

Yale BIDS Technical Assessment Webapp Document

Provided by Niloofar Didar

Niloofar.didar@gmail.com

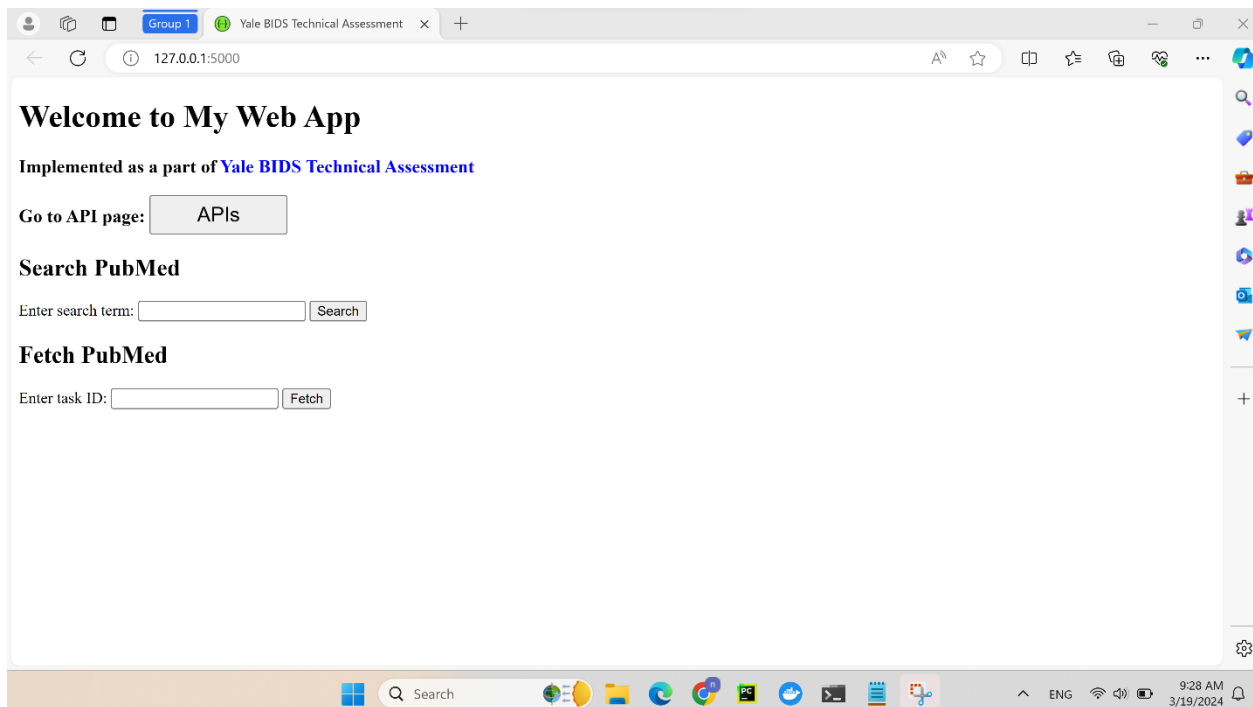
BASIC PROJECT DESCRIPTIONS

Develop a web page designed to enable users to efficiently search for and retrieve information or data corresponding to specific term from PubMed.

Implemented tasks as required:

1. Home page

The homepage, depicted in the figure below, comprises a user interface (UI) facilitating access to the Pubmed website via search and fetch functions. It features a button directing users to the API documentation, along with two additional functions enabling direct term search and task ID retrieval stemming from search tasks.



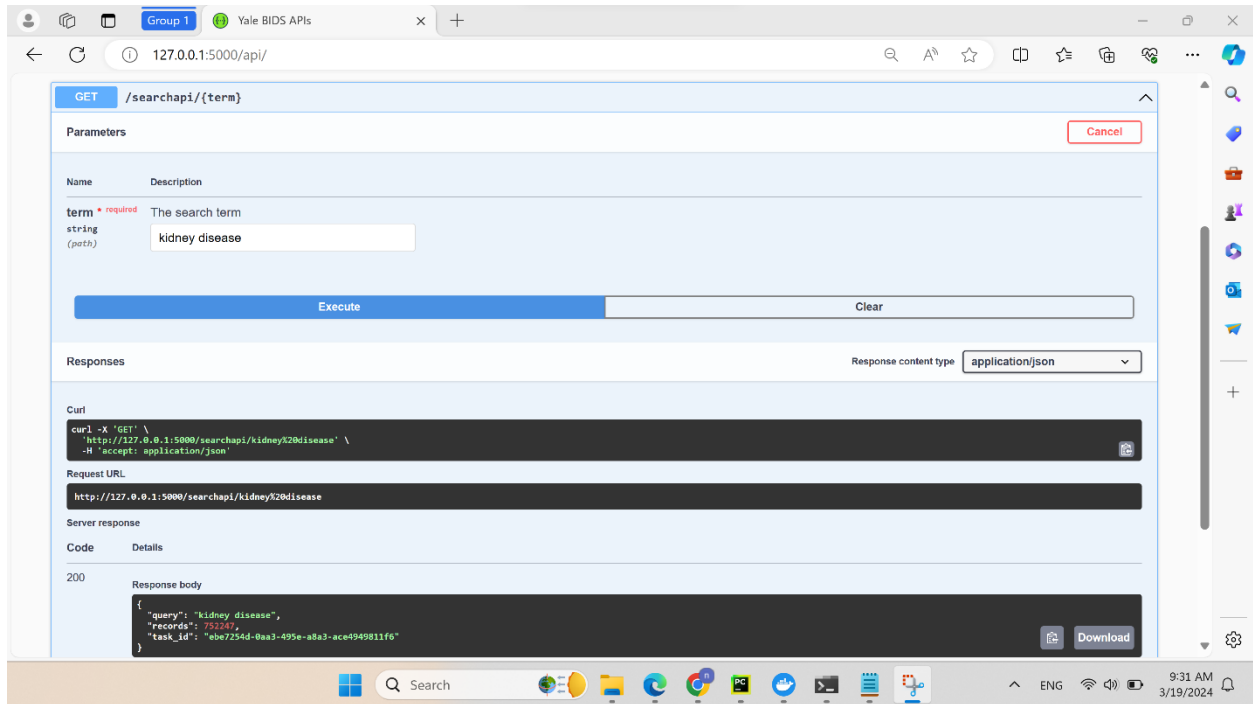
2. API Implementation/Documentation for Search and Fetch:

AS the project requirement, I developed and implemented essential API endpoints to accommodate search queries tailored to user-specified terms

❖ Search:

This API enables users to search for any term (eg, kidney disease as shown in the figure below) on the Pubmed website. The results are displayed in a JSON file format, structured according to the specifications outlined in Yale's documentation provided below.

```
{  
  "records": 212111,  
  "query": "alzheimer disease",  
  "task_id": "7fd381bf3cbe28e892e163db81b9e2cd"  
}
```



❖ Fetch:

The fetch API allows users to retrieve additional details concerning the searched task ID. The results are presented in a JSON file format, containing information aligned with the specifications outlined in Yale's documentation.

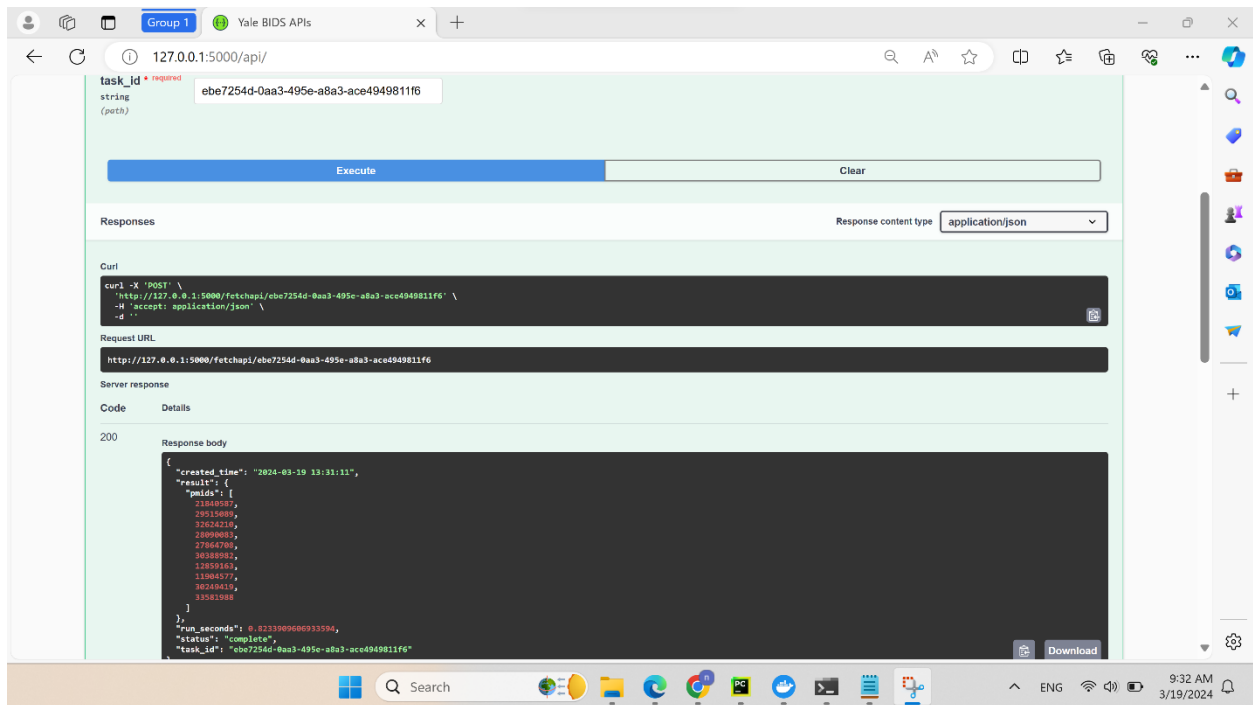
```
{  
  "task_id": "7fd381bf3cbe28e892e163db81b9e2cd"  
  "status": "completed",  
}
```

```

"result":{
"pmids":[7952237,37506310,32397415,...]
},
"created_time":"2024-03-12 13:51:02",
"run_seconds": 75,
}

```

Below is a figure of API result for the fetch function using the kidney disease task ID:



3. UI for User Search and Fetch:

Additionally, this application provides users with the capability to conduct searches for terms without utilizing an API, as depicted in the figure below. For instance, when searching for the term "kidney disease," the results are displayed via the URL <localhost:5000/search?term=kidney+disease> through json data.

```

{
  "query": "kidney disease",
  "records": 752247,
  "task_id": "cae7eae5-a3a3-400e-821b-7ec045c339bb"
}

```

Welcome to My Web App

Implemented as a part of [Yale BIDS Technical Assessment](#)

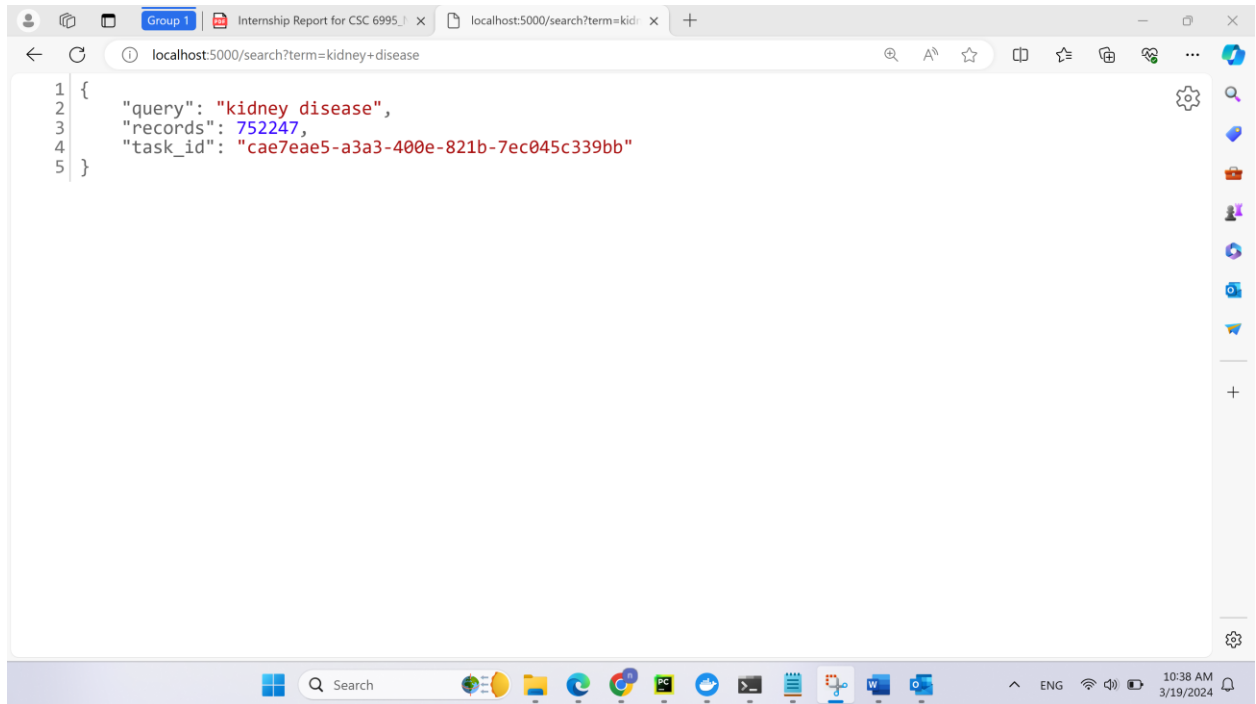
Go to API page:

APIs

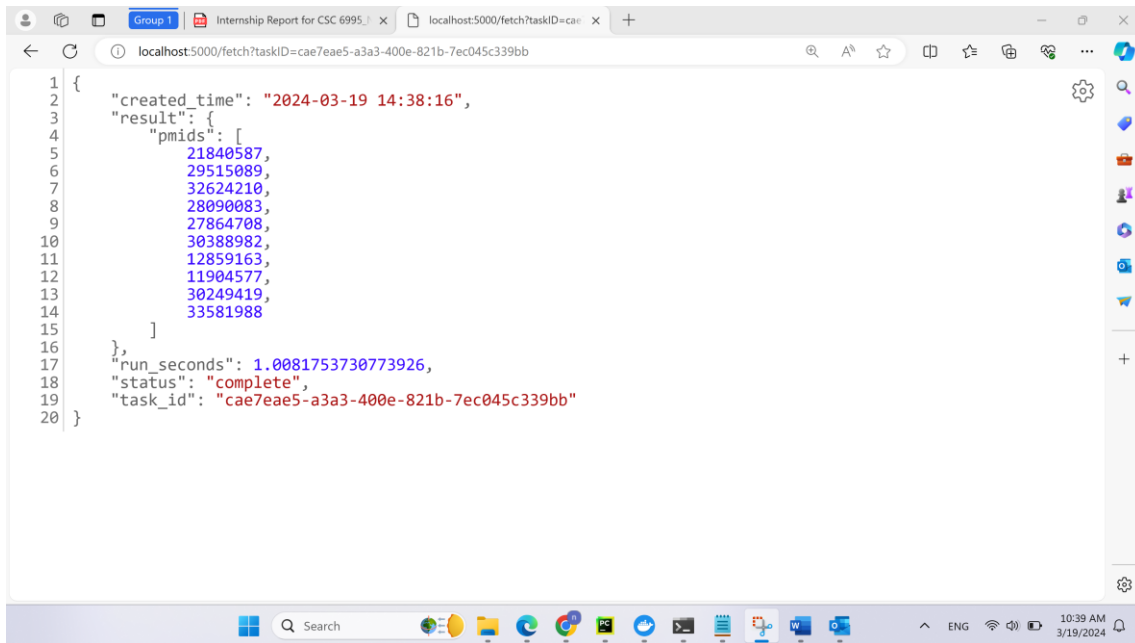
Search PubMed

Enter search term:

Search



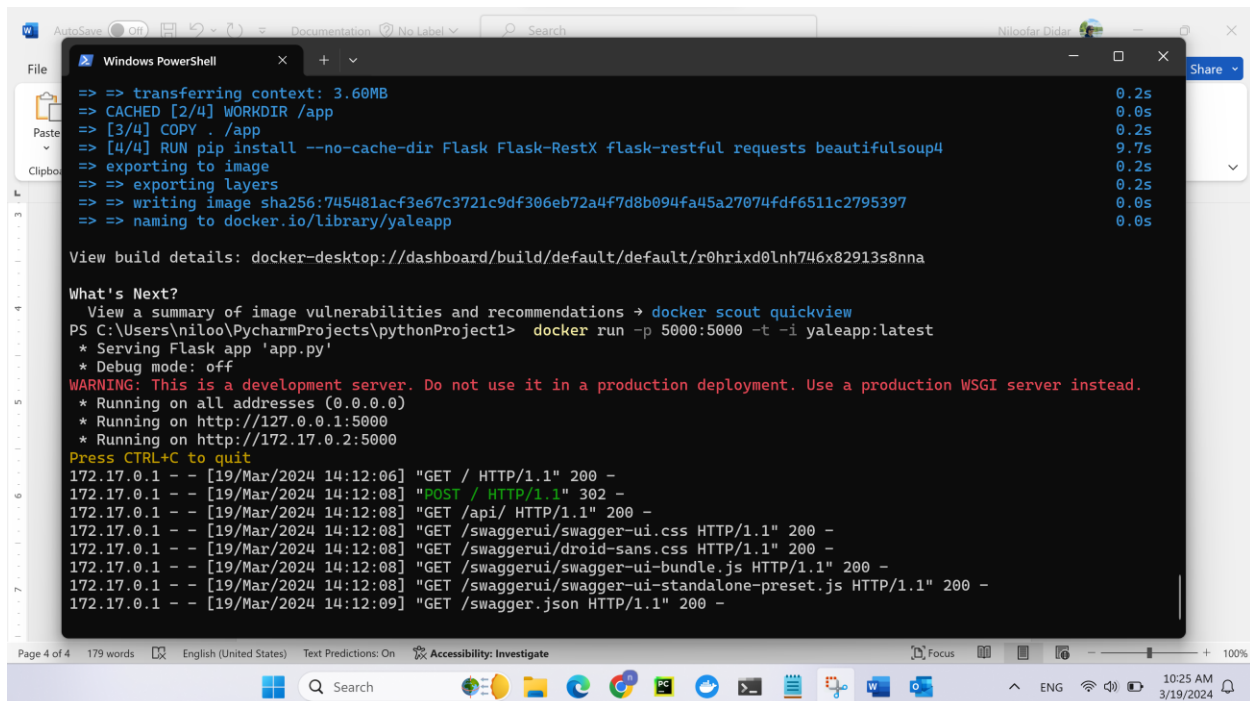
Similarly, when utilizing the task ID "cae7eae5-a3a3-400e-821b-7ec045c339bb" for the search function, the JSON result for that record will be displayed through the following URL format: <localhost:5000/fetch?taskID=cae7eae5-a3a3-400e-821b-7ec045c339bb> as is displayed in the image below:



```
1 {
2   "created_time": "2024-03-19 14:38:16",
3   "result": {
4     "pmids": [
5       21840587,
6       29515089,
7       32624210,
8       28090083,
9       27864708,
10      30388982,
11      12859163,
12      11904577,
13      30249419,
14      33581988
15    ]
16  },
17  "run_seconds": 1.0081753730773926,
18  "status": "complete",
19  "task_id": "cae7eae5-a3a3-400e-821b-7ec045c339bb"
20 }
```

4. Containerization:

The project has been containerized using Docker to facilitate easy deployment. To build your Docker image, open your terminal, navigate to the directory containing the Dockerfile, and execute the commands provided in the Readme.txt file. Further details on running a sample of the application through Docker can be found in the figure below.



```
=> => transferring context: 3.60MB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . /app
=> [4/4] RUN pip install --no-cache-dir Flask Flask-RestX flask-restful requests beautifulsoup4
=> exporting to image
=> exporting layers
=> writing image sha256:745481acf3e67c3721c9df306eb72a4f7d8b094fa45a27074fd6f511c2795397
=> naming to docker.io/library/yaleapp

View build details: docker-desktop://dashboard/build/default/default/r0hrixd0lnh746x82913s8nna

What's Next?
  View a summary of image vulnerabilities and recommendations -> docker scout quickview
PS C:\Users\niloo\PycharmProjects\pythonProject1> docker run -p 5000:5000 -t -i yaleapp:latest
* Serving Flask app 'app.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [19/Mar/2024 14:12:06] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [19/Mar/2024 14:12:08] "POST / HTTP/1.1" 302 -
172.17.0.1 - - [19/Mar/2024 14:12:08] "GET /api/ HTTP/1.1" 200 -
172.17.0.1 - - [19/Mar/2024 14:12:08] "GET /swaggerui/swagger-ui.css HTTP/1.1" 200 -
172.17.0.1 - - [19/Mar/2024 14:12:08] "GET /swaggerui/droid-sans.css HTTP/1.1" 200 -
172.17.0.1 - - [19/Mar/2024 14:12:08] "GET /swaggerui/swagger-ui-bundle.js HTTP/1.1" 200 -
172.17.0.1 - - [19/Mar/2024 14:12:08] "GET /swaggerui/swagger-ui-standalone-preset.js HTTP/1.1" 200 -
172.17.0.1 - - [19/Mar/2024 14:12:09] "GET /swagger.json HTTP/1.1" 200 -
```