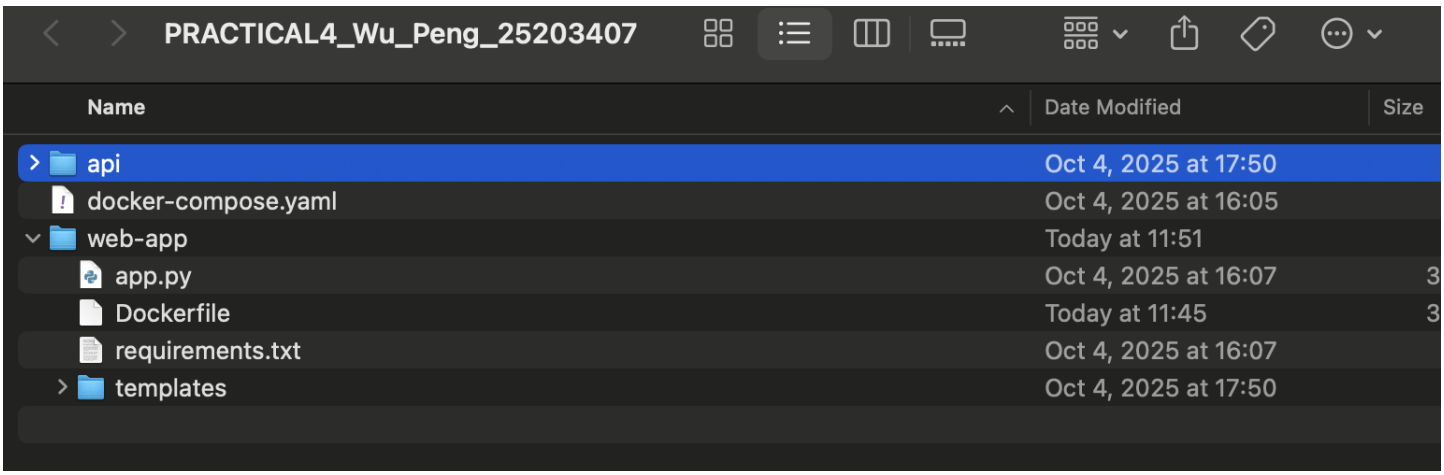
Lab4

1. *Exercise One Web-App setup with Flask*

1.1 1.1 Project Structure

Make sure your folder looks like this:



1.2 Create a Dockerfile in `web_app/`

Go to the folder:

```
pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % ls
api                     docker-compose.yaml     web-app
pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % cd web-app
pen_warren@wus-MacBook-Pro web-app % ls
app.py                  requirements.txt        templates
pen_warren@wus-MacBook-Pro web-app %
```

Create the Dockerfile:

```
pen_warren@wus-MacBook-Pro web-app % touch Dockerfile
pen_warren@wus-MacBook-Pro web-app % ls
Dockerfile              app.py                  requirements.txt        templates
pen_warren@wus-MacBook-Pro web-app % vim Dockerfile
pen_warren@wus-MacBook-Pro web-app % ls
Dockerfile              app.py                  requirements.txt        templates
pen_warren@wus-MacBook-Pro web-app % vim Dockerfile
pen_warren@wus-MacBook-Pro web-app %
```

Then open it and add the following: (The Dockerfile)

```
# Use Python 3.9 slim as base image
FROM python:3.9-slim

# Create a working directory inside the container
WORKDIR /app

# Copy all source files from current directory to the container
COPY . /app

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Expose port 5000 inside the container
EXPOSE 5000

# Run the web app when the container starts
CMD ["python", "app.py"]
```

Verify that the Dockerfile was written successfully

```
pen_warren@wus-MacBook-Pro web-app % cat Dockerfile
# Use Python 3.9 slim as base image
FROM python:3.9-slim

# Create a working directory inside the container
WORKDIR /app

# Copy all source files from current directory to the container
COPY . /app

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Expose port 5000 inside the container
EXPOSE 5000

# Run the web app when the container starts
CMD ["python", "app.py"]

pen_warren@wus-MacBook-Pro web-app %
```

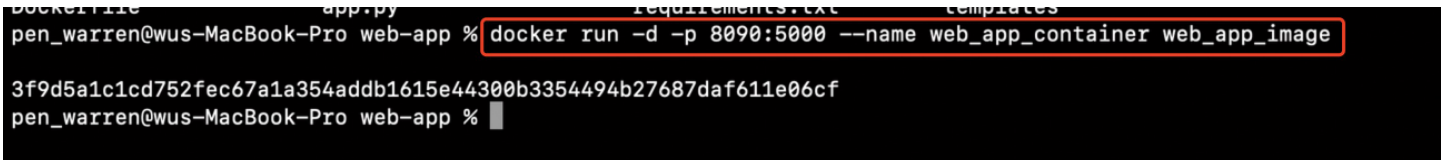
1.3 Build the Docker Image

Run the following command **inside the web_app directory**:

This creates a Docker image called `web_app_image`.

1.4 Run the Container

Now run the container and map ports:



This means:

- Port **8090** on my **host machine** → connects to port **5000** inside the **container**.

1.5 Test the Application

Test in browser or curl:

- http://127.0.0.1:8090/add

Student Information Form

Student ID:

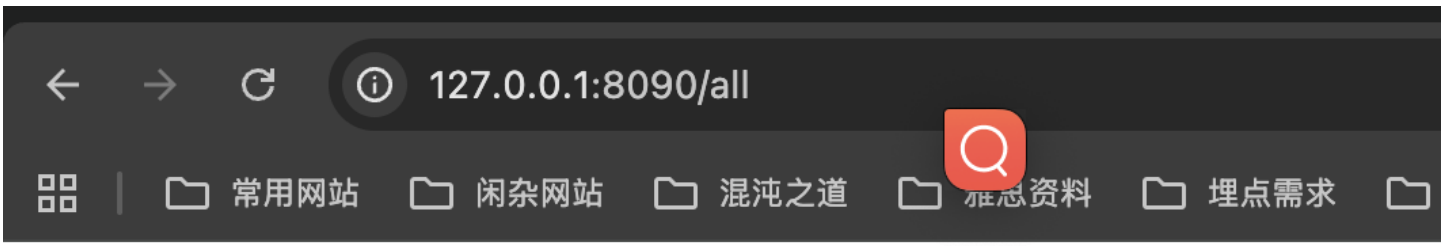
First Name:

Last Name:

Module Code:

Submit

- http://127.0.0.1:8090/all

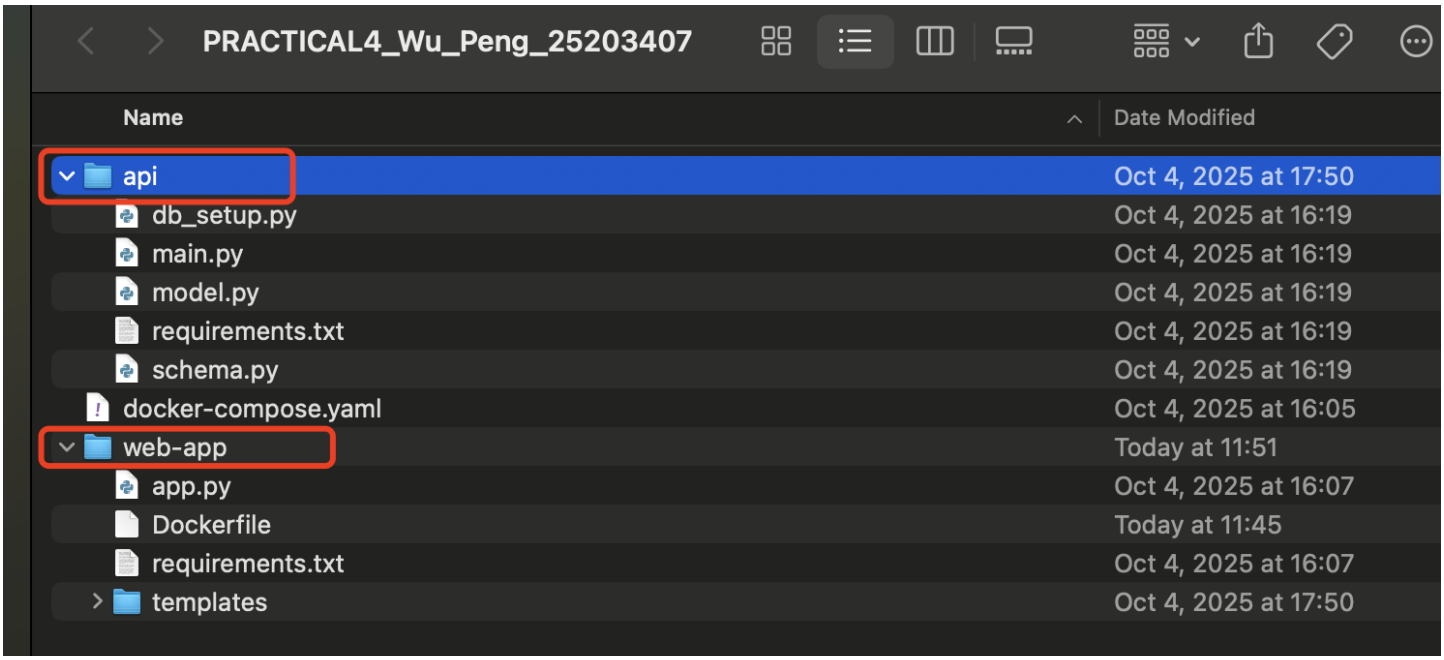


Student Information Form

Details will go here

2. *Exercise Two - API Setup with FastAPI*

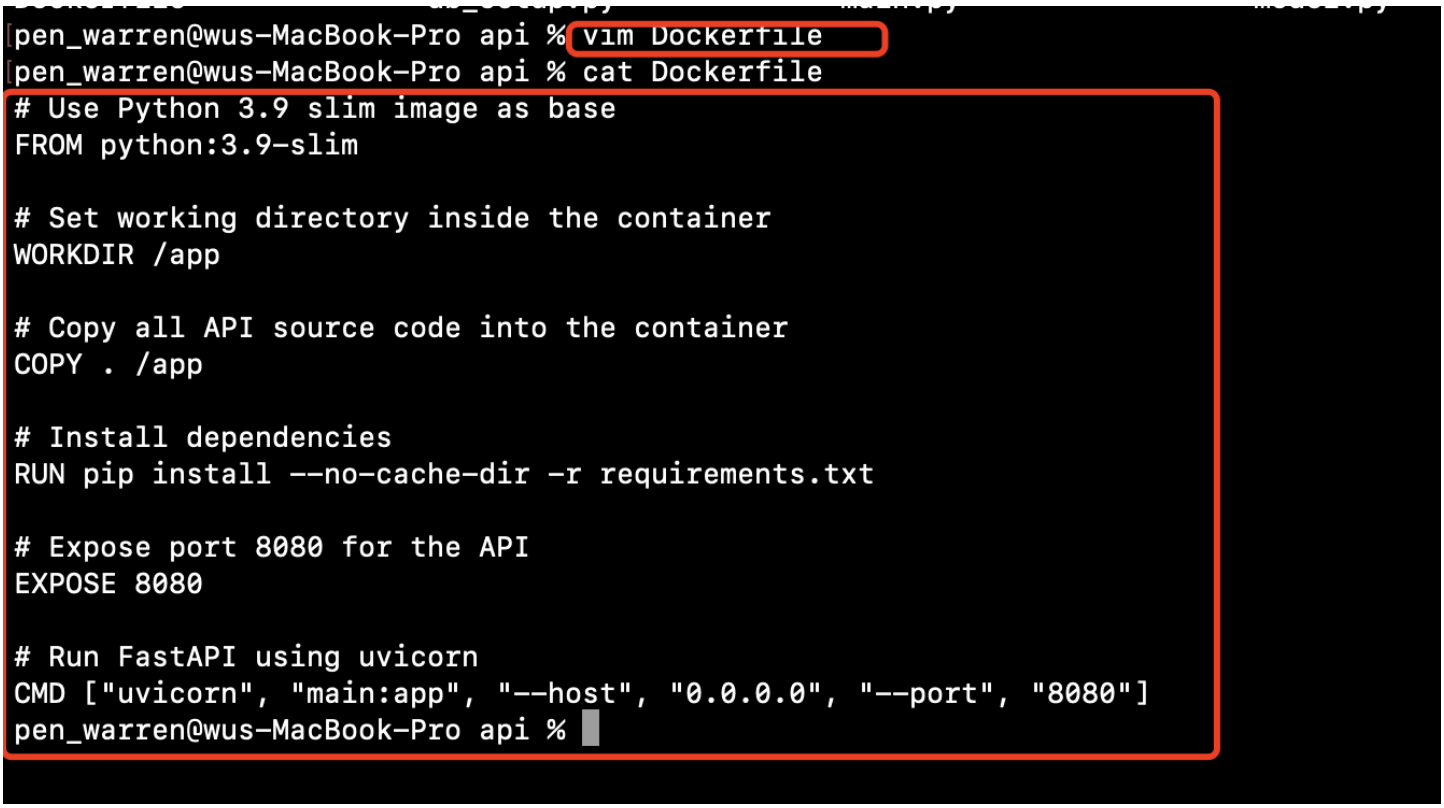
2.1 Project Structure



2.2 Create the `Dockerfile` inside the `api/` folder

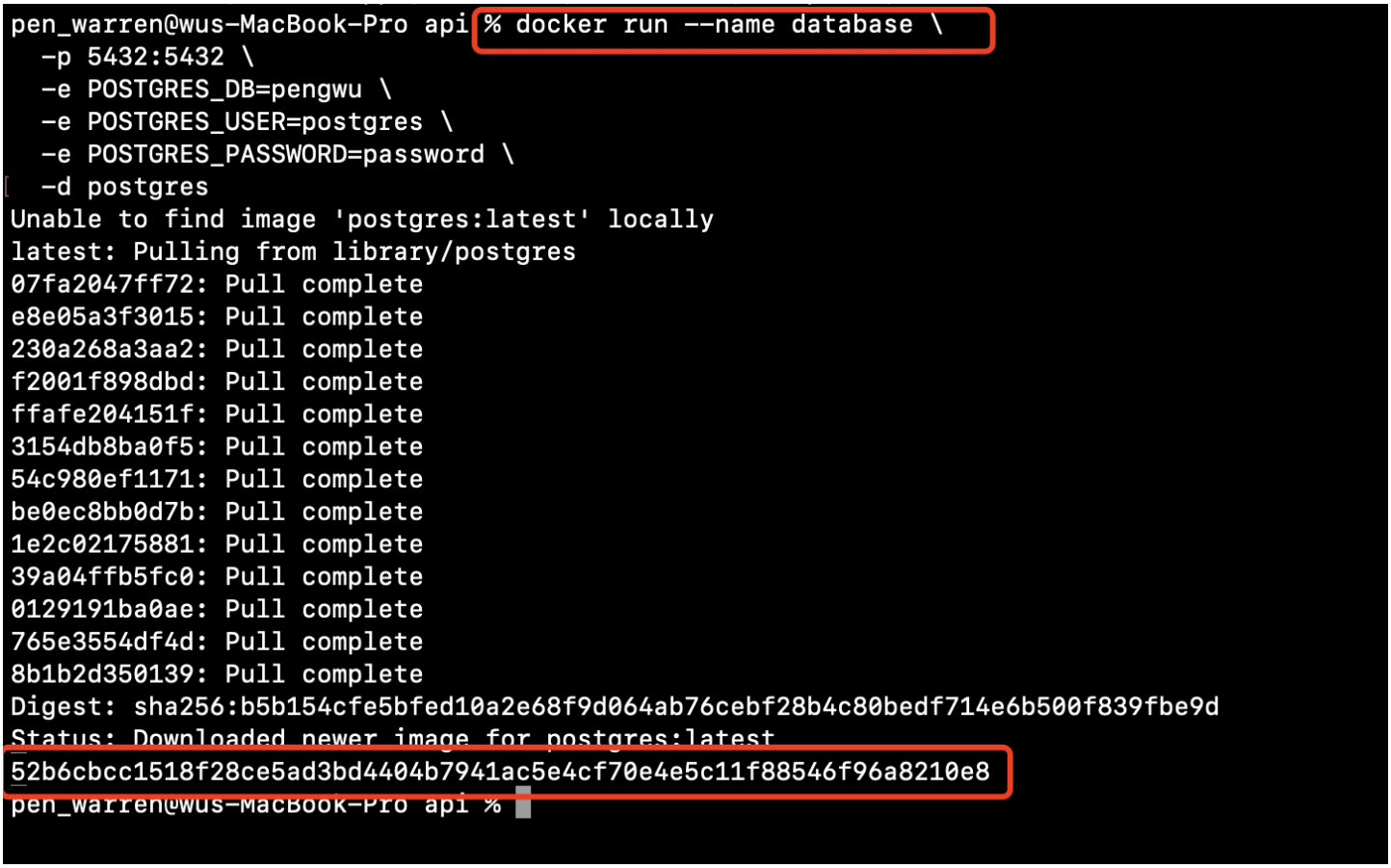
Create a file named `Dockerfile` inside `api/`, and paste this:

Write into Dockerfile:



2.3 Start a temporary PostgreSQL database container

Run this command:



This creates a PostgreSQL database named pengwu on port **5432**.

2.4 Connect FastAPI to the database

Edit `db_setup.py` :

db_setup.py

Users > pen_warren > Desktop > PRACTICAL4_Wu_Peng_25203407 > api > db_setup.py

```python
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

# SQLAlchemy settings for SQLite database
SQLALCHEMY_DATABASE_URL = "postgresql://postgres:password@localhost:5432/pengwu"

# Create the SQLAlchemy engine
engine = create_engine(
    SQLALCHEMY_DATABASE_URL
)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

```

2.5 Create a custom network

Create a network and add the database:

```
pen_warren@wus-MacBook-Pro api % docker network create --driver bridge --attachable app-network
0f815309030c8c6775b3d1dd2756616b744b974152d6596470bae5ee7d4f380b
pen_warren@wus-MacBook-Pro api % docker network inspect app-network
[
    {
        "Name": "app-network",
        "Id": "0f815309030c8c6775b3d1dd2756616b744b974152d6596470bae5ee7d4f380b",
        "Created": "2025-10-21T15:19:51.346987879Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.28.0.0/16",
                    "Gateway": "172.28.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.enable_ipv4": "true",
            "com.docker.network.enable_ipv6": "false"
        },
        "Labels": {}
    }
]
```

Check if it was created successfully

```
pen_warren@wus-MacBook-Pro api % docker network ls

NETWORK ID     NAME                       DRIVER    SCOPE
1f0d12595f5f   app-network                bridge    local
9cd90cb0d6e3   bridge                     bridge    local
0ba2b26f8eba   codu_default               bridge    local
22945d0478d3   elasticsearch_default      bridge    local
781ce76962f7   hippo4j_default            bridge    local
bf2ffdc6da25   host                       host      local
fd105e910ae0   jenkins_default            bridge    local
f47bcd0b70fe   nacos_default              bridge    local
5433c3bbf4c8   nginx_default              bridge    local
b081cf8b892e   none                       null      local
cc5bdd9d614c   rabbitmq_default           bridge    local
a1b52a29bbd4   redis-culster_redis-net    bridge    local
e37e41d29d89   socket_lab_default         bridge    local
6ec15a78561c   xxljob_default             bridge    local
pen_warren@wus-MacBook-Pro api %
```

Delete the database :

```
pen_warren@wus-MacBook-Pro api % docker rm -f database

database
pen_warren@wus-MacBook-Pro api %
```

Recreate:

```
pen_warren@wus-MacBook-Pro api % docker run -d --name database \
  --network app-network \
  -e POSTGRES_DB=pengwu \
  -e POSTGRES_USER=postgres \
  -e POSTGRES_PASSWORD=password \
  -p 5433:5432 \
  postgres

2a25b160264d10f27a947b32108e63dcbfd671efc39877382810d1c2b2110033
pen_warren@wus-MacBook-Pro api %
```

Check :

```
pen_warren@wus-MacBook-Pro api % docker ps -a | grep database

2a25b160264d   postgres                  "docker-entrypoint.s…"   39 seconds ago   Up 39 seconds             0.0.0.0:5433->5432/tcp, [::]:5433->5432/tcp
                      database
pen_warren@wus-MacBook-Pro api %
```

2.6 Build and run your FastAPI container

From the `api/` directory:

```
pen_warren@wus-MacBook-Pro api % docker build -t fastapi-app .
[+] Building 168.6s (9/9) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                0.0s
 => => transferring dockerfile: 429B                                                                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                                                                  1.8s
 => [internal] load .dockerignore                                                                                                   0.0s
 => => transferring context: 2B                                                                                                     0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:69ad475ad27439666dca95aa73cf21b5b5249241e1e0544acf488d5181d91496            0.0s
 => => resolve docker.io/library/python:3.9-slim@sha256:69ad475ad27439666dca95aa73cf21b5b5249241e1e0544acf488d5181d91496            0.0s
 => [internal] load build context                                                                                                   0.0s
 => => transferring context: 9.07kB                                                                                                 0.0s
 => CACHED [2/4] WORKDIR /app                                                                                                       0.0s
 => [3/4] COPY . /app                                                                                                               0.0s
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt                                                                      164.4s
 => exporting to image                                                                                                              2.4s
 => => exporting layers                                                                                                             1.9s
 => => exporting manifest sha256:ba2ae4a896a37d997f6f02e8e283c6d1b4dfee5c7987c4b1a4eaa69f6ae2c78d                                    0.0s
 => => exporting config sha256:e3d46d34e0e5c47813a6f8deaa7c6b34e0f7e57aa02386b5b68640b7a7352a4a                                      0.0s
 => => exporting attestation manifest sha256:bf58d5d8ea3bf109cda7dd6a94d5f8d495fa114640e6c0c55e92826662772607                        0.0s
 => => exporting manifest list sha256:c9899a497fcd7769e3c99b0963c8080a068658094c75d60a619c0a2a5d550af6                              0.0s
 => => naming to docker.io/library/fastapi-app:latest                                                                               0.0s
 => => unpacking to docker.io/library/fastapi-app:latest                                                                            0.4s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/zpr118dmywz0xjy1nvjcwdlef
pen_warren@wus-MacBook-Pro api %
```

Then run it on the same network:

```
pen_warren@wus-MacBook-Pro api % docker run -d --name fastapi-app \
  --network app-network \
  -p 8080:8080 \
  fastapi-app

e976a5997e2f244b737803eb6bba96c462fcbc50dcd6636c243defab6346969b
```

Check the status of fast-app:

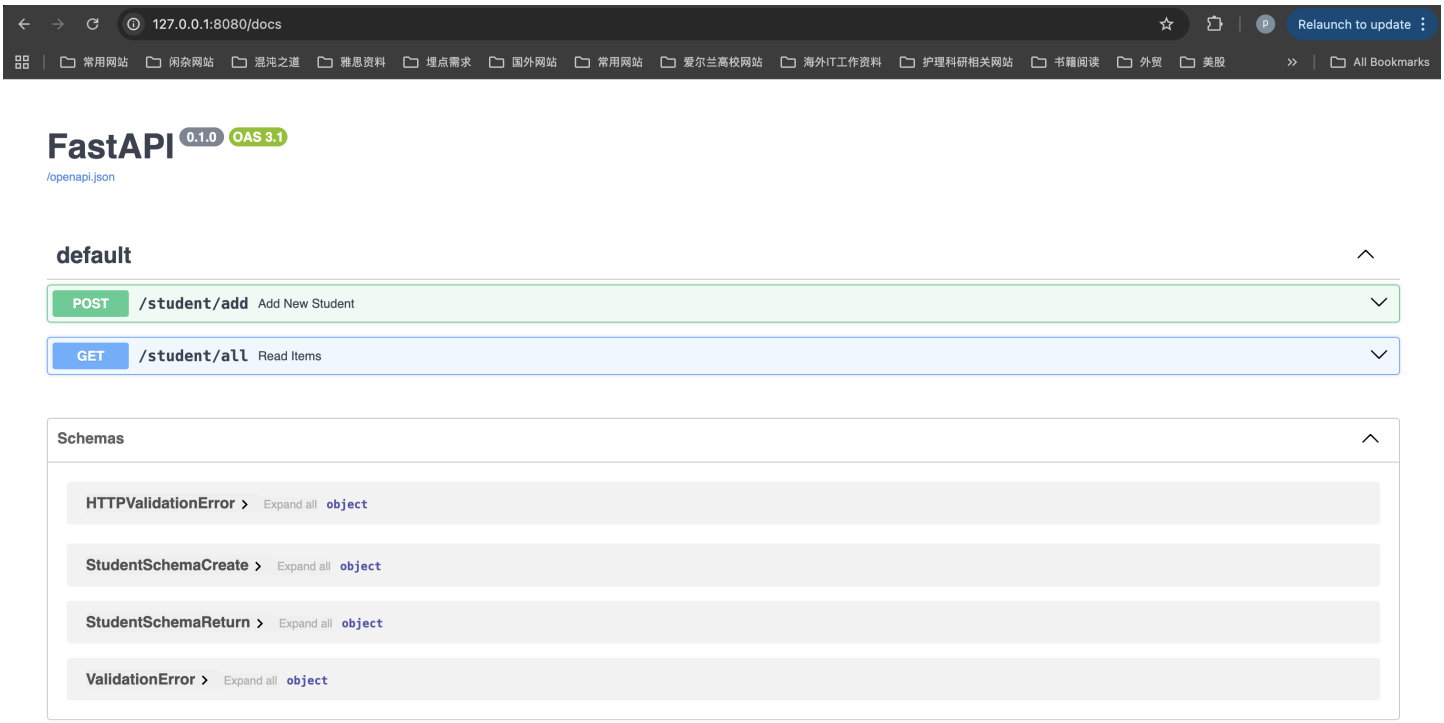
```
pen_warren@wus-MacBook-Pro api % docker ps -a | grep fastapi-app

e976a5997e2f   fastapi-app                   "uvicorn main:app --…"   47 seconds ago   Exited (1) 46 seconds ago
                      fastapi-app
pen_warren@wus-MacBook-Pro api %
```

Check the app-network inspect;

```
is-culster-redis-7006-1
pen_warren@wus-MacBook-Pro api % docker network inspect app-network

[
    {
        "Name": "app-network",
        "Id": "0f815309030c8c6775b3d1dd2756616b744b974152d6596470bae5ee7d4f380b",
        "Created": "2025-10-21T15:19:51.346987879Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.28.0.0/16",
                    "Gateway": "172.28.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "045aafa1b29df556fb73d4cdf9107b219a1004c4660a3833050153316dcb350e": {
                "Name": "fastapi-app",
                "EndpointID": "456cd68924757fb36ea21c965f1d6e62ff30667d23b4ba79e0d263f8b270deac",
                "MacAddress": "ce:95:86:c9:e0:59",
                "IPv4Address": "172.28.0.3/16",
                "IPv6Address": ""
            },
            "97989f5ccecd0a64214a101d7877e53cf33d5d3e40ab77c0648e36fdc78f1aa8": {
                "Name": "database",
                "EndpointID": "51effea0159c517b497e67672a590727e9edf82123586614de73941ad85f1c84",
                "MacAddress": "02:0c:1f:64:99:39",
                "IPv4Address": "172.28.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.enable_ipv4": "true",
            "com.docker.network.enable_ipv6": "false"
        },
        "Labels": {}
    }
]
```

2.7 Verify everything works

check the address: **http://127.0.0.1:8080/docs**



3. *Exercise Three - Database service*

3.1 Run PostgreSQL container

```
-culster-redis-7006-1
pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % docker run --name database -e POSTGRES_PASSWORD=password -d postgres
docker: Error response from daemon: Conflict. The container name "/database" is already in use by container "97989f5ccecd0a64214a101d7877e53cf33d5d3e40ab77c0648e36fdc78f1aa8". You have to
 remove (or rename) that container to be able to reuse that name.

Run 'docker run --help' for more information
pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 %
```

it already started

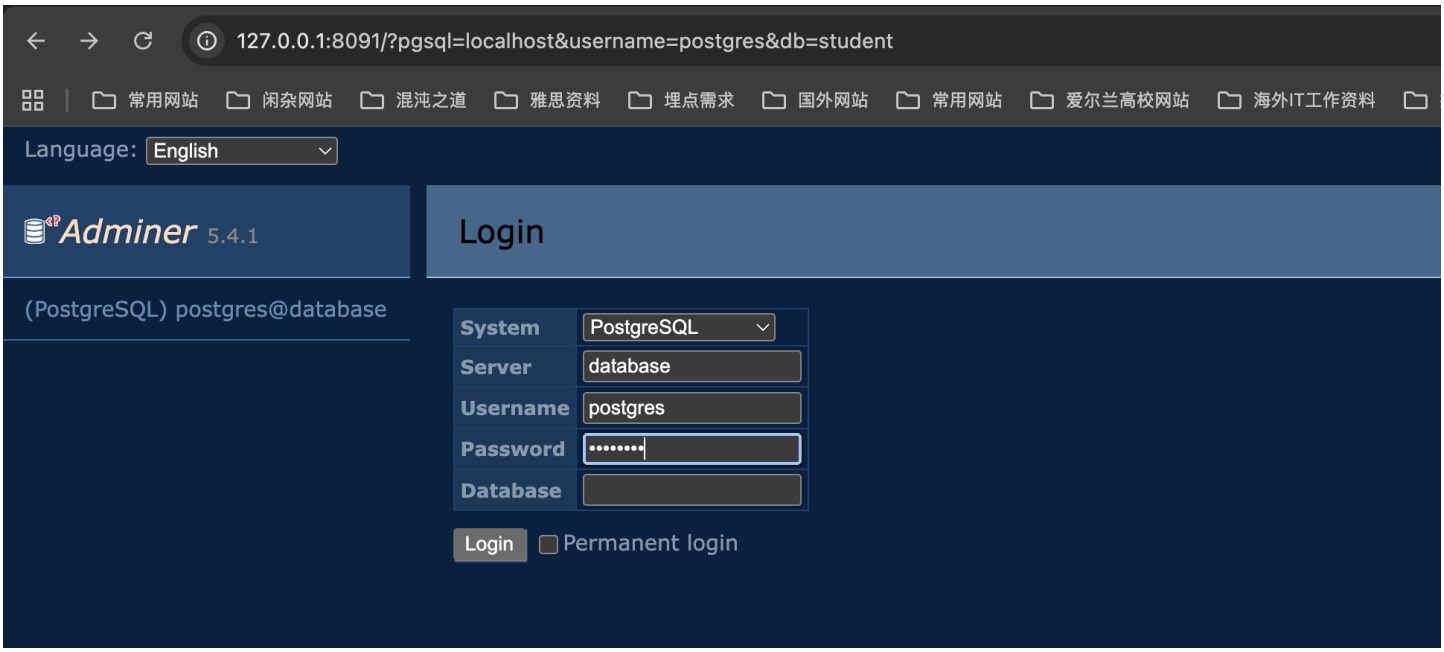
3.2 Run Adminer container

I already finished the network creation at Exercise 2, the network named `app-network`

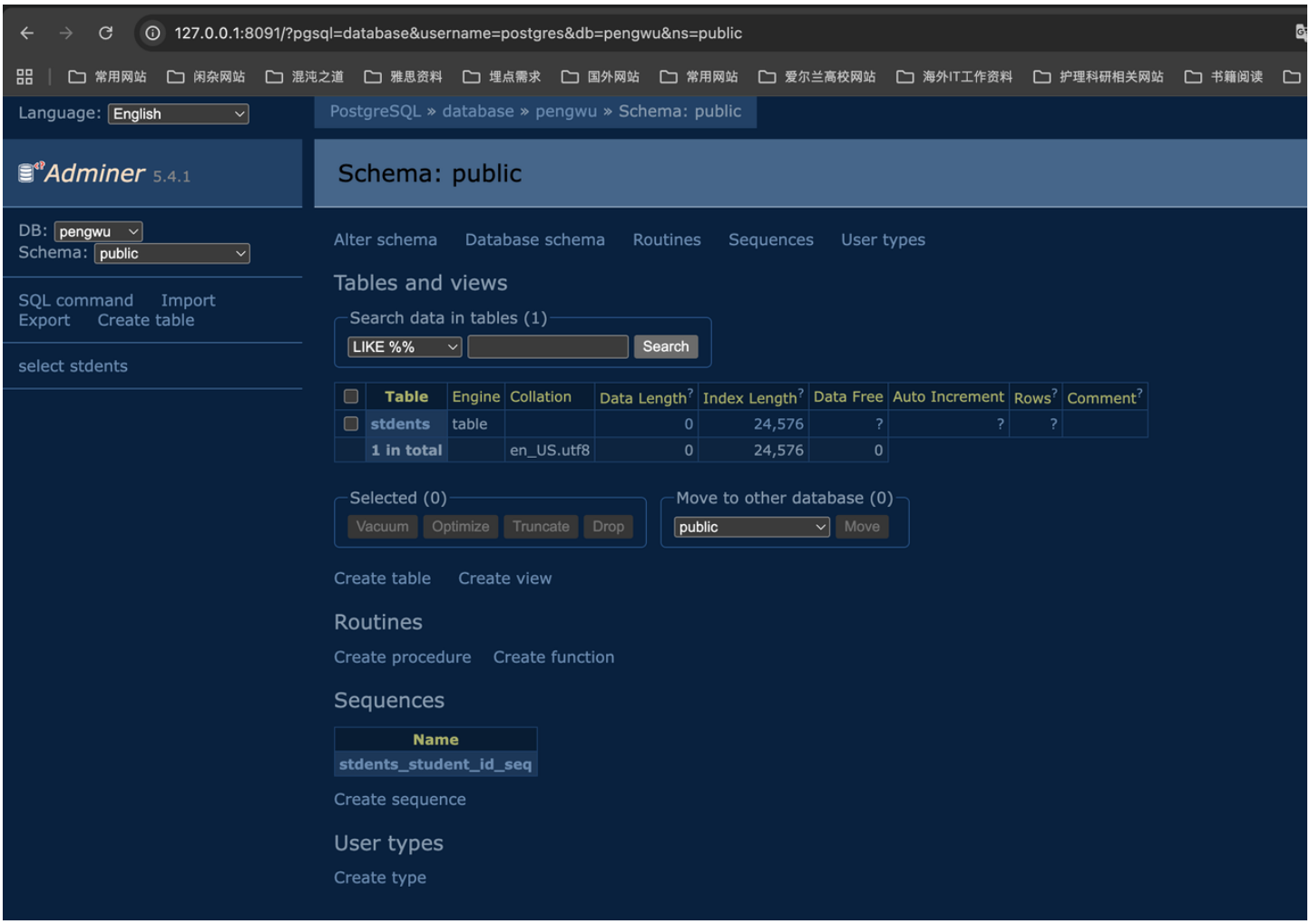
```
6ec15a78561c   xxljob_default            bridge    local
pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % docker run -d \
  -p 8091:8080 \
  --name adminer \
  --network app-network \
  adminer
66c1c25a51ee4db25b761992d2fb673d239a449441c59de7a161be2d7b4e2990
```

3.3 Connect Adminer to PostgreSQL

The login page:



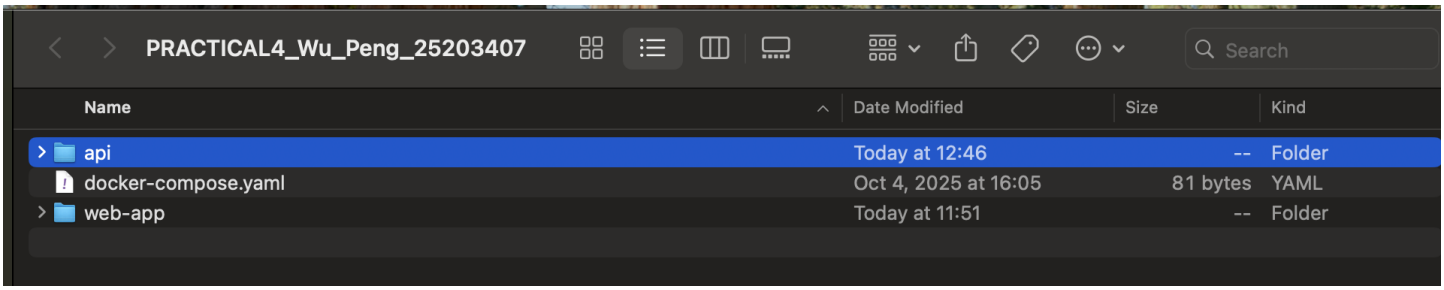
Success login:



4. *Exercise Four - Docker compose*

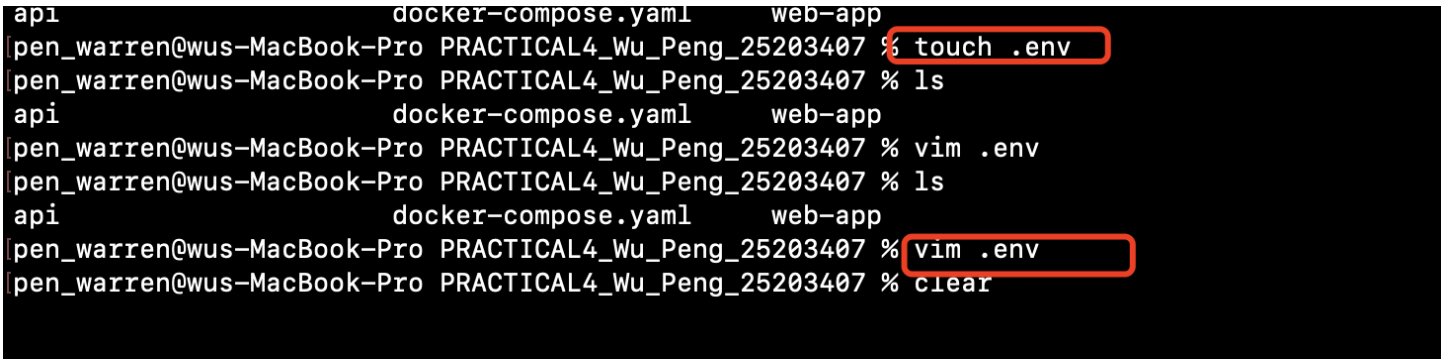
4.1 Prepare Your Project Structure

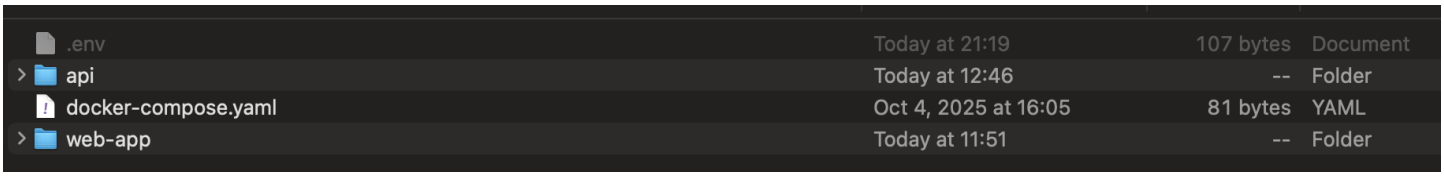
directory structure like this:



4.2 Create an `.env` file

In `project_root/.env`:





4.3 Create `docker-compose.yaml`

In `project_root/docker-compose.yaml`:

Code block

```
services:
  # ======================================
  # Database Service (PostgreSQL)
  # ======================================
  database:
    image: postgres:latest
    container_name: database
    environment:
```

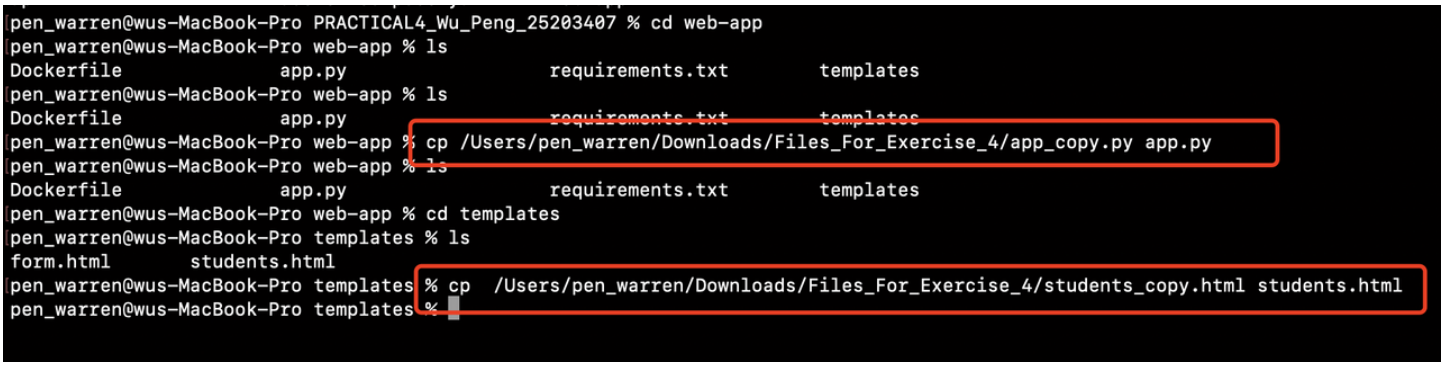
```yaml
      # Read environment variables from .env file
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    volumes:
      # Persist database data to a named volume
      - postgres_data:/var/lib/postgresql/data
    networks:
      # Connect only to the backend network
      - backend
    healthcheck:
      # Health check to ensure the database is fully started
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER}"]
      interval: 10s
      timeout: 5s
      retries: 5

  # ========================================
  # Adminer Service (Database Management UI)
  # ========================================
  adminer:
    image: adminer:latest
    container_name: adminer
    ports:
      # Map to host port 8091
      - "8091:8080"
    networks:
      # Connect only to the backend network
      - backend
    depends_on:
      # Wait until the database service is healthy before starting
      database:
        condition: service_healthy

  # ========================================
  # API Service (FastAPI)
  # ========================================
  api:
    build:
      context: ./api
      dockerfile: Dockerfile
    container_name: api
    environment:
      # Database connection string
      DATABASE_URL:
postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@database:5432/${POSTGRES_DB}
      POSTGRES_DB: ${POSTGRES_DB}
```

```yaml
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    volumes:
      # Mount code directory for hot reloading (no need to restart container after code changes)
      - ./api:/app
    networks:
      # Connect to both backend (access the database) and frontend (accessed by web-app)
      - backend
      - frontend
    depends_on:
      # Wait until the database service is healthy before starting
      database:
        condition: service_healthy
    # Expose internal port (not mapped to host)
    expose:
      - "8080"

  # ========================================
  # Web-App Service (Flask)
  # ========================================
  web-app:
    build:
      context: ./web-app
      dockerfile: Dockerfile
    container_name: web-app
    ports:
      # Map to host port 8090
      - "8090:5000"
    volumes:
      # Mount code directory for hot reloading (no need to restart container after code changes)
      - ./web-app:/app
    networks:
      # Connect only to the frontend network
      - frontend
    depends_on:
      # Wait for database and API to start before running
      - database
      - api
    environment:
      # API service address (use container name)
      API_URL: http://api:8080

# ========================================
# Networks Definition
```

```
# ========================================
networks:
  backend:
    driver: bridge
    # backend network: database, adminer, api
  frontend:
    driver: bridge
    # frontend network: web-app, api

# ========================================
# Volumes Definition
# ========================================
volumes:
  postgres_data:
    driver: local
    # Persistent storage for PostgreSQL data

```

4.4 Copy files

Copy the contents of app_copy.py into app.py and student_copy.html to student.html:



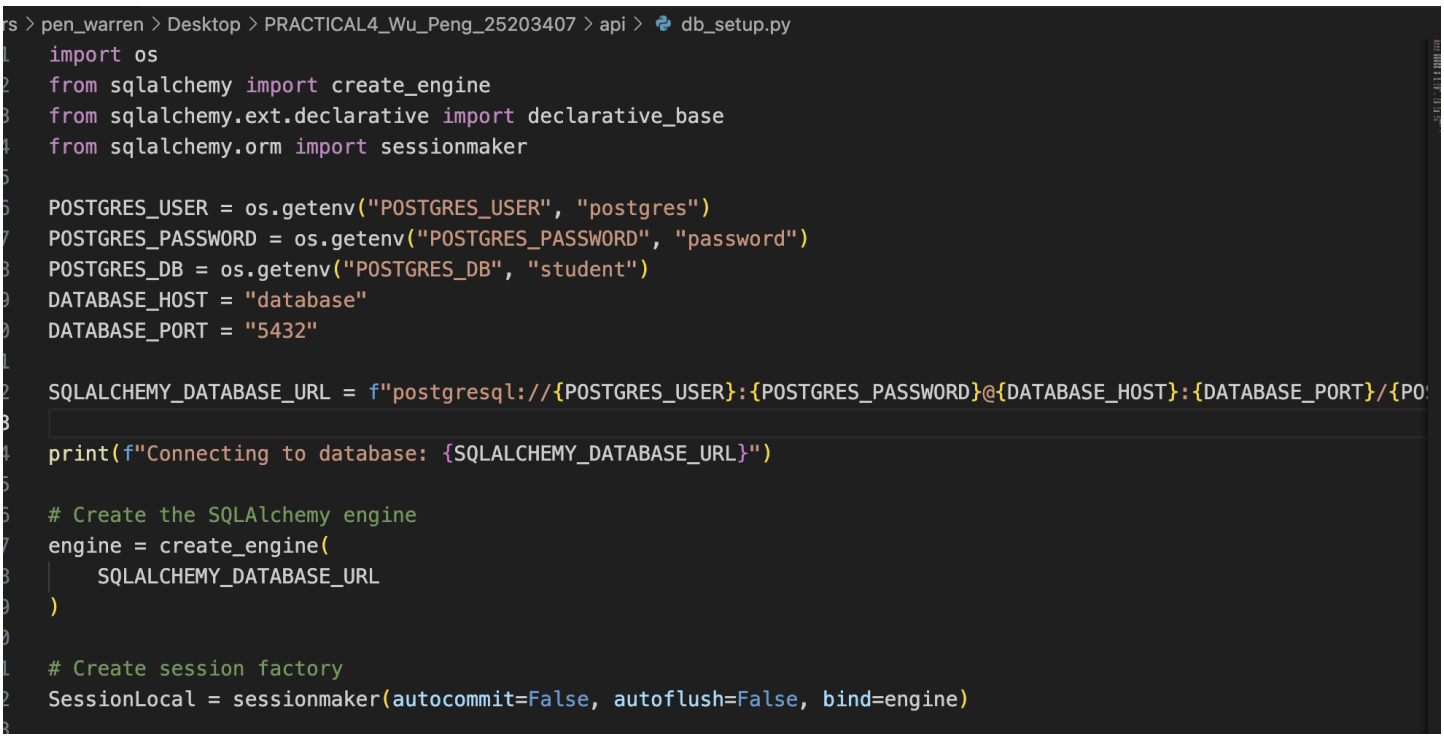
4.5 Update /api/db_setup.py

Update /api/db_setup.py to reflect the database env variable set above.

Code block

```
import os
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

POSTGRES_USER = os.getenv("POSTGRES_USER", "postgres")
```

```python
POSTGRES_PASSWORD = os.getenv("POSTGRES_PASSWORD", "password")
POSTGRES_DB = os.getenv("POSTGRES_DB", "student")
DATABASE_HOST = "database"
DATABASE_PORT = "5432"

SQLALCHEMY_DATABASE_URL = f"postgresql://{POSTGRES_USER}:
{POSTGRES_PASSWORD}@{DATABASE_HOST}:{DATABASE_PORT}/{POSTGRES_DB}"

print(f"Connecting to database: {SQLALCHEMY_DATABASE_URL}")

# Create the SQLAlchemy engine
engine = create_engine(
    SQLALCHEMY_DATABASE_URL
)

# Create session factory
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

# Create base class for models
Base = declarative_base()

# Database dependency function
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```



4.6 Build and Start Containers

Clear the old containers:

```
is-culster-redis-7006-1
pen_warren@wus-MacBook-Pro templates % docker adminer
docker: unknown command: docker adminer

Run 'docker --help' for more information
pen_warren@wus-MacBook-Pro templates % docker stop adminer
adminer
pen_warren@wus-MacBook-Pro templates % docker rm adminer
adminer
pen_warren@wus-MacBook-Pro templates % clear
```

```
ulster-redis-7006-1
pen_warren@wus-MacBook-Pro templates % docker stop fastapi-app
fastapi-app
pen_warren@wus-MacBook-Pro templates % docker rm fastapi-app
fastapi-app
pen_warren@wus-MacBook-Pro templates %
```

```
pen_warren@wus-MacBook-Pro templates % docker stop postgres
Error response from daemon: No such container: postgres
pen_warren@wus-MacBook-Pro templates % docker stop databas
Error response from daemon: No such container: databas
pen_warren@wus-MacBook-Pro templates % docker stop database
database
pen_warren@wus-MacBook-Pro templates % docker rm database
database
pen_warren@wus-MacBook-Pro templates % docker stop web_app_container
web_app_container
pen_warren@wus-MacBook-Pro templates % docker rm web_app_container
```

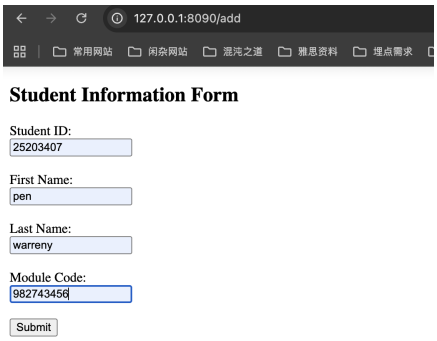
4.7 Build and Start Containers

From `project_root` run:

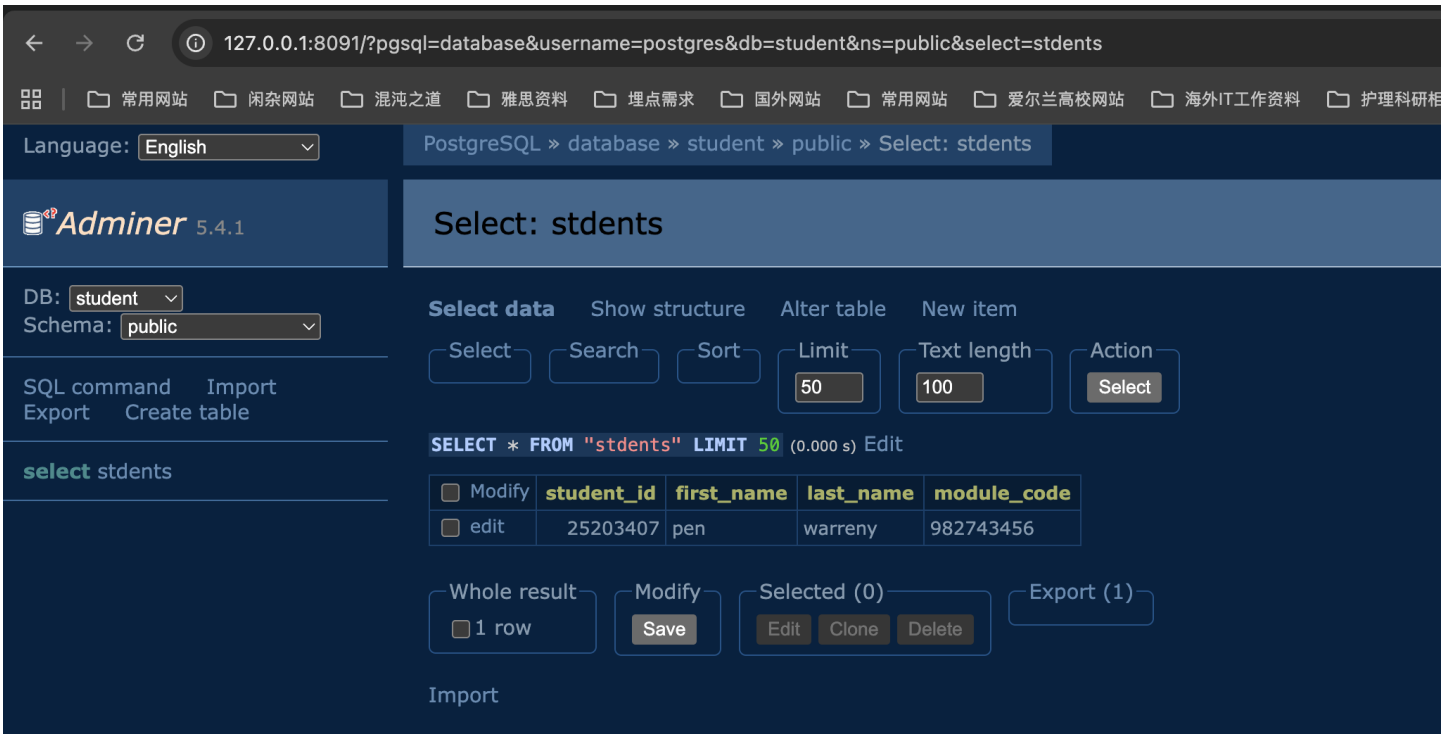
```
pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % docker-compose up -d --build
[+] Building 14.6s (18/18) FINISHED
 => [internal] load local bake definitions
 => => reading from stdin 1.13kB
 => [api internal] load build definition from Dockerfile
 => => transferring dockerfile: 429B
 => [web-app internal] load build definition from Dockerfile
 => => transferring dockerfile: 436B
 => [api internal] load metadata for docker.io/library/python:3.9-slim
 => [api internal] load .dockerignore
 => => transferring context: 2B
```

4.8 Access Your Services

- **Web app**: http://127.0.0.1:8090/add and http://127.0.0.1:8090/all



- **Adminer (DB UI)**: http://127.0.0.1:8091


127.0.0.1:8091/?pgsql=database&username=postgres&db=student&ns=public&select=stdents
常用网站 闲杂网站 混沌之道 雅思资料 埋点需求 国外网站 常用网站 爱尔兰高校网站 海外IT工作资料 护理科研相
Language: English
PostgreSQL » database » student » public » Select: stdents
Adminer 5.4.1
Select: stdents
DB: student
Schema: public
SQL command Import
Export Create table
select stdents
Select data Show structure Alter table New item
Select Search Sort Limit 50 Text length 100 Action Select
SELECT * FROM "stdents" LIMIT 50 (0.000 s) Edit
Modify student_id first_name last_name module_code
edit 25203407 pen warreny 982743456
Whole result 1 row
Modify Save
Selected (0) Edit Clone Delete
Export (1)
Import