README.md

Project Overview

Welcome to our project! Our goal is to develop a suite of intelligent agents to assist students with various academic tasks and provide career guidance. These agents leverage state-of-the-art language models to understand user queries and generate relevant responses. In addition to the agents, we have also created a FastAPI server to expose the agents' functionalities via HTTP endpoints.

In this README, we'll provide a detailed overview of the project, including the functionality of each agent, the structure of the project files, and instructions on how to run the application.

Agents

1. Notes Agent

The Notes Agent is designed to help students prepare comprehensive notes on different subjects and topics. It collects specific details from the user, such as the topic for which notes are needed, the preferred style of notes (short, detailed, or last-minute revision), any reference material, and additional requirements or focus areas. Based on this information, the agent generates tailored notes to assist students in studying effectively.

Fields:

- **Topic**: Specifies the subject matter for which notes are requested.
- **Notes Style**: Determines the preferred style of the notes (short, detailed, or last-minute revision).
- **Reference Material**: Indicates any reference material or sources to be used for note preparation.
- **Additional Requirements**: Specifies any additional requirements or focus areas for the notes.

2. Questions Agent

The Questions Agent provides practice questions and answers to help students prepare for exams and assessments. Users can specify the topic for which they need practice questions, whether they require questions with or without answers, and any additional requirements or preferences they may have. The agent then generates appropriate practice questions tailored to the student's needs.

Fields:

- **Topic**: Specifies the subject matter for which practice questions are requested.
- **With Answers**: Indicates whether the practice questions should include answers.
- **Additional Requirements**: Specifies any additional requirements or preferences for the practice questions.

3. Career Guidance Agent

The Career Guidance Agent offers personalized career guidance and advice to students based on their educational background, interests, and future goals. Users provide details such as their current education level, degree or class, field of interest, and future aspirations. The agent then provides recommendations tailored to the student's profile.

Fields:

- **Education Level**: Indicates whether the student is currently in school or college.
- **Degree/Class**: Specifies the student's degree (if in college) or class (if in school).
- **Field of Interest**: Specifies the student's area of interest (e.g., law, medicine, software engineering).
- **Future Goal**: Indicates whether the student's future goal is further studies or employment.

FastAPI Server

We have created a FastAPI server to expose the functionalities of the agents via HTTP endpoints. The server provides three POST endpoints, each corresponding to one of the agents. Users can make POST requests to these endpoints with the required data models to receive responses generated by the language model.

Endpoints

1. **/notes**: Endpoint for the Notes Agent
2. **/questions**: Endpoint for the Questions Agent
3. **/career-guidance**: Endpoint for the Career Guidance Agent

Example Requests

Notes Agent

Endpoint:

```
POST /notes
```

Request Body:

```
{
  "topic": "Photosynthesis",
  "notes_style": "Detailed",
  "reference_material": "NCERT Textbook of Class Twelve",
  "additional_requirements": "Include detailed explanations of the Calvin cycle and the light-dependent react
}
```

Questions Agent

Endpoint:

```
POST /questions
```

Request Body:

```
{
  "topic": "Data Analysis",
  "with_answers": "Yes",
  "additional_requirements": "Include questions on statistical analysis and machine learning algorithms"
}
```

Career Guidance Agent

Endpoint:

```
POST /career-guidance
```

Request Body:

```
{
  "education_level": "College",
  "degree_or_class": "Computer Science",
  "field_of_interest": "Data Analysis",
  "future_goal": "Employment"
}
```

Accessing the API Using Postman

1. **Open Postman**.
2. **Select** `POST` from the dropdown.
3. **Enter the URL**: For example, `http://localhost:8000/notes` for the Notes Agent.
4. **Go to the** `Body` **tab**.
5. **Select** `raw` and `JSON` from the dropdown.
6. **Paste the JSON** request body into the text area.
7. **Click** `Send` .

File Structure

Our project repository has the following structure:

- **.env**: Environment variables file containing sensitive information (e.g., API keys). This file is not version-controlled.
- **.env.example**: Example environment variables file to guide users in setting up their own environment.
- **.gitignore**: File specifying patterns to be ignored by Git (e.g., `.env` file).
- **career_guidance_agent.py**: Python script defining the Career Guidance Agent model and functionality.
- **main.py**: Main script to run the application.
- **notes_agent.py**: Python script defining the Notes Agent model and functionality.
- **questions_agent.py**: Python script defining the Questions Agent model and functionality.
- **server.py**: Python script defining the FastAPI server.
- **test_agent.py**: Python script containing test code.
- **utils.py**: Python script containing utility functions used in the project.
- **Demo-Videos**: Folder containing demo videos, including `Main.Py-Demo.mp4` .

How to Run It

To run the application, follow these steps:

1. **Set Up Virtual Environment**: Create a virtual environment to isolate project dependencies.

   ```
   python -m venv venv
   ```

2. **Activate Virtual Environment**: Activate the virtual environment.
   - On Windows:

     ```
     venv\Scripts\activate
     ```

   - On Unix or MacOS:

     ```
     source venv/bin/activate
     ```

3. **Install Requirements**: Install the required packages from the `requirements.txt` file.

   ```
   pip install -r requirements.txt
   ```

4. **Run the Main File**: Execute the `main.py` file to start the application.

   ```
   python main.py
   ```

In the `main.py` file, we create a bureau of agents using the `Bureau` class from the `uagents` module. The `Bureau` class manages a collection of agents and orchestrates their execution.

The bureau will handle the coordination and execution of the agents, allowing them to interact with each other and perform their designated tasks. This setup ensures seamless integration and operation of the agents within the system.

Additionally, you can run the FastAPI server to expose the agents' functionalities via HTTP endpoints:

5. **Run the Server File**: Execute the `server.py` file to start the FastAPI server.

   ```
   uvicorn server:app --reload
   ```

You can then use Postman or any other HTTP client to interact with the API as described in the FastAPI Server section.

That's it! You should now have the project up and running, ready to assist students with their academic needs and career aspirations. Enjoy exploring the functionality of our intelligent agents!

Demo Videos

The `Demo-Videos` folder contains demonstration videos showcasing various aspects of the project. These videos provide a visual guide to help you understand how to use and interact with the application. The following videos are available:

- **Main.Py-Demo.mp4**: This video walks through the process of running the main file and interacting with the agents.
- **FastAPI-Server-Demo.mp4**: This video provides a comprehensive guide on how to run the FastAPI server. It shows how to start the server, make POST requests using Postman, and interact with the different endpoints corresponding to the Notes Agent, Questions Agent, and Career Guidance Agent.

These videos are intended to help you get started quickly and effectively with our intelligent agents and FastAPI server.