LSAP HW5

B11705044 資管四 江彥宏

1. Clone the Repository from GitHub

```
git clone https://github.com/namwoam/wiki-assistant.git
```

2. Run the application locally

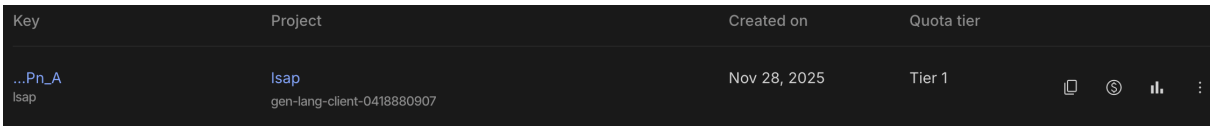
Frontend:

```
(base) jiangyanhong@ZZZZZZZ frontend % poetry run streamlit run chat.py --server.
port 8501 --server.address 0.0.0.0
The currently activated Python version 3.12.9 is not supported by the project (>=
3.13,<3.14).
Trying to find and use a compatible version.
Using python3.13 (3.13.9)

  You can now view your Streamlit app in your browser.

  URL: http://0.0.0.0:8501

  For better performance, install the Watchdog module:

  $ xcode-select --install
  $ pip install watchdog
```

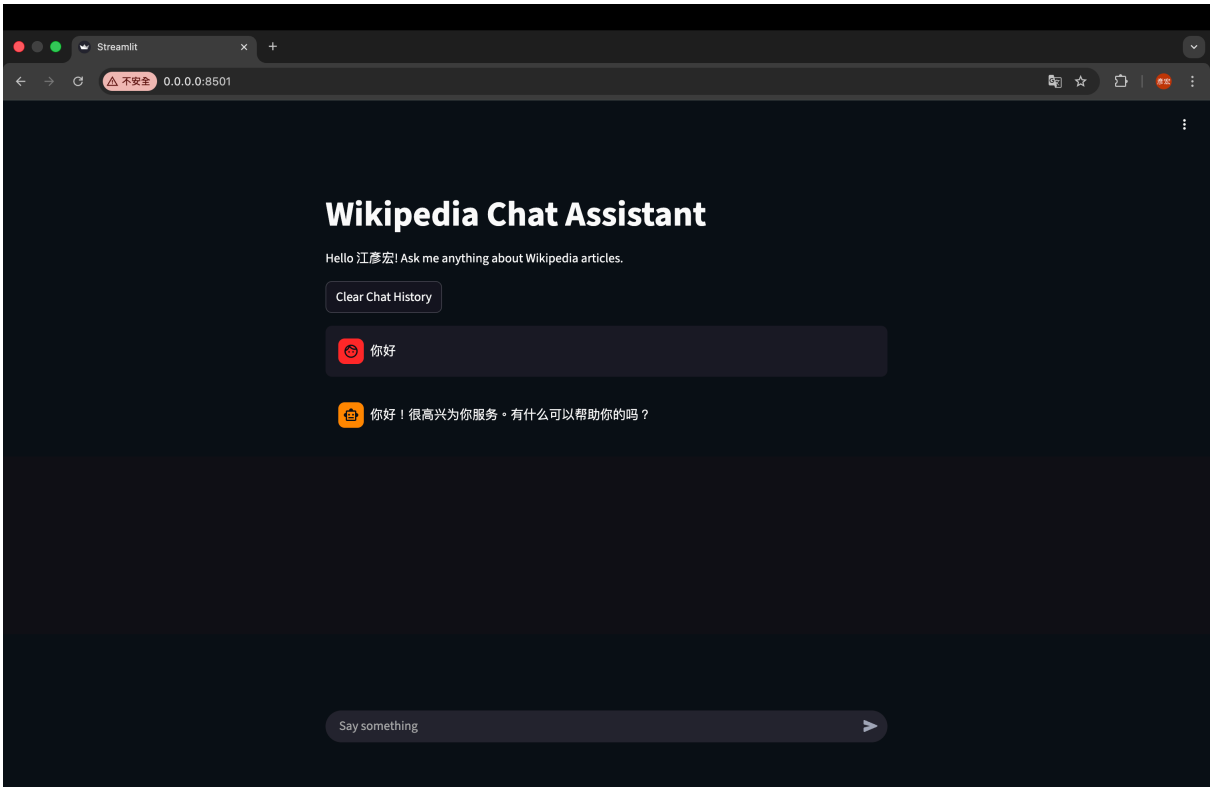
Backend:

```
(base) jiangyanhong@ZZZZZZZ backend % poetry run uvicorn server:app --host 0.0.0
.0 --port 8000
INFO:     Started server process [79994]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     127.0.0.1:50586 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50607 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50635 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50681 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50687 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50736 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50741 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50778 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50793 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50804 - "GET /history HTTP/1.1" 200 OK
INFO:     127.0.0.1:50813 - "POST /history HTTP/1.1" 200 OK
```

3. Obtain a Google AI Studio API Key



4. Modify the application



5. Containerize the application

Dockerfile:

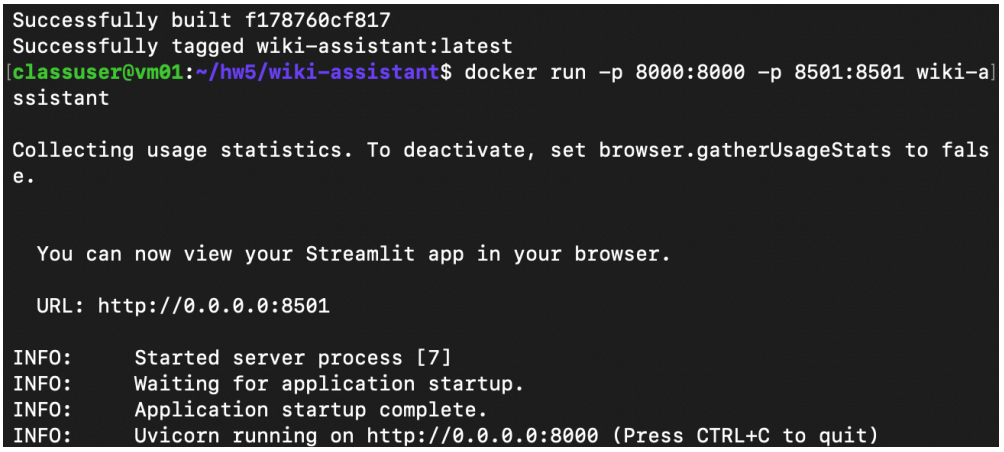
```
FROM python:3.12-slim

RUN apt-get update && apt-get install -y --no-install-recommends \
    curl build-essential \
    && rm -rf /var/lib/apt/lists/*

ENV POETRY_HOME="/opt/poetry"
ENV PATH="$POETRY_HOME/bin:$PATH"
RUN curl -sSL https://install.python-poetry.org | python3 -

ENV POETRY_VIRTUALENVS_IN_PROJECT=true
ENV POETRY_NO_INTERACTION=1
ENV POETRY_CACHE_DIR='/tmp/poetry-cache'

WORKDIR /app

COPY backend/pyproject.toml backend/poetry.lock /app/backend/
COPY frontend/pyproject.toml frontend/poetry.lock /app/frontend/

WORKDIR /app/backend
RUN poetry install --no-root

WORKDIR /app/frontend
RUN poetry install --no-root

WORKDIR /app
COPY backend /app/backend
```

```
COPY frontend /app/frontend

COPY entrypoint.sh /app/entrypoint.sh
RUN chmod +x /app/entrypoint.sh

EXPOSE 8000
EXPOSE 8501

CMD ["/app/entrypoint.sh"]
```

entrypoint.sh:

```
#!/bin/bash

# Start backend
cd /app/backend
poetry run uvicorn server:app --host 0.0.0.0 --port 8000 &

# Start frontend
cd /app/frontend
poetry run streamlit run chat.py \
  --server.headless=true \
  --server.address=0.0.0.0 \
  --server.port=8501 \
  --server.enableWebsocketCompression=false \
  --server.enableCORS=false
```

6. Run the container



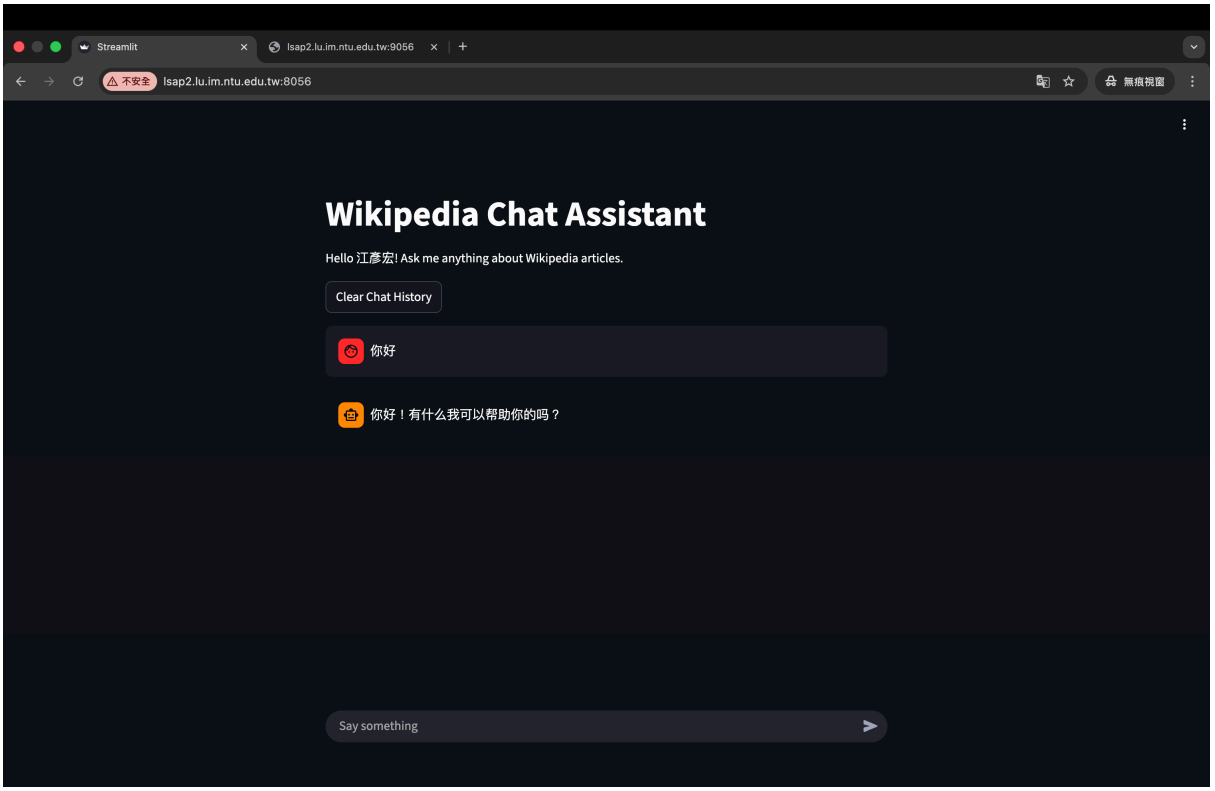
7. Use Nginx as a Reverse Proxy

Nginx site config:

```
map $http_upgrade $connection_upgrade {
  default upgrade;
  '' close;
}

server {
  listen 80;
```

```
    server_name _;

    client_max_body_size 200M;

    # Streamlit websocket + assets
    location /_stcore/ {
        proxy_pass http://localhost:8501;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_buffering off;
        proxy_read_timeout 86400;
        proxy_send_timeout 86400;
    }

    # Default route → Streamlit app
    location / {
        proxy_pass http://localhost:8501;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_buffering off;
        proxy_read_timeout 86400;
        proxy_send_timeout 86400;
    }
}
```

8. Verify External Accessibility



9. Compare Linux distributions

Base Image	**Build Time**	Startup Time
ubuntu:22.04	891s	10.841s
alpine:3.22	137s	5.183s

```
# Dockerfile-ubuntu
FROM ubuntu:22.04

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && apt-get install -y \
    wget curl git build-essential \
    zlib1g-dev libssl-dev libffi-dev libbz2-dev \
    libreadline-dev libsqlite3-dev liblzma-dev \
    ca-certificates \
    && rm -rf /var/lib/apt/lists/*

WORKDIR /tmp
RUN wget https://www.python.org/ftp/python/3.12.2/Python-3.12.2.tgz \
    && tar -xvf Python-3.12.2.tgz \
    && cd Python-3.12.2 \
    && ./configure --enable-optimizations \
    && make -j$(nproc) \
    && make altinstall

RUN ln -sf /usr/local/bin/python3.12 /usr/bin/python3

RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py \
    && python3 get-pip.py \
```

```
    && rm get-pip.py

ENV POETRY_HOME="/opt/poetry"
ENV PATH="$POETRY_HOME/bin:$PATH"
RUN curl -sSL https://install.python-poetry.org | python3

ENV POETRY_VIRTUALENVS_IN_PROJECT=true
ENV POETRY_NO_INTERACTION=1
ENV POETRY_CACHE_DIR='/tmp/poetry-cache'

WORKDIR /app

COPY backend/pyproject.toml backend/poetry.lock /app/backend/
COPY frontend/pyproject.toml frontend/poetry.lock /app/frontend/

WORKDIR /app/backend
RUN poetry install --no-root

WORKDIR /app/frontend
RUN poetry install --no-root

WORKDIR /app
COPY backend /app/backend
COPY frontend /app/frontend

COPY entrypoint.sh /app/entrypoint.sh
RUN chmod +x /app/entrypoint.sh

EXPOSE 8000 8501

CMD ["/app/entrypoint.sh"]
```

```
# Dockerfile-alpine
FROM alpine:3.22

RUN apk update && apk add --no-cache \
    python3 \
    py3-pip \
    curl \
    bash \
    build-base \
    libffi-dev \
    openssl-dev \
    python3-dev \
    linux-headers \
    git \
    py3-numpy \
    py3-pandas

RUN rm -f /usr/lib/python3.12/EXTERNALLY-MANAGED

ENV POETRY_HOME="/opt/poetry"
ENV PATH="$POETRY_HOME/bin:$PATH"
ENV POETRY_VIRTUALENVS_CREATE=false
```

```
RUN curl -sSL https://install.python-poetry.org | python3 -

WORKDIR /app

COPY backend/pyproject.toml backend/poetry.lock /app/backend/
COPY frontend/pyproject.toml frontend/poetry.lock /app/frontend/

WORKDIR /app/backend
RUN rm -f poetry.lock && poetry install --no-root

WORKDIR /app/frontend
RUN poetry install --no-root

WORKDIR /app
COPY backend /app/backend
COPY frontend /app/frontend

COPY entrypoint.sh /app/entrypoint.sh
RUN chmod +x /app/entrypoint.sh

EXPOSE 8000
EXPOSE 8501

CMD ["/bin/bash", "/app/entrypoint.sh"]
```

Alpine 因為系統更精簡、套件多為預先編譯版本，不需要像 Ubuntu 那樣重新編譯 Python 和科學套件，因此不論建置時間或啟動時間都明顯較快。

10. Compare container build speed

把 source code 放在最前面會讓每次程式碼變動都破壞 Docker cache，導致依賴套件必須重新安裝，因此重建容器的速度會比原本方式慢非常多。

11. Persistent chat history

Demo video: https://youtu.be/vx1YWfF9cpM

Docker run command:

```
docker run -d \
  -p 8000:8000 \
  -p 8501:8501 \
  -v /home/classuser/hw5/chatdata:/app/backend/chatdata \
  wiki-assistant
```

Modified `server.py`:

```
from typing import Union
from fastapi import FastAPI
import json
import os

from utils import (
    get_wiki_text_from_url,
    split_text_into_chunks,
    query_chunks_with_query,
```

```
    search_for_wikipedia_page_url,
)

app = FastAPI()

BASE_DIR = os.path.dirname(os.path.abspath(__file__))         # /app/backend
DATA_DIR = os.path.join(os.path.dirname(BASE_DIR), "chatdata") # /app/chatdata
DB_FILE = os.path.join(DATA_DIR, "chat_history.json")

os.makedirs(DATA_DIR, exist_ok=True)

def load_history_from_file():
    if os.path.exists(DB_FILE):
        with open(DB_FILE, "r", encoding="utf-8") as f:
            return json.load(f)
    return []

def save_history_to_file(history):
    with open(DB_FILE, "w", encoding="utf-8") as f:
        json.dump(history, f, ensure_ascii=False, indent=2)

# Load initial history on startup
chat_history = load_history_from_file()

@app.get("/")
def read_root() -> dict[str, str]:
    return {"message": "Welcome to the Wiki Assistant API"}

@app.get("/query")
def query_wiki(url: str, query: str) -> dict[str, Union[list[str] | str, None]]:
    try:
        wiki_text = get_wiki_text_from_url(url)
        chunks = split_text_into_chunks(wiki_text)
        relevant_chunks = query_chunks_with_query(chunks, query)
        return {"relevant_chunks": relevant_chunks}
    except Exception as e:
        return {"error": str(e)}

@app.get("/explore")
def explore_relevant_wiki_pages(query: str) -> dict[str, Union[list[str] | str, None]]:
    try:
        page_urls = search_for_wikipedia_page_url(query)
        if page_urls is None:
            return {"page_urls": []}
        return {"page_urls": page_urls}
    except Exception as e:
        return {"error": str(e)}

@app.get("/history")
def get_history() -> dict[str, list[dict[str, str]]]:
    return {"chat_history": chat_history}

@app.post("/history")
def modify_history(history: list[dict[str, str]]) -> dict[str, str]:
    global chat_history
```

```
    chat_history = history
    save_history_to_file(chat_history)
    return {"message": "Chat history updated successfully"}

@app.delete("/history")
def clear_history() -> dict[str, str]:
    global chat_history
    chat_history = []
    save_history_to_file(chat_history)
    return {"message": "Chat history cleared successfully"}
```

12. Multiple containers

```
[classuser@vm01:~/hw5/wiki-assistant$ docker compose ps                                    ]
NAME            IMAGE                     COMMAND                  SERVICE    CR
EATED          STATUS          PORTS
wiki-backend    wiki-assistant-backend    "poetry run uvicorn …"   backend    19
 minutes ago   Up 6 minutes    0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp
wiki-frontend   wiki-assistant-frontend   "poetry run streamli…"   frontend   15
 minutes ago   Up 12 minutes   0.0.0.0:8501->8501/tcp, [::]:8501->8501/tcp
```

Dockerfile.frontend:

```
FROM python:3.12-slim

RUN apt-get update && apt-get install -y --no-install-recommends \
    curl build-essential \
    && rm -rf /var/lib/apt/lists/*

ENV POETRY_HOME="/opt/poetry"
ENV PATH="$POETRY_HOME/bin:$PATH"
RUN curl -sSL https://install.python-poetry.org | python3 -

ENV POETRY_VIRTUALENVS_IN_PROJECT=true
ENV POETRY_NO_INTERACTION=1
ENV POETRY_CACHE_DIR='/tmp/poetry-cache'

WORKDIR /app/frontend

COPY frontend/pyproject.toml frontend/poetry.lock ./

RUN poetry install --no-root
COPY frontend /app/frontend

EXPOSE 8501
CMD ["poetry", "run", "streamlit", "run", "chat.py", \
    "--server.headless=true", \
    "--server.address=0.0.0.0", \
    "--server.port=8501", \
    "--server.enableWebsocketCompression=false", \
    "--server.enableCORS=false"]
```

Dockerfile.backend:

```
FROM python:3.12-slim
```

```
RUN apt-get update && apt-get install -y --no-install-recommends \
    curl build-essential \
    && rm -rf /var/lib/apt/lists/*

ENV POETRY_HOME="/opt/poetry"
ENV PATH="$POETRY_HOME/bin:$PATH"
RUN curl -sSL https://install.python-poetry.org | python3 -

ENV POETRY_VIRTUALENVS_IN_PROJECT=true
ENV POETRY_NO_INTERACTION=1
ENV POETRY_CACHE_DIR='/tmp/poetry-cache'

WORKDIR /app/backend
COPY backend/pyproject.toml backend/poetry.lock ./
RUN poetry install --no-root
COPY backend /app/backend

EXPOSE 8000
CMD ["poetry", "run", "uvicorn", "server:app", \
       "--host", "0.0.0.0", "--port", "8000"]
```

`docker-compose.yml`:

```
services:
  backend:
    build:
      context: .
      dockerfile: Dockerfile.backend
    container_name: wiki-backend
    ports:
      - "8000:8000"
    volumes:
      - ./backend:/app/backend:rw
      - /app/backend/.venv
    restart: unless-stopped

  frontend:
    build:
      context: .
      dockerfile: Dockerfile.frontend
    container_name: wiki-frontend
    ports:
      - "8501:8501"
    environment:
      BACKEND_URL: "http://backend:8000"
    volumes:
      - ./frontend:/app/frontend:rw
      - /app/frontend/.venv
    depends_on:
      - backend
    restart: unless-stopped
```