Class Activity: Setting Up a Simple Text Generation FastAPI Project

This guide will help you set up a FastAPI project using UV Package Manager. Docker is optional for containerization and deployment.

Prerequisites

Before getting started, ensure you have the following installed:

- **Python 3.7+**
- **UV Package Manager**: If you haven't installed UV, follow the installation instructions here
- **Docker (Optional)**: Only needed for containerization. Install from here

Steps to Set Up the FastAPI Project

1. Navigate to Your Project Directory

Navigate to your existing project directory (created in Module 1):

```
cd sps_genai
```

2. Create the Application Directory

Create a new `app` directory to store your FastAPI application code:

```
mkdir app
cd app
```

3. Create the Model and Main Application Files

As part of this activity you will move the appropriate functions from lecture code to set this up.

Use a code editor (such as VS Code) to create and edit the necessary Python files:

```
code bigram_model.py
code main.py
```

- `bigram_model.py`: This file will contain the logic for processing bigrams. You are responsible for editing this file. Use are free to use LLM based code assistants such as GitHub Copilot to help you transform the notebook code from Module 2 and make it available for your API.
- `main.py`: This is the entry point for your FastAPI application.

```
from typing import Union
from fastapi import FastAPI
from pydantic import BaseModel
from app.bigram_model import BigramModel

app = FastAPI()

# Sample corpus for the bigram model
corpus = [
    "The Count of Monte Cristo is a novel written by Alexandre Dumas. \
It tells the story of Edmond Dantès, who is falsely imprisoned and later seeks revе
    "this is another example sentence",
    "we are generating text based on bigram probabilities",
    "bigram models are simple but effective"
]

bigram_model = BigramModel(corpus)

class TextGenerationRequest(BaseModel):
    start_word: str
    length: int

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.post("/generate")
def generate_text(request: TextGenerationRequest):
    generated_text = bigram_model.generate_text(request.start_word, request.length
    return {"generated_text": generated_text}
```

4. Install Required Dependencies

Navigate back to the root project directory (if you're still in the `app` directory):

```
cd ..
```

Install the necessary libraries for your FastAPI application:

```
uv add ...
```

5. Sync Dependencies

Ensure that the dependencies are properly synchronized:

```
uv sync
```

6. Run the Application

You can run the application directly with UV:

```
uv run fastapi dev app/main.py
```

Access your FastAPI application at: `http://127.0.0.1:8000`

Optional: Docker Deployment

If you want to containerize your application for deployment, you can use Docker.

7. Create a Dockerfile

Create a `Dockerfile` for containerizing the application:

```
code Dockerfile
```

In the `Dockerfile`, add the following content:

```
# Use an official Python runtime as a parent image
FROM python:3.12-slim-bookworm

# The installer requires curl (and certificates) to download the release archive
RUN apt-get update && apt-get install -y --no-install-recommends curl ca-certifica

# Download the latest installer
ADD https://astral.sh/uv/install.sh /uv-installer.sh

# Run the installer then remove it
RUN sh /uv-installer.sh && rm /uv-installer.sh

# Ensure the installed binary is on the `PATH`
ENV PATH="/root/.local/bin/:$PATH"

# Set the working directory
WORKDIR /code

# Copy the pyproject.toml and uv.lock files
COPY pyproject.toml uv.lock /code/

# Install dependencies using uv
RUN uv sync --frozen

# Copy the application code
COPY ./app /code/app

# Command to run the application
CMD ["uv", "run", "fastapi", "run", "app/main.py", "--port", "80"]
```

8. Build and Run the Docker Image

Build the Docker image:

```
docker build -t sps-genai .
```

After the build completes, you can run the container using:

```
docker run -p 8000:80 sps-genai
```

Access your containerized FastAPI application at: `http://127.0.0.1:8000`

View the interactive API documentation at: `http://127.0.0.1:8000/docs`

Note: Although the container runs on port 80 internally, the port mapping (`-p 8000:80`) makes it accessible on port 8000 from your host machine, keeping the same access URL as the development server.