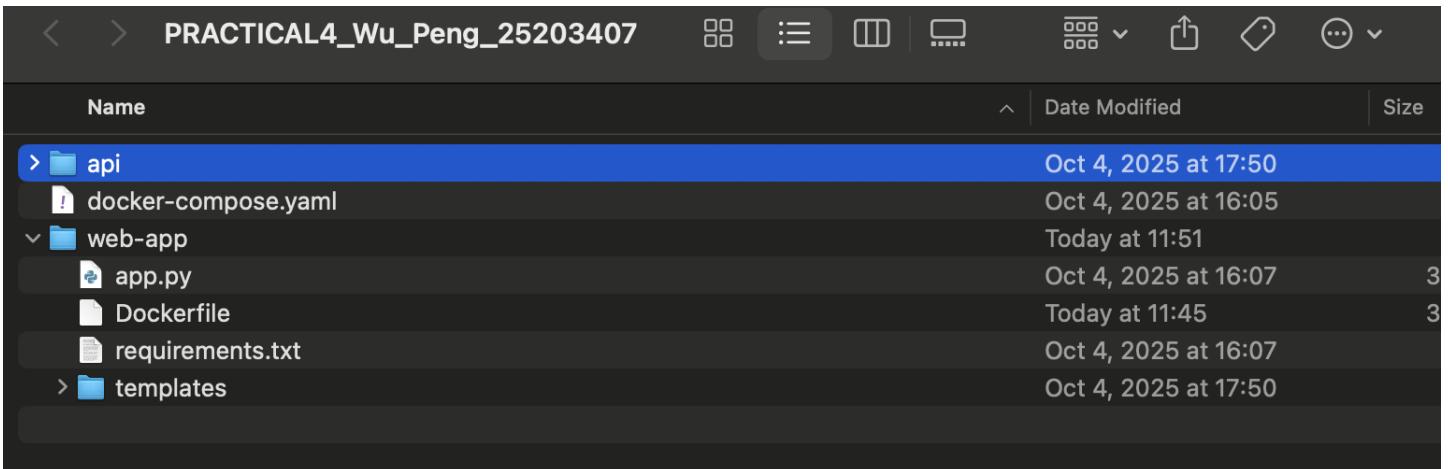


Lab4

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**:

```

pen_warren@wus-MacBook-Pro web-app % docker build -t web_app_image .

[+] Building 15.5s (9/9) FINISHED
--> [internal] load build definition from Dockerfile
--> => transferring dockerfile: 436B
--> [internal] load metadata for docker.io/library/python:3.9-slim
--> [internal] load .dockerignore
--> => transferring context: 2B
--> [1/4] FROM docker.io/library/python:3.9-slim@sha256:69ad475ad27439666dca95aa73cf21b5b5249241e1e0544acf488d5181d91496
--> => resolve docker.io/library/python:3.9-slim@sha256:69ad475ad27439666dca95aa73cf21b5b5249241e1e0544acf488d5181d91496
--> => sha256:a16e51192670581ec8359c70ab9eebf8f2af5468ff79b3ad4f9ce21b366f47 30.14MB / 30.14MB
--> => sha256:a9589d33c90e6245c66601264da8aae995538c5ef979af12cb0111cfb3b4ae37 13.82MB / 13.82MB
--> => sha256:c0f8ad3620a9a9591dc84459475fe458a9d8ef84830017ce792a98e2b854de10e 1.27MB / 1.27MB
--> => sha256:7c9f8bae26b9f2099a18e802900c9c0460a5a613858973cb2a7a975a88d30f97 250B / 250B
--> => extracting sha256:a16e51192670581ec8359c70ab9eebf8f2af5468ff79b3ad4f9ce21b366f47
--> => extracting sha256:9f0ad3620a9a9591dc84459475fe458a9d8ef84830017ce702a98e2b854de10e
--> => extracting sha256:c0f8ad3620a9a9591dc84459475fe458a9d8ef84830017ce702a98e2b854de10e
--> => extracting sha256:7c9f8bae26b9f2099a18e802900c9c0460a5a613858973cb2a7a975a88d30f97
--> [internal] load build context
--> => transferring context: 8.37kB
--> [2/4] WORKDIR /app
--> [3/4] COPY . /app
--> [4/4] RUN pip install --no-cache-dir -r requirements.txt
--> exporting to image
--> => exporting layers
--> => exporting manifest sha256:f62a761cdbd37afc97f65b74a437e225e7666cdc6ea1eb675714bc3cf0b500ea
--> => exporting config sha256:61fb1176379e081cda6ad928f6b33f7bc0bdbc9a1db3b2e5ee94a1d3d7eca810
--> => exporting attestation manifest sha256:12197822acc65abb480227eecf1390217416ae266a91ba13dbaeb09a6c9b7d4e
--> => exporting manifest list sha256:8b27940cf41f466bc1881897d7560f781cecc2128fc4ba3027e9ecabd213fdc
--> => naming to docker.io/library/web_app_image:latest
--> => unpacking to docker.io/library/web_app_image:latest

```

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/lrz8351fd57x93al6nbn9x6yp
pen_warren@wus-MacBook-Pro web-app %

This creates a Docker image called `web_app_image`.

1.4 Run the Container

Now run the container and map ports:

```

Dockerfile          app.py           requirements.txt      templates
pen_warren@wus-MacBook-Pro web-app % docker run -d -p 8090:5000 --name web_app_container web_app_image

```

```

3f9d5a1c1cd752fec67a1a354addb1615e44300b3354494b27687daf611e06cf
pen_warren@wus-MacBook-Pro web-app %

```

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:

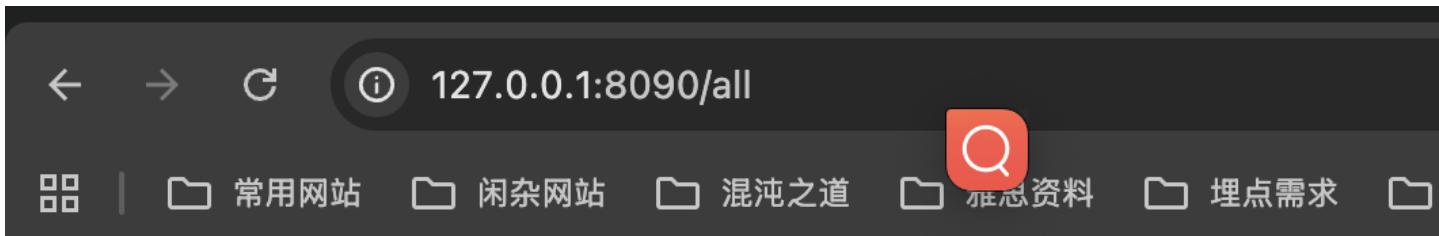
First Name:

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

Name	Date Modified
api	Oct 4, 2025 at 17:50
db_setup.py	Oct 4, 2025 at 16:19
main.py	Oct 4, 2025 at 16:19
model.py	Oct 4, 2025 at 16:19
requirements.txt	Oct 4, 2025 at 16:19
schema.py	Oct 4, 2025 at 16:19
docker-compose.yaml	Oct 4, 2025 at 16:05
web-app	Today at 11:51
app.py	Oct 4, 2025 at 16:07
Dockerfile	Today at 11:45
requirements.txt	Oct 4, 2025 at 16:07
templates	Oct 4, 2025 at 17:50

2.2 Create the Dockerfile inside the api/ folder

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

```
pen_warren@wus-MacBook-Pro api % touch Dockerfile
pen_warren@wus-MacBook-Pro api % ls
Dockerfile      db_setup.py      main.py      model.py      requirements.txt      schema.py
pen_warren@wus-MacBook-Pro api %
```

Write into Dockerfile:

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

# Set working directory inside the container
WORKDIR /app

# Copy all API source code into the container
COPY . /app

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

# Expose port 8080 for the API
EXPOSE 8080

# Run FastAPI using uvicorn
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8080"]
pen_warren@wus-MacBook-Pro api %
```

2.3 Start a temporary PostgreSQL database container

Run this command:

```
pen_warren@wus-MacBook-Pro api % docker run --name database \
-p 5432:5432 \
-e POSTGRES_DB=pengwu \
-e POSTGRES_USER=postgres \
-e POSTGRES_PASSWORD=password \
-d postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
07fa2047ff72: Pull complete
e8e05a3f3015: Pull complete
230a268a3aa2: Pull complete
f2001f898dbd: Pull complete
ffafe204151f: Pull complete
3154db8ba0f5: Pull complete
54c980ef1171: Pull complete
be0ec8bb0d7b: Pull complete
1e2c02175881: Pull complete
39a04ffb5fc0: Pull complete
0129191ba0ae: Pull complete
765e3554df4d: Pull complete
8b1b2d350139: Pull complete
Digest: sha256:b5b154cf5bfed10a2e68f9d064ab76cebf28b4c80bedf714e6b500f839fbe9d
Status: Downloaded newer image for postgres:latest
52b6cbcc1518f28ce5ad3bd4404b7941ac5e4cf70e4e5c11f88546f96a8210e8
pen_warren@wus-MacBook-Pro api %
```

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  
1  from sqlalchemy import create_engine  
2  from sqlalchemy.ext.declarative import declarative_base  
3  from sqlalchemy.orm import sessionmaker  
4  
5  # SQLAlchemy settings for SQLite database  
6  SQLALCHEMY_DATABASE_URL = "postgresql://postgres:password@localhost:5432/pengwu"  
7  
8  # Create the SQLAlchemy engine  
9  engine = create_engine(  
10     SQLALCHEMY_DATABASE_URL  
11 )  
12  
13 SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)  
14 Base = declarative_base()  
15
```

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": {}
  }
]

```

1

Check if it was created successfully

```

[pen_warren@wus-MacBook-Pro api % docker network connect app-network database
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
bf2fffdc6da25	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:

```
database
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
```

```
2a25b160264d10f27a947b32108e63dcfd671efc39877382810d1c2b2110033
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
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
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 429B
=> [internal] load metadata for docker.io/library/python:3.9-slim
=> [internal] load .dockerrignore
=> transferring context: 2B
=> [1/4] FROM docker.io/library/python:3.9-slim@sha256:69ad475ad27439666dca95aa73cf21b5b5249241e1e0544acf488d5181d91496
=> => resolve docker.io/library/python:3.9-slim@sha256:69ad475ad27439666dca95aa73cf21b5b5249241e1e0544acf488d5181d91496
=> [internal] load build context
=> transferring context: 9.07kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . /app
=> [4/4] RUN pip install --no-cache-dir -r requirements.txt
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:ba2ae4a896a37d997f6f02e8e283c6d1b4dfee5c7987c4b1a4eaa69f6ae2c78d
=> => exporting config sha256:e3d46d34e0e5c47813a6f8deaa7c6b34e0f7e57aa82386b5b68640b7a7352a4a
=> => exporting attestation manifest sha256:bfs8d5d8ea3bf109cd47dd6a94d5fb8d495fa114640e6c8c56e9282662772607
=> => exporting manifest list sha256:c9899a497fc7769e3c99b0963c8080a068658094c75d60a619c0a2a5d550af6
=> => naming to docker.io/library/fastapi-app:latest
=> => unpacking to docker.io/library/fastapi-app:latest
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/zprl18dmywz0xjy1nvjcwlef
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
```

```
e976a5997e2f244b737803eb6bba96c462fcbe50dc6636c243defab6346969b
pen_warren@wus-MacBook-Pro api %
```

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
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": {
      "045aaafa1b29df556fb73d4cdf9107b219a1004c4660a3833050153316dcb350e": {
        "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>

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

3. Exercise Three - Database service

3.1 Run PostgreSQL container

```
-custer-redis-7000-1
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 "97989f5ccced0a64214a101d7877e53cf33d5d3e40ab77c0648e36fdc78f1aa8". 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

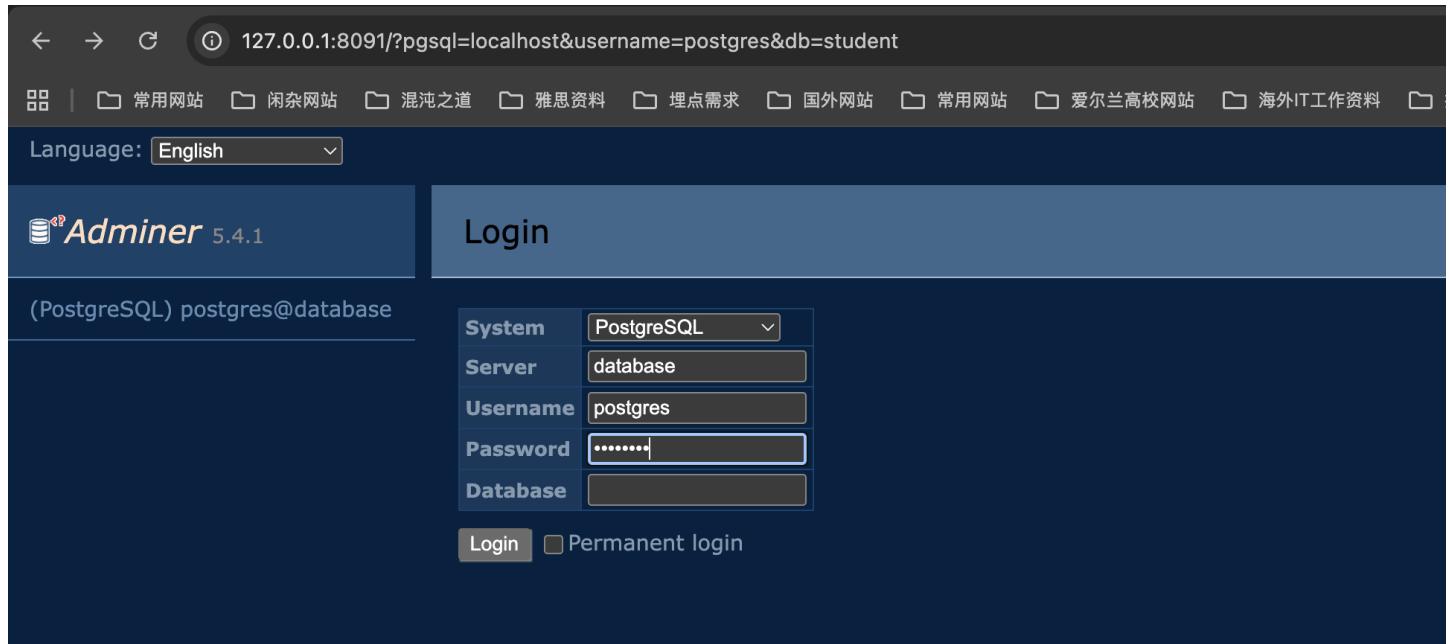
3.2 Run Adminer container

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:

The screenshot shows the Adminer 5.4.1 schema browser for the public schema of the database pengwu. The URL in the address bar is 127.0.0.1:8091/?pgsql=database&username=postgres&db=pengwu&ns=public. The language is set to English. The main page title is Schema: public. The top navigation bar includes links for Alter schema, Database schema, Routines, Sequences, and User types. Below the navigation is a search bar for tables. A table lists one table named stdents with details: Engine: table, Collation: en_US.utf8, Data Length: 0, Index Length: 24,576, Data Free: ?, Auto Increment: ?, Rows: 0, and Comment: ?. At the bottom, there are buttons for Selected (0) (Vacuum, Optimize, Truncate, Drop) and Move to other database (0) (public, Move).

4. Exercise Four - Docker compose

4.1 Prepare Your Project Structure

directory structure like this:

Name	Date Modified	Size	Kind
api	Today at 12:46	--	Folder
docker-compose.yaml	Oct 4, 2025 at 16:05	81 bytes	YAML
web-app	Today at 11:51	--	Folder

4.2 Create an `.env` file

In `project_root/.env`:

```
api           docker-compose.yaml    web-app
[pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % touch .env]
[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 % vim .env
[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 % vim .env
[pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % clear
```

.env	Today at 21:19	107 bytes	Document
api	Today at 12:46	--	Folder
docker-compose.yaml	Oct 4, 2025 at 16:05	81 bytes	YAML
web-app	Today at 11:51	--	Folder

4.3 Create `docker-compose.yaml`

In `project_root/docker-compose.yaml`:

Code block

```
1 services:
2   # =====
3   # Database Service (PostgreSQL)
4   # =====
5   database:
6     image: postgres:latest
7     container_name: database
8     environment:
```

```

9      # Read environment variables from .env file
10     POSTGRES_DB: ${POSTGRES_DB}
11     POSTGRES_USER: ${POSTGRES_USER}
12     POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
13   volumes:
14     # Persist database data to a named volume
15     - postgres_data:/var/lib/postgresql/data
16   networks:
17     # Connect only to the backend network
18     - backend
19   healthcheck:
20     # Health check to ensure the database is fully started
21     test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER}"]
22     interval: 10s
23     timeout: 5s
24     retries: 5
25
26   # =====
27   # Adminer Service (Database Management UI)
28   # =====
29   adminer:
30     image: adminer:latest
31     container_name: adminer
32     ports:
33       # Map to host port 8091
34       - "8091:8080"
35     networks:
36       # Connect only to the backend network
37       - backend
38     depends_on:
39       # Wait until the database service is healthy before starting
40       database:
41         condition: service_healthy
42
43   # =====
44   # API Service (FastAPI)
45   # =====
46   api:
47     build:
48       context: ./api
49       dockerfile: Dockerfile
50     container_name: api
51     environment:
52       # Database connection string
53       DATABASE_URL:
54         postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@database:5432/${POSTGRES_DB}
55         POSTGRES_DB: ${POSTGRES_DB}

```

```

55      POSTGRES_USER: ${POSTGRES_USER}
56      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
57  volumes:
58      # Mount code directory for hot reloading (no need to restart container
59      # after code changes)
60          - ./api:/app
61  networks:
62      # Connect to both backend (access the database) and frontend (accessed
63      # by web-app)
64          - backend
65          - frontend
66  depends_on:
67      # Wait until the database service is healthy before starting
68      database:
69          condition: service_healthy
70  # Expose internal port (not mapped to host)
71  expose:
72      - "8080"
73
74      # =====
75      # Web-App Service (Flask)
76      # =====
77  web-app:
78      build:
79          context: ./web-app
80          dockerfile: Dockerfile
81          container_name: web-app
82      ports:
83          # Map to host port 8090
84          - "8090:5000"
85      volumes:
86          # Mount code directory for hot reloading (no need to restart container
87          # after code changes)
88          - ./web-app:/app
89  networks:
90      # Connect only to the frontend network
91          - frontend
92  depends_on:
93      # Wait for database and API to start before running
94          - database
95          - api
96  environment:
97      # API service address (use container name)
98      API_URL: http://api:8080
99
100     # =====
101     # Networks Definition

```

```

99  # =====
100 networks:
101   backend:
102     driver: bridge
103     # backend network: database, adminer, api
104   frontend:
105     driver: bridge
106     # frontend network: web-app, api
107
108 # =====
109 # Volumes Definition
110 # =====
111 volumes:
112   postgres_data:
113     driver: local
114     # Persistent storage for PostgreSQL data
115

```

4.4 Copy files

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

```

[pen_warren@wus-MacBook-Pro PRACTICAL4_Wu_Peng_25203407 % cd web-app
[pen_warren@wus-MacBook-Pro web-app % ls
Dockerfile          app.py           requirements.txt      templates
[pen_warren@wus-MacBook-Pro web-app % ls
Dockerfile          app.py           requirements.txt      templates
[pen_warren@wus-MacBook-Pro web-app % cp /Users/pen_warren/Downloads/Files_For_Exercise_4/app_copy.py app.py
[pen_warren@wus-MacBook-Pro web-app %
Dockerfile          app.py           requirements.txt      templates
[pen_warren@wus-MacBook-Pro web-app % cd templates
[pen_warren@wus-MacBook-Pro templates % ls
form.html        students.html
[pen_warren@wus-MacBook-Pro templates % cp /Users/pen_warren/Downloads/Files_For_Exercise_4/students_copy.html students.html
[pen_warren@wus-MacBook-Pro templates %

```

4.5 Update /api/db_setup.py

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

Code block

```

1 import os
2 from sqlalchemy import create_engine
3 from sqlalchemy.ext.declarative import declarative_base
4 from sqlalchemy.orm import sessionmaker
5
6 POSTGRES_USER = os.getenv("POSTGRES_USER", "postgres")

```

```

7 POSTGRES_PASSWORD = os.getenv("POSTGRES_PASSWORD", "password")
8 POSTGRES_DB = os.getenv("POSTGRES_DB", "student")
9 DATABASE_HOST = "database"
10 DATABASE_PORT = "5432"
11
12 SQLALCHEMY_DATABASE_URL = f"postgresql://{{POSTGRES_USER}}:
13 {{POSTGRES_PASSWORD}}@{{DATABASE_HOST}}:{{DATABASE_PORT}}/{{POSTGRES_DB}}"
14 print(f"Connecting to database: {{SQLALCHEMY_DATABASE_URL}}")
15
16 # Create the SQLAlchemy engine
17 engine = create_engine(
18     SQLALCHEMY_DATABASE_URL
19 )
20
21 # Create session factory
22 SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
23
24 # Create base class for models
25 Base = declarative_base()
26
27 # Database dependency function
28 def get_db():
29     db = SessionLocal()
30     try:
31         yield db
32     finally:
33         db.close()

```

```

rs > pen_warren > Desktop > PRACTICAL4_Wu_Peng_25203407 > api > db_setup.py
1 import os
2 from sqlalchemy import create_engine
3 from sqlalchemy.ext.declarative import declarative_base
4 from sqlalchemy.orm import sessionmaker
5
6 POSTGRES_USER = os.getenv("POSTGRES_USER", "postgres")
7 POSTGRES_PASSWORD = os.getenv("POSTGRES_PASSWORD", "password")
8 POSTGRES_DB = os.getenv("POSTGRES_DB", "student")
9 DATABASE_HOST = "database"
10 DATABASE_PORT = "5432"
11
12 SQLALCHEMY_DATABASE_URL = f"postgresql://{{POSTGRES_USER}}:{{POSTGRES_PASSWORD}}@{{DATABASE_HOST}}:{{DATABASE_PORT}}/{{PO:
13
14 print(f"Connecting to database: {{SQLALCHEMY_DATABASE_URL}}")
15
16 # Create the SQLAlchemy engine
17 engine = create_engine(
18     SQLALCHEMY_DATABASE_URL
19 )
20
21 # Create session factory
22 SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

```

4.6 Build and Start Containers

Clear the old containers:

```
is-culster-redis-7006-1
[pen_warren@wus-MacBook-Pro templates % docker adminer
adminer: 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 database
Error response from daemon: No such container: database
[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) FTNTSHFD
=> [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>

Student Information Form

Student ID	First Name	Last Name	Module Code
25203407	pen	warreny	982743456

- Adminer (DB UI): <http://127.0.0.1:8091>

Adminer 5.4.1

Login

System	PostgreSQL
Server	database
Username	postgres
Password
Database	student

Login Permanent login

← → ⌂ 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

DB: student Schema: public

SQL command Import Export Create table

select stdents

Select data Show structure Alter table New item

Select Search Sort Limit Text length Action

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 Modify Selected (0) Export (1)

1 row Save Edit Clone Delete

Import