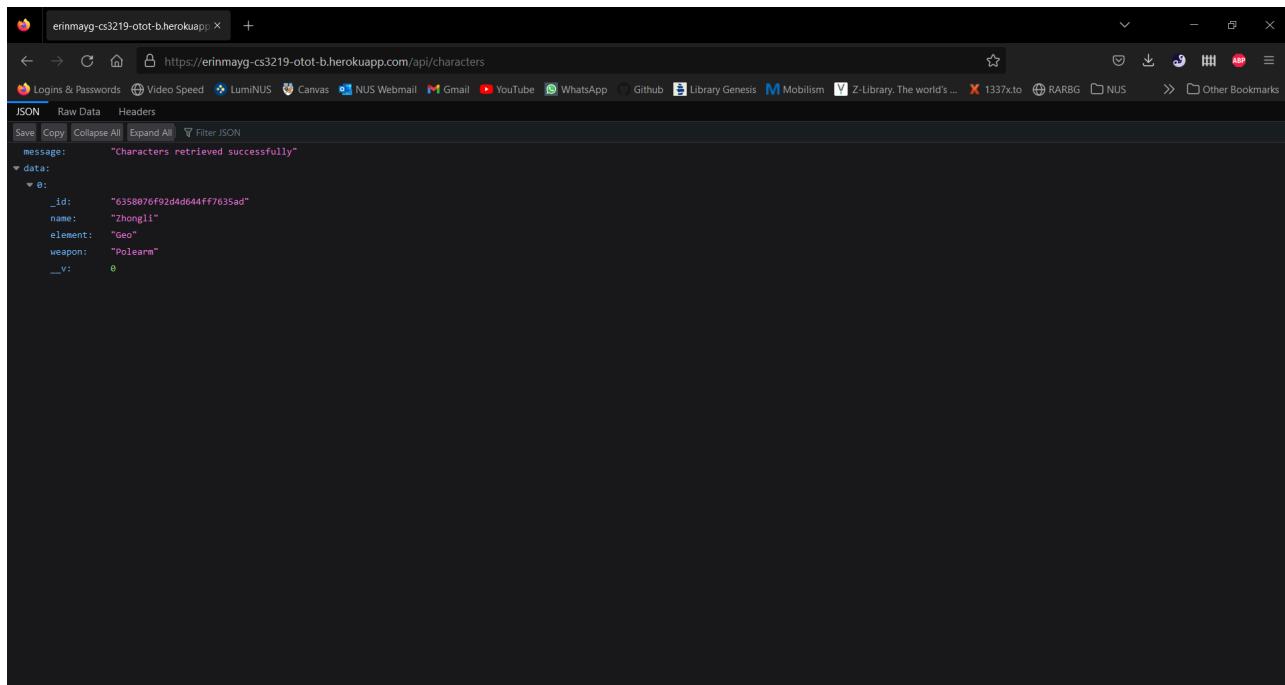
2. It is also done through the CI in GitHub Actions triggered via a Pull Request or a Push to the **main** branch.

The screenshot shows the GitHub Actions page for the repository 'erinmayg/OTOT-B'. The 'Actions' tab is selected. On the left, there's a sidebar with sections like 'Actions', 'All workflows', 'Backend CD', 'Backend CI', 'Build and deploy Node.js project to Azure...', 'Management', and 'Caches'. The main area displays 'All workflows' with 62 workflow runs. A table lists the runs, each with a green checkmark icon, the job name 'Enable multiselect (#5)', the event (e.g., 'Backend CD #20: Commit 56d265a pushed by erinmayg'), the branch (e.g., 'main' or 'nusmods'), the time (e.g., '3 days ago'), and a '30s' duration. There are also three-dot ellipsis icons for each row.

Task B2.2

1. Access the deployed backend API through this [link](#)

It should display the following page



```
message: "Characters retrieved successfully"
data:
  0:
    _id: "6358076f92d4d644ff7635ad"
    name: "ZhongLi"
    element: "Geo"
    weapon: "Polearm"
    __v: 0
```

2. To test whether the deployed backend is working correctly, you can check via the [Postman Collection](#).

By default, the `base_url` in the collection is set to the [deployed backend link](#).

You can open the Postman documentation to see more details on the API requests and responses.

Task B3

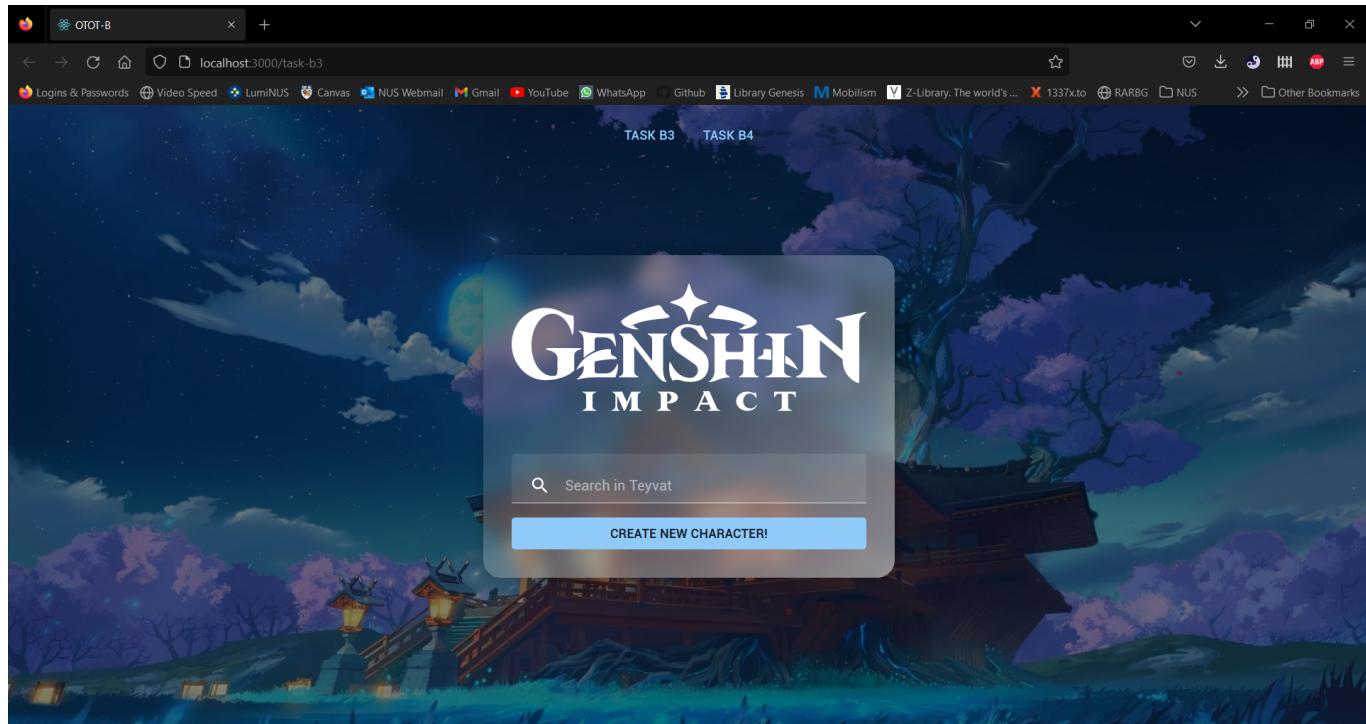
To run the frontend run the following commands:

```
# From the project root
# navigate to the frontend directory
cd frontend

# Start the frontend
npm run start
```

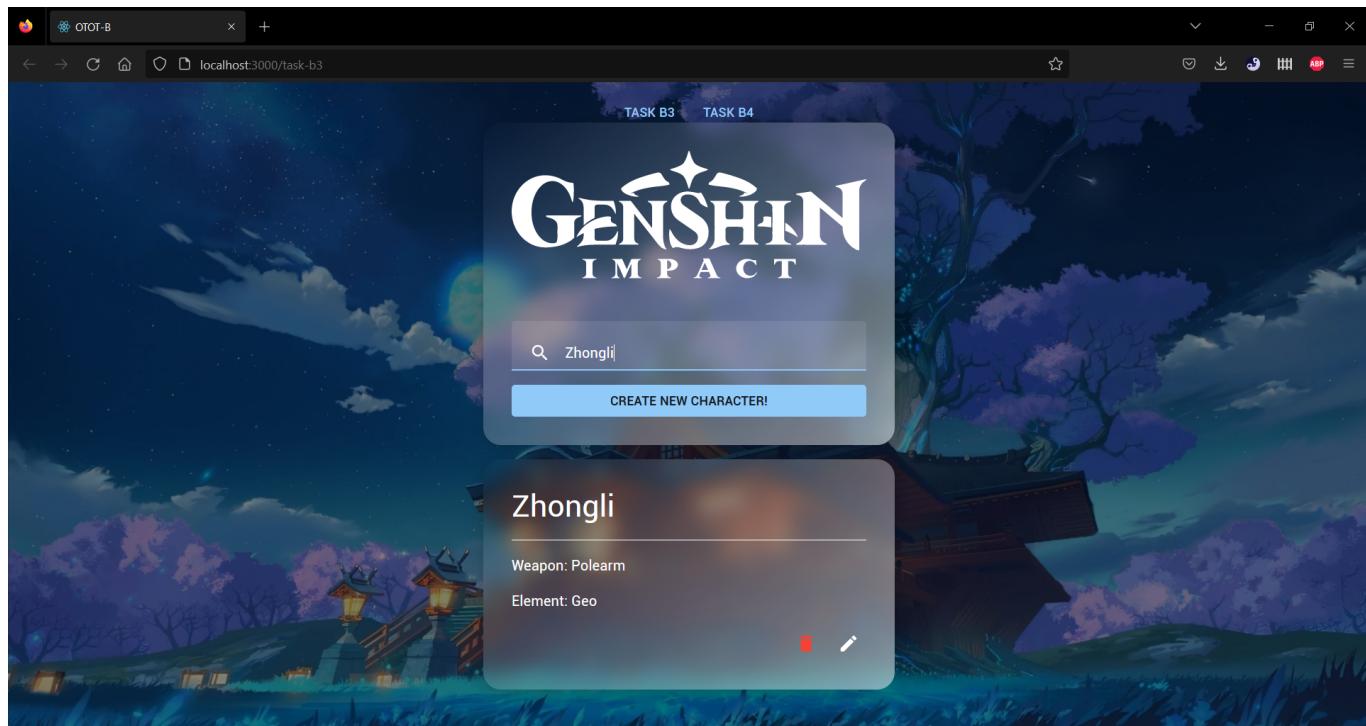
By default, it should run in port `3000`, you can open <http://localhost:3000> from the browser.

It should display the following page

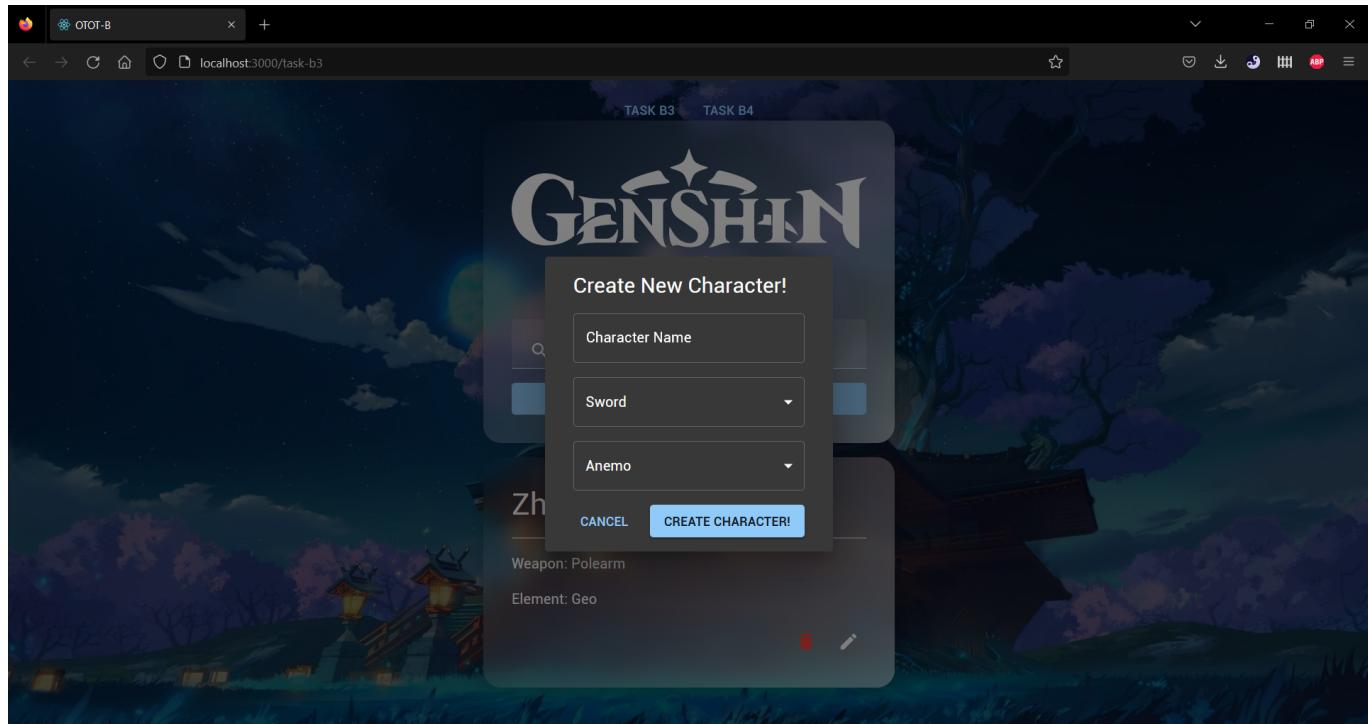


Note that the search function is case-sensitive

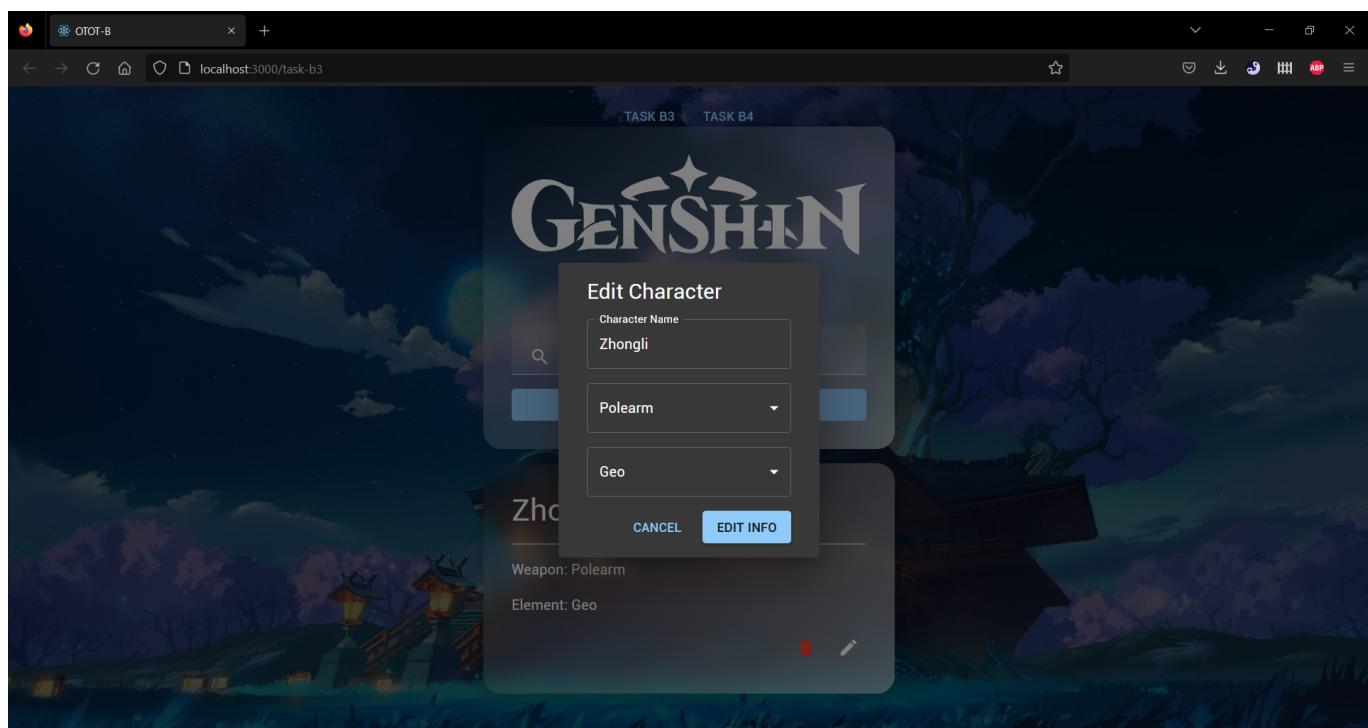
GET



POST



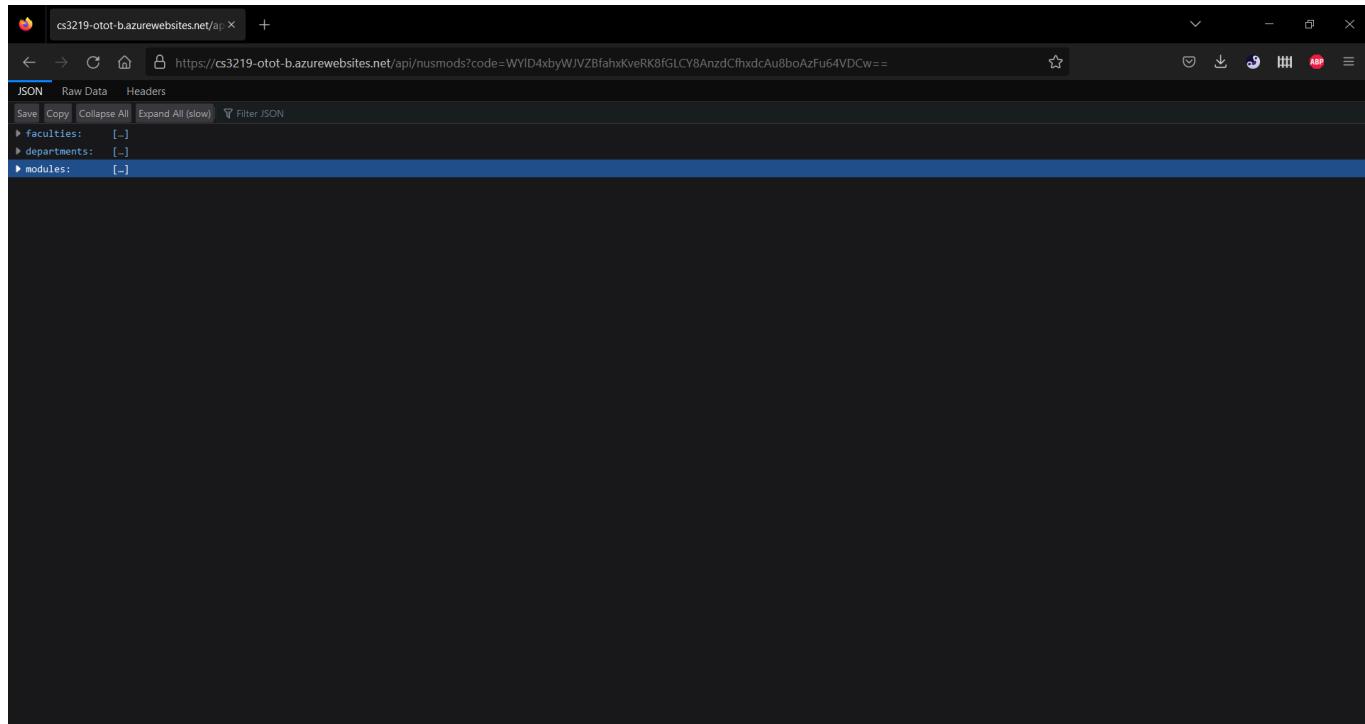
PUT



Task B4

The serverless function is deployed in [Azure Functions](#) and can be accessed via this [link](#).

It should display the following website



It displays all faculties, departments, and modules in NUS (retrieved via the [NUSMods API](#)) in the AY 2022-2023 (set by default).

And it is able to filter the modules via the department and/or faculties. It employs **AND** filtering between params, and **OR** filtering within it.

A query example would be: <[azure_functions_link](#)>&faculty=Computing,Science

Frontend

To run the frontend, follow the steps listed in [Task B3](#). Once done, you can navigate to the following [link](#) to view the Frontend for the serverless function.

It should display the following webpage

NUSMODS

Filters

AC5001 Architectural History of Singapore
Architecture • 4 MCs
This module introduces the architectural history of Singapore from the pre-colonial era to the contemporary period from an interdisciplinary perspective. It explores the social and cultural role of architects, and to how that role has been interpreted and has evolved. It also offers critical views on the role of architecture in constructing the national identity of Singapore.

AC5002 Conservation Approaches and Philosophies
Architecture • 4 MCs
This module aims to introduce students to current conservation approaches such as Historic Urban Landscape, Heritage Impact Assessment, Conservation Management Plan, Historic Area Assessments approach and applications and their application and relevance in the Asian context. The focus will be to understand how and why these concepts and methods have evolved and applied in practice in the Singapore context. Students will be strongly encouraged to contextualise this with other international heritage conservation scenarios in the region and worldwide to reflect broader shifts in conservation approaches.

AC5003 Urban Conservation and Regeneration
Architecture • 4 MCs
This module will equip students with a comprehensive understanding of the key theoretical debates, challenges, and practices of urban conservation and regeneration. In addition to examining the architectural issues, the module will also investigate the social, economic and environmental problems which have resulted in current Asian urban and suburban landscapes, and how government policy can affect change to these landscapes for urban and economic renewal. By

You can use the drop down to filter the modules

NUSMODS

Filters

IT5100B High-Throughput Stream Programming
Computer Science • 2 MCs
The global availability of data has reached a level where aggregating data into generic, general-purpose "stores" is no longer feasible. Having data collections statically available for querying by interested parties on demand is increasingly becoming the way of the past. Instead, a new paradigm, called Data Streaming, has emerged recently. In this paradigm, data is bundled into high-throughput "streams" that are sharded efficiently across a large number of network nodes. Consumers, sometimes counted in hundreds of thousands, or millions, "subscribe" to data subsets and are notified when new data becomes available, being under the obligation to process it immediately, or lose it. Consequently, data storage is no longer centralized, but rather distributed into many smaller-sized abstract collections. This new approach to "Big Data" requires a new set of tools, platforms, and solution patterns. In this course we propose to explore several facets of this new paradigm: • The Stream paradigm introduced in Java 8. • Platforms that implement Data Streaming, such as Kafka, and the Java bindings in the library KafkaConnect. • Computing paradigms for stream processing, such as Reactive Programming, and the library RxJava. • High-performance stream computing platforms, such as Flink. The course will be using Java as the main vehicle for introducing concepts and showcasing examples.

MA1100 Basic Discrete Mathematics
Mathematics • 4 MCs
This is the entry-level module for a sound education in modern mathematics, to prepare students for higher level mathematics courses. The first goal is to build the necessary mathematical foundation by introducing the basic language, concepts, and methods of contemporary mathematics, with focus on discrete and