Understanding the Automated Course Planner Project

The goal of this project is to develop an automated system that helps university students plan their semester course registrations. This system would act as an academic advisor, recommending courses based on the student's major, courses they've already completed, and the ones they're currently enrolled in. To achieve this, the project explores implementing a Large Language Model (LLM), specifically using OpenAI's API, to generate plans dynamically.

Research and Resource Selection

To begin, extensive research was conducted to understand how to best incorporate AI into the course planning system. Because OpenAI's platform provides strong language models that can comprehend data, and produce responses it was recognized as a reliable option.

References Include:

- https://platform.openai.com/docs/overview
- https://www.coltsteele.com/tips/understanding-openai-s-temperature-parameter
- https://snipcart.com/blog/integrating-apis-introduction
- https://technologyadvice.com/blog/information-technology/how-to-use-an-api/
- https://youtu.be/OB99E7Y1cMA?si=FXuDrCaAEvj-IXl4
- https://youtu.be/4qNwoAAfnk4?si=B21vXDdopK5MPIPM
- https://youtu.be/Lag9Pj_33hM?si=am5zcSyGbHGKhopp

Technical Setup and Challenges

A major technical obstacle encountered was the need to pay for an OpenAI API key, which was required to test the code with OpenAI's servers. Without access to an active API key, it wasn't possible to validate if the code functioned as intended, making it challenging to confirm the success of our setup. Despite this limitation, a mock version of the code was written to simulate the anticipated interactions with OpenAI's API.

Code Structure

The code for the planner is organized into three main files: main.py, prompts.py, and .env. Each of these files serves a specific function:

> **The .env File**: This file stores sensitive information, specifically the OpenAI API key. By keeping the API key in a separate .env file, we enhance security, ensuring sensitive data is not directly embedded in the main code files. An example entry in the .env file is as follows:

```
API_KEY=
"sk-proj-enrrUc3KoA-FPqgpgE5BjOcFcGzGLYFHnWOI2jpYH1TawxiUVSjbeWUexEHPRKgqu
mOvpK0WtVT3BlbkFJoLZQPK3_BhdFs7qU1Khs8Zob5PTT5XfZk5YNgdhAk9OvdrZ6Is_hArulU
p-1zNlCagefZLVocA"
```

The main.py File: This file contains the core logic for creating a course registration plan. Here's a breakdown of the code and its main components:

```
import os
import openai
from dotenv import load_dotenv, find_dotenv
import prompts

#Load environment variables
_ = load_dotenv(find_dotenv())
openai.api_key = os.environ.get('API_KEY')

#Configuration
model = "gpt-4o-mini"
temperature = 0.3
max_tokens = 500

def create_plan(student_name, major, previous_courses, current_courses):
    #Generate the prompt based on the provided student data
    prompt = prompts.generate_prompt(student_name, major,
previous_courses, current_courses)

    #Define messages
    messages = [
        {"role": "system", "content": prompts.system_message},
        {"role": "user", "content": prompt}
    ]

    #Create a chat completion with OpenAI (using the synax form OpenAI
website)
    completion = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=temperature,
        max_tokens=max_tokens,
```

```
    )
    return completion.choices[0].message.content #According to the OpenAI
syntax this is where the AI sending out the reply

# Example Sample Data
student_name = "Zeina Elsawy"
major = "Computer Science"
previous_courses = ["CSCE 2202", "CSCE 2203", "MACT 2123", "CSCE 2303",
"CSCE 2301", "CSCE 2302"]
current_courses = ["MACT 3211", "CSCE 2501", "CSCE 4930", "CSCE 3701"]

#Output Plan
print(create_plan(student_name, major, previous_courses, current_courses))
```

- **Environment Setup**: The dotenv library is used to load the API key from .env.
- **API Configuration**: Parameters are set to adjust the model type ("gpt-4o-mini"), response temperature (0.3), and maximum token count (500). These settings influence how creative or concise the AI's responses will be.
- **Course Plan Generation Function**: The function create_plan is defined to take a student's name, major, previously completed courses, and currently registered courses as input. Using this data, it formulates a prompt that will be passed to OpenAI's language model, instructing it to generate a course plan.
- **Prompt Creation and API Interaction**: The create_plan function utilizes the prompts.py file to assemble a detailed message for the AI model. The response from the OpenAI API (simulating the academic advisor's advice) is then printed as the course plan.

  The prompts.py File: This file defines two main components: system_message and generate_prompt:

```
system_message = """
You are an academic advisor AI helping university students create a
semester course registration plan.
Consider each student's major, previously registered courses, and
currently registered courses.
Check the prerequisites and recommend a semester plan with the best
sequence of courses to help them progress smoothly toward their degree
requirements.
"""
```

```
def generate_prompt(student_name, major, previous_courses,
current_courses):
    return f"""
    Generate a semester course plan for {student_name}, who is majoring in
{major}.
    The student has completed the following courses: {previous_courses}.
They are currently registered in these courses: {current_courses}.
    Based on the prerequisite requirements, recommend the best courses for
the next semester to ensure steady progress toward graduation.
    Prioritize courses that are prerequisites for other major courses, and
suggest electives if needed.
    """
```

- **System Message:** The system_message sets up the context for the AI, instructing it to act as an academic advisor. It details the purpose of the model and how it should approach generating a course plan by considering prerequisites, electives, and the student's progress toward their degree.
- **Prompt Generation:** The generate_prompt function dynamically builds a specific prompt for each student based on their name, major, previously completed courses, and current enrollment. This tailored approach allows the AI to generate a semester plan that is both relevant and practical for each student's needs.