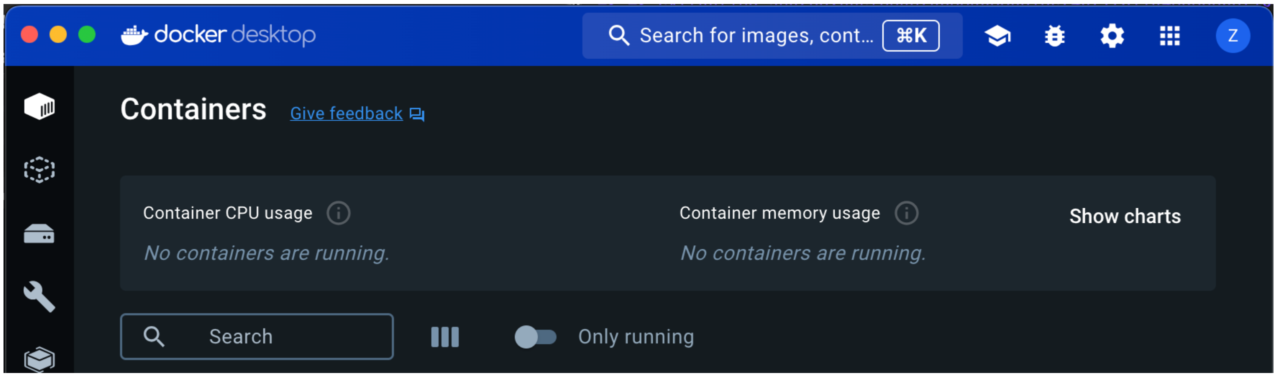
19953 Zhiyu Zhang

CS571: Week 11: Homework 3: GenAI – Develop your containerized app

1. Prerequisites

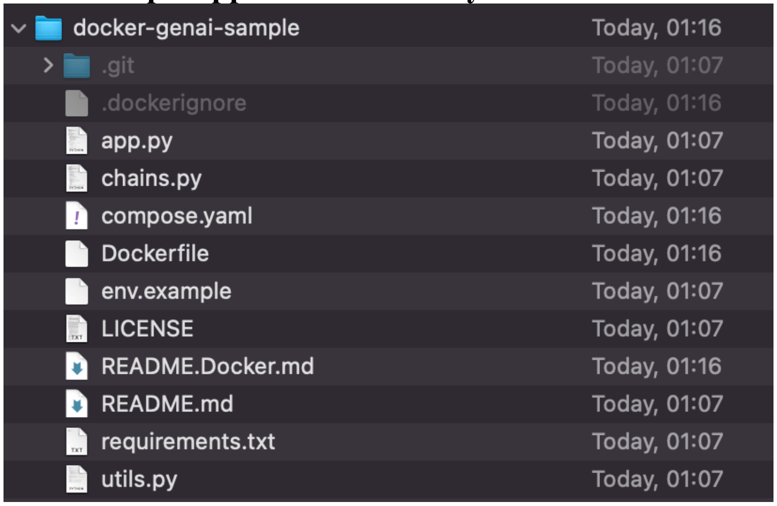
Complete Containerize your app.

Make sure your Docker Desktop is running:



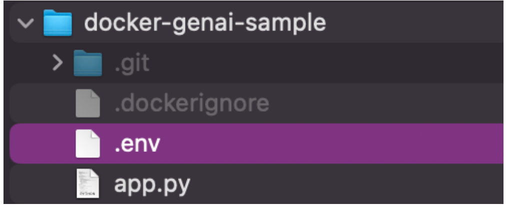
2. Add a local database

2-1 Open the sample application directory



docker-genai-sample	Today, 01:16
.git	Today, 01:07
.dockerignore	Today, 01:16
app.py	Today, 01:07
chains.py	Today, 01:07
compose.yaml	Today, 01:16
Dockerfile	Today, 01:16
env.example	Today, 01:07
LICENSE	Today, 01:07
README.Docker.md	Today, 01:16
README.md	Today, 01:07
requirements.txt	Today, 01:07
utils.py	Today, 01:07

2-2 Rename "env.example" to ".env"



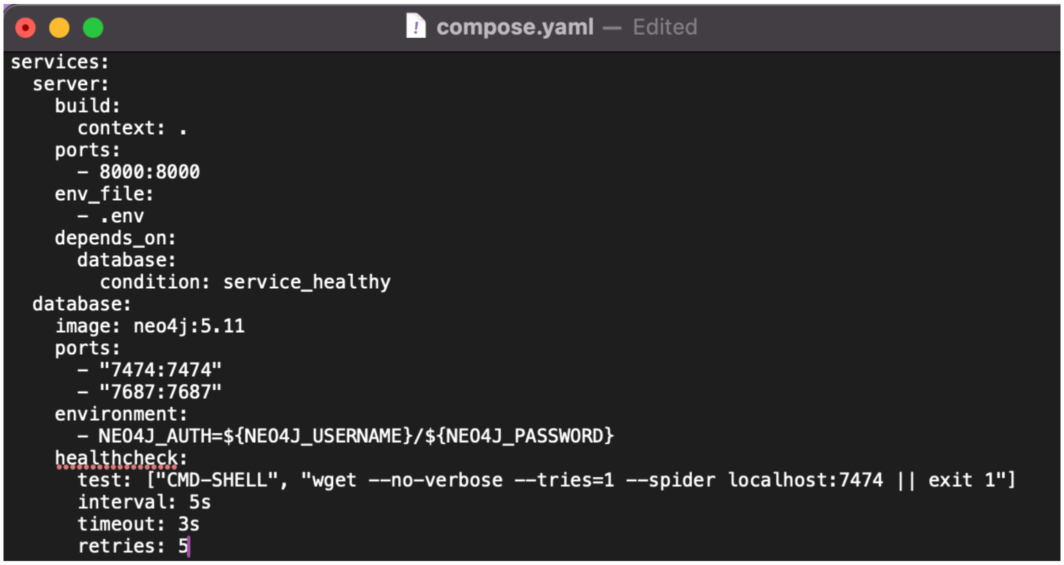
docker-genai-sample
.git
.dockerignore
.env
app.py

This file contains the environment variables that the containers will use.

2-3 Edit "compose.yaml"

Add instructions to run a Neo4j database, and specify the environment file under the server service in order to pass in the environment variables for the connection. Following is the updated compose.yaml, with all comments removed:

```
services:
  server:
    build:
      context: .
    ports:
      - 8000:8000
    env_file:
      - .env
    depends_on:
      database:
        condition: service_healthy
  database:
    image: neo4j:5.11
    ports:
      - "7474:7474"
      - "7687:7687"
    environment:
      - NEO4J_AUTH=${NEO4J_USERNAME}/${NEO4J_PASSWORD}
    healthcheck:
      test: ["CMD-SHELL", "wget --no-verbose --tries=1 --spider localhost:7474 ||
exit 1"]
      interval: 5s
      timeout: 3s
      retries: 5
```



```
services:
  server:
    build:
      context: .
    ports:
      - 8000:8000
    env_file:
      - .env
    depends_on:
      database:
        condition: service_healthy
  database:
    image: neo4j:5.11
    ports:
      - "7474:7474"
      - "7687:7687"
    environment:
      - NEO4J_AUTH=${NEO4J_USERNAME}/${NEO4J_PASSWORD}
    healthcheck:
      test: ["CMD-SHELL", "wget --no-verbose --tries=1 --spider localhost:7474 || exit 1"]
      interval: 5s
      timeout: 3s
      retries: 5
```

2-4 Run the application

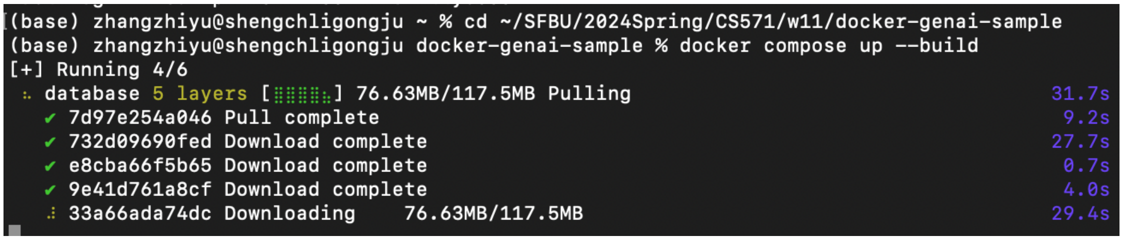
In terminal, go to the sample application directory:

```
% cd ~/SFBU/2024Spring/CS571/w11/docker-genai-sample
```

Build the application:

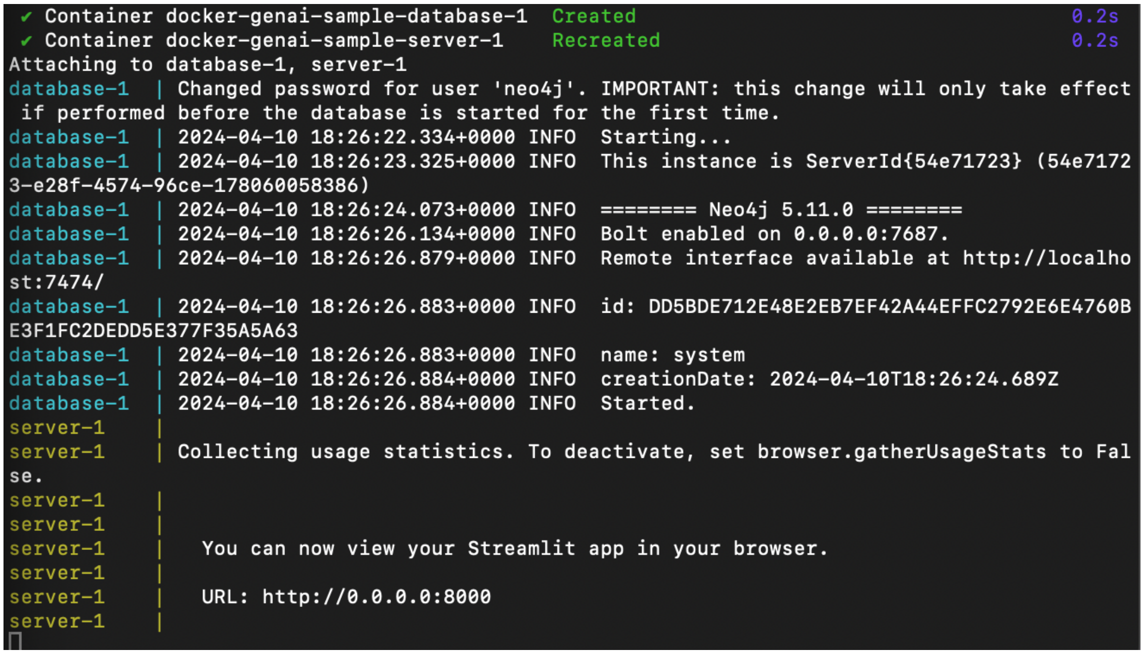
```
% docker compose up --build
```

You will see something starting like this while Docker is building the application:



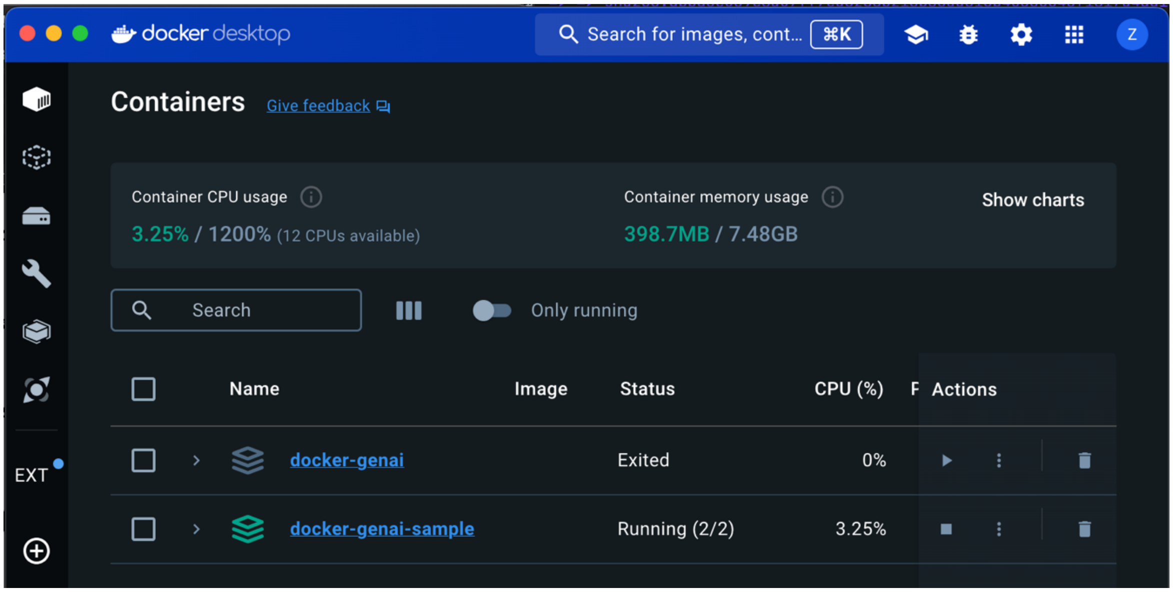
```
(base) zhangzhiyu@shengchligongju ~ % cd ~/SFBU/2024Spring/CS571/w11/docker-genai-sample
(base) zhangzhiyu@shengchligongju docker-genai-sample % docker compose up --build
[+] Running 4/6
 ⠼ database 5 layers [⣿⣿⣿⣿⣀] 76.63MB/117.5MB Pulling                        31.7s
   ✔ 7d97e254a046 Pull complete                                                9.2s
   ✔ 732d09690fed Download complete                                           27.7s
   ✔ e8cba66f5b65 Download complete                                            0.7s
   ✔ 9e41d761a8cf Download complete                                            4.0s
   ⠼ 33a66ada74dc Downloading    76.63MB/117.5MB                              29.4s
```

You'll see something like the following when the application is running:



```
 ✔ Container docker-genai-sample-database-1  Created                                    0.2s
 ✔ Container docker-genai-sample-server-1    Recreated                                  0.2s
Attaching to database-1, server-1
database-1  | Changed password for user 'neo4j'. IMPORTANT: this change will only take effect
 if performed before the database is started for the first time.
database-1  | 2024-04-10 18:26:22.334+0000 INFO  Starting...
database-1  | 2024-04-10 18:26:23.325+0000 INFO  This instance is ServerId{54e71723} (54e7172
3-e28f-4574-96ce-178060058386)
database-1  | 2024-04-10 18:26:24.073+0000 INFO  ======== Neo4j 5.11.0 ========
database-1  | 2024-04-10 18:26:26.134+0000 INFO  Bolt enabled on 0.0.0.0:7687.
database-1  | 2024-04-10 18:26:26.879+0000 INFO  Remote interface available at http://localho
st:7474/
database-1  | 2024-04-10 18:26:26.883+0000 INFO  id: DD5BDE712E48E2EB7EF42A44EFFC2792E6E4760B
E3F1FC2DEDD5E377F35A5A63
database-1  | 2024-04-10 18:26:26.883+0000 INFO  name: system
database-1  | 2024-04-10 18:26:26.884+0000 INFO  creationDate: 2024-04-10T18:26:24.689Z
database-1  | 2024-04-10 18:26:26.884+0000 INFO  Started.
server-1    |
server-1    | Collecting usage statistics. To deactivate, set browser.gatherUsageStats to Fal
se.
server-1    |
server-1    |
server-1    |   You can now view your Streamlit app in your browser.
server-1    |
server-1    |   URL: http://0.0.0.0:8000
server-1    |
```

You can also see the container running in your Docker Desktop:



Containers

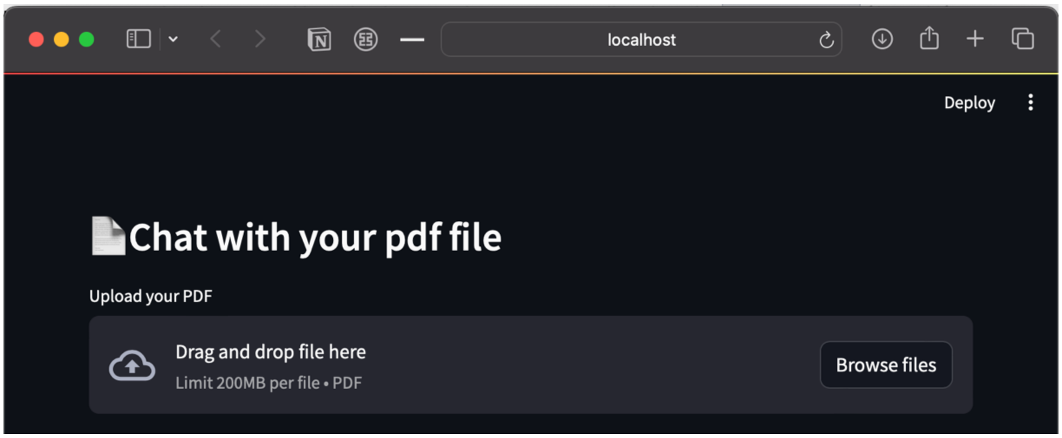
Container CPU usage 3.25% / 1200% (12 CPUs available)

Container memory usage 398.7MB / 7.48GB

□	Name	Image	Status	CPU (%)	F	Actions
□	docker-genai		Exited	0%		
□	docker-genai-sample		Running (2/2)	3.25%		

2-5 Access the application

Open a browser and view the application at http://localhost:8000:

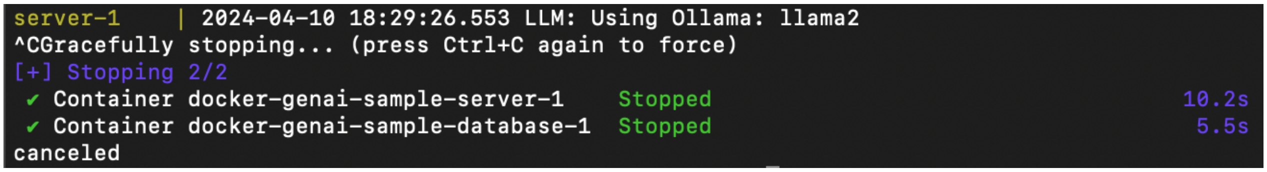


Chat with your pdf file

We can’t use the application yet because the LLM service specified in the .env file isn’t running yet.

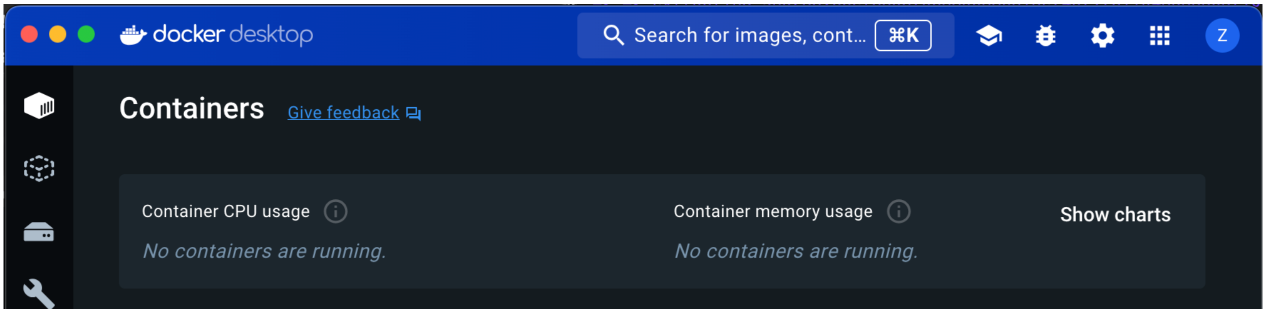
2-6 Stop the application

In terminal, press ctrl + C to stop the application:



```
server-1     | 2024-04-10 18:29:26.553 LLM: Using Ollama: llama2
^CGracefully stopping... (press Ctrl+C again to force)
[+] Stopping 2/2
 ✔ Container docker-genai-sample-server-1    Stopped                                 10.2s
 ✔ Container docker-genai-sample-database-1  Stopped                                  5.5s
canceled
```

You can also see the container stop running in Docker Desktop:



Containers Give feedback

Container CPU usage

No containers are running.

Container memory usage

No containers are running.

Show charts

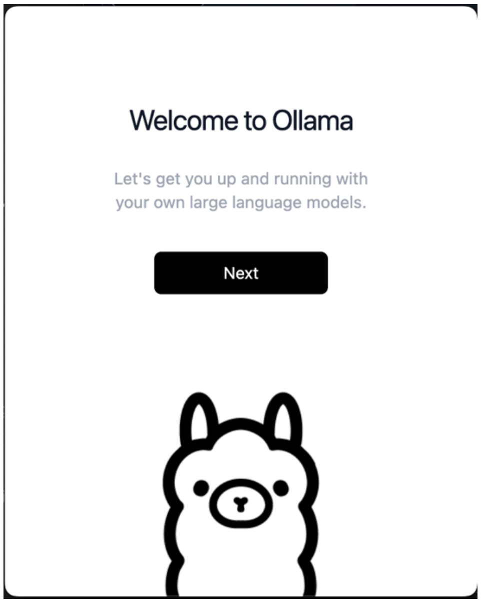
3. Add a local or remote LLM service

For this application, you can use Ollama or OpenAI as the LLM service. I use Ollama in this documentation. I'm using macOS environment, so I'll run Ollama outside of a container. Check more options for different environment here.

3-1 Install Ollama

You can download or manually install Ollama here.

Download Ollama for macOS and open the app, then simply follow its prompt to finish the setup:



Welcome to Ollama

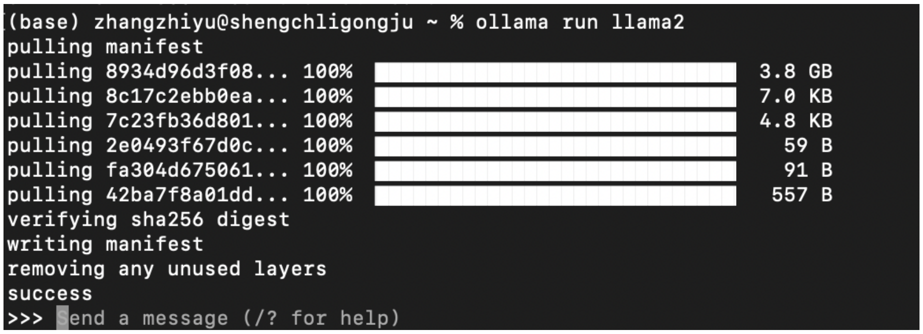
Let's get you up and running with your own large language models.

Then go to terminal, go to your main directory:

```
% cd ~/
```

Run the following command to finish setup and run Ollama, this will take a few minites:

```
% ollama run llama2
```

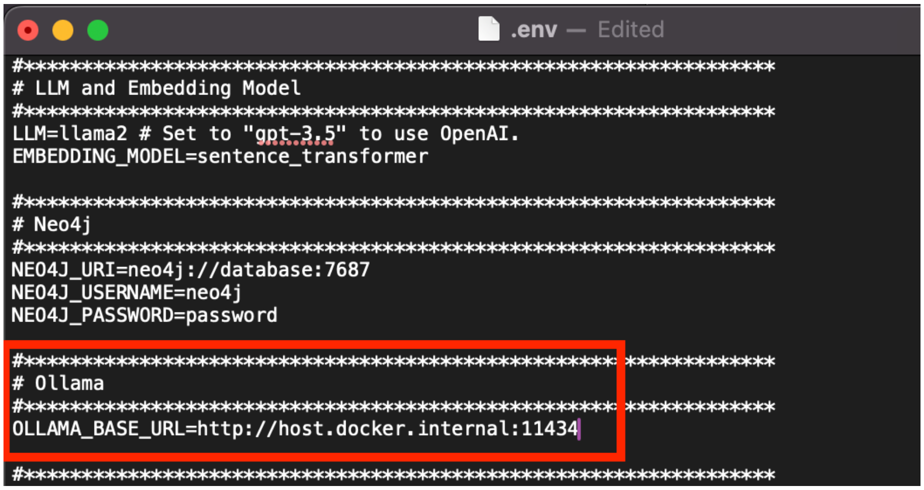


```
(base) zhangzhiyu@shengchligongju ~ % ollama run llama2
pulling manifest
pulling 8934d96d3f08... 100% ▕████████████████████████████▏ 3.8 GB
pulling 8c17c2ebb0ea... 100% ▕████████████████████████████▏ 7.0 KB
pulling 7c23fb36d801... 100% ▕████████████████████████████▏ 4.8 KB
pulling 2e0493f67d0c... 100% ▕████████████████████████████▏   59 B
pulling fa304d675061... 100% ▕████████████████████████████▏   91 B
pulling 42ba7f8a01dd... 100% ▕████████████████████████████▏  557 B
verifying sha256 digest
writing manifest
removing any unused layers
success
>>> Send a message (/? for help)
```

Use ctrl + D or /bye to exit.

3-2 Update ".env" in the application directory

Update the OLLAMA_BASE_URL value in your .env file to "http://host.docker.internal:11434":



```
#*****************************************************************
# LLM and Embedding Model
#*****************************************************************
LLM=llama2 # Set to "gpt-3.5" to use OpenAI.
EMBEDDING_MODEL=sentence_transformer

#*****************************************************************
# Neo4j
#*****************************************************************
NEO4J_URI=neo4j://database:7687
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=password

#*****************************************************************
# Ollama
#*****************************************************************
OLLAMA_BASE_URL=http://host.docker.internal:11434

#*****************************************************************
```

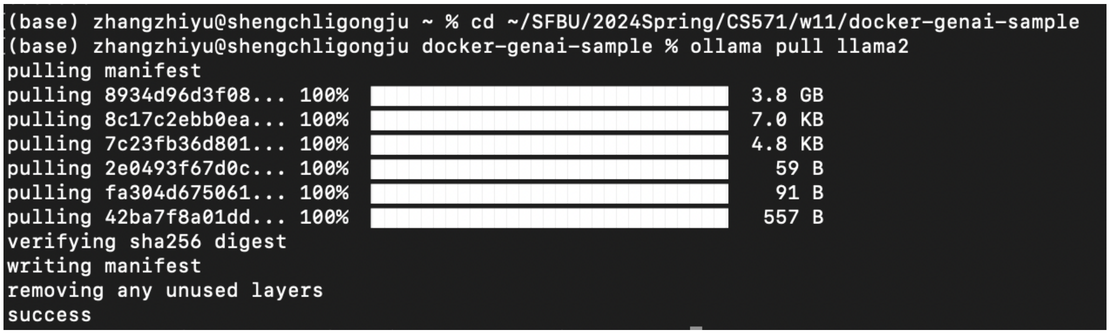
3-3 Back in terminal, pull the model to Ollama

Go back to the application directory in terminal:

```
% cd ~/SFBU/2024Spring/CS571/w11/docker-genai-sample
```

Then pull the model:

```
% ollama pull llama2
```



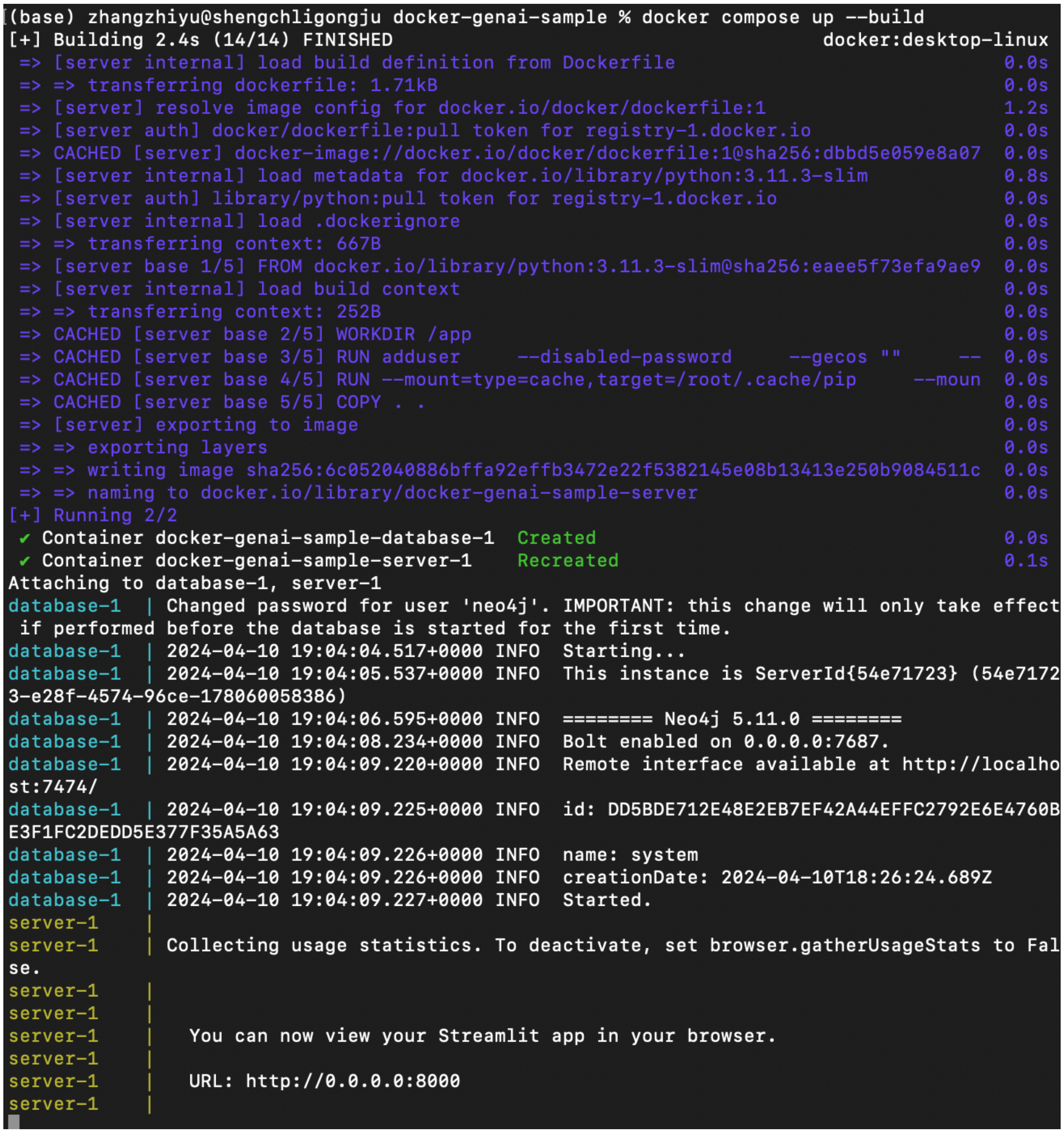
```
(base) zhangzhiyu@shengchligongju ~ % cd ~/SFBU/2024Spring/CS571/w11/docker-genai-sample
(base) zhangzhiyu@shengchligongju docker-genai-sample % ollama pull llama2
pulling manifest
pulling 8934d96d3f08... 100% ▕████████████████████████████▏ 3.8 GB
pulling 8c17c2ebb0ea... 100% ▕████████████████████████████▏ 7.0 KB
pulling 7c23fb36d801... 100% ▕████████████████████████████▏ 4.8 KB
pulling 2e0493f67d0c... 100% ▕████████████████████████████▏   59 B
pulling fa304d675061... 100% ▕████████████████████████████▏   91 B
pulling 42ba7f8a01dd... 100% ▕████████████████████████████▏  557 B
verifying sha256 digest
writing manifest
removing any unused layers
success
```

4. Use the GenAI application

4-1 Build the application

Now that we set up the LLM service, build and run the GenAI application again:

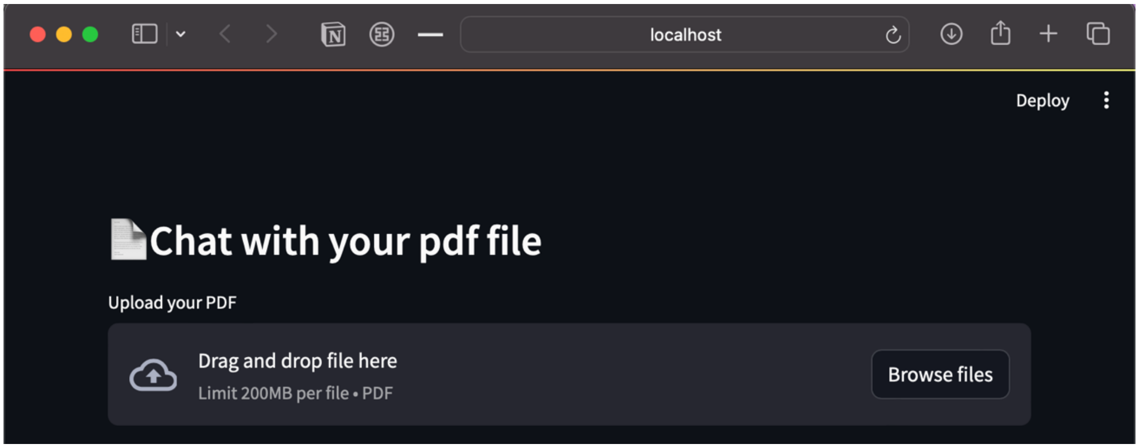
```
% docker compose up —build
```



```
(base) zhangzhiyu@shengchligongju docker-genai-sample % docker compose up --build
[+] Building 2.4s (14/14) FINISHED                                          docker:desktop-linux
 => [server internal] load build definition from Dockerfile                                 0.0s
 => => transferring dockerfile: 1.71kB                                                      0.0s
 => [server] resolve image config for docker.io/docker/dockerfile:1                         1.2s
 => [server auth] docker/dockerfile:pull token for registry-1.docker.io                     0.0s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07      0.0s
 => [server internal] load metadata for docker.io/library/python:3.11.3-slim                0.8s
 => [server auth] library/python:pull token for registry-1.docker.io                        0.0s
 => [server internal] load .dockerignore                                                    0.0s
 => => transferring context: 667B                                                           0.0s
 => [server base 1/5] FROM docker.io/library/python:3.11.3-slim@sha256:eaee5f73efa9ae9      0.0s
 => [server internal] load build context                                                    0.0s
 => => transferring context: 252B                                                           0.0s
 => CACHED [server base 2/5] WORKDIR /app                                                   0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --      0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --moun      0.0s
 => CACHED [server base 5/5] COPY . .                                                       0.0s
 => [server] exporting to image                                                             0.0s
 => => exporting layers                                                                     0.0s
 => => writing image sha256:6c052040886bffa92effb3472e22f5382145e08b13413e250b9084511c      0.0s
 => => naming to docker.io/library/docker-genai-sample-server                               0.0s
[+] Running 2/2
 ✔ Container docker-genai-sample-database-1  Created                                        0.0s
 ✔ Container docker-genai-sample-server-1    Recreated                                      0.1s
Attaching to database-1, server-1
database-1  | Changed password for user 'neo4j'. IMPORTANT: this change will only take effect
 if performed before the database is started for the first time.
database-1  | 2024-04-10 19:04:04.517+0000 INFO  Starting...
database-1  | 2024-04-10 19:04:05.537+0000 INFO  This instance is ServerId{54e71723} (54e7172
3-e28f-4574-96ce-178060058386)
database-1  | 2024-04-10 19:04:06.595+0000 INFO  ======== Neo4j 5.11.0 ========
database-1  | 2024-04-10 19:04:08.234+0000 INFO  Bolt enabled on 0.0.0.0:7687.
database-1  | 2024-04-10 19:04:09.220+0000 INFO  Remote interface available at http://localho
st:7474/
database-1  | 2024-04-10 19:04:09.225+0000 INFO  id: DD5BDE712E48E2EB7EF42A44EFFC2792E6E4760B
E3F1FC2DEDD5E377F35A5A63
database-1  | 2024-04-10 19:04:09.226+0000 INFO  name: system
database-1  | 2024-04-10 19:04:09.226+0000 INFO  creationDate: 2024-04-10T18:26:24.689Z
database-1  | 2024-04-10 19:04:09.227+0000 INFO  Started.
server-1    |
server-1    | Collecting usage statistics. To deactivate, set browser.gatherUsageStats to Fal
se.
server-1    |
server-1    |
server-1    |   You can now view your Streamlit app in your browser.
server-1    |
server-1    |   URL: http://0.0.0.0:8000
server-1    |
```

4-2 Access the application

Open a browser and view the application at http://localhost:8000:



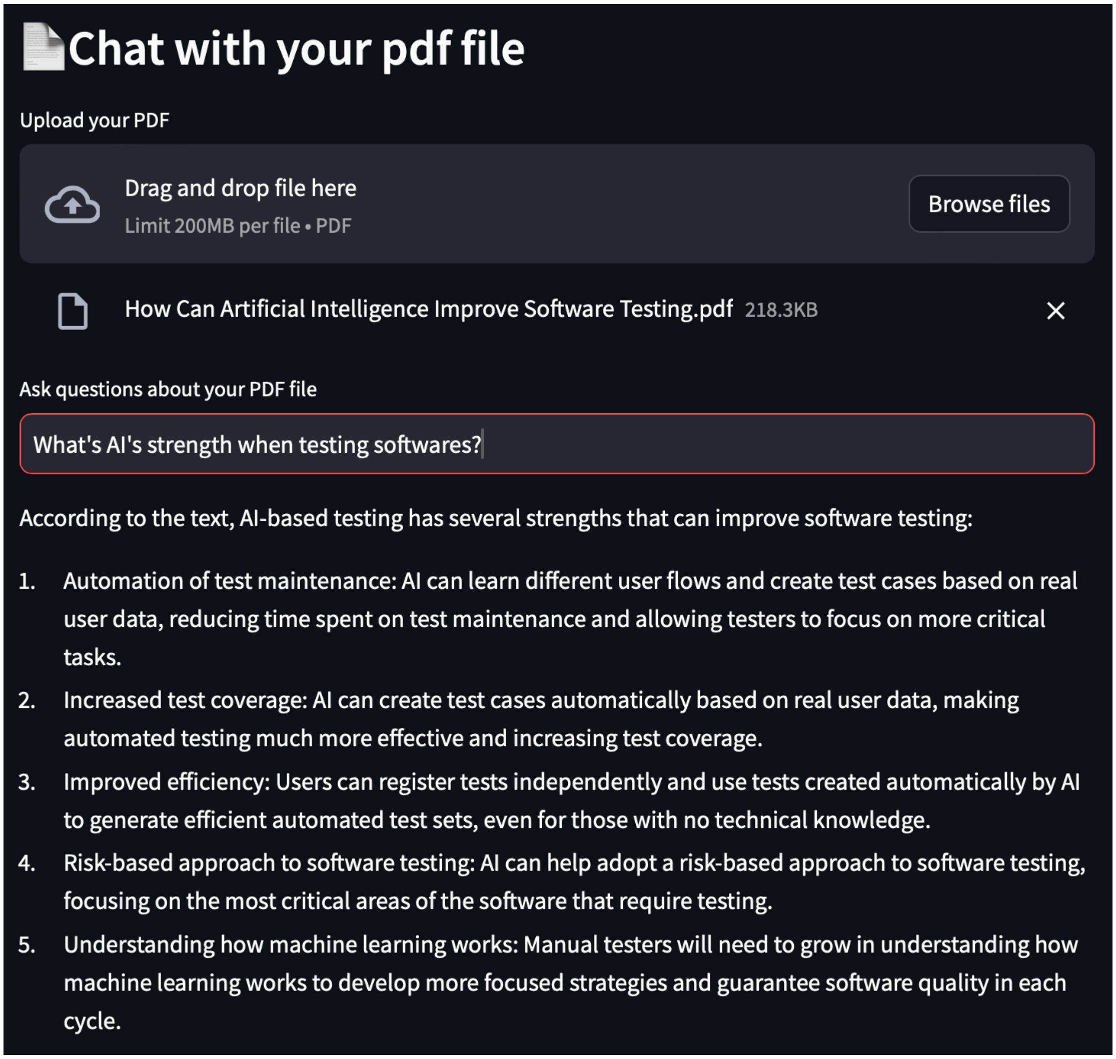
📄Chat with your pdf file

Upload your PDF

Drag and drop file here

Limit 200MB per file • PDF

Now we can upload a PDF file and ask questions about the file. Depending on your system and the LLM service you chose, it may take a couple minutes to generate the answer:


Chat with your pdf file
Upload your PDF
Drag and drop file here
Limit 200MB per file • PDF
Browse files
How Can Artificial Intelligence Improve Software Testing.pdf 218.3KB
Ask questions about your PDF file

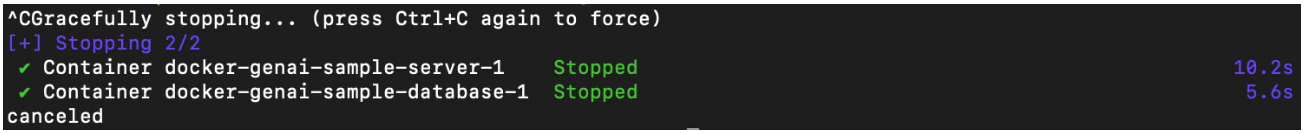
What's AI's strength when testing softwares?

According to the text, AI-based testing has several strengths that can improve software testing:

1. Automation of test maintenance: AI can learn different user flows and create test cases based on real user data, reducing time spent on test maintenance and allowing testers to focus on more critical tasks.
2. Increased test coverage: AI can create test cases automatically based on real user data, making automated testing much more effective and increasing test coverage.
3. Improved efficiency: Users can register tests independently and use tests created automatically by AI to generate efficient automated test sets, even for those with no technical knowledge.
4. Risk-based approach to software testing: AI can help adopt a risk-based approach to software testing, focusing on the most critical areas of the software that require testing.
5. Understanding how machine learning works: Manual testers will need to grow in understanding how machine learning works to develop more focused strategies and guarantee software quality in each cycle.

4-3 Stop the application

Once you finish using the application, go back to terminal and press ctrl + C to stop the application:



```
^CGracefully stopping... (press Ctrl+C again to force)
[+] Stopping 2/2
 ✔ Container docker-genai-sample-server-1    Stopped                                              10.2s
 ✔ Container docker-genai-sample-database-1  Stopped                                               5.6s
canceled
```