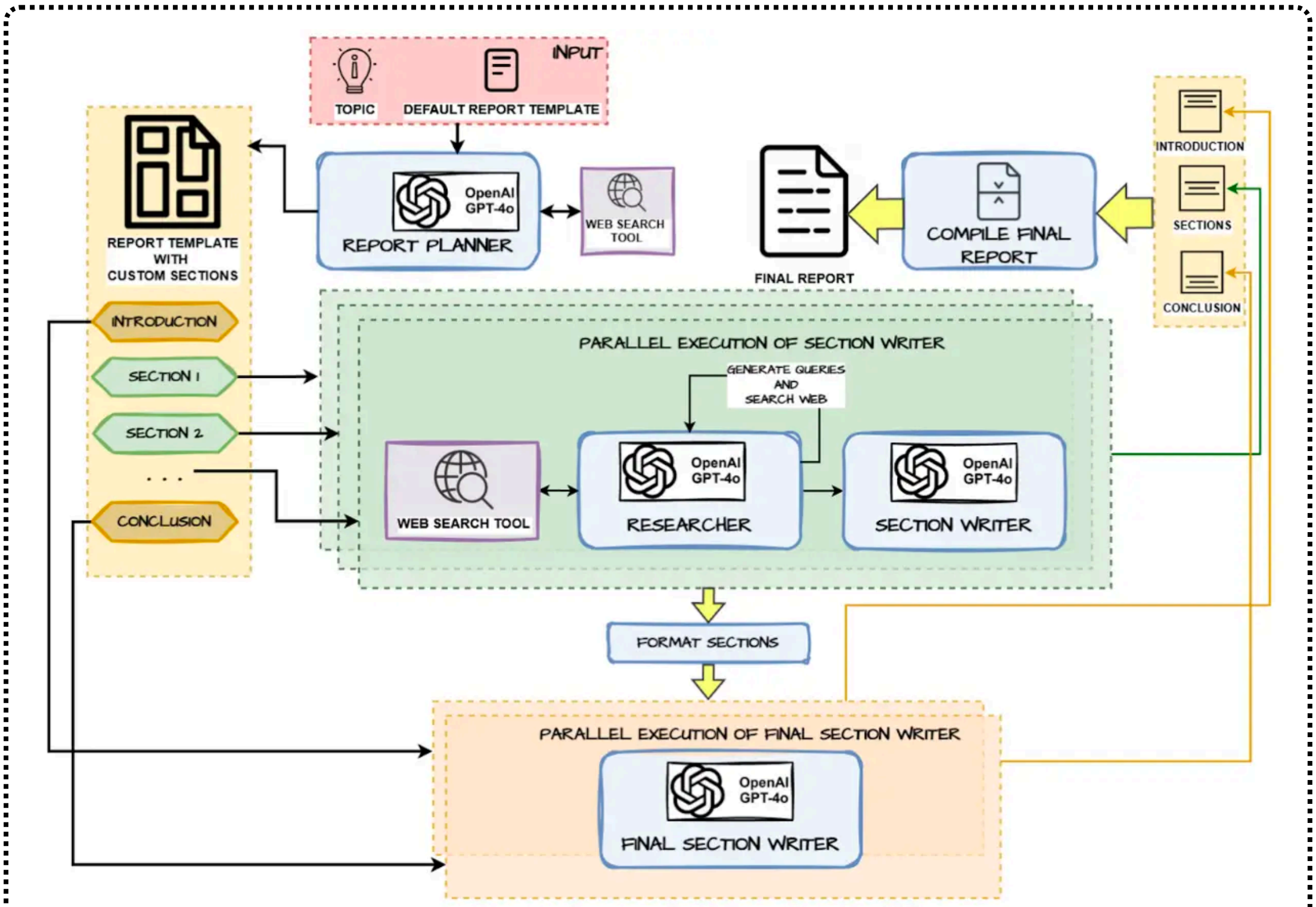
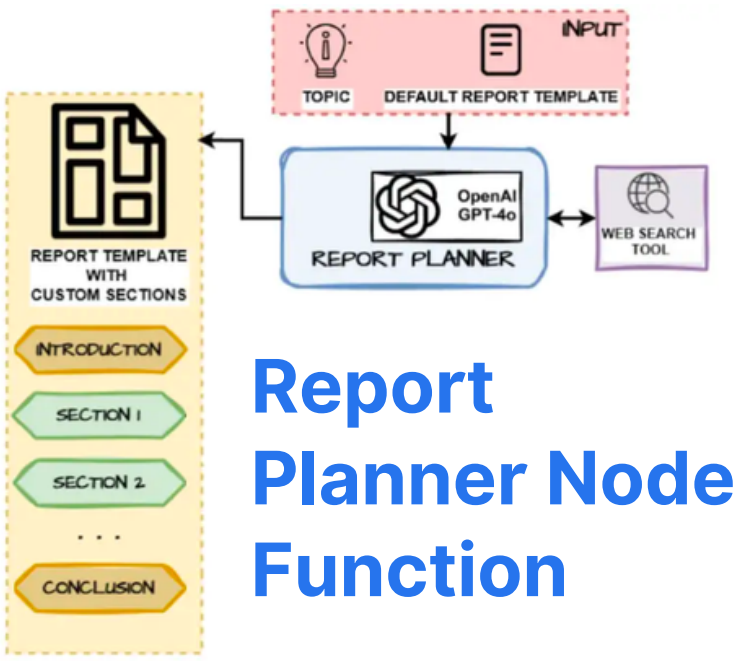
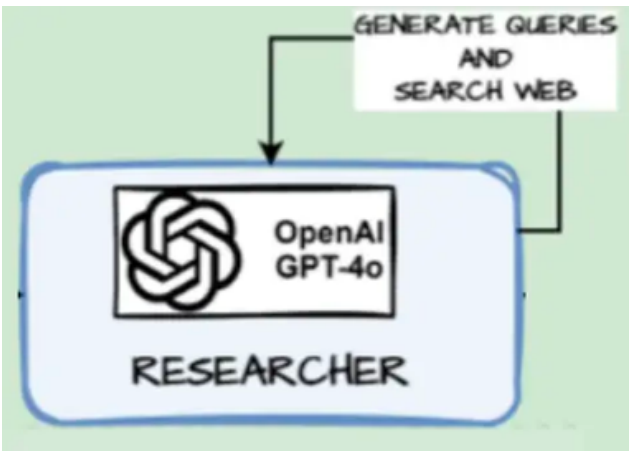
Analytics Vidhya

Build a Deep Research AI Agent

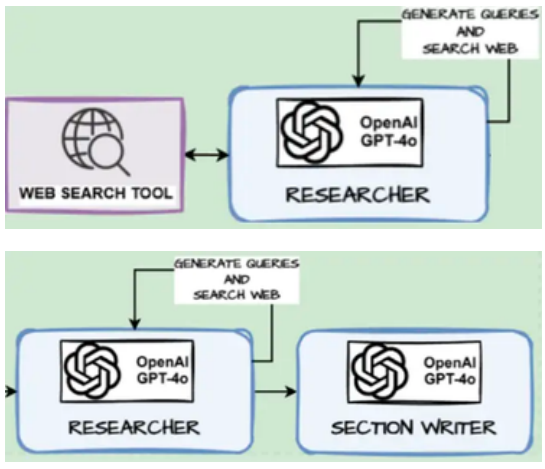




Report Planner Node Function



Query Generator Node Function



Free Course Inside

Deep Research & Structured Report Generation Planning Agentic AI System Architecture

The following figure shows the overall architecture of our system which we will be implementing with LangChain's LangGraph open-source framework for building stateful agentic systems with ease and control.

The key components which will power the above system include:

- A powerful Large Language Model which is good in reasoning. We are using GPT-4o which is not super expensive and fast, however, you can even use LLMs like Llama 3.2 or other open-source alternatives.

- LangGraph for building our agentic system as it is an excellent framework for building cyclical graph-based systems which can maintain state variables throughout the workflow and help build agentic feedback loops easily.

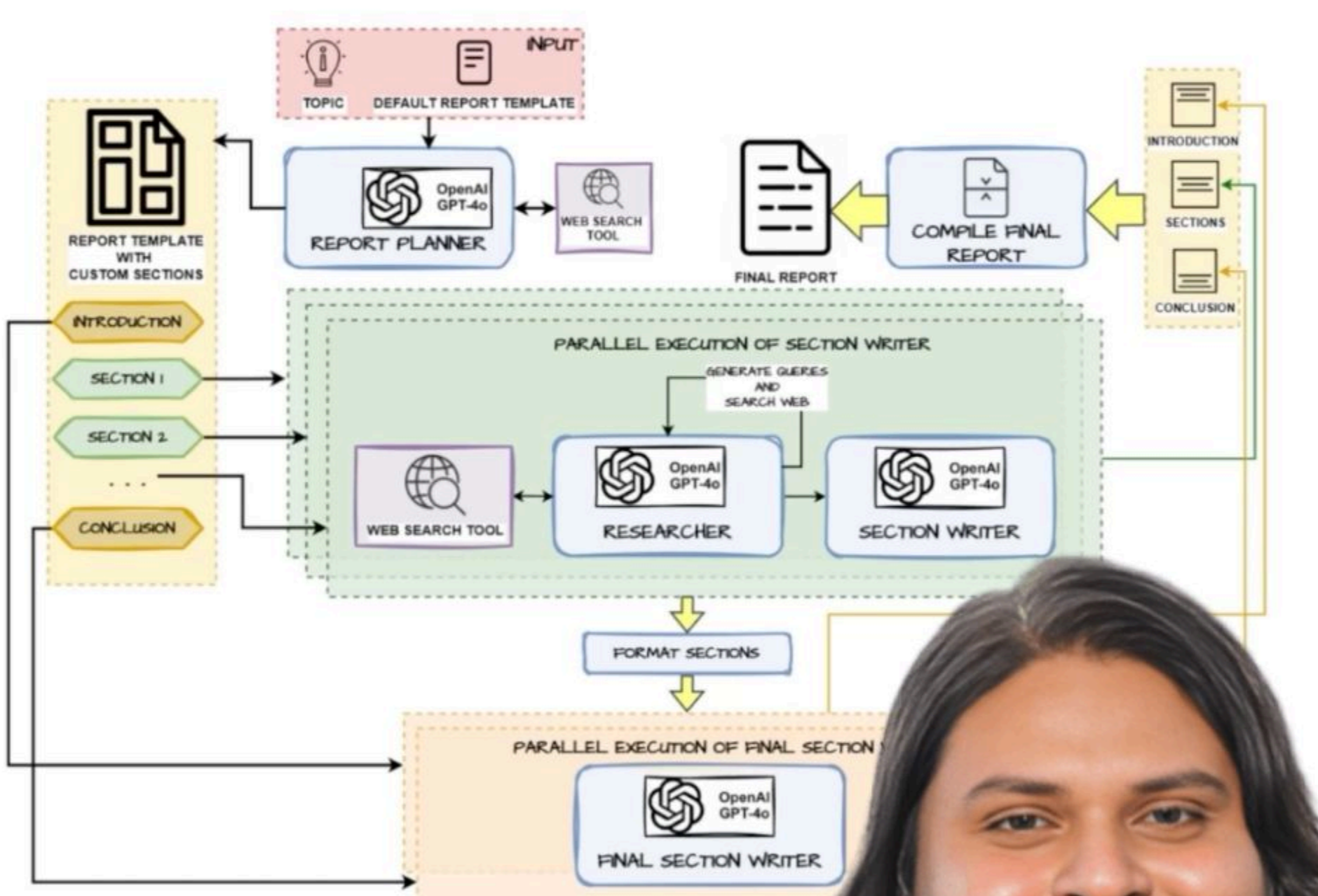
- Tavily AI is an excellent AI-powered search engine, perfect for web research and getting data from websites to power our Deep Research System.

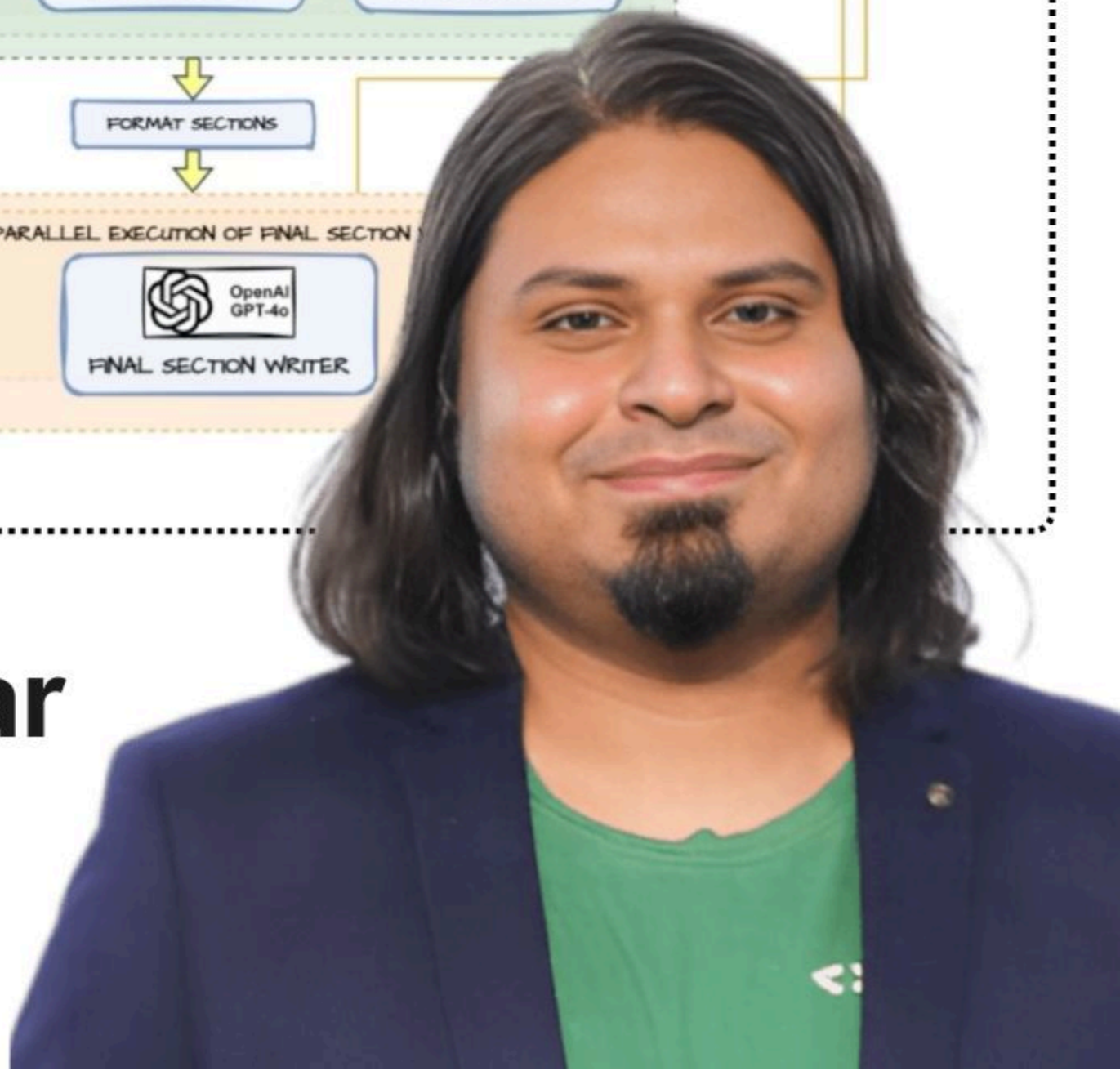
This project focuses on building a Planning Agent for Deep Research and Structured Report Generation as an alternative to OpenAI's Deep Research. The agent follows the popular Planning Agent Design Pattern and automates the process of analyzing a user-defined topic, performing deep web research, and generating a well-structured report.

Analytics Vidhya

Free Course on

Building a Deep Research AI Agent





Dipanjan Sarkar

Head of Community &
Principal AI Scientist
Analytics Vidhya

The workflow is actually inspired by LangChain's own Report mAIstro so full credit to them for coming up with the workflow, I took that as my baseline inspiration and then built out this system, which includes the following components:

1. **Report Planning**:

- The agent analyzes the user-provided topic and default report template to create a custom plan for the report.
- Sections such as Introduction, Key Sections, and Conclusion are defined based on the topic.
- A web search tool is used to collect the information required before deciding on the main sections.

2. **Parallel Execution for Research and Writing**:

- The agent uses parallel execution to efficiently perform:
  - Web Research: Queries are generated for each section and executed via the web search tool to retrieve up-to-date information.
  - Section Writing: The retrieved data is used to write content for each section, with the following process:

    - The Researcher gathers relevant data from the web.
    - The Section Writer uses the data to generate structured content for the assigned section.

3. **Formatting Completed Sections**

- Once all sections are written, they are formatted to ensure consistency and adherence to the report structure.

4. **Introduction and Conclusion Writing**

- After the main sections are completed and formatted:
    - The Introduction and Conclusion are written based on the content of the remaining sections (in parallel)
    - This process ensures that these sections align with the overall flow and insights of the report.

5. **Final Compilation**

- All completed sections are compiled together to generate the final report.
- The final output is a comprehensive and structured report in the style of Wiki docs.

Let’s now start building out these components step-by-step with LangGraph and Tavily.

Hands-on Implementation

We will now implement the end-to-end workflow for our Deep Research Report Generator Agentic AI System based on the architecture we discussed in detail in the previous section step-by-step with detailed explanations, code and outputs.

Install Dependencies

We start by installing the necessary dependencies which are going to be the libraries we will be using to build our system. This includes langchain, LangGraph and also rich for generating nice markdown reports.

```
!pip install langchain==0.3.14
!pip install langchain-openai==0.3.0
!pip install langchain-community==0.3.14
!pip install langgraph==0.2.64
!pip install rich
```

Enter Open AI API Key

We enter our Open AI key using the getpass() function so we don’t accidentally expose our key in the code.

```
from getpass import getpass
OPENAI_KEY = getpass('Enter Open AI API Key: ')
```

Enter Tavily Search API Key

We enter our Tavily Search key using the getpass() function so we don’t accidentally expose our key in the code. You can get the key from here and they have a generous free tier.

```
TAVILY_API_KEY = getpass('Enter Tavily Search API Key: ')
```

For more information, kindly visit this article

Advanced AI Agents Generative AI LLMs

Build a Deep Research Agent: $1 Alternative to $200 OpenAI's Tool

Build your own Deep Research & Report Generation Agent for under $1! A hands-on guide to creating an OpenAI Deep Research alternative.

Dipanjan (DJ) Sarkar 26 Mar, 2025