# Agent-Based Task Execution System

## Overview

This project implements an agent-based system that utilizes language models and web search APIs to execute complex tasks. The system comprises specialized agents that work together to gather and synthesize information based on user-provided task descriptions.

## Features

- \*\*Google Search Integration\*\*: Uses SerpAPI to perform Google searches and retrieve the top 10 links.  
- \*\*YouTube Video Analysis\*\*: Fetches video information using the YouTube Data API, providing links and descriptions.  
- \*\*Domain Expert Advice\*\*: Leverages OpenAI's GPT model to provide expert insights in legal and medical fields.  
- \*\*User-Friendly Interface\*\*: Built with Streamlit for an interactive user experience.

## Setup and Installation

### Prerequisites

- Python 3.7 or higher  
- API keys for OpenAI, SerpAPI, and YouTube Data API

### Installation

1. Clone the repository:  
 ```bash  
 git clone https://github.com/your-username/agent-based-task-execution.git  
 cd agent-based-task-execution  
 ```  
2. Install required packages:  
 ```bash  
 pip install -r requirements.txt  
 ```  
3. Set up your API keys in the `.env` file:  
 - `OPENAI\_API\_KEY`: Your OpenAI API key  
 - `SERPAPI\_KEY`: Your SerpAPI key  
 - `YOUTUBE\_API\_KEY`: Your YouTube API key

### Running the Application

1. Use Streamlit to run the Python script:  
 ```bash  
 streamlit run main.py  
 ```  
2. Enter a task description in the input field and click "Execute Task" to see the results.

## Approach and Implementation

### Agent Design

The system uses a modular approach, where each agent is responsible for a specific task:  
- \*\*GoogleSearch Agent\*\*: Performs web searches and retrieves top links using SerpAPI.  
- \*\*YouTubeVideo Agent\*\*: Searches YouTube for relevant videos and returns details.  
- \*\*LegalAdvisor Agent\*\*: Provides legal advice using OpenAI's language model.  
- \*\*MedicalAdvisor Agent\*\*: Offers medical insights using OpenAI's language model.

### Techniques and Models

- \*\*OpenAI GPT-4\*\*: Used for generating expert-level insights and summaries.  
- \*\*SerpAPI\*\*: Provides access to Google search results.  
- \*\*YouTube Data API\*\*: Retrieves video data from YouTube.

### Error Handling

The system includes error handling for API requests, ensuring graceful failure and informative error messages.

## Best Practices

- The code is organized with clear function definitions and class structures.  
- Comments are added throughout the code to explain the purpose and functionality of each component.  
- Modular design allows for easy scalability and maintenance.

## Dependencies

- `openai`: For interacting with OpenAI's language model API.  
- `requests`: For making HTTP requests to external APIs.  
- `streamlit`: For creating an interactive user interface.  
- `google-search-results`: For accessing Google search results via SerpAPI.  
- `python-dotenv`: For loading environment variables from the `.env` file.

## Future Improvements

- Add more agents to cover additional domains and tasks.  
- Enhance the error handling mechanism for better reliability.  
- Improve user interface with more interactive features.