Cloud Computing Assignment (Practical 1)

Name: Sudhanshu Ghuge
Student ID: 24213171

Exercise One: Web-App setup with Flask

1. Project Setup

1.1 Folder Structure

The project folder was named as follows:

- PROJECT1_Sudhanshu_Ghuge_24213171/

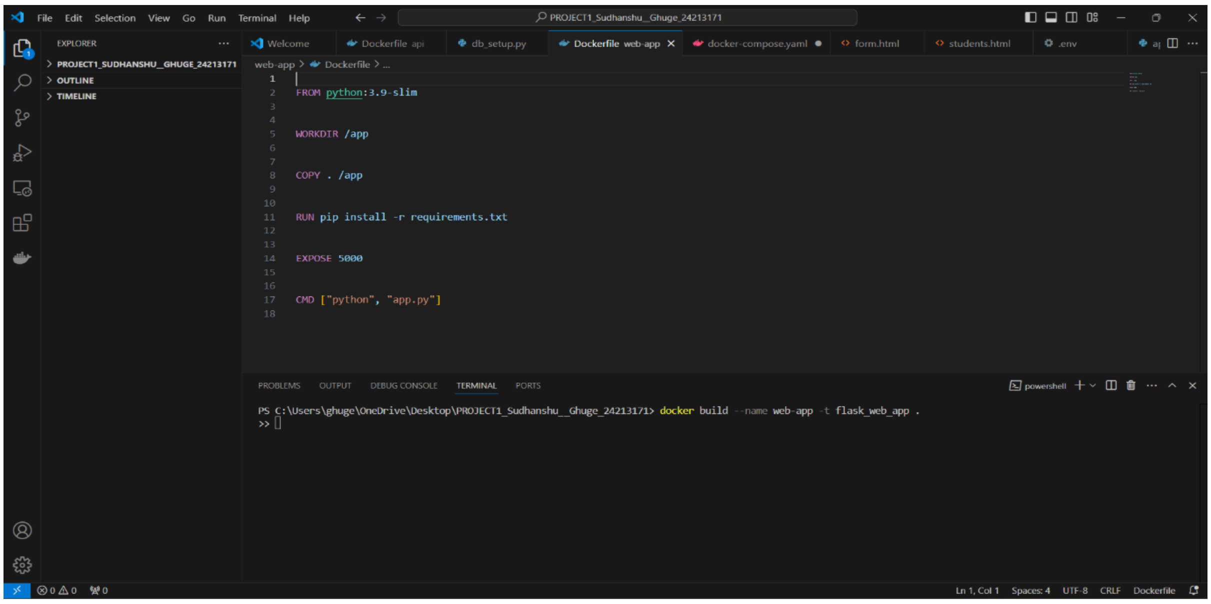
After downloading and extracting the **web_app.zip** file from Brightspace, the folder structure looked like this:

- PROJECT1_Sudhanshu_Ghuge_24213171/

```
├── web_app/
        ├── app.py
        ├── requirements.txt
        └── templates
```

1.2 Creating the Dockerfile

Inside the web-app/ directory, I created a new Dockerfile with the following content:



```
FROM python:3.9-slim


WORKDIR /app


COPY . /app


RUN pip install -r requirements.txt


EXPOSE 5000


CMD ["python", "app.py"]
```

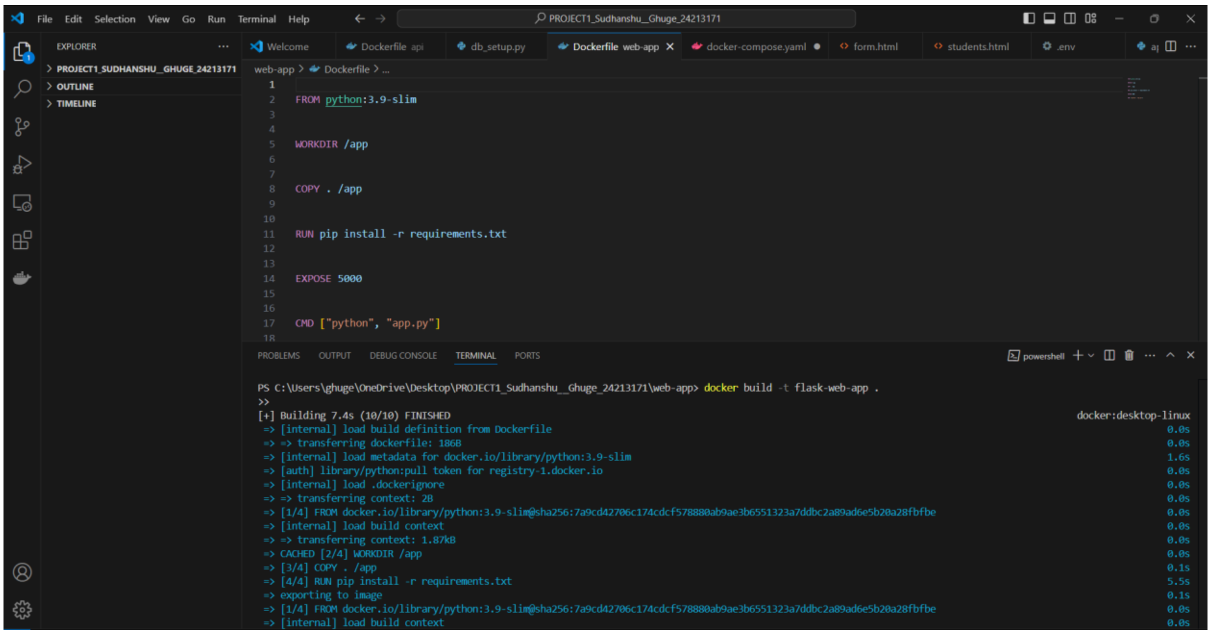
```
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171> docker build --name web-app -t flask_web_app .
>> 
```

2. Building and Running the Docker Container

2.1 Building the Docker Image

To build the Docker image, I navigated to the web_app directory in the terminal and executed the following command:

Command: docker build -t flask-web-app.



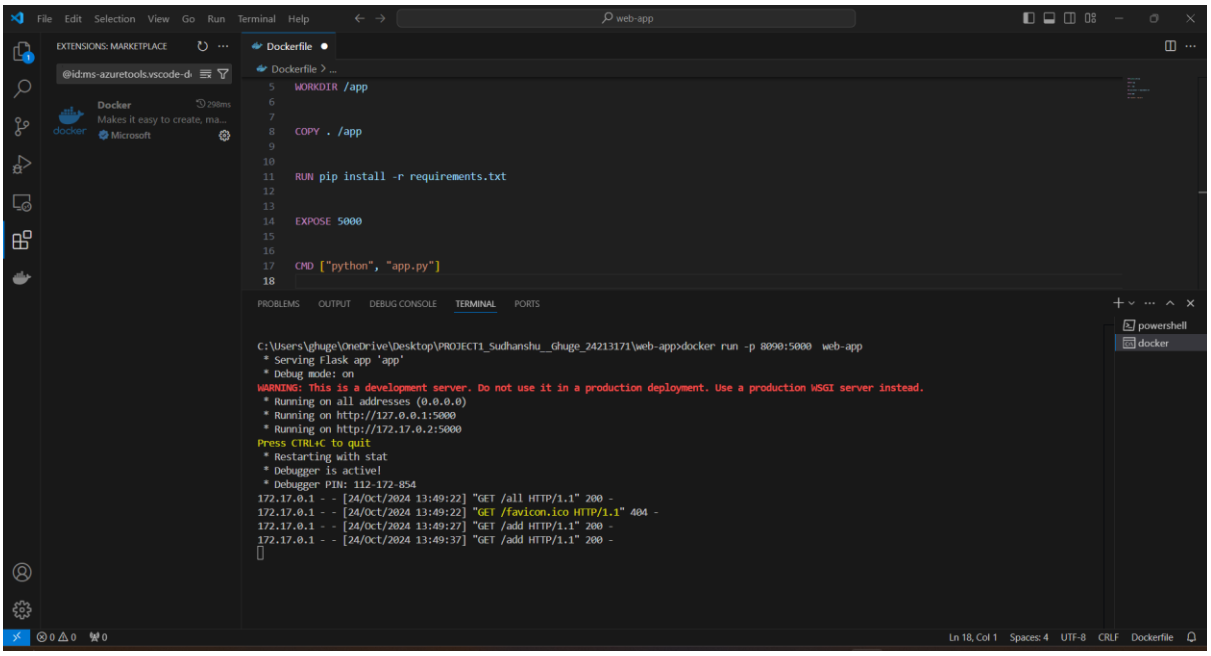
```
FROM python:3.9-slim

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

EXPOSE 5000

CMD ["python", "app.py"]
```

```
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171\web-app> docker build -t flask-web-app .
>>
[+] Building 7.4s (10/10) FINISHED                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                             0.0s
 => => transferring dockerfile: 186B                                                                             0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                               1.6s
 => [auth] library/python:pull token for registry-1.docker.io                                                    0.0s
 => [internal] load .dockerignore                                                                                0.0s
 => => transferring context: 2B                                                                                  0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:7a9cd42706c174cdcf578880ab9ae3b6551323a7ddbc2a89ad6e5b20a28fbfbe  0.0s
 => [internal] load build context                                                                                0.0s
 => => transferring context: 1.87kB                                                                              0.0s
 => CACHED [2/4] WORKDIR /app                                                                                    0.0s
 => [3/4] COPY . /app                                                                                            0.1s
 => [4/4] RUN pip install -r requirements.txt                                                                    5.5s
 => exporting to image                                                                                           0.1s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:7a9cd42706c174cdcf578880ab9ae3b6551323a7ddbc2a89ad6e5b20a28fbfbe  0.0s
 => [internal] load build context                                                                                0.0s
```

2.2 Running the Docker Container

After the image was built, I ran the container with the following command:

Command: docker run -p 8090:5000 flask-web-app

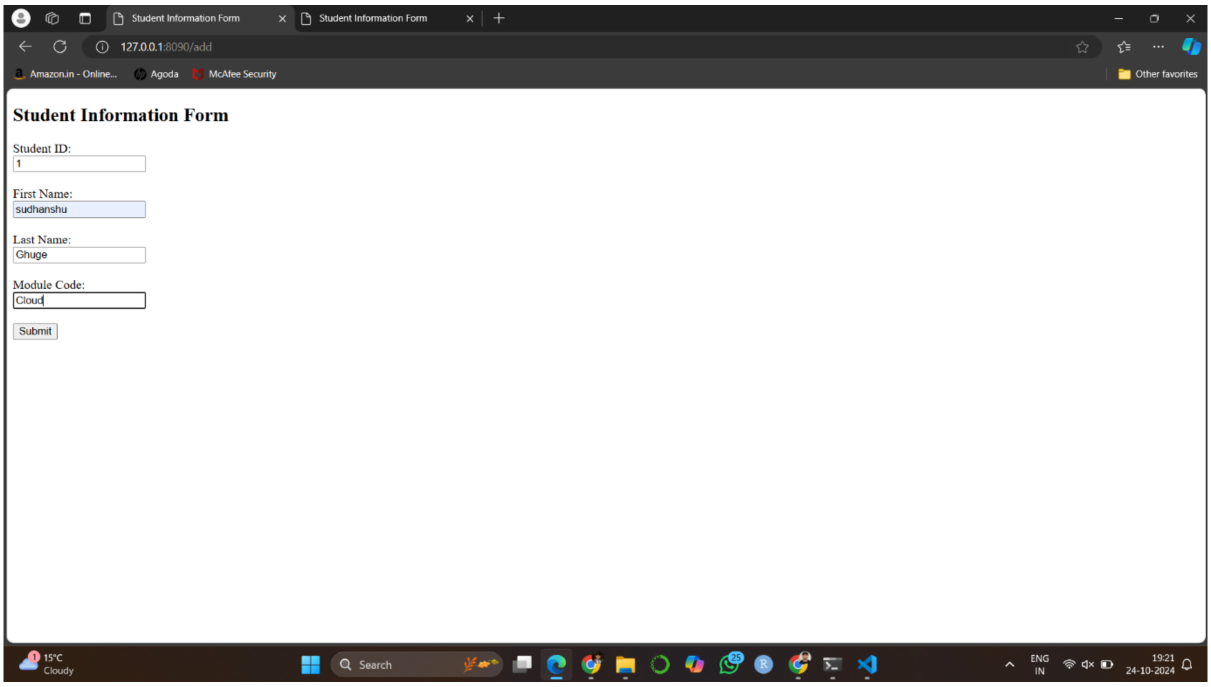


```
C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171\web-app>docker run -p 8090:5000  web-app
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (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
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 112-172-854
172.17.0.1 - - [24/Oct/2024 13:49:22] "GET /all HTTP/1.1" 200 -
172.17.0.1 - - [24/Oct/2024 13:49:22] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [24/Oct/2024 13:49:27] "GET /add HTTP/1.1" 200 -
172.17.0.1 - - [24/Oct/2024 13:49:37] "GET /add HTTP/1.1" 200 -
```

3. Accessing the Web Application

To verify that the application was running correctly, I navigated to the following URLs in a web browser:

- http://127.0.0.1:8090/add



127.0.0.1:8090/add

Student Information Form

Student ID: 1

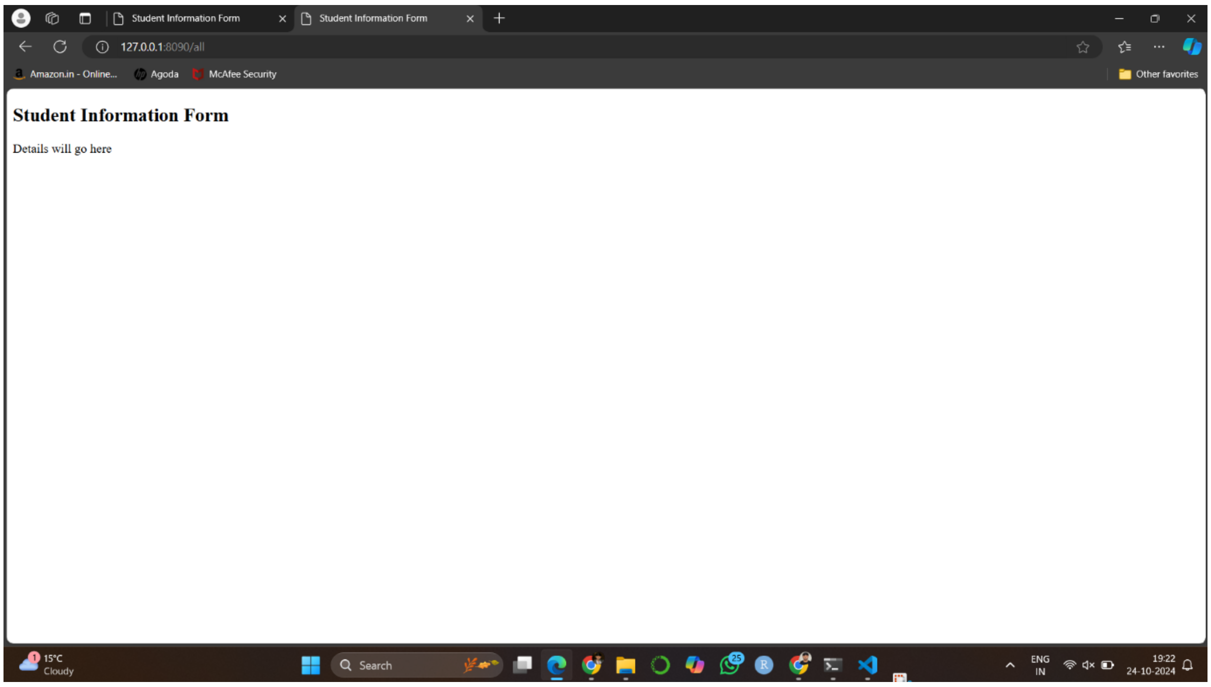
First Name: sudhanshu

Last Name: Ghuge

Module Code: Cloud

Submit

- http://127.0.0.1:8090/all



127.0.0.1:8090/all

Student Information Form

Details will go here

Exercise Two: API Setup with FastAPI

1. Project Setup

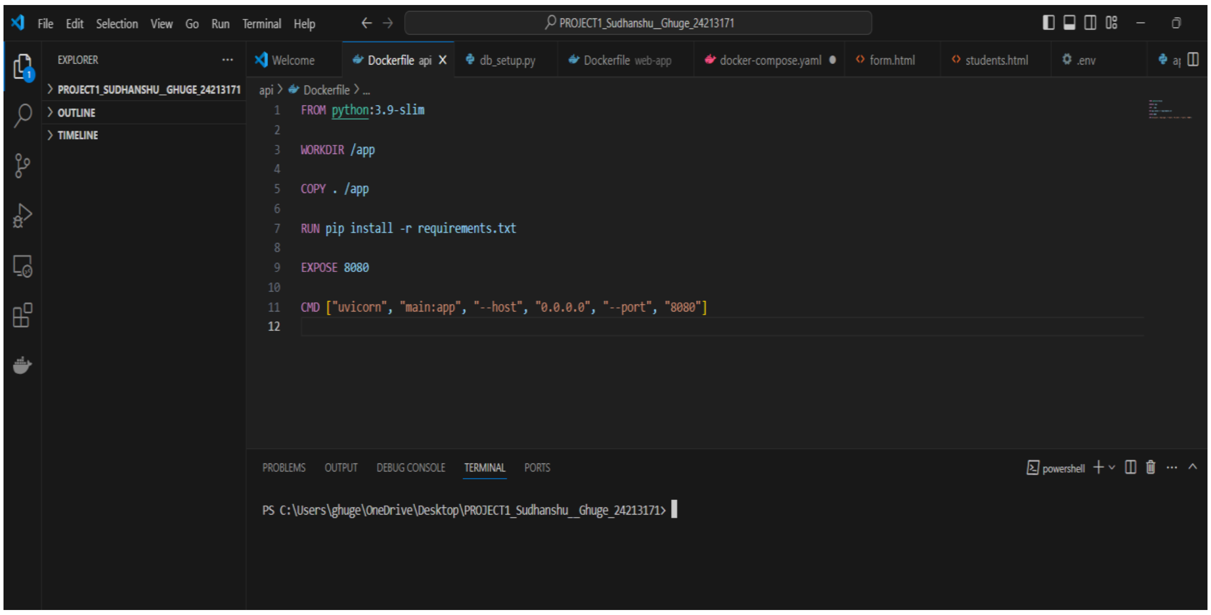
1.1 Folder Structure

After downloading and extracting the api.zip file from Brightspace, I ensured both the web-app and api folders were in the same directory:

```
├── api/
    ├── main.py
    ├── requirements.txt
    ├── .DS_Store
    ├── schema.py
    ├── db_setup.py
    └── model.py
```

1.2 Creating the Dockerfile

Inside the api/ directory, I created a new Dockerfile with the following content:



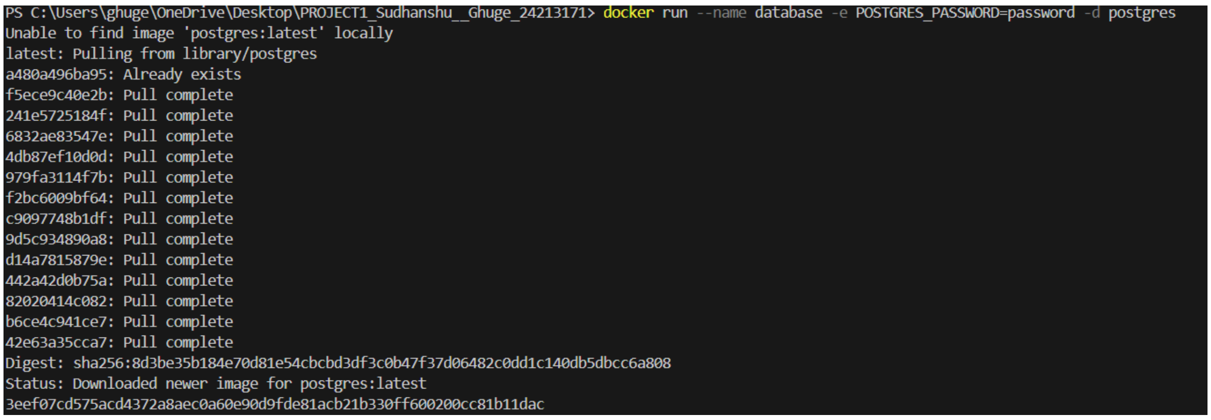
```
FROM python:3.9-slim

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

EXPOSE 8080

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8080"]
```

```
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu_Ghuge_24213171>
```

2. Setting Up the Temporary Database

To set up a temporary PostgreSQL database, I executed the following command in the terminal:

Command: docker run --name database -p 5432:5432 -e POSTGRES_DB=student -e POSTGRES_PASSWORD=password -e POSTGRES_USER=postgres -d postgres



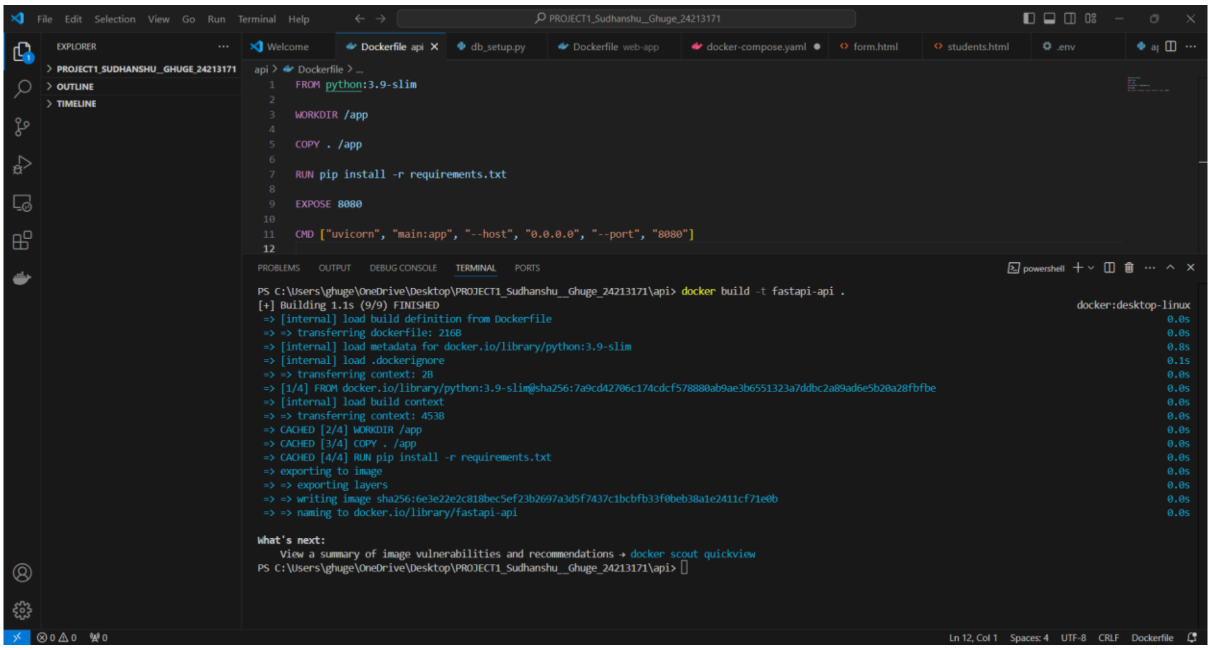
```
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171> docker run --name database -e POSTGRES_PASSWORD=password -d postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
a480a496ba95: Already exists
f5ece9c40e2b: Pull complete
241e5725184f: Pull complete
6832ae83547e: Pull complete
4db87ef10d0d: Pull complete
979fa3114f7b: Pull complete
f2bc6009bf64: Pull complete
c9097748b1df: Pull complete
9d5c934890a8: Pull complete
d14a7815879e: Pull complete
442a42d0b75a: Pull complete
82020414c082: Pull complete
b6ce4c941ce7: Pull complete
42e63a35cca7: Pull complete
Digest: sha256:8d3be35b184e70d81e54cbcbd3df3c0b47f37d06482c0dd1c140db5dbcc6a808
Status: Downloaded newer image for postgres:latest
3eef07cd575acd4372a8aec0a60e90d9fde81acb21b330ff600200cc81b11dac
```

3. Building and Running the API Container

3.1 Building the Docker Image

To build the Docker image for the API, I navigated to the api directory and executed the following command:

Command: docker build -t fastapi-api



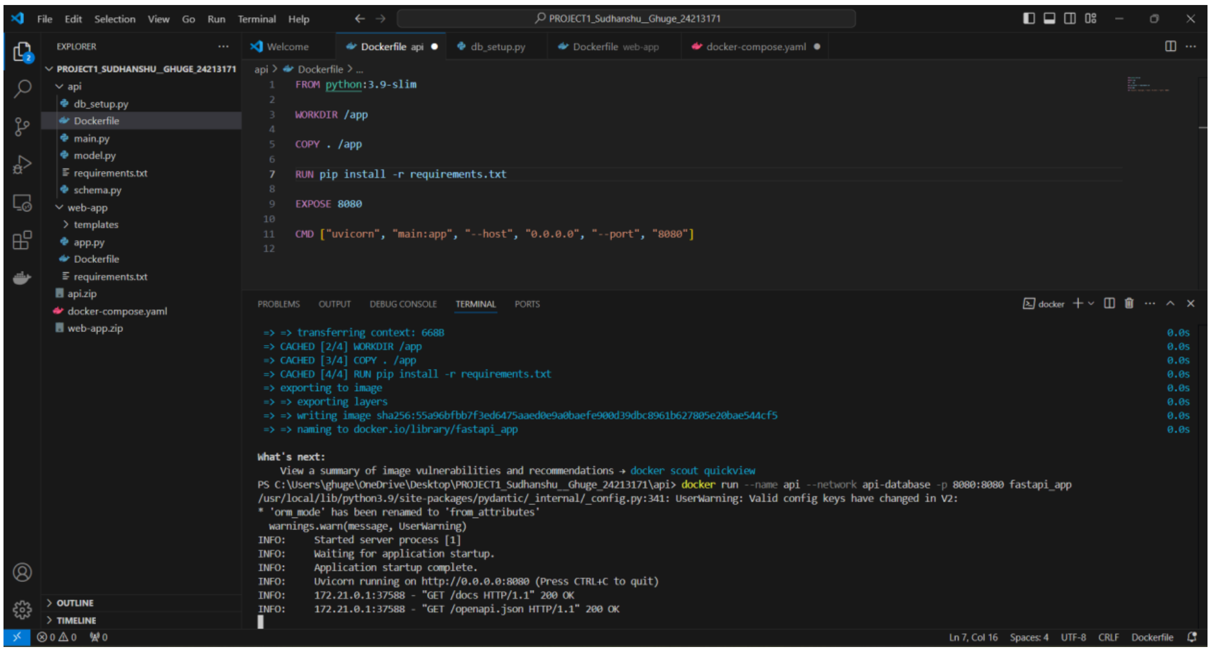
```
FROM python:3.9-slim

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

EXPOSE 8080

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8080"]
```

```
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171\api> docker build -t fastapi-api .
[+] Building 1.1s (9/9) FINISHED                                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 216B                                                                    0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                      0.8s
 => [internal] load .dockerignore                                                                       0.1s
 => => transferring context: 2B                                                                         0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:7a9cd42706c174cdcf578880ab9ae3b6551323a7ddbc2a89ad6e5b20a28fbfbe  0.0s
 => [internal] load build context                                                                       0.0s
 => => transferring context: 453B                                                                       0.0s
 => CACHED [2/4] WORKDIR /app                                                                           0.0s
 => CACHED [3/4] COPY . /app                                                                            0.0s
 => CACHED [4/4] RUN pip install -r requirements.txt                                                    0.0s
 => exporting to image                                                                                  0.0s
 => => exporting layers                                                                                 0.0s
 => => writing image sha256:6e3e22e2c818bec5ef23b2697a3d5f7437c1bcbfb33f0beb38a1e2411cf71e0b            0.0s
 => => naming to docker.io/library/fastapi-api                                                          0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171\api>
```

3.2 Running the Docker Container

After the image was built, I ran the container with the following command:

Command: docker run –-name api network api-database -p 8080:8080 fastapi_app



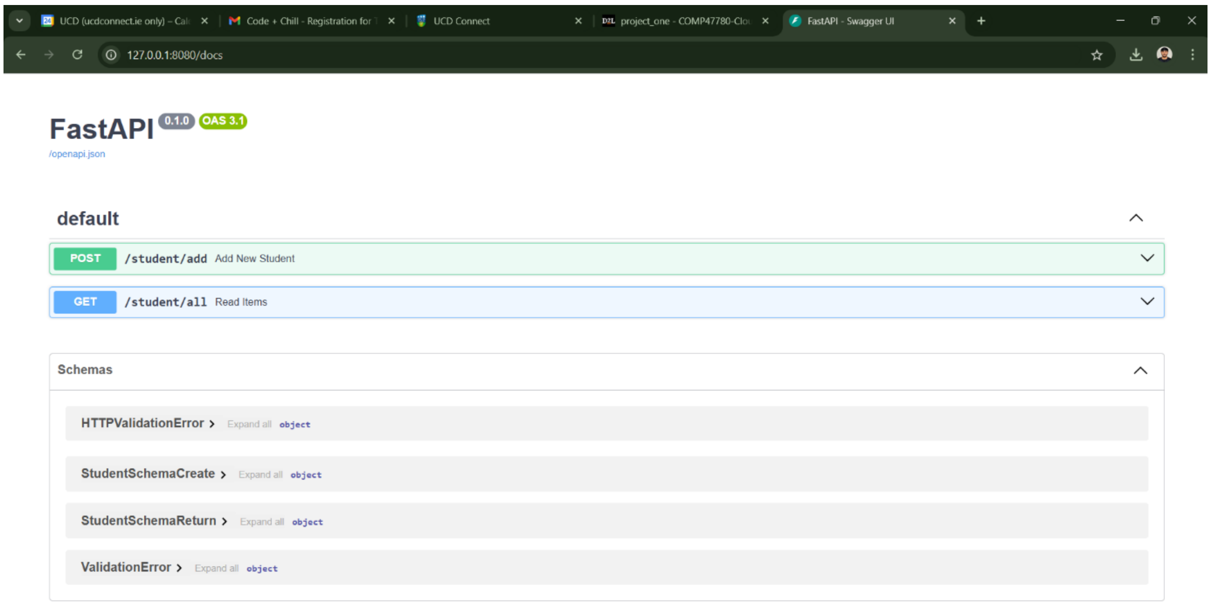
```
FROM python:3.9-slim

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

EXPOSE 8080

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8080"]
```

```
 => => transferring context: 668B                                                                  0.0s
 => CACHED [2/4] WORKDIR /app                                                                      0.0s
 => CACHED [3/4] COPY . /app                                                                       0.0s
 => CACHED [4/4] RUN pip install -r requirements.txt                                               0.0s
 => exporting to image                                                                             0.0s
 => => exporting layers                                                                            0.0s
 => => writing image sha256:55a96bfbb7f3ed6475aaed0e9a0baefe900d39dbc8961b627805e20bae544cf5       0.0s
 => => naming to docker.io/library/fastapi_app                                                     0.0s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171\api> docker run --name api --network api-database -p 8080:8080 fastapi_app
/usr/local/lib/python3.9/site-packages/pydantic/_internal/_config.py:341: UserWarning: Valid config keys have changed in V2:
* 'orm_mode' has been renamed to 'from_attributes'
  warnings.warn(message, UserWarning)
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
INFO:     172.21.0.1:37588 - "GET /docs HTTP/1.1" 200 OK
INFO:     172.21.0.1:37588 - "GET /openapi.json HTTP/1.1" 200 OK
```

4. Browsing the API

To verify that the API was running correctly, I navigated to the following URL in a web browser:

- http://127.0.0.1:8080/docs



FastAPI 0.1.0 OAS 3.1
/openapi.json

default

POST /student/add Add New Student

GET /student/all Read Items

Schemas

HTTPValidationError > Expand all object

StudentSchemaCreate > Expand all object

StudentSchemaReturn > Expand all object

ValidationError > Expand all object

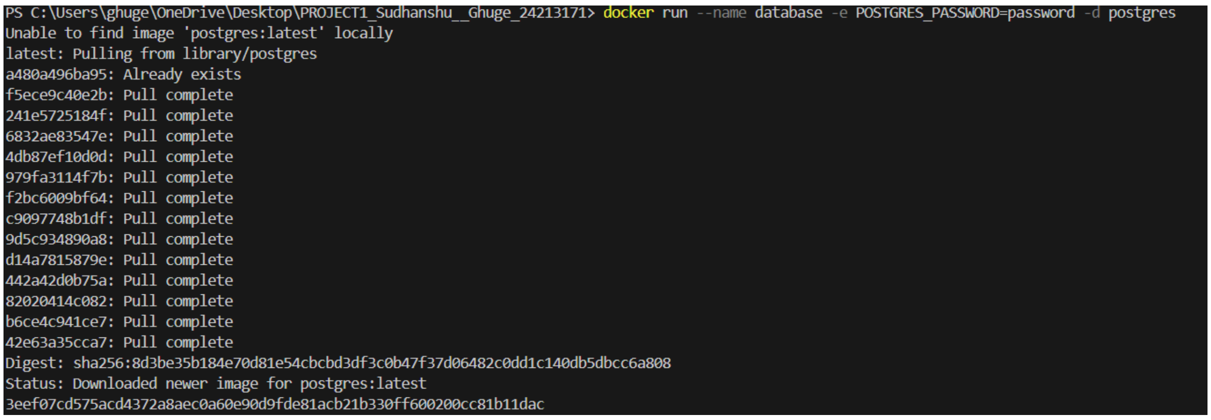
Exercise Three: Database Service

1. Setting Up the PostgreSQL Database

1.1 Running the PostgreSQL Container

To create and run a new instance of the PostgreSQL database, I executed the following command:

command: docker run --name database -e POSTGRES_PASSWORD=password -d postgres



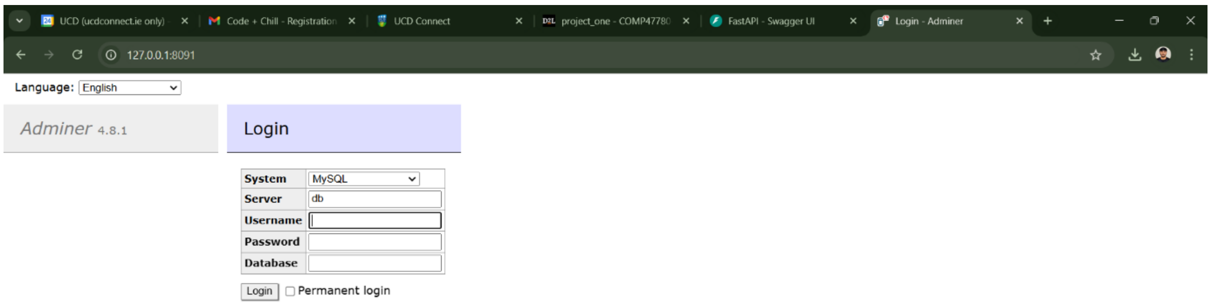
```
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171> docker run --name database -e POSTGRES_PASSWORD=password -d postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
a480a496ba95: Already exists
f5ece9c40e2b: Pull complete
241e5725184f: Pull complete
6832ae83547e: Pull complete
4db87ef10d0d: Pull complete
979fa3114f7b: Pull complete
f2bc6009bf64: Pull complete
c9097748b1df: Pull complete
9d5c934890a8: Pull complete
d14a7815879e: Pull complete
442a42d0b75a: Pull complete
82020414c082: Pull complete
b6ce4c941ce7: Pull complete
42e63a35cca7: Pull complete
Digest: sha256:8d3be35b184e70d81e54cbcbd3df3c0b47f37d06482c0dd1c140db5dbcc6a808
Status: Downloaded newer image for postgres:latest
3eef07cd575acd4372a8aec0a60e90d9fde81acb21b330ff600200cc81b11dac
```

2. Setting Up the Adminer Service

2.1 Running the Adminer Container

To start a new instance of Adminer and map it to port 8091 on the host, I executed the following command:

Command: docker run -p 8091:8080 adminer



127.0.0.1:8091

Language: English

System	MySQL
Server	db
Username	
Password	
Database	

Login ☐ Permanent login

2.2 Accessing Adminer

I navigated to the following URL to connect to the database:

- http://127.0.0.1:8091

However, I encountered an error when trying to connect to the database due to the isolation of containers.

3. Configuring the Network

3.1 Creating a New Network

To resolve the connection issue, I stopped all running instances of the containers and created a new network called backend using the following command:

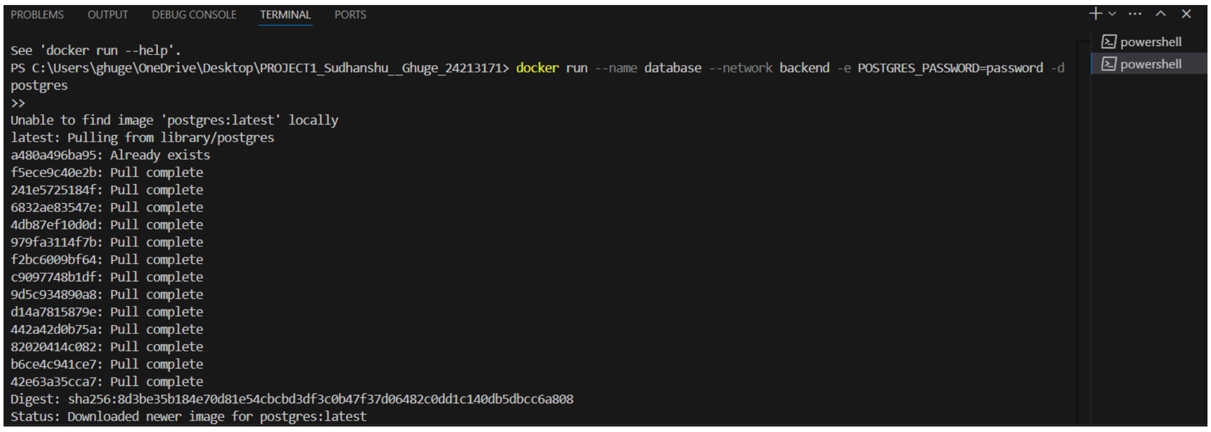
Command: docker network create backend

3.2 Re-running the Containers

I re-ran both the PostgreSQL and Adminer containers while attaching them to the backend network.

- Running the PostgreSQL Container

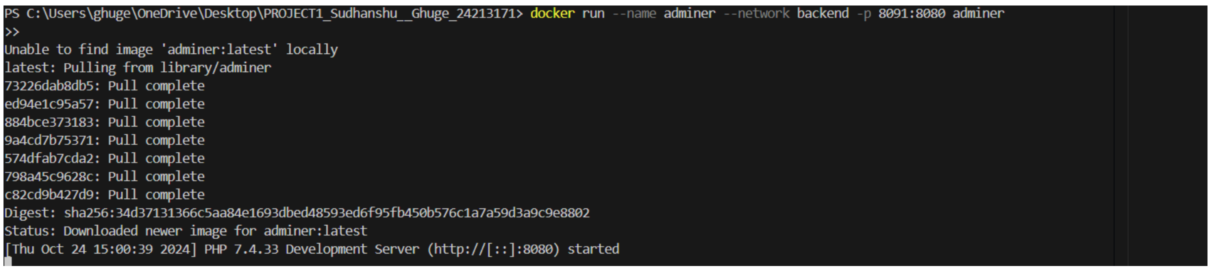
Command: docker run --name database --network backend -e POSTGRES_PASSWORD=password -d postgres



```
See 'docker run --help'.
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171> docker run --name database --network backend -e POSTGRES_PASSWORD=password -d
postgres
>>
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
a480a496ba95: Already exists
f5ece9c40e2b: Pull complete
241e5725184f: Pull complete
6832ae83547e: Pull complete
4db87ef10d0d: Pull complete
979fa3114f7b: Pull complete
f2bc6009bf64: Pull complete
c9097748b1df: Pull complete
9d5c934890a8: Pull complete
d14a7815879e: Pull complete
442a42d0b75a: Pull complete
82020414c082: Pull complete
b6ce4c941ce7: Pull complete
42e63a35cca7: Pull complete
Digest: sha256:8d3be35b184e70d81e54cbcbd3df3c0b47f37d06482c0dd1c140db5dbcc6a808
Status: Downloaded newer image for postgres:latest
```

- Running the Adminer Container

Command: docker run -p 8091:8080 --network backend adminer

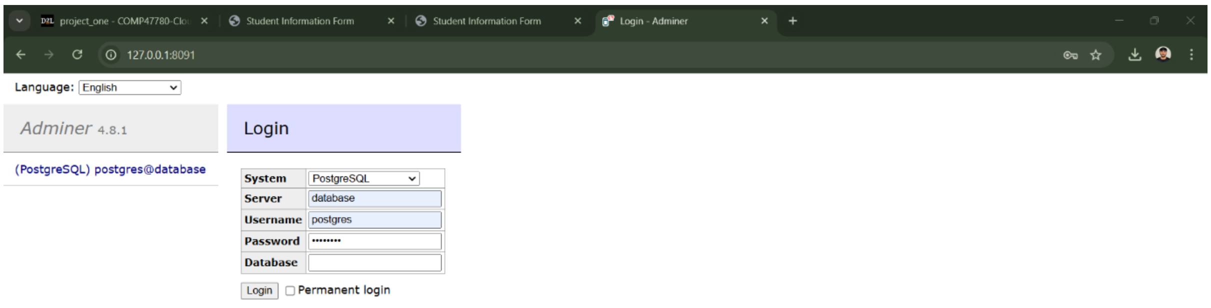


```
PS C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171> docker run --name adminer --network backend -p 8091:8080 adminer
>>
Unable to find image 'adminer:latest' locally
latest: Pulling from library/adminer
73226dab8db5: Pull complete
ed94e1c95a57: Pull complete
884bce373183: Pull complete
9a4cd7b75371: Pull complete
574dfab7cda2: Pull complete
798a45c9628c: Pull complete
c82cd9b427d9: Pull complete
Digest: sha256:34d37131366c5aa84e1693dbed48593ed6f95fb450b576c1a7a59d3a9c9e8802
Status: Downloaded newer image for adminer:latest
[Thu Oct 24 15:00:39 2024] PHP 7.4.33 Development Server (http://[::]:8080) started
```

4. Connecting to the Database

Now that both containers are on the same network, I used the container name as the server in Adminer to connect to the PostgreSQL database.

- **Server:** database
- **Username:** postgres
- **Password:** password



127.0.0.1:8091

Language: English

Adminer 4.8.1

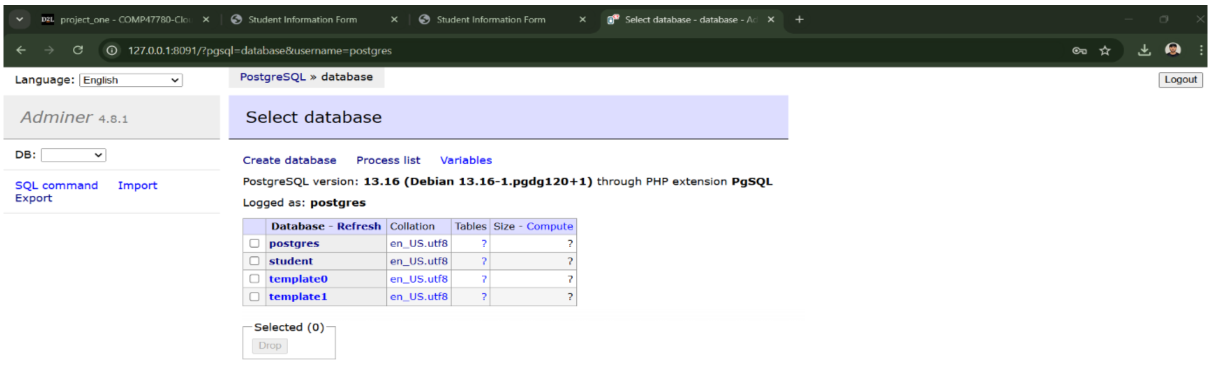
(PostgreSQL) postgres@database

Login

System	PostgreSQL
Server	database
Username	postgres
Password	••••••••
Database	

Login ☐ Permanent login

After logging in:



127.0.0.1:8091/?pgsql=database&username=postgres

Language: English

PostgreSQL » database

Logout

Adminer 4.8.1

DB:

SQL command Import Export

Select database

Create database Process list Variables

PostgreSQL version: **13.16 (Debian 13.16-1.pgdg120+1)** through PHP extension **PgSQL**

Logged as: **postgres**

	Database - Refresh	Collation	Tables	Size - Compute
☐	postgres	en_US.utf8	?	?
☐	student	en_US.utf8	?	?
☐	template0	en_US.utf8	?	?
☐	template1	en_US.utf8	?	?

Selected (0)

Drop

Exercise Four: Docker Compose Setup

1. Setup Process

1. 1 Downloading the Docker Compose File

I downloaded docker-compose.yaml from Brightspace and placed it in the root of my working directory. The directory structure now looks as follows:

/PROJECT1_Sudhanshu_Ghuge_24213171/

/api

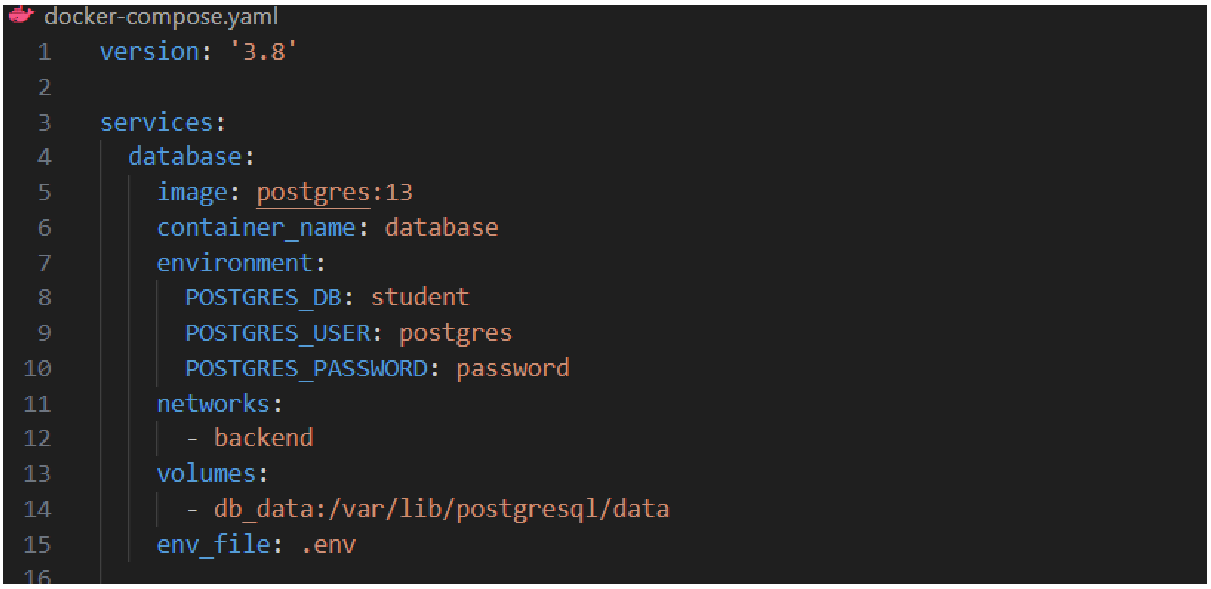
/web_app

/docker-compose.yaml

1.2 Configuring the docker-compose.yaml File

I modified the docker-compose.yaml file to set up the required services based on the provided constraints.

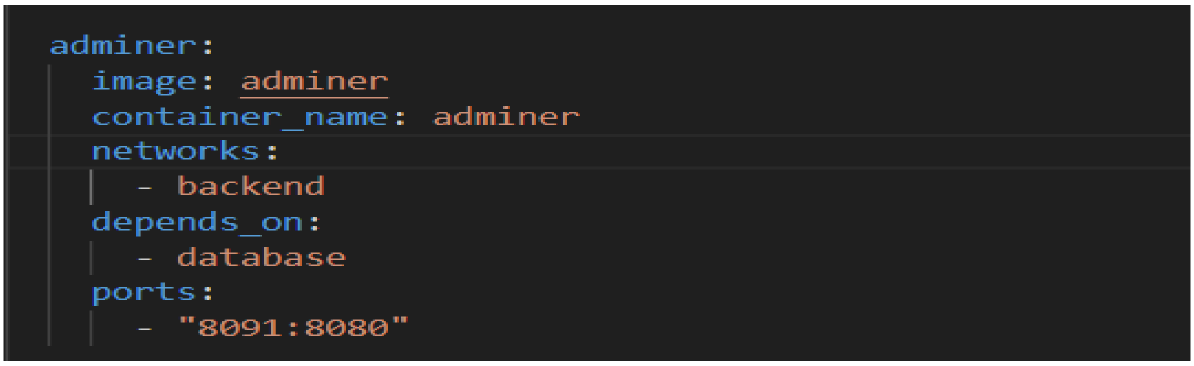
a. Database Service



docker-compose.yaml

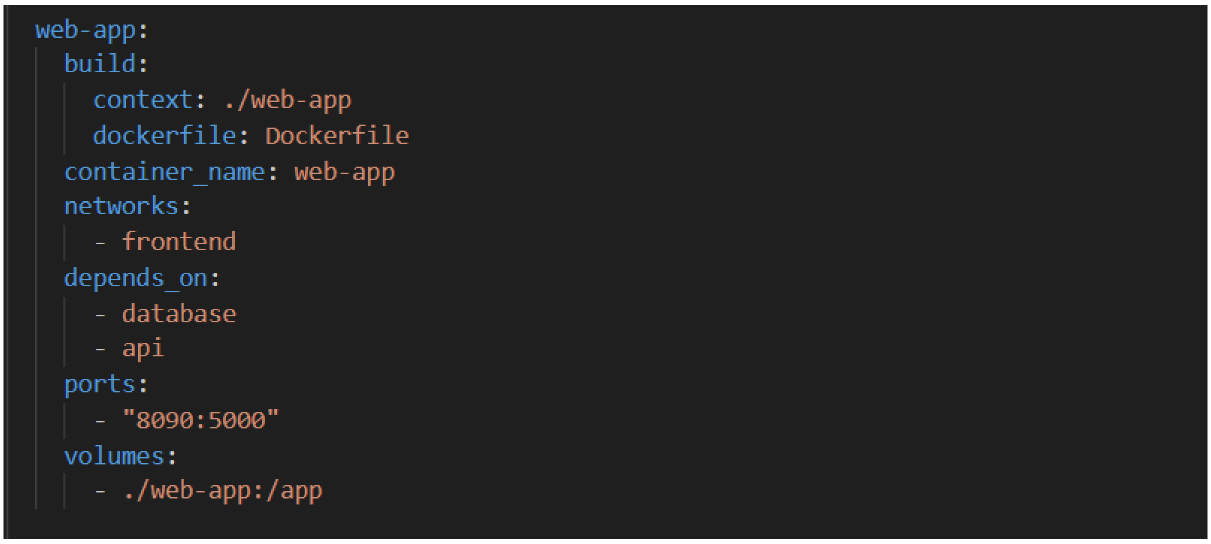
```
version: '3.8'

services:
  database:
    image: postgres:13
    container_name: database
    environment:
      POSTGRES_DB: student
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: password
    networks:
      - backend
    volumes:
      - db_data:/var/lib/postgresql/data
    env_file: .env
```

b. Adminer



```
adminer:
  image: adminer
  container_name: adminer
  networks:
    - backend
  depends_on:
    - database
  ports:
    - "8091:8080"
```

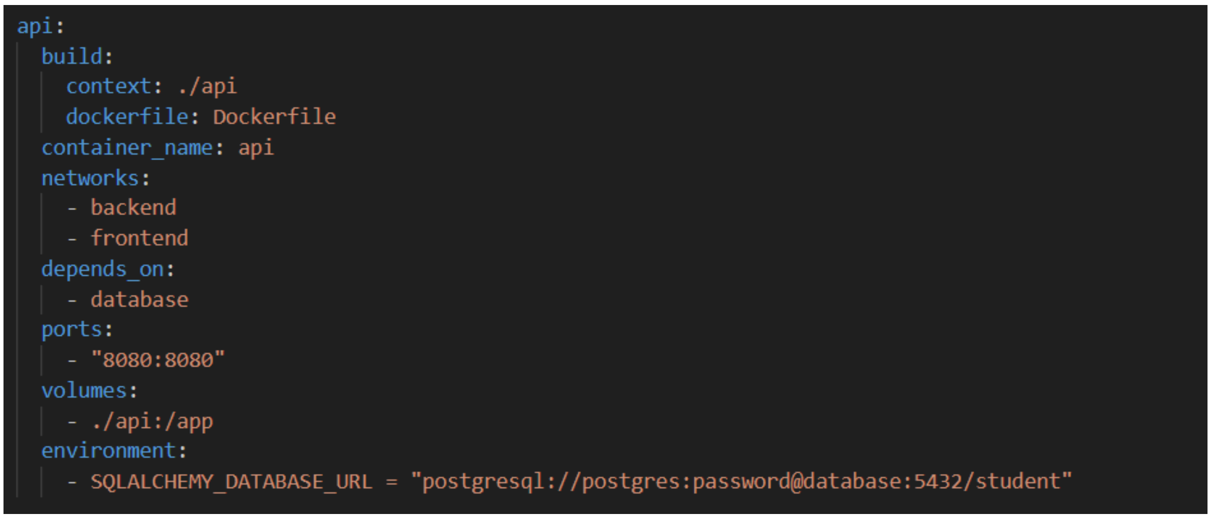
c. web-app



```yaml
web-app:
  build:
    context: ./web-app
    dockerfile: Dockerfile
  container_name: web-app
  networks:
    - frontend
  depends_on:
    - database
    - api
  ports:
    - "8090:5000"
  volumes:
    - ./web-app:/app
```

d. api

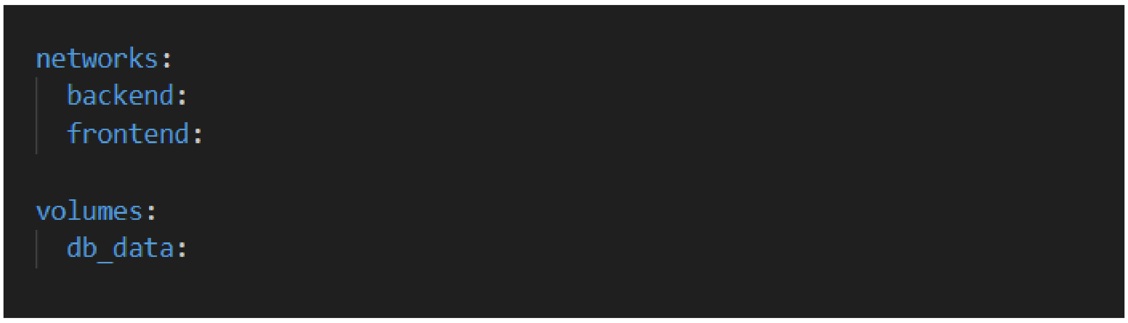
Copied the contents of app_copy.py into app.py and student_copy.html into student.html manually.



```yaml
api:
  build:
    context: ./api
    dockerfile: Dockerfile
  container_name: api
  networks:
    - backend
    - frontend
  depends_on:
    - database
  ports:
    - "8080:8080"
  volumes:
    - ./api:/app
  environment:
    - SQLALCHEMY_DATABASE_URL = "postgresql://postgres:password@database:5432/student"
```

1.3 Network Configuration

Defined two networks, backend and frontend, in the docker-compose.yaml file to ensure proper isolation and communication between services.

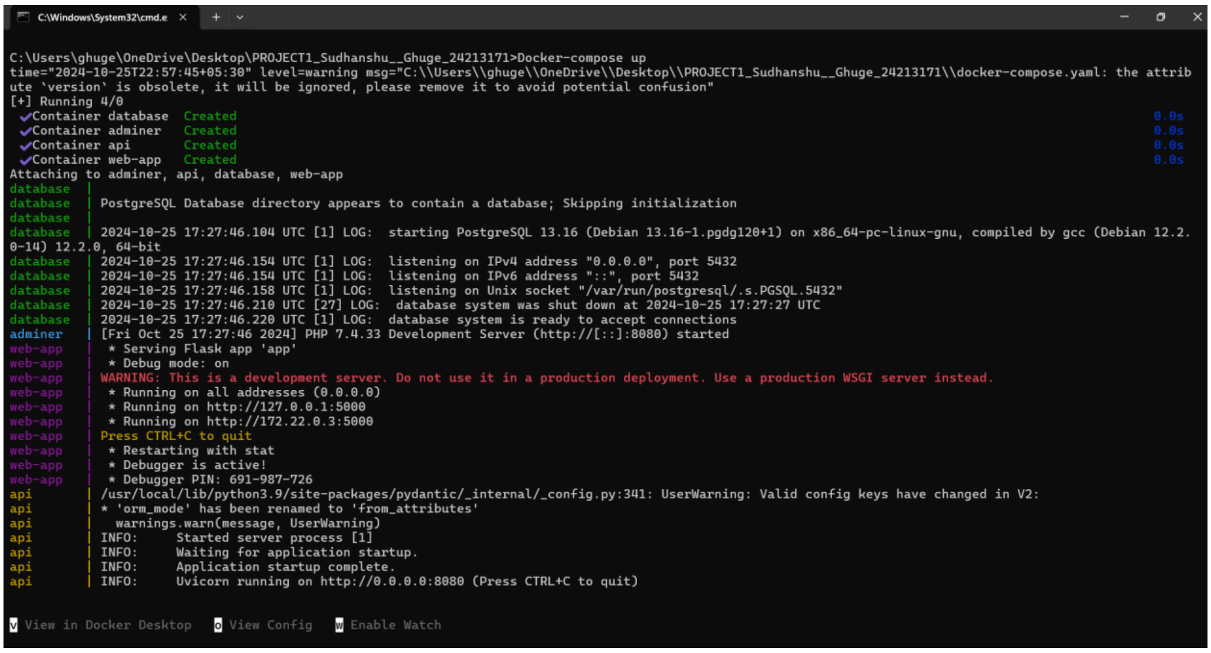


```yaml
networks:
  backend:
  frontend:

volumes:
  db_data:
```

2. Running Docker Compose

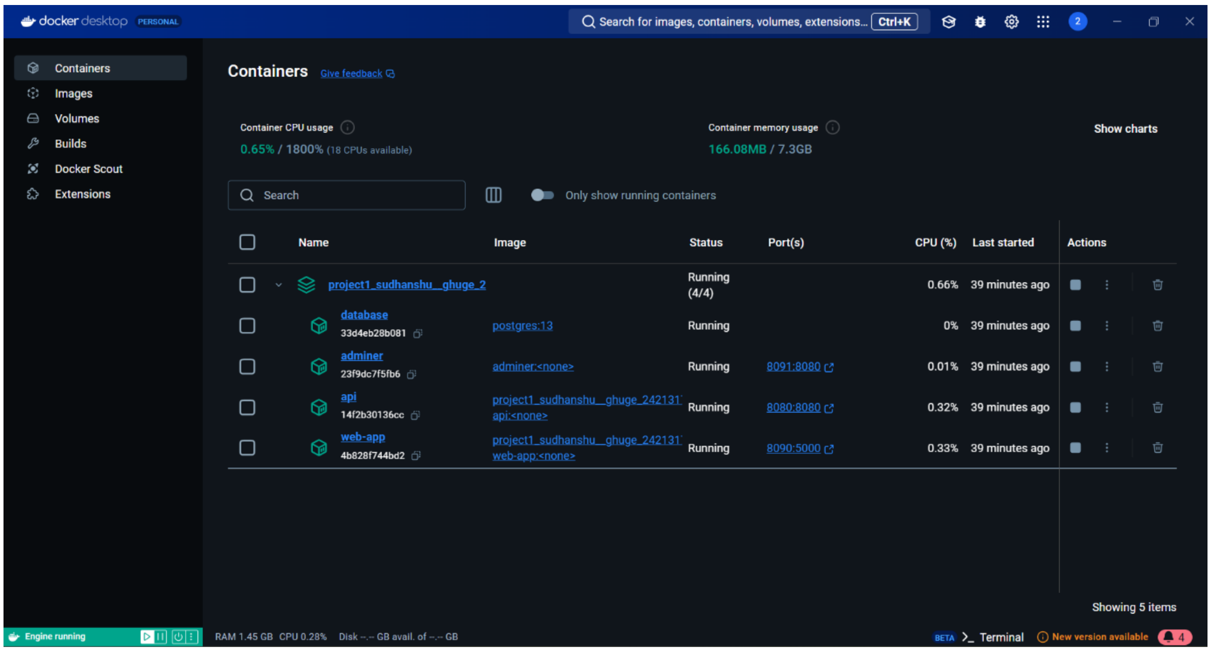
After completing the configuration, I executed the following command to start the services:

Command: docker-compose up



```
C:\Users\ghuge\OneDrive\Desktop\PROJECT1_Sudhanshu__Ghuge_24213171>Docker-compose up
time="2024-10-25T22:57:45+05:30" level=warning msg="C:\\Users\\ghuge\\OneDrive\\Desktop\\PROJECT1_Sudhanshu__Ghuge_24213171\\docker-compose.yaml: the attrib
ute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 4/0
 ✔Container database  Created                                                                                                                         0.0s
 ✔Container adminer   Created                                                                                                                         0.0s
 ✔Container api       Created                                                                                                                         0.0s
 ✔Container web-app   Created                                                                                                                         0.0s
Attaching to adminer, api, database, web-app
database  |
database  | PostgreSQL Database directory appears to contain a database; Skipping initialization
database  |
database  | 2024-10-25 17:27:46.104 UTC [1] LOG:  starting PostgreSQL 13.16 (Debian 13.16-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.
0-14) 12.2.0, 64-bit
database  | 2024-10-25 17:27:46.154 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
database  | 2024-10-25 17:27:46.154 UTC [1] LOG:  listening on IPv6 address "::", port 5432
database  | 2024-10-25 17:27:46.158 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
database  | 2024-10-25 17:27:46.210 UTC [27] LOG:  database system was shut down at 2024-10-25 17:27:27 UTC
database  | 2024-10-25 17:27:46.220 UTC [1] LOG:  database system is ready to accept connections
adminer   | [Fri Oct 25 17:27:46 2024] PHP 7.4.33 Development Server (http://[::]:8080) started
web-app   |  * Serving Flask app 'app'
web-app   |  * Debug mode: on
web-app   | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
web-app   |  * Running on all addresses (0.0.0.0)
web-app   |  * Running on http://127.0.0.1:5000
web-app   |  * Running on http://172.22.0.3:5000
web-app   | Press CTRL+C to quit
web-app   |  * Restarting with stat
web-app   |  * Debugger is active!
web-app   |  * Debugger PIN: 691-987-726
api       | /usr/local/lib/python3.9/site-packages/pydantic/_internal/_config.py:341: UserWarning: Valid config keys have changed in V2:
api       | * 'orm_mode' has been renamed to 'from_attributes'
api       |   warnings.warn(message, UserWarning)
api       | INFO:     Started server process [1]
api       | INFO:     Waiting for application startup.
api       | INFO:     Application startup complete.
api       | INFO:     Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)

v View in Docker Desktop   o View Config   w Enable Watch
```

Docker Containers:



Containers

Container CPU usage 0.65% / 1800% (18 CPUs available)

Container memory usage 166.08MB / 7.3GB

	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
	project1_sudhanshu__ghuge_2		Running (4/4)		0.66%	39 minutes ago	
	database 33d4eb28b081	postgres:13	Running		0%	39 minutes ago	
	adminer 23f9dc7f5fb6	adminer:<none>	Running	8091:8080	0.01%	39 minutes ago	
	api 14f2b30136cc	project1_sudhanshu__ghuge_242131 api:<none>	Running	8080:8080	0.32%	39 minutes ago	
	web-app 4b828f744bd2	project1_sudhanshu__ghuge_242131 web-app:<none>	Running	8090:5000	0.33%	39 minutes ago	

Showing 5 items

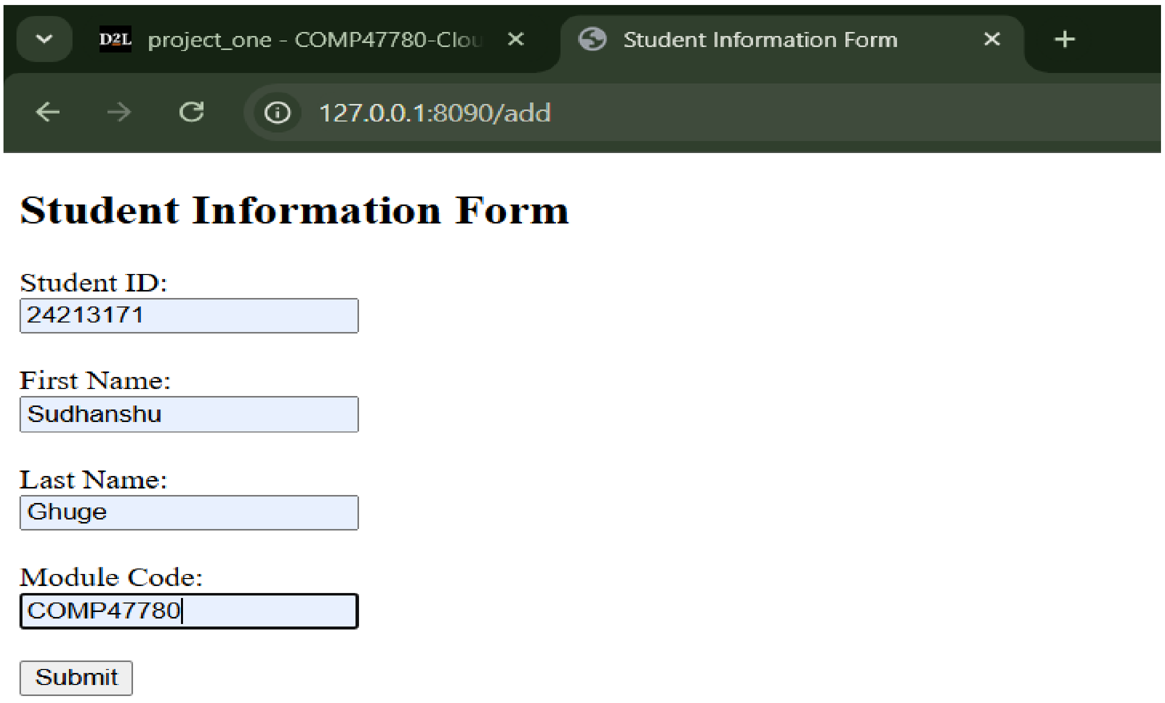
RAM 1.45 GB CPU 0.28% Disk --.-- GB avail. of --.-- GB

BETA Terminal New version available 4

3. Browsing the service

I navigated to the following URLs to interact with the web application and Adminer:

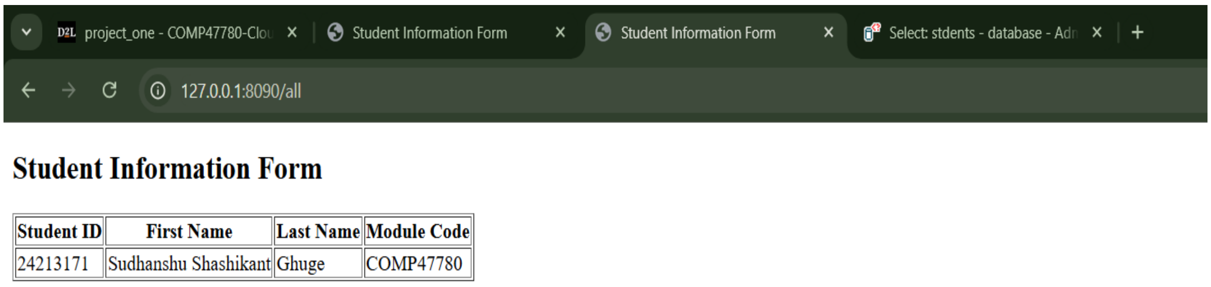
- Web-App: add
  - I filled in the form on the web application and submitted it.
  - http://127.0.0.1:8090/add



Student Information Form

Student ID:	24213171
First Name:	Sudhanshu
Last Name:	Ghuge
Module Code:	COMP47780

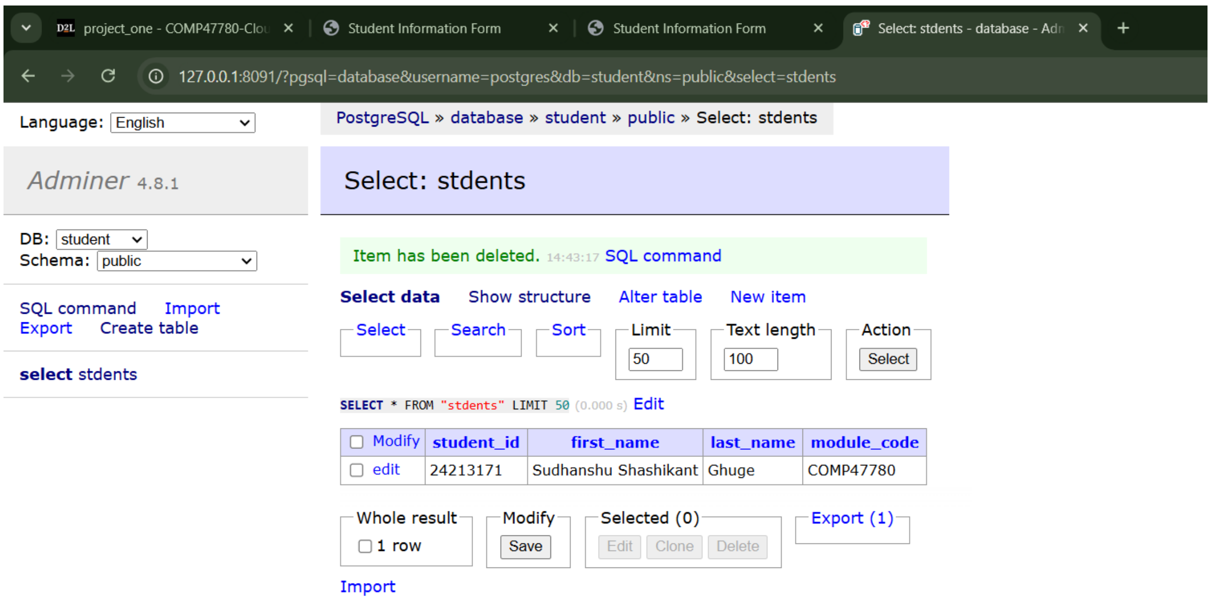
- Web-app: all
  - The data was displayed in the form.
  - https://127.0.0.1:8090/all



Student Information Form

Student ID	First Name	Last Name	Module Code
24213171	Sudhanshu Shashikant	Ghuge	COMP47780

- Adminer
  - Checked Adminer to verify the saved data.
  - http://127.0.0.1:8091/



Select: stdents

Item has been deleted. 14:43:17 SQL command

```
SELECT * FROM "stdents" LIMIT 50 (0.000 s)
```

Modify	student_id	first_name	last_name	module_code
edit	24213171	Sudhanshu Shashikant	Ghuge	COMP47780

1 row

Conclusion:

To summarize, this project was a practical journey through setting up and managing a multi-container environment with Docker and Docker Compose. Creating separate containers for a Flask web app, FastAPI API, PostgreSQL, and Adminer taught key skills in container setup and networking. Docker Compose made it easy to manage these services together, allowing them to work smoothly as a cohesive system. Each exercise demonstrated how Docker helps scale and organize complex applications. Overall, it was a solid foundation in deploying efficient, containerized solutions.