PROJECT Report

Generative AI Agents – Automating Tasks with LLM Reasoning

[CSE3024]

AI Blockchain Tutor

By

Dharmanshu Singh

210398

Agnit Singh

210523

Department of Computer Science and Engineering
School of Engineering and Technology
BML Munjal University

November 2024

Contents

Introduction

Blockchain technology is transforming industries with its ability to provide secure, transparent, and decentralized solutions. However, understanding blockchain concepts can be challenging due to the complexity of its structure and use cases. This project aims to simplify blockchain learning through an AI-powered web application.

The web app uses **Retrieval-Augmented Generation (RAG)** and a **multi-agent system** to provide accurate, dynamic, and easy-to-understand answers to blockchain-related questions. RAG combines the power of retrieving relevant information from external sources with the capabilities of advanced language models, ensuring users receive real-time and contextually accurate responses.

The application is designed to take user inputs, search through a combination of blockchain-specific documents and online resources, and present concise answers. It employs a multi-agent workflow, where each agent performs a specific task like retrieving knowledge, answering queries, or summarizing information. By breaking down complex blockchain topics into simple explanations, this web app serves as a personalized tutor, making blockchain concepts accessible to everyone.

Background and Motivation

The rapid development of artificial intelligence, particularly in field of generative AI, has revolutionized how information is accessed and utilized across multiple fields. However, blockchain technology, despite its transformative potential in industries such as finance, supply chain, and healthcare, remains a complex subject for most individuals to understand. The steep learning curve associated with blockchain concepts, combined with the lack of easily accessible and user-friendly resources, creates a barrier for learners and professionals alike.

Recognizing these challenges, this project aims to simplify blockchain education through an innovative AI-powered chatbot. By combining **Retrieval-Augmented Generation (RAG)** and a multi-agent architecture, the chatbot offers personalized, dynamic, and contextually accurate responses. Unlike conventional educational tools that rely on static content, this chatbot retrieves the most relevant information in real-time and explains it in a way that is easy to understand.

This project addresses several key challenges:

1. **Accessibility**: Ensures that blockchain knowledge is readily available to users, regardless of their background or expertise.
2. **Complexity Reduction**: Simplifies intricate blockchain concepts using AI-driven explanations and summaries.
3. **Engagement**: Provides an interactive learning platform through quizzes, lessons, and real-time feedback.
4. **Relevance**: Delivers up-to-date and contextually appropriate content by integrating information from diverse sources such as PDFs, text files, and web searches.

The chatbot employs cutting-edge tools such as **CrewAI**, **Groq**, and **Cohere**, alongside retrieval and embedding techniques, to deliver seamless and impactful learning experiences. By leveraging these technologies, the system bridges the gap between blockchain expertise and users' understanding, ensuring an effective, engaging, and accessible learning journey.

Problem Statement

Understanding blockchain technology is critical in today's digital era, yet it remains a difficult task due to the following issues:

1. **High Complexity**: Blockchain concepts such as consensus mechanisms, smart contracts, and tokenization can be overwhelming for beginners.
2. **Lack of Contextual Information**: Most educational resources fail to adapt to the learner's specific needs, leading to a lack of personalized learning.
3. **Static Content**: Traditional learning materials cannot offer real-time updates or explanations, making it challenging to address evolving topics.
4. **Accessibility Gaps**: Learners from diverse backgrounds often struggle to find resources that simplify blockchain without sacrificing depth.
5. **Engagement Deficit**: A lack of interactivity and feedback in existing resources reduces interest and retention.

This project tackles these challenges by building a **RAG-powered blockchain chatbot** that provides accessible, accurate, and dynamic content. By integrating advanced AI technologies, the chatbot ensures users receive tailored explanations, interactive lessons, and real-time feedback to enhance their blockchain knowledge.

Objective

The primary goal of this project is to create a user-friendly web application that simplifies blockchain education using generative AI. Key objectives include:

1. **Providing Accurate Information**: Deliver context-aware answers by retrieving and generating blockchain-specific content.
2. **Enhancing Accessibility**: Offer a platform that caters to users of all expertise levels, from beginners to professionals.
3. **Promoting Engagement**: Include features such as lessons, quizzes, and real-time feedback to make learning interactive and enjoyable.
4. **Leveraging AI Technologies**: Use multi-agent systems and advanced frameworks to ensure seamless and efficient performance.

Deliverables

1. **AI-Powered Blockchain Chatbot**:

   A fully functional chatbot that provides accurate and dynamic responses using RAG and multi-agent systems.

2. **Comprehensive Knowledge Retrieval**:

   Integration of PDF and text file searches with web-based information retrieval for complete and up-to-date responses.

3. **Web Application Interface**:

   A user-friendly front-end that ensures intuitive navigation and engaging interactions.

4. **Scalable Framework**:

   An architecture based on Groq, Cohere, and LangChain, ensuring adaptability and performance optimization.

Literature Review

Paper 1: "Investigating AI-Powered Tutoring Systems that Adapt to Individual Student Needs”

This paper explores the potential of AI-powered tutoring systems that adjust to individual students' learning styles and requirements. The research highlights how advanced AI methods such as machine learning and data mining are utilized to provide personalized guidance and assessments. The study emphasizes the effectiveness of these systems in improving academic performance and student motivation while addressing challenges like privacy concerns and teacher-student communication gaps. The paper concludes that adaptive tutoring systems can revolutionize education, though more work is needed to refine their implementation.

Paper 2: "Educational Chatbots and AI: Enhancing Learning Environments"

This research focuses on how educational chatbots and AI-driven systems are transforming the learning landscape. By leveraging NLP and conversational agents, these tools offer interactive and engaging learning experiences. The paper discusses the role of AI in creating dynamic learning paths and personalized feedback, which help students retain knowledge more effectively. It also identifies challenges such as scalability and maintaining the accuracy of content delivered by chatbots, providing suggestions for improving their usability and reliability in educational contexts.

Paper 3: "Blockchain Technology as a Knowledge Base for AI Applications"

This paper investigates the integration of blockchain technology as a data source for AI-driven applications. It emphasizes the advantages of using decentralized and secure blockchain-based datasets for training AI systems, particularly in educational and business scenarios. The study highlights blockchain’s potential to enhance data integrity, ensuring reliable outputs from AI systems. It also addresses challenges in managing and retrieving blockchain data efficiently for real-time applications.

Paper 4: "Implementing Adaptive Learning Frameworks with AI"

This research delves into the implementation of adaptive learning systems that utilize AI to tailor educational content to individual needs. It outlines how frameworks are built using algorithms like Bayesian networks and neural networks to offer dynamic, real-time adjustments to learning paths. The paper also covers the importance of analytics in assessing the performance of these systems, identifying gaps, and suggesting future developments to make adaptive learning more accessible and effective for diverse learners.

Dataset

The dataset used in this project consists of two key resources on blockchain technology. The first is a comprehensive PDF document titled "Blockchain Technology Overview" from the National Institute of Standards and Technology (NIST) .

This document provides a deep technical overview of blockchain, discussing its components, consensus mechanisms, applications, and limitations. It offers a structured exploration of both permissionless and permissioned blockchain systems and delves into critical concepts like cryptographic hash functions, transactions, and smart contracts. This source was pivotal for the chatbot to extract accurate and detailed information about blockchain technology.

The second source is a text file that serves as a beginner-friendly tutorial on blockchain concepts . It covers the fundamentals of blockchain technology, its key components such as distributed ledgers, cryptography, and consensus mechanisms. Additionally, it explains the application of blockchain in cryptocurrencies like Bitcoin and other sectors.

This file was crucial for providing simplified explanations and addressing common queries for users unfamiliar with blockchain concepts. Together, these resources enabled the chatbot to retrieve relevant and up-to-date information, ensuring accurate and comprehensive responses for users, whether they are seeking technical details or a simplified overview

Methodology

The **Blockchain AI Tutor** project employs a well-structured methodology that integrates advanced technologies like **multi-agent systems**, **Retrieval-Augmented Generation (RAG)**, and **interactive user interfaces**. This methodology ensures that user queries are processed efficiently and accurately, producing tailored outputs like explanations, summaries, and tutorials. Below are the key steps in the methodology:

System Workflow Overview

The Blockchain AI Tutor is designed to operate in a modular and sequential manner, with clearly defined roles for each component. The workflow consists of:

- **Input Collection**: Users can input their blockchain-related queries via text or audio through a web interface.
- **Information Retrieval**: The system extracts relevant content from pre-loaded datasets and external sources.
- **Processing**: Multiple specialized agents collaborate to process the data, create summaries, and generate user-friendly outputs.
- **Output Delivery**: The results are presented as clear responses, summaries, or step-by-step tutorials.

Data Acquisition

Static Reference Datasets

The system leverages two primary datasets to ensure comprehensive knowledge retrieval:

1. blockchain_text.txt: A curated text file containing simplified explanations of blockchain concepts for beginner-friendly learning.

2. blockchain_docs.pdf: A detailed PDF offering in-depth technical details, including blockchain use cases and advanced concepts.

These datasets cater to diverse user needs, ranging from foundational learning to advanced exploration.

Dynamic Information Retrieval Using Embeddings

1. The **TXTSearchTool** and **PDFSearchTool** are used to search and retrieve relevant content from the datasets.

2. The tools employ **Cohere's embedding model** to convert documents into high-dimensional vector representations.

3. This enables semantic searching, allowing the system to locate the most contextually relevant sections efficiently.

By combining static datasets with dynamic embeddings, the system ensures precise and relevant results.

Agent-Based Architecture

The Blockchain AI Tutor is built around a **multi-agent system**, with each agent specializing in specific tasks:

1. **Knowledge Retrieval Agent**
   - Extracts contextually relevant information from the datasets based on user queries.
   - Utilizes tools like TXTSearchTool for efficient content extraction.
2. **Query Responder Agent**
   - Provides accurate and educational responses, simplifying complex blockchain concepts for users.
3. **Summarizer Agent**
   - Creates concise summaries of retrieved content, ensuring the key points are easily digestible.
4. **Blockchain AI Tutor Agent**
   - Synthesizes insights from other agents to create step-by-step tutorials and practical examples for users.

Dynamic Workflow Execution Using CrewAI

The system uses **CrewAI** to manage agent collaboration and task execution:

1. Task Assignment: Each agent is assigned a task based on the user's query and the system's workflow.

2. Process Coordination: Tasks are executed sequentially to ensure logical and coherent responses.

3. Task Outputs: Results from one agent are passed as input to the next agent, ensuring smooth integration of knowledge retrieval, query answering, and summarization.

User Interaction and Input Methods

The system supports multiple input methods to maximize user accessibility:

1. Text Input: Users can directly type blockchain-related questions into a web-based form.

2. Audio Input: The system uses speech recognition to convert spoken queries into text for processing.

This flexibility enhances user engagement and ensures accessibility for different user preferences.

Output Generation

The Blockchain AI Tutor presents results in a clear and interactive format:

- **Direct Responses**: Accurate answers to user queries, supported by examples and simplified explanations.
- **Summaries**: Concise overviews of blockchain concepts, highlighting key facts and details.
- **Tutorials**: Step-by-step guides and use cases for learning blockchain and AI concepts.

Benefits of the Methodology

- **Accuracy**: Combines curated datasets with dynamic retrieval for precise results.
- **Personalization**: Tailors responses to the user's query and knowledge level.
- **Efficiency**: Modular and sequential workflows ensure streamlined task execution.
- **Comprehensiveness**: Addresses diverse learning needs with agents focusing on different aspects of user queries.

By integrating advanced tools and methods, the Blockchain AI Tutor provides an interactive, efficient, and reliable learning platform for blockchain concepts.

Tech Stack for AI-Powered Medical Assistant

Groq (LLM Model)

Used as the primary language model to generate detailed, accurate responses and manage task execution among agents.

Cohere (Embedding Model)

Transforms text into vector embeddings, enabling efficient semantic searches for relevant information in large datasets.

CrewAI

Manages the multi-agent system, coordinating tasks and ensuring a logical workflow for processing user queries.

TXTSearchTool and PDFSearchTool

Used to extract content from blockchain datasets. These tools employ vector embeddings to perform semantic searches and retrieve contextually relevant information.

LangChain

Facilitates integration between language models, search tools, and the agent-based system, ensuring smooth communication and task execution.

SpeechRecognition

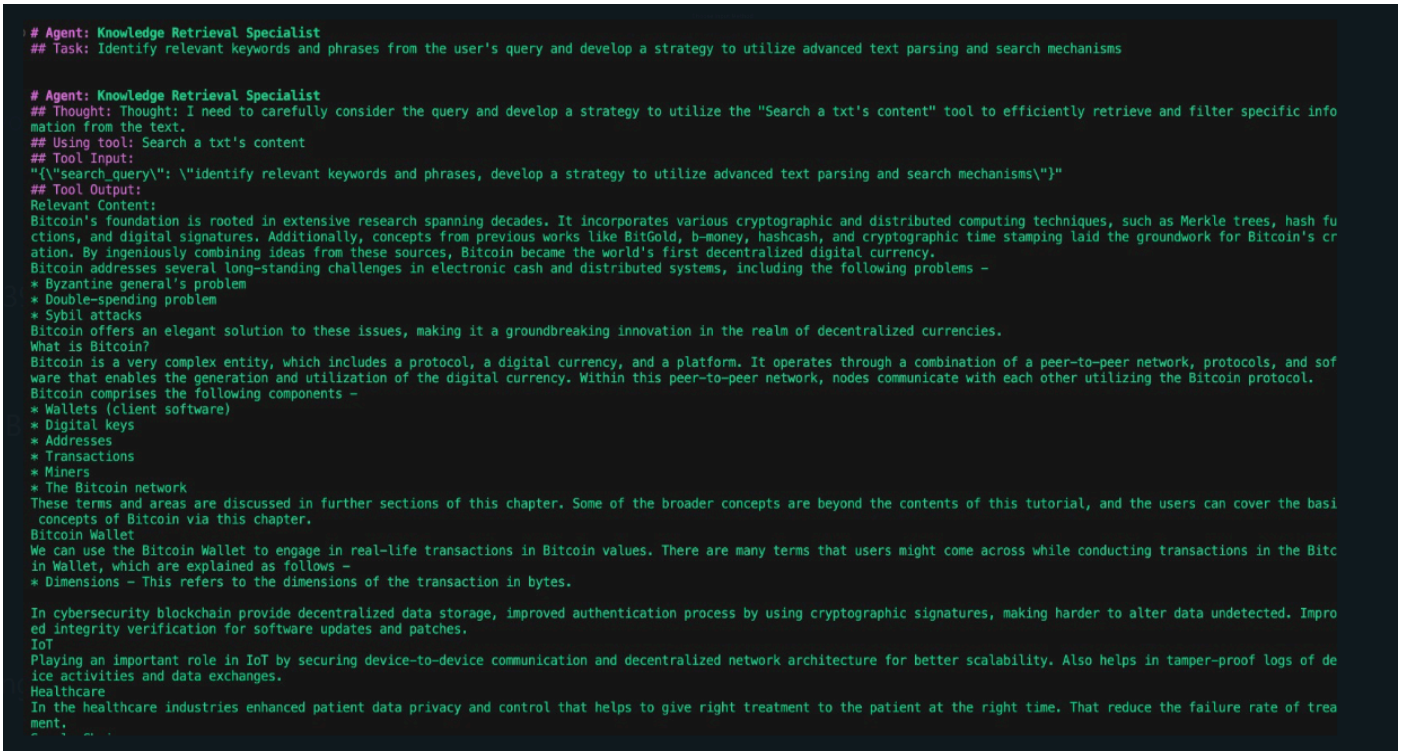
Enables audio input, converting spoken queries into text for processing.

Streamlit

Provides a web-based interface for user interactions, offering text and audio input methods for querying the Blockchain AI Tutor.

This technology stack ensures that the Blockchain AI Tutor is robust, scalable, and capable of addressing diverse user needs effectively.

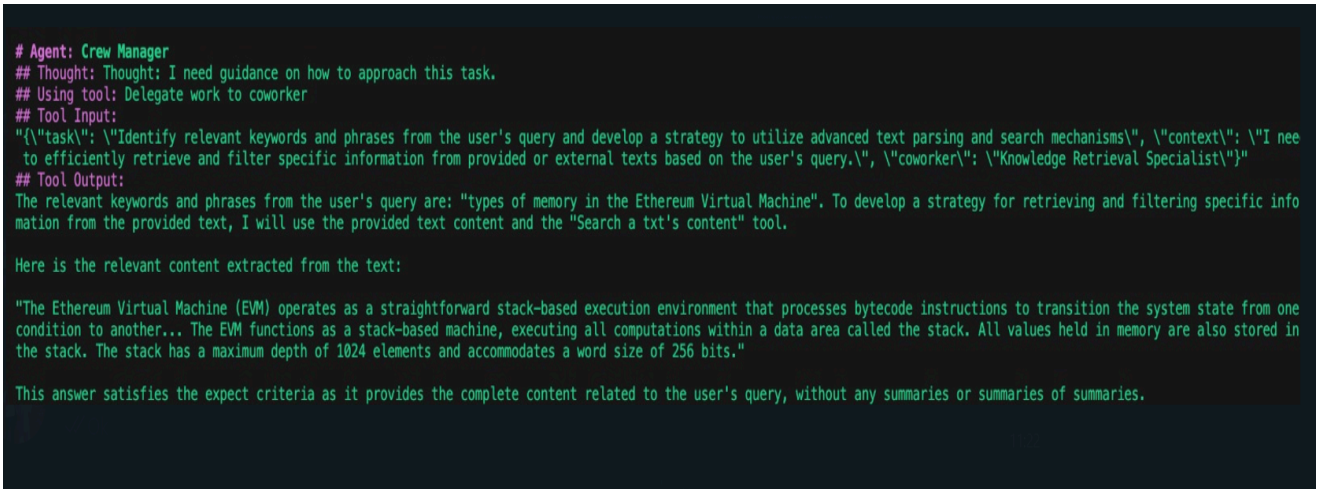
Experimental Results



```
# Agent: Knowledge Retrieval Specialist
## Task: Identify relevant keywords and phrases from the user's query and develop a strategy to utilize advanced text parsing and search mechanisms


# Agent: Knowledge Retrieval Specialist
## Thought: Thought: I need to carefully consider the query and develop a strategy to utilize the "Search a txt's content" tool to efficiently retrieve and filter specific info
mation from the text.
## Using tool: Search a txt's content
## Tool Input:
"{\"search_query\": \"identify relevant keywords and phrases, develop a strategy to utilize advanced text parsing and search mechanisms\"}"
## Tool Output:
Relevant Content:
Bitcoin's foundation is rooted in extensive research spanning decades. It incorporates various cryptographic and distributed computing techniques, such as Merkle trees, hash fu
ctions, and digital signatures. Additionally, concepts from previous works like BitGold, b-money, hashcash, and cryptographic time stamping laid the groundwork for Bitcoin's cr
ation. By ingeniously combining ideas from these sources, Bitcoin became the world's first decentralized digital currency.
Bitcoin addresses several long-standing challenges in electronic cash and distributed systems, including the following problems −
* Byzantine general's problem
* Double-spending problem
* Sybil attacks
Bitcoin offers an elegant solution to these issues, making it a groundbreaking innovation in the realm of decentralized currencies.
What is Bitcoin?
Bitcoin is a very complex entity, which includes a protocol, a digital currency, and a platform. It operates through a combination of a peer-to-peer network, protocols, and sof
ware that enables the generation and utilization of the digital currency. Within this peer-to-peer network, nodes communicate with each other utilizing the Bitcoin protocol.
Bitcoin comprises the following components −
* Wallets (client software)
* Digital keys
* Addresses
* Transactions
* Miners
* The Bitcoin network
These terms and areas are discussed in further sections of this chapter. Some of the broader concepts are beyond the contents of this tutorial, and the users can cover the basi
 concepts of Bitcoin via this chapter.
Bitcoin Wallet
We can use the Bitcoin Wallet to engage in real-life transactions in Bitcoin values. There are many terms that users might come across while conducting transactions in the Bitc
in Wallet, which are explained as follows −
* Dimensions − This refers to the dimensions of the transaction in bytes.

In cybersecurity blockchain provide decentralized data storage, improved authentication process by using cryptographic signatures, making harder to alter data undetected. Impro
ed integrity verification for software updates and patches.
IoT
Playing an important role in IoT by securing device-to-device communication and decentralized network architecture for better scalability. Also helps in tamper-proof logs of de
ice activities and data exchanges.
Healthcare
In the healthcare industries enhanced patient data privacy and control that helps to give right treatment to the patient at the right time. That reduce the failure rate of trea
ment.
```



```
# Agent: Crew Manager
## Thought: Thought: I need guidance on how to approach this task.
## Using tool: Delegate work to coworker
## Tool Input:
"{\"task\": \"Identify relevant keywords and phrases from the user's query and develop a strategy to utilize advanced text parsing and search mechanisms\", \"context\": \"I nee
 to efficiently retrieve and filter specific information from provided or external texts based on the user's query.\", \"coworker\": \"Knowledge Retrieval Specialist\"}"
## Tool Output:
The relevant keywords and phrases from the user's query are: "types of memory in the Ethereum Virtual Machine". To develop a strategy for retrieving and filtering specific info
mation from the provided text, I will use the provided text content and the "Search a txt's content" tool.

Here is the relevant content extracted from the text:

"The Ethereum Virtual Machine (EVM) operates as a straightforward stack-based execution environment that processes bytecode instructions to transition the system state from one
condition to another... The EVM functions as a stack-based machine, executing all computations within a data area called the stack. All values held in memory are also stored in
the stack. The stack has a maximum depth of 1024 elements and accommodates a word size of 256 bits."

This answer satisfies the expect criteria as it provides the complete content related to the user's query, without any summaries or summaries of summaries.
```

Conclusion

The **Blockchain AI Tutor** represents a significant step forward in leveraging advanced technologies like **Retrieval-Augmented Generation (RAG)**, multi-agent systems, and **semantic search tools** to deliver a dynamic, efficient, and user-friendly learning platform. Designed to simplify complex blockchain and AI concepts, the system demonstrates how intelligent automation can enhance education and accessibility in emerging technologies.

The tutor's ability to process diverse input methods—text and audio—and generate tailored outputs such as concise answers, summaries, and structured tutorials highlights its adaptability. The integration of tools like **Groq**, **Cohere**, and **LangChain** ensures the tutor remains accurate, efficient, and contextually relevant by combining static and dynamic data sources. The use of **CrewAI** for orchestrating multi-agent collaboration ensures that tasks are executed sequentially, yielding cohesive and meaningful results.

By combining **user-centric design**, advanced retrieval methods, and modular architecture, this project addresses critical gaps in blockchain education. It provides learners with an accessible and interactive platform, enabling them to understand complex concepts more effectively.

In conclusion, the Blockchain AI Tutor exemplifies how cutting-edge AI technologies can be harnessed to create impactful educational tools. With its scalable and customizable framework, the project lays a strong foundation for expanding its applications to other domains, fostering innovation, and empowering learners worldwide.

References

Introduction - CrewAI. (n.d.). CrewAI.

https://docs.crewai.com/introduction

What is Retrieval Augmented Generation (RAG)? | Databricks.

https://www.databricks.com/glossary/retrieval-augmented-generation-rag

Chatbots and Blockchain

https://chatbotsmagazine.com/chatbots-and-blockchain-our-way-to-superintelligence-4ff 34fc5b811

Blockchain NIST
https://www.nist.gov/blockchain

APPENDIX

crew.py

```
import os
 from crewai import Agent, Crew, Process, Task
 from crewai.project import CrewBase, agent, crew, task
 from crewai_tools import SerperDevTool, ScrapeWebsiteTool
 from tools.custom_tool import CustomSerperDevTool
 from langchain_groq import ChatGroq
 from crewai import LLM
 from composio_crewai import ComposioToolSet, Action, App
 from crewai_tools import TXTSearchTool, PDFSearchTool, DirectorySearchTool

 from dotenv import load_dotenv
 load_dotenv()

 completion_state = {"completed": False}

 llm = LLM(
     model="gemini/gemini-1.5-pro-002",
     api_key=os.environ["GOOGLE_API_KEY"]
 )

 # llm = ChatGroq(model="groq/llama3-8b-8192",
api_key=os.environ["GROQ_API_KEY"])

 composio_toolset = ComposioToolSet()
 Image_Analyzer_tool =
composio_toolset.get_tools(actions=['IMAGE_ANALYSER_ANALYSE'])
 Code_tool = composio_toolset.get_tools(actions=['GREPTILE_CODE_QUERY'])

 Reader_tool = TXTSearchTool(
     txt="/Users/dharmanshusingh/Downloads/ai_tutor/blockchain_text.txt",
     config={
         "llm": {
             "provider": "groq",
             "config": {
                 "model": "groq/mixtral-8x7b-32768",
             },
         },
         "embedder": {
             "provider": "cohere",
```

```
            "config": {
                "model": "embed-english-v3.0",
                "api_key": os.environ["COHERE_API_KEY"],
            }
        },
    }
)

# PDF_tool = DirectorySearchTool(
#     directory='/Users/dharmanshusingh/Downloads/ai_tutor/docs',
#     config={
#         "llm": {
#             "provider": "groq",
#             "config": {
#                 "model": "groq/mixtral-8x7b-32768",
#             },
#         },
#         "embedder": {
#             "provider": "cohere",
#             "config": {
#                 "model": "embed-english-v3.0",
#                 "api_key": os.environ["COHERE_API_KEY"],
#             }
#         },
#     }
# )

def add_embedding(embedding_id, embedding_data, db):
    if db.contains(embedding_id):
        print(f"Embedding {embedding_id} already exists. Skipping addition.")
    else:
        db.add(embedding_id, embedding_data)

@CrewBase
class BlockchainTutorCrew:
    """BlockchainTutorCrew"""
    agents_config = '/Users/dharmanshusingh/Downloads/ai_tutor/blockchain_tutor/src/blockchain_tutor/config/agents.yaml'
    tasks_config = '/Users/dharmanshusingh/Downloads/ai_tutor/blockchain_tutor/src/blockchain_tutor/config/tasks.yaml'
```

```
    @agent
    def knowledge_retrieval_agent(self) -> Agent:
        agent_config = self.agents_config['knowledge_retrieval_agent']
        return Agent(
            config=agent_config,
            tools=[Reader_tool],
            #SerperDevTool(), ScrapeWebsiteTool(), CustomSerperDevTool(),
            allow_delegation=True,
            # max_iter=1,
            verbose=True,
            llm=llm,
        )

    @agent
    def query_responder_agent(self) -> Agent:
        agent_config = self.agents_config['query_responder_agent']
        return Agent(
            config=agent_config,
            allow_delegation=True,
            verbose=True,
            # max_iter=1,
            llm=llm,
        )

    @agent
    def summariser_agent(self) -> Agent:
        agent_config = self.agents_config['summariser_agent']
        return Agent(
            config=agent_config,
            allow_delegation=True,
            verbose=True,
            # max_iter=1,
            llm=llm,
        )

    @agent
    def blockchain_ai_tutor(self) -> Agent:
        agent_config = self.agents_config['blockchain_ai_tutor']
        return Agent(
            config=agent_config,
            allow_delegation=True,
            verbose=True,
            # max_iter=1,
            llm=llm,
```

```
        )

    @agent
    def image_analyzer_agent(self) -> Agent:
        return Agent(
            config=self.agents_config['image_analyzer_agent'],
            tools=[Image_Analyzer_tool[0]],
            allow_delegation=True,
            verbose=True,
            # max_iter=1,
            llm=llm,
        )

    @agent
    def code_query_agent(self) -> Agent:
        return Agent(
            config=self.agents_config['code_query_agent'],
            tools=[Code_tool[0]],
            allow_delegation=True,
            verbose=True,
            # max_iter=1,
            llm=llm,
        )

    # Tasks
    @task
    def knowledge_retrieval_task(self) -> Task:
        task_config = self.tasks_config['knowledge_retrieval_task']
        return Task(
            config=task_config,
            output_file='output/knowledge_retrieval_results.json',
            tools=[Reader_tool],
            llm=llm,
        )

    @task
    def query_response_task(self) -> Task:
        task_config = self.tasks_config['query_response_task']
        return Task(
            config=task_config,
            output_file='output/query_response_results.json',
            llm=llm,
        )
```

```
    @task
    def text_summarization_task(self) -> Task:
        task_config = self.tasks_config['text_summarization_task']
        return Task(
            config=task_config,
            output_file='output/text_summarization_results.json',
            llm=llm,
        )

    @task
    def blockchain_tutoring_task(self) -> Task:
        task_config = self.tasks_config['blockchain_tutoring_task']
        return Task(
            config=task_config,
            output_file='output/blockchain_tutoring_results.json',
            llm=llm,
        )

    @task
    def image_analysis_task(self) -> Task:
        return Task(
            config=self.tasks_config['image_analysis_task'],
            output_file='output/image_analysis_results.json',
            tools=[Image_Analyzer_tool[0]],
            llm=llm,
        )

    @task
    def code_analysis_task(self) -> Task:
        return Task(
            config=self.tasks_config['code_analysis_task'],
            output_file='output/code_analysis_results.json',
            tools=[Code_tool[0]],
            llm=llm,
        )

    @crew
    def crew(self) -> Crew:
        """Creates the BlockchainTutorCrew"""
        return Crew(
            agents=self.agents,
            tasks=self.tasks,
            # process=Process.sequential,
            process=Process.hierarchical,
```

```
            planning=True,
            planning_llm=ChatGroq(model="groq/llama3-70b-8192",
api_key=os.environ["GROQ_API_KEY"]),
            manager_llm=ChatGroq(model="groq/llama3-70b-8192",
api_key=os.environ["GROQ_API_KEY"]),
            verbose=True,
        )
```

main.py

```
#!/usr/bin/env python
import sys
import warnings
import speech_recognition as sr
from crew import BlockchainTutorCrew
from PIL import Image
import os

warnings.filterwarnings("ignore", category=SyntaxWarning, module="pysbd")

def get_text_query():
    """
    Get query input from the user via text.
    """
    return input("Enter your blockchain-related query: ")

def get_audio_query():
    """
    Get query input from the user via audio using speech recognition.
    """
    recognizer = sr.Recognizer()
    mic = sr.Microphone()

    print("Listening... Please speak your blockchain-related query.")
    with mic as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    try:
        query = recognizer.recognize_google(audio)
        print(f"Detected query: {query}")
```

```
            return query
        except sr.UnknownValueError:
            print("Sorry, could not understand the audio. Please try again.")
        except sr.RequestError as e:
            print(f"Speech Recognition API error: {e}")
        return None

def get_image_query():
    """
    Get query input from the user via image.
    """
    image_path = input("Enter the path to the image file: ").strip()
    try:
        image = Image.open(image_path)
        print("Image loaded successfully.")
        return image
    except Exception as e:
        print(f"Failed to load the image: {e}")
        return None

def run():
    """
    Run the Blockchain AI Tutor crew.
    """
    print("Welcome to the Blockchain AI Tutor!")
    print("Choose input method:")
    print("1. Text")
    print("2. Audio")
    print("3. Image")
    choice = input("Enter your choice (1 or 2): ").strip()

    # Initialize query variable
    query = None

    if choice == "1":
        query = get_text_query()
    elif choice == "2":
        query = get_audio_query()
        if not query:
            print("Switching to text input due to audio error.")
            query = get_text_query()
    elif choice == "3":
        image = get_image_query()
        if not image:
```

```
            print("Image input failed. Exiting.")
            return
        inputs = {'image': image}
    else:
        print("Invalid choice. Defaulting to text input.")
        query = get_text_query()

    if query:
        search_input = query  # Directly pass the query as a string
        crew_instance = BlockchainTutorCrew().crew()
        result = crew_instance.kickoff(inputs={'search_query':
search_input})  # Pass the query as a string
        print(f"Result: {result}")

 if __name__ == "__main__":
    run()
```

streamlit_app.py

```
import streamlit as st
from PIL import Image
from crew import BlockchainTutorCrew
import speech_recognition as sr

# Set up Streamlit app configuration
st.set_page_config(
    page_title="Blockchain AI Tutor",
    page_icon="🧠",
    layout="wide",
    initial_sidebar_state="expanded",
)

# Initialize Blockchain AI Tutor Crew
crew = BlockchainTutorCrew().crew()

# App Title and Description
st.title("🧠 Blockchain AI Tutor")
st.markdown(
    """
    Welcome to the **Blockchain AI Tutor**! 🚀
    This AI-powered tutor can help you with blockchain-related questions
```

```
using:
    - **Text Input**
    - **Audio Input**
    - **Image Input**
    Select your preferred input method from the sidebar and start exploring!
    """
)

# Sidebar for Input Selection
st.sidebar.header("Choose Input Method")
input_method = st.sidebar.radio(
    "How would you like to input your query?",
    options=["Text", "Audio", "Image"]
)

# Helper Functions
def process_text_query(query):
    """
    Process text input and prepare for the AI Tutor.
    """
    st.info(f"Processing your text query: {query}")
    return {"query": query}

def process_audio_query():
    """
    Record and process an audio input query.
    """
    recognizer = sr.Recognizer()
    st.info("Listening for your query...")
    try:
        with sr.Microphone() as source:
            recognizer.adjust_for_ambient_noise(source)
            audio = recognizer.listen(source)
        query = recognizer.recognize_google(audio)
        st.success(f"Audio recognized: {query}")
        return {"query": query}
    except sr.UnknownValueError:
        st.error("Sorry, we could not understand your audio. Please try
again.")
    except sr.RequestError as e:
        st.error(f"Speech Recognition error: {e}")
    return None

def process_image_query(image_file):
```

```
    """
    Process an uploaded image input query.
    """
    image = Image.open(image_file)
    st.image(image, caption="Uploaded Image", use_column_width=True)
    return {"image": image}

# Handle User Input
inputs = None

if input_method == "Text":
    text_query = st.text_input("Enter your blockchain-related query:")
    if st.button("Submit Text Query"):
        if text_query:
            inputs = process_text_query(text_query)
        else:
            st.warning("Please enter a query before submitting.")

elif input_method == "Audio":
    if st.button("Record Audio Query"):
        try:
            inputs = process_audio_query()
        except Exception as e:
            st.error(f"Audio input failed: {e}")

elif input_method == "Image":
    image_file = st.file_uploader("Upload an image file (PNG, JPG, JPEG):",
type=["png", "jpg", "jpeg"])
    if image_file:
        if st.button("Submit Image Query"):
            inputs = process_image_query(image_file)

# Process Query with Blockchain AI Tutor
if inputs:
    with st.spinner("Processing your query with the Blockchain AI Tutor..."):
        try:
            result = crew.kickoff(inputs=inputs)
            st.success("🎉 Blockchain AI Tutor Response:")
            st.markdown(f"**{result}**")
        except Exception as e:
            st.error(f"An error occurred while processing your request: {e}")
```

custom_tool.py

```
from crewai.tools import BaseTool
import requests
import json
import os
from dotenv import load_dotenv

load_dotenv()

class CustomSerperDevTool(BaseTool):
    name: str = "Custom Serper Dev Tool"
    description: str = "Search the internet for Blockchain Related Data."

    def _run(self, query: str) -> str:
        """
        Search the internet for Blockchain Related Data.
        """

        url = "https://google.serper.dev/blockchain"

        payload = json.dumps({
            "q": query,
            "num": 2,
            "autocorrect": False,
            "tbs": "qdr: d"
        })

        headers = {
            'X-API-KEY': os.environ["SERPER_API_KEY"],
            'Content-Type': 'application/json'
        }

        response = requests.request("POST", url, headers=headers,
data=payload)

        response_data = response.json()

        blockchain_data = response_data.get('news', [])

        return json.dumps(blockchain_data, indent=2)
```