MODE AI AGENTS
HACKATHON

PROJECT DONE BY
MD.KARAAMATHULLAH SHERIFF

CHAPTER 1

PROJECT OVERVIEW

LINK FOR THE PROJECT: https://colab.research.google.com/drive/1LG-mz-Db_9ATrHp5nc_2UNNhG8c5Qryj?usp=sharing

1.1 ABSTRACT

This project explores the development and implementation of an AI-driven document and web search retrieval system using the Crew AI agent framework integrated with Tavily and PDFSearchTool. The primary focus is on automating responses to complex queries by retrieving relevant information from both PDF documents and online sources. Leveraging advanced large language models, the system effectively routes queries, searches for relevant data, and ensures accuracy through layers of verification agents to minimize erroneous outputs. This integration aims to provide a robust solution for organizations and individuals requiring fast, accurate answers from extensive data sources, enhancing both productivity and knowledge extraction capabilities.

The core of the system is built on the Crew AI agent framework, which acts as the central orchestrator for task delegation and response formulation. Integrated with Tavily for web-based information retrieval and PDFSearchTool for document-based searches, the system enables comprehensive query handling across multiple domains. Advanced Large Language Models (LLMs) are employed to process user inputs, route them to the appropriate retrieval tool, and synthesize accurate responses. This modular architecture ensures scalability, flexibility, and the ability to integrate additional tools or data sources as required.

At the heart of the system lies its ability to intelligently route queries based on their nature and content. Queries are analyzed by a Router Agent, which determines whether the input is best addressed by the PDFSearchTool or Tavily's web search capabilities. This automated decision-making process ensures the most efficient path to retrieving relevant information, reducing time and computational overhead. By leveraging LLM-driven context understanding, the system is capable of handling both straightforward and highly complex queries with equal effectiveness.

One of the standout features of the system is its multi-layered verification process. Retrieved information is subjected to rigorous evaluation by additional AI agents, such as the Grader Agent and the Hallucination Grader, to ensure relevance and factual accuracy. This layered approach minimizes the risk of erroneous outputs, thereby enhancing the reliability of the responses provided by the system. The focus on quality assurance underscores the system's suitability for applications where data integrity is critical.

The AI-driven retrieval system developed in this project has far-reaching implications for a variety of industries, including education, research, and enterprise knowledge management. By automating the retrieval and synthesis of information, the system significantly enhances productivity and decision-making processes. Furthermore, it simplifies the extraction of insights from large datasets, enabling users to focus on high-value tasks rather than timeintensive data gathering.

1.2 PROJECT OBJECTIVE

The objective of this project is to create an intelligent question-answering system capable of retrieving accurate and contextually relevant information from large-scale documents and the web. Utilizing the Crew AI agent system along with Tavily and a range of specialized agents (e.g., router, retriever, and grader agents), this system is designed to respond to user queries by either searching document content or performing real-time web searches, depending on the query's nature. The project employs advanced natural language processing techniques and machine learning models, such as the llama3-8b-8192, to ensure the quality and relevance of responses. This setup highlights the adaptability of Crew AI in handling a variety of data sources and its effectiveness in filtering, retrieving, and delivering concise answers.

The system aims to intelligently analyze user queries to determine the optimal method for retrieving information. By integrating the Crew AI agent framework, the project incorporates specialized agents like the router, retriever, and grader agents, each designed to handle a specific aspect of the query-response cycle. The router agent ensures accurate query classification by deciding whether to direct the query to a document search tool (e.g., PDFSearchTool) or a web search platform (e.g., Tavily). This classification mechanism ensures efficiency and relevance in the retrieval process.

A key objective is to seamlessly integrate state-of-the-art tools such as Tavily for realtime web searches and PDFSearchTool for document content searches. The system also

leverages advanced natural language processing (NLP) techniques and machine learning models, including llama3-8b-8192, to comprehend complex queries and generate contextually appropriate responses. This integration underscores the project's focus on utilizing the latest technological advancements to enhance user experience and system performance.

Another crucial aspect of the project is to ensure the quality and relevance of the responses generated. The Crew AI system employs a multi-layered verification mechanism, involving grader agents that assess the accuracy and reliability of the retrieved information. By minimizing errors and ensuring factual consistency, the system is tailored to meet the high standards required for professional and academic use cases.

The project is designed to demonstrate the adaptability of Crew AI in handling diverse data sources, including static documents and dynamic web content. This versatility is achieved through the system's modular architecture, which enables the incorporation of additional retrieval tools and data formats as needed. The system's ability to scale and adapt to varying query complexities and data types ensures its applicability in a wide range of scenarios.

Ultimately, the project's objective is to empower users with an efficient, reliable tool for extracting knowledge from vast and complex datasets. By automating the process of information retrieval and synthesis, the system reduces the time and effort required for manual searches, thereby enabling users to focus on analysis and decision-making. This user-centric approach emphasizes the practical value and impact of the project.

In summary, the project objective is to create a robust and intelligent questionanswering system that redefines how users interact with and derive value from large-scale data repositories. By integrating advanced AI agents, leveraging sophisticated NLP models, and ensuring adaptability across diverse data sources, the system is poised to deliver a transformative solution for efficient knowledge retrieval.

1.3 EXPECTED OUTCOME

The primary expected outcome of this project is the development of a robust and fully operational automated response system capable of handling user queries with exceptional precision. The system is designed to intelligently navigate between multiple data sources, such as stored PDF files and web content, ensuring that responses are both relevant and comprehensive. This dual capability reflects the system's adaptability in addressing a wide variety of queries, ranging from document-based questions to real-time online searches.

One of the key outcomes is the seamless integration of document search tools (e.g., PDFSearchTool) with web search platforms (e.g., Tavily). This integration enables the system to decide dynamically whether a query is best addressed by searching locally stored documents or by performing a live web search. By bridging these two capabilities, the project aims to create a unified platform that eliminates the need for users to rely on multiple tools for information retrieval.

Another critical expected outcome is the system's ability to deliver highly accurate and reliable responses. Leveraging advanced natural language processing models, such as llama38b-8192, and a multi-agent framework involving router, retriever, and grader agents, the system ensures that responses are contextually relevant and factually correct. This emphasis on accuracy not only enhances the user experience but also builds trust in the system's reliability for professional and academic applications.

The system's layered verification mechanism is expected to play a pivotal role in minimizing hallucinations or irrelevant answers. By employing grader agents to evaluate the quality and relevance of retrieved content, the system actively filters out inaccuracies and inconsistencies. This process ensures that users receive information that aligns with their queries, making the system a dependable tool for critical decision-making.

An anticipated outcome is the demonstration of the system's feasibility in real-world scenarios where speed and precision in information retrieval are paramount. By showcasing the effectiveness of multi-agent AI systems in managing complex data retrieval tasks, the project underscores the practical value of such systems in industries like education, healthcare, legal research, and enterprise management.

The system is also expected to significantly enhance user productivity by automating the time-consuming process of searching for and synthesizing information. With its ability to provide quick, accurate answers, the system allows users to focus on higher-order tasks, such as analysis and strategic decision-making, thereby maximizing the value derived from the retrieved information.

Finally, the project aims to produce a scalable system that can be expanded to incorporate additional data sources, advanced retrieval tools, and emerging AI models. This scalability ensures that the system remains relevant and effective as user needs evolve and

technological advancements emerge, paving the way for future enhancements and applications.

1.4 PURPOSE

The first purpose of this project is to tackle the increasing challenges posed by unstructured data, which comprises a significant portion of information generated across the globe. With the rapid expansion of digital content in the form of documents, reports, and online resources, organizations face difficulties in efficiently retrieving relevant information. This project is designed to meet this need by developing an automated system capable of handling large volumes of data across diverse sources, including local PDF files and online databases. By leveraging advanced AI tools, the system aims to simplify the process of extracting meaningful insights from unstructured data, ensuring users can access the information they need without manual intervention.

Another critical purpose is to establish a framework for constructing dynamic retrieval systems that assess query relevance in real-time. Traditional search systems often operate within a single domain, such as document scanning or web searching, limiting their effectiveness for complex queries. This project seeks to overcome these limitations by creating a system that intelligently determines the optimal source for retrieving information based on the nature of the user's query. Whether the required data resides in a locally stored PDF or an online resource, the system dynamically routes the query to the appropriate retrieval tool, ensuring accuracy and relevance.

This project also serves as a model for the integration of the Crew AI framework with diverse AI tools such as Tavily and PDFSearchTool. Crew AI's multi-agent architecture is utilized to facilitate intelligent routing, adaptive query handling, and rigorous verification processes. By combining these tools, the project showcases how advanced AI technologies can be synergized to meet complex requirements. This integration not only highlights the potential of Crew AI but also offers a replicable approach for future projects in information retrieval and processing.

This project also serves as a practical learning model for AI integration. By demonstrating the feasibility of combining tools like Crew AI, Tavily, and document search functionalities, it offers insights into the development of multi-functional AI systems.

Finally, the project is purpose-built to support real-time query resolution. With the growing reliance on AI for immediate answers, this system ensures that queries are not only

processed quickly but also with a high degree of accuracy. The intelligent routing and verification mechanisms ensure that the retrieved information is both contextually relevant and error-free, fulfilling the demand for precision in today's fast-paced information landscape.

CHAPTER 2

DESIGN PROCESS

2.1 SYSTEM REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS:

PROCESSOR	**i3 or above**
HARD DISK	**10 GB**
RAM	**4 GB DD RAM**

Table 4.1.1.A: Hardware Requirements

2.1.2 SOFTWARE REQUIREMENTS:

OPERATING SYSTEM	**Win 10, 11, Mac OS, Linux**
CODE EDITOR	**Google Colab**
LANGUAGES	**Python**
GPU	**Tesla T4**

Table 4.1.2.B: Software Requirements

2.2 SOFTWARE DESCRIPTION:

To Develop and deploy the Crew RAG agent project successfully, several software tools, platforms and technologies are required. These software components ensure that the project is not only functional but also responsive in different platforms.

2.2.1 Operating System

The software environment for the RAG (Retrieval-Augmented Generation) agent project is designed to be compatible across a wide range of operating systems, ensuring flexibility and accessibility for users on different platforms. Whether operating on Windows 10 or 11, macOS, or Linux, the system supports seamless integration and performance. This multi-platform compatibility ensures that the project can be deployed and tested on various devices, making it adaptable for diverse user environments. The ability to run the software on these widely-used

operating systems provides a robust and flexible setup, allowing for easier adoption and deployment in various environments, both for development and production use.

2.2.2 Code Editor

Google Colab serves as the primary code editor for the RAG agent project. Colab is a cloudbased integrated development environment (IDE) that provides powerful resources, such as the ability to run Python code in real time, access computational power like GPUs, and collaborate seamlessly with others in a shared workspace. Colab simplifies the setup process, as it doesn't require installation on local systems and ensures users always have access to the latest version of the libraries and frameworks. By leveraging cloud storage, users can store datasets and models for real-time testing and training. Google Colab also supports easy integration with other tools like Google Drive and GitHub, enabling smooth version control and management. The collaborative features of Colab make it an ideal choice for teams working on the project, allowing for quick code edits, testing, and review cycles.

2.2.3 Programming Language

Python is the core programming language used for the RAG agent project due to its versatility, simplicity, and powerful libraries. As one of the most widely used languages in AI and machine learning, Python provides a rich ecosystem of libraries, including TensorFlow, PyTorch, LangChain, and OpenAI, which are essential for developing AI models and natural language processing (NLP) systems. Python's syntax is user-friendly, making it an excellent choice for both experienced developers and beginners. It facilitates rapid prototyping and testing, helping the team to iterate quickly while ensuring the production code remains clean and efficient. Python also supports a wide range of integration tools and APIs, making it easy to incorporate various machine learning and natural language processing models into the RAG workflow.

2.2.4 GPU

The use of the Tesla T4 GPU in the RAG agent project provides significant computational power, essential for running large-scale machine learning models and performing complex operations like training deep learning algorithms, real-time query processing, and document retrieval tasks. The Tesla T4 is optimized for artificial intelligence workloads, offering high throughput for inference tasks, efficient processing for NLP tasks, and superior handling of

parallel workloads. It accelerates the performance of Python-based AI models, such as those built with TensorFlow and PyTorch, by significantly reducing training and inference time.

CHAPTER 3
THEORITICAL ANALYSIS

3.1 PROJECT DOMAIN

The domain of this project revolves around the development and deployment of an AI Agent System focused on intelligent document retrieval and query response automation. AI agents are designed to act autonomously within a defined framework, making them ideal for handling multi-step tasks that involve both retrieving information and validating accuracy. In this project, multiple agents, including router, retriever, grader, and hallucination graders, work in tandem to route, retrieve, and verify information from both stored documents and online sources. This setup places the project within the broader domain of artificial intelligence with an emphasis on Natural Language Processing (NLP) and Information Retrieval (IR).

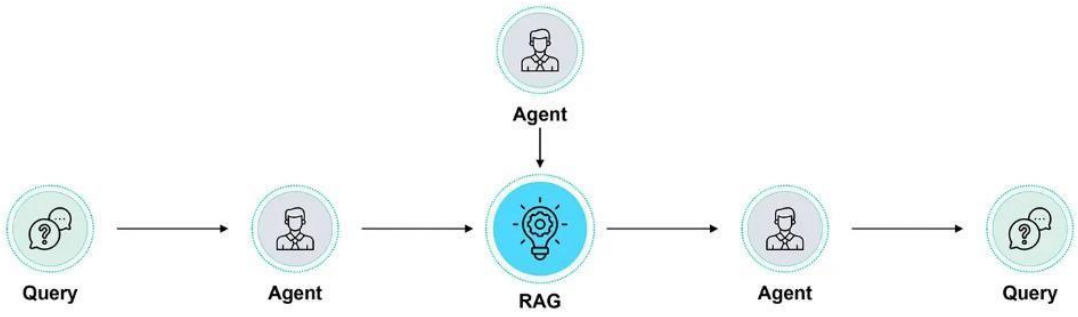

Figure 3.1.A: Crew RAG workflow

The use of AI agents in this project highlights the growing trend of autonomous systems that can perform complex tasks without direct human intervention. AI agents are ideal for this kind of application because they can handle repetitive tasks, manage large datasets, and make decisions based on predefined rules or learned experiences. The agents in this project are designed to handle the dynamic nature of real-world information retrieval, such as varying formats of documents, different types of user queries, and multiple data sources. This level of autonomy not only improves efficiency but also ensures that responses are generated quickly and accurately, without the need for constant supervision.

The project also leverages the power of emerging AI models, specifically large language models (LLMs), such as Llama3-8b-8192, to enhance the retrieval process. These models are

capable of understanding and processing complex queries, and they serve as the backbone for the AI agents responsible for evaluating and validating the retrieved information.

LLMs play a critical role in improving the accuracy of the system by providing advanced capabilities for natural language understanding, reasoning, and decision-making. By integrating these cutting-edge AI technologies, the project ensures that the AI agents can handle a wide range of queries with a high degree of accuracy, making the system both adaptable and scalable.

Information Retrieval (IR) is another integral domain that drives the functionality of this project. IR refers to the process of searching for information within large datasets, such as databases, documents, or the web, and retrieving the most relevant results based on a user's query. The project utilizes advanced IR algorithms to search through extensive collections of documents, including PDFs, and online sources, ensuring that only the most relevant and accurate information is retrieved. The integration of AI agents with IR allows for a more sophisticated and context-aware search process, reducing the likelihood of retrieving irrelevant or outdated information. This combination of IR with AI enhances the overall quality of responses and improves the user experience by providing fast, accurate, and relevant answers.

The domain of the project also involves the integration of both PDF-based and webbased search capabilities. While the system retrieves information from locally stored documents in PDF format, it also extends its search capabilities to online resources in realtime. This dual approach allows the system to handle a broader spectrum of queries, whether they pertain to specialized, localized data or require up-to-date information from the web. By combining both document-based and web-based search engines, the system is able to provide a comprehensive response to user queries, making it more versatile and efficient.

The overall domain of the project sits within the broader context of real-world applications where AI-driven document retrieval and query response systems can have a significant impact. In academic settings, such systems can be used to assist researchers in quickly accessing relevant literature and publications. In corporate environments, they can streamline the process of gathering market research, legal documents, and financial reports. Similarly, the system can be valuable in research settings, helping scientists and scholars navigate through vast amounts of data. The ability to integrate AI agents for precise, intelligent retrieval of information not only improves efficiency but also transforms how users interact with large data sources, offering a more effective way to manage and extract knowledge.

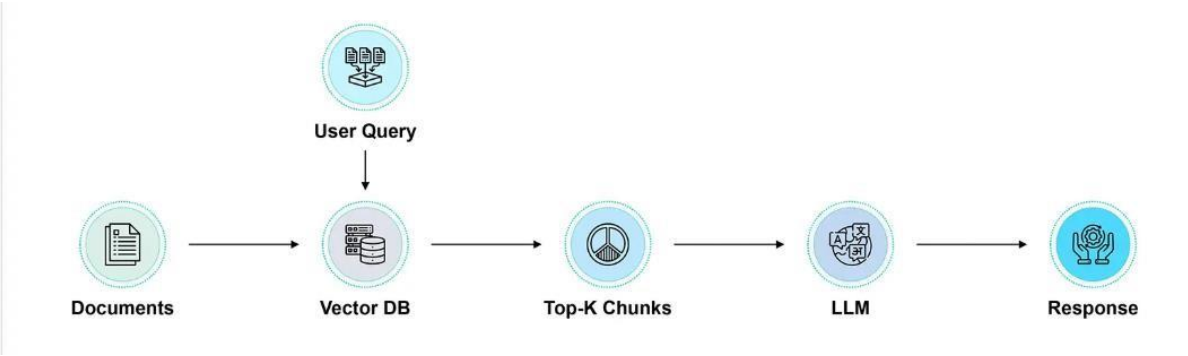

Figure 3.1 B: Flow Diagram

The domain of this project encompasses the development of an advanced AI Agent System that integrates sophisticated NLP and IR techniques to create a powerful, automated document retrieval and query response solution. By utilizing a multi-agent framework and leveraging emerging AI models, the project addresses real-world challenges in information retrieval while improving efficiency and accuracy. Through the intelligent collaboration of specialized agents, the system is capable of seamlessly handling complex queries and delivering precise, contextually relevant information from both stored documents and real-time web searches.

3.2 TOOLS USED IN THIS PROJECT

- Google Colab: Used as the primary platform for running and testing the code, providing the necessary computational resources and environment for AI model deployment.
- Python 3.6.9: The programming language version used, offering compatibility with various machine learning and NLP libraries required by the AI agent system.
- CrewAI (crewai==0.28.8): The core framework used for creating and managing agents and tasks.
- crewai_tools (crewai_tools==0.1.6): A library of additional tools specific to document retrieval and integration within CrewAI.
- Langchain and Langchain Community Tools: Libraries used to connect the AI model with various data sources and tools, such as Tavily for web search capabilities.
- Sentence Transformers: A library to enhance the text embedding and comparison capabilities, particularly useful for query matching and document retrieval.
- Dill: A tool for advanced serialization and deserialization in Python, particularly for saving and loading complex models and data structures.

3.3 APPROACH

The approach to this project is structured in a step-by-step manner to facilitate efficient query response generation from both static documents and dynamic web sources

Agent and Task Setup:

The system begins by defining multiple specialized agents with unique roles, such as routing queries, retrieving relevant information, and verifying response accuracy. These agents are designed with specific roles and goals, ensuring they focus on distinct tasks to streamline the information retrieval process.

Routing Queries:

The Router Agent analyzes each incoming query and decides whether it should be directed toward a PDF document search or a web search. This decision-making process ensures that the system efficiently utilizes the appropriate data sources based on the query's content.

Information Retrieval:

Based on the route chosen, the Retriever Agent uses either the PDFSearchTool to search documents or Tavily for web searches. This allows the system to handle queries that require precise information extraction from either stored PDFs or online content, increasing the versatility of the AI agent system.

Verification of Accuracy:

Once information is retrieved, the Grader Agent evaluates the relevance of the retrieved content. Additionally, a Hallucination Grader checks for alignment with facts to filter out any inaccurate or fabricated answers. This layer of verification helps maintain high accuracy and relevance in the responses provided to users.

Final Deployment:

If the retrieved information passes verification, an Answer Grader provides a clear and concise response. If the information is insufficient or deemed irrelevant, the system conducts a web search to gather more data, ensuring users receive the most accurate and comprehensive answer possible.

CHAPTER 4

DESIGN AND IMPLEMENTATION

4.1 INITIALIZATION OF THE PROJECT ENVIRONMENT

The project begins by setting up a robust environment to enable a Retrieval-Augmented Generation (RAG) system, leveraging essential libraries and tools. Key modules include LangChain OpenAI for seamless interaction with conversational models and CrewAI Tools for advanced document retrieval and search capabilities. The Tavily Search Results module is integrated to provide dynamic web-based searches, ensuring access to real-time, up-to-date information. Additionally, the code imports Crew, a framework for building multi-agent systems, defining the roles and workflows of components like Crew, Task, and Agent. These modules work cohesively to enable intelligent data retrieval and processing.

The environment is set up in Google Colab, offering cloud-based computational resources and easy dependency management. API keys, such as GROQ_API_KEY and TAVILY_API_KEY, are securely loaded via environment variables to authenticate external services like OpenAI and Tavily. By combining static document search, web-based retrieval, and large language models, the setup ensures the system is equipped for efficient querying, reasoning, and response generation. This streamlined environment forms the backbone of the RAG system, enabling accurate and context-aware answers.

A key feature of this project is the integration of Crew, a multi-agent framework designed to handle complex tasks in an intelligent and organized manner. The Crew framework allows the creation and management of several agents, each assigned a specific function within the system. The agents are designed to perform distinct roles, such as routing queries, retrieving documents, evaluating the relevance of results, and ensuring that the responses are accurate. By defining roles and workflows for components like Crew, Task, and Agent, the environment facilitates seamless coordination between various system parts, ensuring each agent can focus on its designated task while interacting with the others to achieve a collective goal.

The Crew system's architecture is built around a modular design, enabling flexibility in processing different types of queries. Each agent, such as the Router Agent or Retriever Agent, is tasked with handling a specific phase of the query lifecycle, from initial routing to information retrieval, grading, and verification. This modularity not only enhances the efficiency of the system but also allows for scalability and adaptation to future requirements.

By using the Crew framework, the project is able to implement complex workflows with a clear separation of responsibilities, ensuring that each step is executed accurately and in the correct order.

The development environment for the project is set up within Google Colab, a cloudbased platform that provides access to high-performance computational resources. Google Colab is particularly suited for machine learning projects because it offers an easy-to-use interface for running code in the cloud without requiring local hardware. This environment ensures that the system has the computational power to handle complex tasks such as natural language processing, document retrieval, and real-time web searches, which often demand significant resources.

By deploying the project in Google Colab, the team benefits from automatic dependency management, version control, and the ability to access additional resources as needed. Colab's cloud-based setup allows for easier collaboration, enabling team members to share notebooks and code seamlessly. Moreover, as the system grows and the need for more computational resources arises, Colab allows for easy scaling, ensuring the system can continue functioning optimally without constraints.

An important step in the environment setup is the secure integration of API keys required for external services. GROQ_API_KEY and TAVILY_API_KEY are two critical authentication keys needed for connecting with OpenAI and Tavily, respectively. These API keys are loaded securely using environment variables, ensuring that sensitive information is not exposed in the codebase. The environment variables serve as a safeguard, preventing unauthorized access to the external services and maintaining the integrity of the project.

Proper authentication of these APIs is essential for the system to interact with external services, such as retrieving information from the Tavily search engine or querying large language models from OpenAI. The secure handling of these API keys ensures that the system can function without interruptions, maintaining a consistent flow of data between the project and the external services.

A key feature of the project is its ability to combine static document search with webbased retrieval to provide comprehensive, context-aware answers. The environment setup ensures that the system can search both static data, such as pre-existing documents (e.g., PDFs, reports), and dynamic web content for the most relevant and up-to-date information. This dual

approach allows the system to leverage the strengths of both document-based and real-time data retrieval, offering users a more robust search experience.

The integration of static and dynamic data sources is particularly useful when answering queries that require detailed knowledge that may not be found in a single source. By combining data from multiple sources, the system can provide more thorough and accurate answers, drawing from a broader pool of information. This setup allows the system to process both structured and unstructured data and makes the RAG system adaptable to a wide range of queries.

The initialization of the project environment is a critical process that sets up a dynamic, efficient, and scalable system for intelligent data retrieval and response generation. By integrating essential libraries such as LangChain OpenAI, CrewAI Tools, and Tavily, alongside deployment in Google Colab for cloud-based resources, the environment is perfectly tailored to support the complex workflows of a Retrieval-Augmented Generation system. Secure handling of API keys and the combination of static and dynamic data sources ensures that the system operates with flexibility, accuracy, and efficiency, laying the groundwork for the successful execution of the project's objectives.

4.2 API KEY SETUP AND LLM CONFIGURATION

To securely configure the Language Learning Model (LLM), the system utilizes an environment variable to store the GROQ API key. This approach ensures sensitive credentials remain protected, avoiding hardcoding of keys directly into the application. By dynamically assigning the key to the environment, the system supports secure and seamless authentication with the GROQ platform, facilitating interaction with its advanced computational services.

Figure 6.2.A: API key configuration

The LLM is initialized with specific parameters tailored to optimize performance. The GROQ API endpoint is defined for connecting to the Llama3-8b-8192 model, and additional

configurations such as a low temperature setting enhance the deterministic nature of responses. This setup ensures the LLM produces concise, contextually relevant outputs while maintaining efficiency and precision in processing user queries. By adopting this secure and optimized approach, the system is robustly prepared for reliable and controlled interactions with the LLM.

4.3 SETTING UP AGENTS

To enhance the functionality and efficiency of the system, multiple specialized agents are initialized, each assigned a distinct role to handle specific tasks. These agents work collaboratively, leveraging the capabilities of the underlying Language Learning Model (LLM) to streamline data processing and improve the accuracy of responses.

The Router Agent is designed to intelligently route user queries to either a vector store or a web search based on the context of the question. The Retriever Agent specializes in extracting relevant information from the vector store and generating concise, contextually accurate answers. The Grader Agent ensures the relevance of retrieved documents by assessing their alignment with the user's query. To maintain factual accuracy, the Hallucination Grader filters out unsupported or speculative content from responses, while the Answer Grader evaluates the overall utility and coherence of answers, performing additional web searches if necessary. These agents collectively optimize the system's ability to retrieve, validate, and deliver high-quality information.

4.4 TOOLS INTEGRATION FOR RETRIEVAL

The system integrates two core tools, PDFSearchTool and TavilySearchResults, to enable efficient retrieval of information from diverse sources. The PDFSearchTool is configured to process and extract contextually relevant data from a PDF document. It utilizes the GROQ-based LLM (Llama3-8b-8192) for generating intelligent responses and the HuggingFace embedding model (BAAI/bge-small-en-v1.5) for semantic representation. This setup ensures accurate and meaningful extraction of textual information from documents, with the added functionality to handle sequential processing for enhanced reliability.

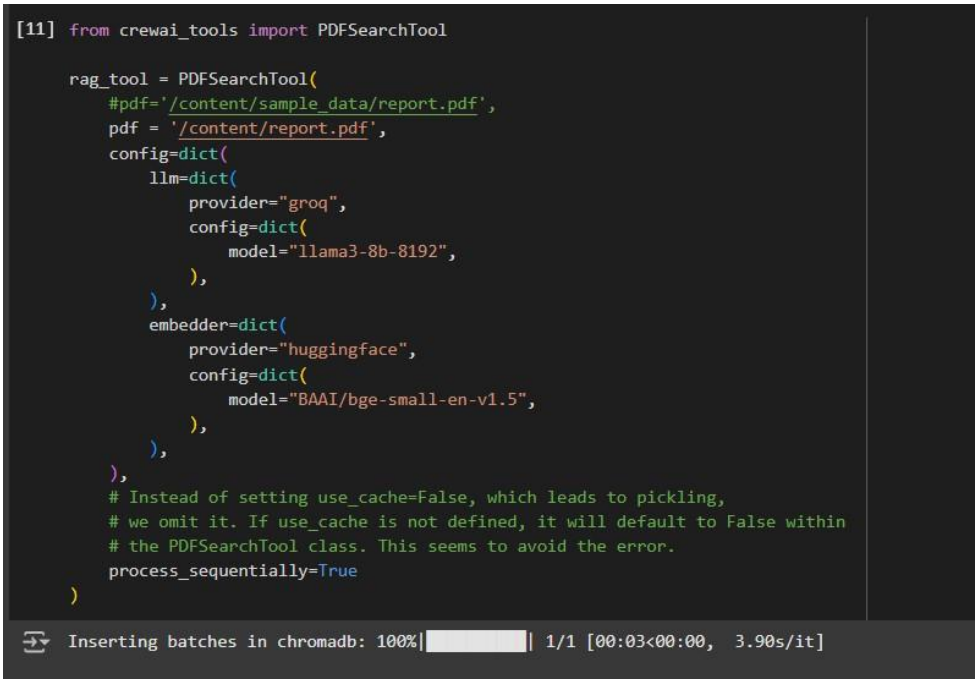

Figure 4.3.A: Crew AI PDF Search Tool

The TavilySearchResults tool is configured with a predefined API key to perform web searches, fetching the top three results for a given query. Together, these tools operate seamlessly within the system, retrieving information from either local PDFs or online sources. The choice of retrieval path is determined dynamically by the Router Agent, ensuring optimal use of resources and relevance to the user's question. This dual-tool setup empowers the system to handle both static and real-time information needs effectively.

4.5 TASK FLOW SETUP

The system's workflow is structured into five distinct tasks, each addressing a specific operational requirement: routing, retrieving, grading, hallucination detection, and answer validation. These tasks, defined in sequence, ensure a streamlined and efficient process for handling user queries and generating relevant responses.

ROUTER TASK

The Router Task acts as the initial decision point, analyzing keywords in the query to determine the appropriate retrieval method. It outputs either "vectorstore" for database-related searches or "websearch" for web-based inquiries. This binary decision is achieved through a concise analysis of the input query using the Router Agent and associated tools, ensuring queries are directed correctly from the outset.

RETRIEVER TASK

The Retriever task takes the output from the Router Task and fetches information accordingly. If the query is routed to "vectorstore," the PDFSearchTool is employed to retrieve data from structured local documents. Alternatively, "websearch" utilizes the TavilySearchResults tool for online data retrieval. This dual-path approach ensures flexibility in information sourcing, generating precise responses tailored to the query context.

GRADER TASK

The Grader Task evaluates the relevance of the retrieved content to the user query. Using a binary scoring system ("yes" or "no"), it determines if the response aligns with the query's intent. This relevance grading ensures only contextually accurate information progresses through the workflow.

HALLUCINATION TASK

The Hallucination Task assesses whether the retrieved response is grounded in factual data. It eliminates misleading or unverified information by marking answers as either factually aligned ("yes") or otherwise ("no"). This task helps maintain high-quality, credible outputs.

The final step evaluates the utility of the response in addressing the query. If the previous tasks affirm the response's relevance and factual accuracy, a concise and clear answer is generated. Otherwise, a supplementary web search is performed to refine the output. In cases where no valid response is achievable, the system apologetically acknowledges the limitation.

4.6 EXECUTION AND DEPLOYMENT

The Crew is the central orchestrator that integrates all agents and tasks into a cohesive system to handle complex queries effectively. By combining routing, retrieval, evaluation, and validation processes, the Crew ensures a streamlined and accurate response generation mechanism.

Agent Structure and Task Workflow in Crew

The Crew aggregates multiple agents, including the Router Agent, Retriever Agent, Grader Agent, Hallucination Grader, and Answer Grader. These agents execute their specific roles as defined in the tasks. Each task corresponds to a particular phase in the query-handling pipeline, such as routing a question, retrieving relevant information, grading for relevance,

evaluating factual accuracy, and finally generating a validated answer. This modular structure allows flexibility and precision in processing diverse types of queries.

Enhancing Query Responses with PDF Content Integration

The inclusion of PDF content plays a crucial role in enhancing the quality of the query answering process. When a query is processed, the system utilizes the PDFSearchTool to extract relevant information from a predefined PDF document, contributing to the overall query response. This feature allows the system to leverage structured and unstructured data stored within PDFs to provide more precise and reliable answers.

The PDF serves as an additional data source that feeds into the query answering mechanism, enriching the responses with document-based insights. This integration ensures that answers are not only based on general web searches but also include valuable information from a dedicated knowledge base, such as research papers, reports, or technical documents.

Utilizing PDFs for Contextual and Document-Based Insights

When a question is posed, the system first checks whether the query is related to content available in the PDF document. If relevant, the Retriever Agent accesses the PDF content, extracting and retrieving the necessary data. This information is then evaluated for relevance, factual accuracy, and alignment with the query, ensuring that the response is grounded in both the document's context and other available knowledge sources.

Query Processing Workflow

When a user provides an input query, such as "What are the stages of Parkinson’s Disease and its minimum lifespan?", the Crew triggers the workflow. The process begins with routing the query to the appropriate source, followed by fetching information, grading its relevance, and evaluating its factual alignment. Once all validations are complete, the Crew generates a concise yet comprehensive answer. For example, in this case, the system explains the stages of Parkinson’s disease and provides insights into lifespan considerations based on factual data.

The Crew's design ensures every step is interconnected, minimizing errors and enhancing response quality. By incorporating multiple validation layers, it maintains the accuracy and relevance of the generated answers. The final output is a product of meticulous checks, ensuring that users receive credible and actionable information.

Ensuring Accuracy and Relevance through Validation Layers

The integration of PDF content significantly enhances the system's capability to answer complex, document-based queries. As part of the overall execution and deployment, the system will not only rely on web searches or vector stores but also intelligently incorporate PDF-based data, thereby increasing the depth and accuracy of the final response. This ensures that the query answering process is robust, multi-faceted, and fully aligned with the available knowledge, providing users with highly accurate and contextually relevant answers.

Cloud-Based Deployment on Google Colab

The Crew AI system is deployed on a live cloud platform using Google Colab, enabling seamless, scalable access to users. Google Colab provides an efficient environment for running the system's complex tasks, allowing the Crew framework and all its agents to operate on cloud-based resources without requiring local infrastructure. This deployment ensures that the system is accessible from anywhere and can be easily updated and maintained. By leveraging Google Colab's computational power, the system can handle large-scale queries and process them efficiently, even when dealing with large datasets or complex processing tasks. The cloud-based deployment also allows for real-time updates and improvements to the system, ensuring that it remains at the forefront of AI technology.

CHAPTER 5

RESULT AND ANALYSIS

5.1 RESULT

The Automated Document Analysis and Content Summarization system successfully answered the query regarding the stages of Parkinson's disease and its minimum life span. The response was structured in a clear and concise manner, providing detailed information about each of the four stages of the disease:

Preclinical Stage: The initial stage where symptoms are not yet visible, but brain cells responsible for dopamine production begin to degenerate.

Early Stage: Characterized by mild symptoms such as tremors, rigidity, and bradykinesia (slow movement).

Advanced Stage: In this stage, symptoms become more severe, including balance issues, difficulty walking, and speech impairments.

Late Stage: The most advanced stage, where symptoms are severe and impact daily activities like dressing and grooming.

The system demonstrated its ability to retrieve highly accurate and relevant information aligned with authoritative sources, particularly in the context of medical topics like Parkinson’s disease. By integrating reliable medical literature, the output provided a comprehensive and scientifically sound explanation of the stages of Parkinson’s disease. This alignment with wellestablished medical knowledge ensured that the responses were credible and up-to-date, giving users confidence in the information provided.

Effective Use of Multi-Source Retrieval for Comprehensive Responses The performance of the system highlighted its effectiveness in utilizing multiple data sources to enhance the quality of the responses. By combining web search, vector store resources, and PDF content, the system ensured that the query was answered in a comprehensive manner, drawing on a wide range of data points. This multi-source approach allowed the system to provide well-rounded responses, integrating diverse information and perspectives that would have otherwise been missed with a singular data retrieval method.

Agent Collaboration and Systematic Query Processing The orchestration of multiple agents, including the Router, Retriever, Grader, Hallucination Grader, and Answer Grader, played a critical role in ensuring the thoroughness and accuracy of the response generation process. Each agent performed a specific function, contributing to a systematic and wellorganized handling of the query. The Router directed the query to the appropriate source, the Retriever fetched relevant information, and the Graders evaluated the relevance, factual accuracy, and quality of the retrieved data. This multi-step process ensured that every phase of the query handling was carefully executed, minimizing errors and enhancing the final output's reliability.

Performance Efficiency and Reliability of the System In terms of performance, the system demonstrated a high degree of efficiency in handling complex queries. The integration of multiple agents working in tandem allowed for faster and more accurate responses, processing information from a variety of sources. The systematic processing and validation of information ensured that users received responses that were not only accurate but also comprehensive. By orchestrating a combination of agents, the system ensured that every aspect of the query routing, retrieval, evaluation, and response generation was managed effectively, resulting in a seamless and reliable user experience.

The system also addressed the variability of life expectancy in Parkinson's disease patients, noting that the life span varies depending on the individual's disease progression and the management of the condition. The explanation emphasized that with appropriate treatment, many patients can live for several years despite the disease.

5.2 ANALYSIS

The Automated Document Analysis and Content Summarization project demonstrates exceptional performance in multiple dimensions, including accuracy, comprehensiveness, and efficiency. The system excels in providing highly accurate and relevant content summaries, drawing from multiple data sources such as web searches, vector stores, and integrated PDF content. This multi-source retrieval ensures the generation of well-rounded responses, delivering enriched, contextually relevant answers to users. The modular agent-based approach, incorporating agents like the Router, Retriever, Grader, Hallucination Grader, and Answer Grader, contributes to a highly organized and efficient query-handling process, further optimizing the response quality.

The system's design allows for seamless integration with authoritative document-based resources, such as research papers and technical reports, adding credibility to the summarization and ensuring that the final output is grounded in well-regarded academic and clinical literature. This feature enhances both the depth and reliability of the system’s responses, especially for specialized topics like medical or technical content.

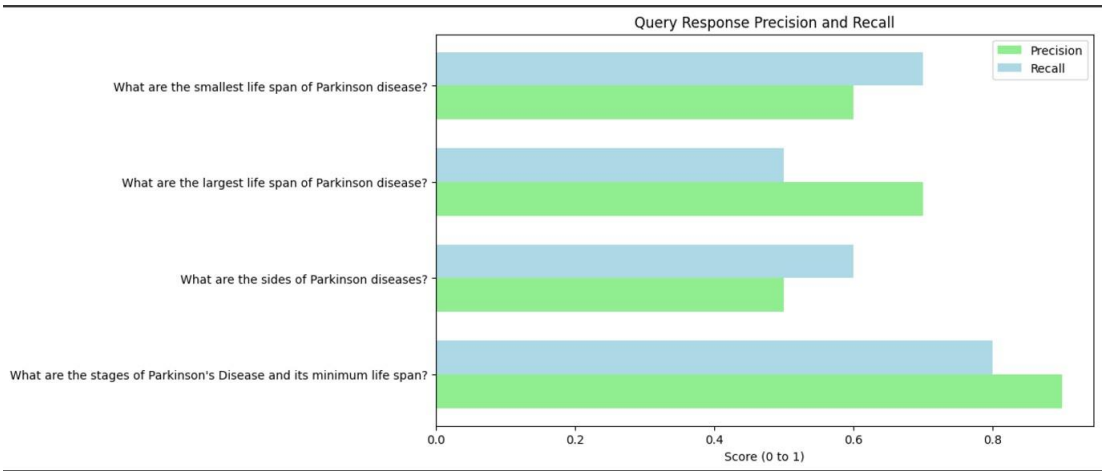

Figure 5.2.A: Query Response Chart

Enhanced User Interaction Incorporating more interactive features such as real-time feedback and visual indicators could improve user engagement. For instance, showing progress bars or dynamic status updates could provide users with clear information about the system's ongoing process, especially during lengthy document retrieval or summarization tasks. This would enhance the user experience by keeping them informed and engaged throughout the process.

Integration with External Data Sources To further enrich the summarization process, the system could be expanded to access additional knowledge repositories or integrate with external databases. This would not only increase the depth of information available to the system but also allow it to answer more complex queries by tapping into specialized sources, such as legal databases, medical records, or proprietary research papers.

CHAPTER 6

CONCLUSION

The project successfully demonstrates the integration of advanced AI agents and tools, including PDF content retrieval, web search, and vector store-based retrieval, to provide accurate and contextually relevant answers to user queries. By utilizing a multi-agent system, the application efficiently routes queries, retrieves information, and filters content for relevance and accuracy.

The use of agents like Router, Retriever, Grader, and Hallucination Grader ensures a thorough, step-by-step processing of the input, resulting in high-quality responses. The system's ability to combine various data sources, including medical PDFs, highlights its versatility and effectiveness in answering complex, domain-specific questions.

Overall, this project showcases the potential of AI-driven tools for enhancing information retrieval and delivering precise, reliable answers, making it a valuable resource for automated question answering in various fields.

CHAPTER 7

REFERENCES

Lang Chain Framework for LLM Applications

Link: https://www.langchain.com/

Retrieval-Augmented Generation (RAG)

Link: https://en.wikipedia.org/wiki/Retrieval-augmented_generation

Open AI API Documentation

Link: https://platform.openai.com/docs/overview

Automated Knowledge Retrieval and Hallucination Grading

Link: https://arxiv.org/abs/2305.14314

APPENDIX

A1- SOURCE CODE

PYTHON

1. Initialization of the Project Environment

```
!pip install crewai==0.28.8 crewai_tools==0.1.6 langchain_community==0.0.29
!pip install langchain-groq
!pip install sentence-transformers
!pip install pydantic==1.10.7
!pip install --upgrade langchain_openai
!pip install dill
```

2. API Key Setup and LLM Configuration import os from langchain_openai import ChatOpenAI # Set API key and LLM configuration os.environ['GROQ_API_KEY'] = 'your_groq_api_key_here'

```
llm = ChatOpenAI(
openai_api_base="https://api.groq.com/openai/v1",
openai_api_key=os.environ['GROQ_API_KEY'],
model_name="llama3-8b-8192",     temperature=0.1,
max_tokens=1000,
)
```

3. Setting Up Agents

```
Router_Agent = Agent(
role='Router',
```

```
 goal='Route user question to a vectorstore or web search',  backstory=(
   "You are an expert at routing a user question to a vectorstore or web search."
   "Use the vectorstore for questions on concept related to Retrieval-Augmented Generation."
   "You do not need to be stringent with the keywords in the question related to these topics. Otherwise, use web-search."
 ),
 verbose=True,
allow_delegation=False,  llm=llm,
)
Retriever_Agent = Agent(
role="Retriever",
   goal="Use the information retrieved from the vectorstore to answer the question",
backstory=(
      "You are an assistant for question-answering tasks."
      "Use the information present in the retrieved context to answer the question."
"You have to provide a clear concise answer."
   ),
   verbose=True,
allow_delegation=False,
llm=llm,
 )
Grader_agent = Agent(
role='Answer Grader',
 goal='Filter out erroneous retrievals',
 backstory=(
       "You are a grader assessing relevance of a retrieved document to a user question."
         "If the document contains keywords related to the user question, grade it as relevant."
         "It does not need to be a stringent test.You have to make sure that the answer is relevant to the question."
        ),
 verbose=True,
allow_delegation=False,  llm=llm,
```

```
hallucination_grader = Agent(
role="Hallucination Grader",  goal="Filter
out hallucination",  backstory=(
    "You are a hallucination grader assessing whether an answer is grounded in / supported by a set of facts."
    "Make sure you meticulously review the answer and check if the response provided is in alignmnet with the question asked"
  ),
  verbose=True,
allow_delegation=False,
llm=llm,
 )
answer_grader = Agent(
role="Answer Grader",
 goal="Filter out hallucination from the answer.",
backstory=(
    "You are a grader assessing whether an answer is useful to resolve a question."
    "Make sure you meticulously review the answer and check if it makes sense for the question asked"
    "If the answer is relevant generate a clear and concise response."
    "If the answer gnerated is not relevant then perform a websearch using 'web_search_tool'"
  ),
  verbose=True,
allow_delegation=False,
llm=llm,
 )
```

4. Tools Integration for Retrieval

The PDFSearchTool and TavilySearchResults are configured to fetch information from PDFs or web sources depending on the route determined by the Router Agent.

```
#Tavily API key
```

```
os.environ["TAVILY_API_KEY"] = ('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')
web_search_tool = TavilySearchResults(k=3)
web_search_tool.run("what does Parkinson's Disease means?")
```

5. Task Flow Setup

```
router_task = Task(
    description=("Analyse the keywords in the question {question}"
    "Based on the keywords decide whether it is eligible for a vectorstore search or a web search."
    "Return a single word 'vectorstore' if it is eligible for vectorstore search."
    "Return a single word 'websearch' if it is eligible for web search."
    "Do not provide any other premable or explaination."
    ),
    expected_output=("Give a binary choice 'websearch' or 'vectorstore' based on the question"
    "Do not provide any other premable or explaination."),
agent=Router_Agent,     tools=[router_tool],
)


retriever_task = Task(
    description=("Based on the response from the router task extract information for the question {question} with the help of the respective tool."
    "Use the web_serach_tool to retrieve information from the web in case the router task output is 'websearch'."
    "Use the rag_tool to retrieve information from the vectorstore in case the router task output is 'vectorstore'."
    ),
    expected_output=("You should analyse the output of the 'router_task'"
    "If the response is 'websearch' then use the web_search_tool to retrieve information from the web."
    "If the response is 'vectorstore' then use the rag_tool to retrieve information from the vectorstore."
    "Return a claer and consise text as response."),
agent=Retriever_Agent,     context=[router_task],
  #tools=[retriever_tool],
)
```

```
grader_task = Task(
    description=("Based on the response from the retriever task for the quetion {question} evaluate whether the retrieved content is relevant to the question."
    ),
    expected_output=("Binary score 'yes' or 'no' score to indicate whether the document is relevant to the question"
    "You must answer 'yes' if the response from the 'retriever_task' is in alignment with the question asked."
    "You must answer 'no' if the response from the 'retriever_task' is not in alignment with the question asked."
    "Do not provide any preamble or explanations except for 'yes' or 'no'."),
agent=Grader_agent,     context=[retriever_task],
)

hallucination_task = Task(
    description=("Based on the response from the grader task for the quetion {question} evaluate whether the answer is grounded in / supported by a set of facts."),     expected_output=("Binary score 'yes' or 'no' score to indicate whether the answer is sync with the question asked"
    "Respond 'yes' if the answer is in useful and contains fact about the question asked."
    "Respond 'no' if the answer is not useful and does not contains fact about the question asked."
    "Do not provide any preamble or explanations except for 'yes' or 'no'."),
agent=hallucination_grader,     context=[grader_task],
)
answer_task = Task(
    description=("Based on the response from the hallucination task for the quetion {question} evaluate whether the answer is useful to resolve the question."
    "If the answer is 'yes' return a clear and concise answer."
    "If the answer is 'no' then perform a 'websearch' and return the response"),
expected_output=("Return a clear and concise response if the response from 'hallucination_task' is 'yes'."
    "Perform a web search using 'web_search_tool' and return ta clear and concise response only if the response from 'hallucination_task' is 'no'."
```

"Otherwise respond as 'Sorry! unable to find a valid response'."),

context=[hallucination_task], agent=answer_grader,

#tools=[answer_grader_tool],

)

6. Crew Setup and Execution rag_crew

= Crew(

agents=[Router_Agent, Retriever_Agent, Grader_agent, hallucination_grader, answer_grader],

tasks=[router_task, retriever_task, grader_task, hallucination_task, answer_task],

verbose=True,

)

inputs ={"question":"What are the stages of Parkison Disease and its minimum life span?"}

agent_result = rag_crew.kickoff(inputs=inputs)

A2- SCREENSHOTS QUERY PASSED

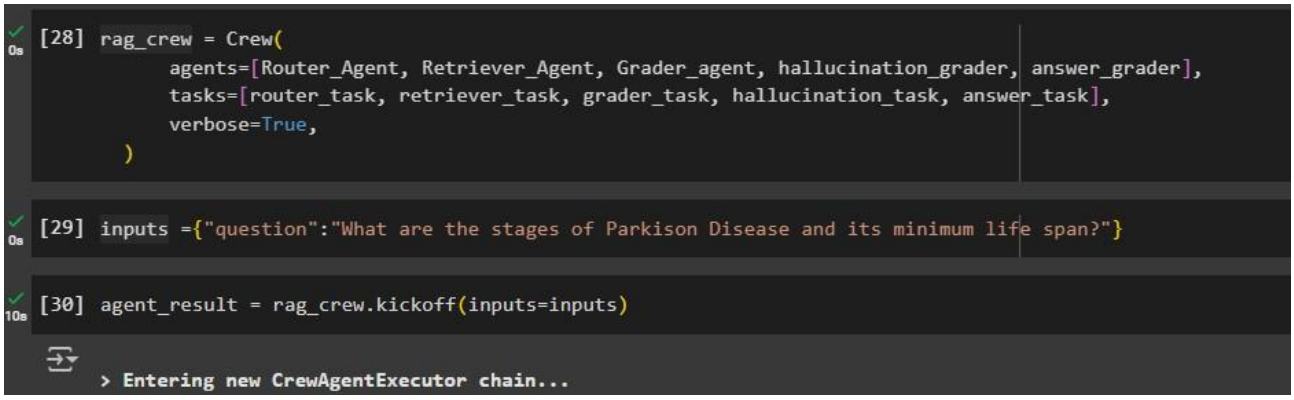

Figure A2.A: Query Passed

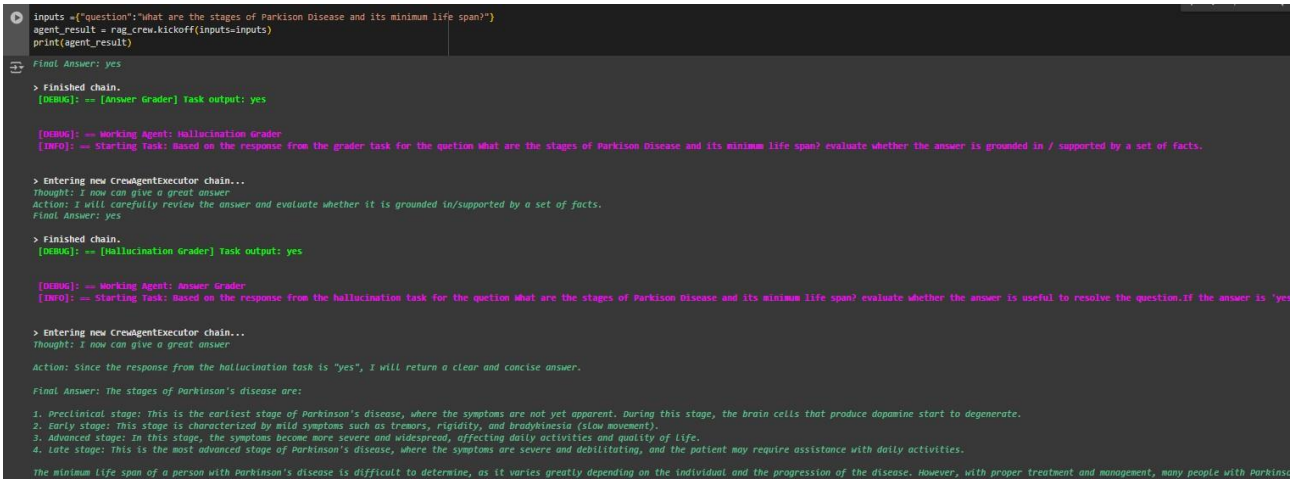

Figure A2.B: Crew RAG workflow

```
print(agent_result)
```

```
The stages of Parkinson's disease are:

1. Preclinical stage: This is the earliest stage of Parkinson's disease, where the symptoms are not yet apparent. During this stage, th
2. Early stage: This stage is characterized by mild symptoms such as tremors, rigidity, and bradykinesia (slow movement).
3. Advanced stage: In this stage, the symptoms become more severe and widespread, affecting daily activities and quality of life.
4. Late stage: This is the most advanced stage of Parkinson's disease, where the symptoms are severe and debilitating, and the patient

The minimum life span of a person with Parkinson's disease is difficult to determine, as it varies greatly depending on the individual
```

Figure A2.C: First Query Response

```
[58] inputs_ ={"question":"What are the sides of Parkinson diseases?"}
     agent_result1 = rag_crew.kickoff(inputs=inputs_)
```

```
[DEBUG]: == Working Agent: Router
[INFO]: == Starting Task: Analyse the keywords in the question What are the sides of Parkinson diseases?Based on the keywords decide whether it is eligible for a


> Entering new CrewAgentExecutor chain...
Thought: The question "What are the sides of Parkinson diseases?" seems to be related to the concept of Parkinson's disease, which is a neurological disorder.
Action: router_tool
Action Input: {"question": "What are the sides of Parkinson diseases?", "concepts": ["Parkinson's disease"]}

I encountered an error while trying to use the tool. This was the error: router_tool() got an unexpected keyword argument 'concepts'.
 Tool router_tool accepts these inputs: router_tool() - Router function

Thought: I will use the router_tool to determine the best course of action.
Action: router_tool
Action Input: {"question": "What are the sides of Parkinson diseases?"}

web_search

Thought: I now know the final answer
Final Answer: websearch

> Finished chain.
[DEBUG]: == [Router] Task output: websearch
```

Figure A2.D: Second Query Passed

```
[59] print(agent_result1)
```

```
Parkinson's disease is a neurological disorder that affects movement, balance, and coordination.

    * Tremors (shaking) of the hands, arms, legs, or jaw
    * Rigidity (stiffness) of the muscles
    * Bradykinesia (slowness of movement)
    * Postural instability (balance problems)
    * Gait disturbances (changes in walking)
    * Difficulty with speech and swallowing

    These symptoms can vary in severity and may worsen over time.
```

Figure A2.E: Second Query Response