



**International Centre for Education and Research (ICER)
VIT-Bangalore**

COBOT - THE CARBON TRADE ALLY

CS7610 – CAPSTONE PROJECT

REPORT

Submitted by

DHARSHINI M – 24MSP3070

In partial fulfilment for the award of the degree of

POST GRADUATE PROGRAMME

INTERNATIONAL CENTRE FOR HIGHER EDUCATION AND RESEARCH

VIT BANGALORE

June, 2025



**International Centre for Education and Research (ICER)
VIT-Bangalore**

BONAFIDE CERTIFICATE

Certified that this project report “**COBOT - THE CARBON TRADE ALLY**” is the bonafide record of work done by “**DHARSHINI M – 24MSP3070**” who carried out the project work under my supervision.

Signature of the Supervisor

Signature of Director

Dr. Shiyamala Gowri

Prof. Prema M

Associate Professor,

Director,

ICER

ICER

VIT Bangalore

VIT Bangalore.

Evaluation Date: 20th June 2025



**International Centre for Education and Research (ICER)
VIT-Bangalore**

ACKNOWLEDGEMENT

I express my sincere gratitude to our director of ICER **Prof. Prema M.** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor, **Dr. Shiyamala Gowri, Associate Professor**, ICER for extending help and encouragement throughout the project. Without her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of ICER, my beloved parents, and friends for extending the support, who helped us to overcome obstacles in the study.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	5
	LIST OF FIGURES	6
	LIST OF TABLES	7
1	INTRODUCTION	8
	1.1. CARBON TRADING	8
	1.2. CARBON CREDITS	8
	1.3. RETRIEVAL-AUGMENTED GENERATION	9
2	LITERATURE REVIEW	11
3	OBJECTIVE	13
	3.1. GENERAL OBJECTIVE	13
	3.2. SPECIFIC OBJECTIVES	13
4	PROPOSED METHODOLOGY	14
5	TOOLS AND TECHNIQUES	16
6	IMPLEMENTATION	18
	6.1. ABOUT THE DATASET	18
	6.2. PREPROCESSING	18
	6.3. FUNCTIONAL COMPONENTS AND QUERY PROCESSING	19
	6.4. UI RENDERING AND DEPLOYMENT	19
	6.5. PERFORMANCE AND RESPONSE GENERATION FLOW	19
7	RESULTS AND DISCUSSIONS	21
8	CONCLUSION	22
9	FUTURE ENHANCEMENT	23
10	APPENDICES	24
	Appendix-1: Code – Full Code and Outputs	24
11	REFERENCES	30
12	WORKLOG	31

ABSTRACT

As carbon trading markets are developing so quickly, people and organisations need easy-to-use, real-time tools to navigate market data and regulatory frameworks. To meet this need, CoBot—The Carbon Trading Ally—is an AI-powered chatbot built with Retrieval-Augmented Generation (RAG) architecture. The system provides precise, context-aware answers to user queries by combining Google's Generative AI models with FAISS-based vector retrieval. The Streamlit interface allows users to interact with the bot through simple question input and comprehensive responses derived from carefully selected carbon trading documents. Four reputable PDFs from international financial and environmental organisations are currently included in the knowledge base. For participants in the carbon market, CoBot facilitates sustainability planning, compliance comprehension, and market insights. CoBot illustrates the possibilities of combining document retrieval, natural language processing, and interactive design. CoBot shows how artificial intelligence (AI) can be used to advance climate literacy and ethical trading by fusing natural language processing, document retrieval, and interactive design.

LIST OF FIGURES

Figure No.	Figure Name	Pg. No.
Fig. 1.1	Explaining Carbon credits with Balloon Analogy	9
Fig. 1.2	Basic Working of RAG System	10
Fig. 4.1	Architecture Diagram	14
Fig. 7.1	Output of Interface Created	21

LIST OF TABLES

Table No.	Table Name	Pg. No.
Table. 5.1	Toolchain Summary Table	16
Table. 6.1	Workflow Table	20

CHAPTER 1

INTRODUCTION

1.1. CARBON TRADING

A market-driven strategy called carbon trading uses financial incentives to lower greenhouse gas emissions. Using a cap-and-trade model, it allows organisations to purchase or sell carbon credits based on whether their emissions are below or above a set limit. This encourages investment in cleaner technologies as well as the reduction of emissions. Governments and businesses can take part thanks to important mechanisms like the European Union Emissions Trading System (EU ETS) and other voluntary markets. The system is one of the most useful instruments for promoting environmental responsibility and sustainable industrial practices since it supports the global climate goals established by accords like the Kyoto Protocol and Paris Agreement.

Navigating the carbon trading ecosystem, however, calls for knowledge of project certifications, compliance procedures, regulatory policies, and volatile market prices. The crucial data required for trading or compliance is frequently dispersed throughout numerous sources and hidden deep within lengthy PDF documents. This makes it difficult for people, small enterprises, and even institutional stakeholders to effectively participate in carbon markets. CoBot—The Carbon Trading Ally—fills this void by offering an easy-to-use, AI-powered solution that transforms static carbon trading knowledge into interactive advice. It makes carbon trading understandable, precise, and conversational by using a Retrieval-Augmented Generation (RAG) framework to extract pertinent content from reliable documents and provide real-time answers to user queries.

1.2. CARBON CREDITS

The right to emit a specific quantity of carbon dioxide or its equivalent is represented by carbon credits, which are market-based financial instruments. One metric tonne of CO₂ is typically equivalent to one carbon credit. Verified projects that lower, prevent, or eliminate greenhouse gas emissions—such as methane capture, reforestation, or renewable energy—are the source of these credits. Credits can be earned and sold in the market by organisations that reduce emissions above and beyond what is required by law. As a result, businesses are financially motivated to invest in sustainable practices.

Government-regulated compliance markets and voluntary markets, which are motivated by environmental and corporate social responsibility objectives, are two markets in which carbon credits can be traded. Carbon credits are primarily intended to assist in economically achieving global climate goals.

By funding environmental projects abroad, they allow nations and businesses to offset emissions. This system aids in the transfer of clean technology to developing nations and promotes sustainable development. However, accurate verification, openness, and avoiding double counting are necessary for carbon credit systems to be effective.

Carbon credits continue to be an essential part of international emission reduction plans, despite complaints about their legitimacy and unequal pricing. As organisations commit to net-zero goals and governments tighten climate policies, their role is anticipated to grow even more. Anyone involved in or analysing carbon markets must have a thorough understanding of their types, functions, and trading platforms.

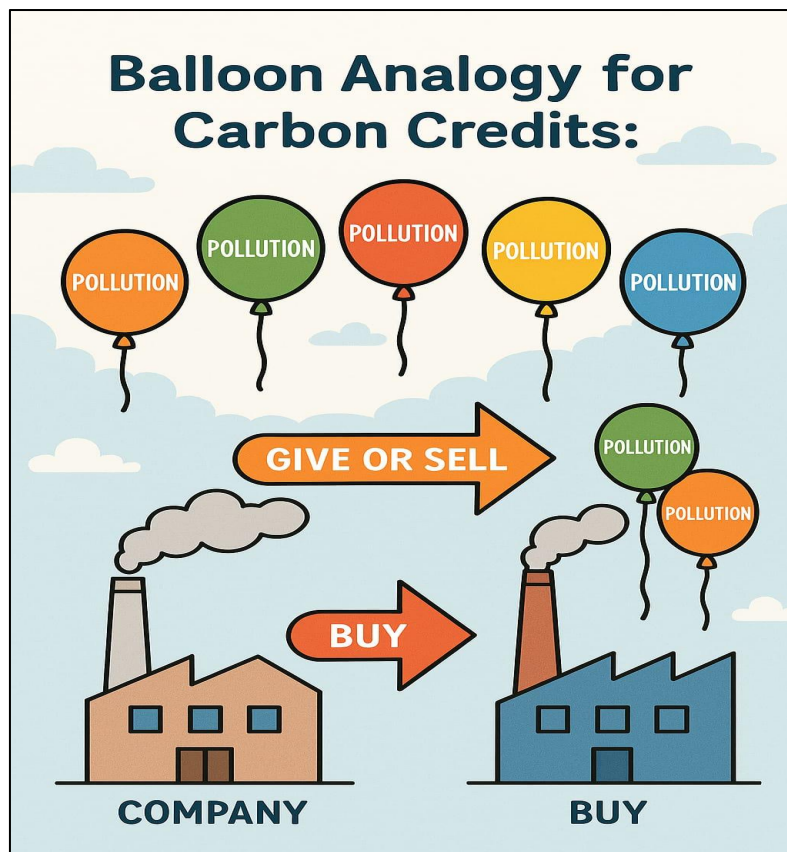


Fig. 1.1 Explaining Carbon credits with Balloon Analogy

1.3 RETRIEVAL-AUGMENTED GENERATION

A hybrid AI architecture called Retrieval-Augmented Generation (RAG) blends the advantages of natural language generation and information retrieval.

Conventional language models produce answers based on the parameters they have learnt, but they frequently have trouble answering questions about real-time or domain-specific data. In order to solve this, RAG retrieves pertinent external documents during query time from a knowledge base. The generative model uses the factual foundation these documents offer to generate well-informed and contextually aware responses. RAG is perfect for applications that need current or document-specific outputs, like legal support systems, medical advice, or climate policy, because it combines retrieval and generation.

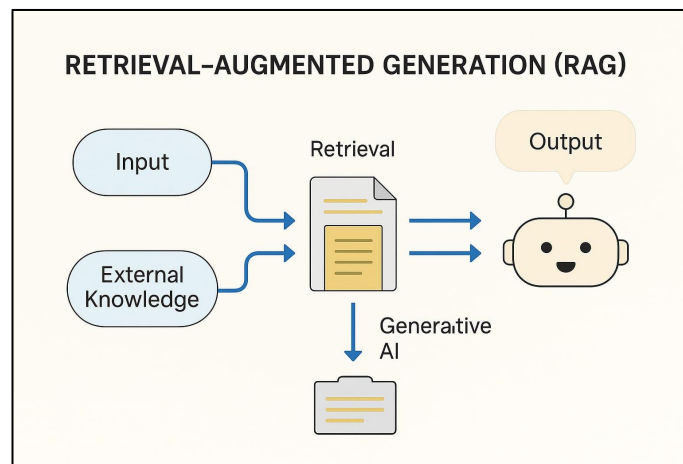


Fig. 1.2 Basic Working of RAG System

Compared to static LLMs, RAG offers a number of benefits in the context of knowledge systems. It guarantees verifiable responses, enables customisation for specialised domains using document collections, and keeps systems up to date without frequent retraining. Fast semantic search across embedded document chunks is made possible by RAG's use of vector stores like FAISS or Pinecone. RAG systems can mimic human-like comprehension of sizable unstructured datasets when paired with prompt templates and optimised models. Because of this, RAG is a crucial tool for AI assistants that have to interpret academic papers, legal documents, technical documents, or climate reports. Users can ask real-time carbon trading questions using CoBot's architecture, which uses Gemini AI to produce precise and understandable answers while retrieving pertinent information from uploaded PDFs.

2. LITERATURE REVIEW

New opportunities for sustainable digital solutions have been created by the combination of domain-specific knowledge retrieval systems and artificial intelligence, especially large language models (LLMs). This collaboration is essential for facilitating precise, situation-specific decision-making in the context of carbon trading and emission analysis. Retrieval-augmented generation (RAG) frameworks, AI-based chat systems, and regulatory insight tools that support climate-conscious policies have all been developed as a result of numerous research initiatives. This review of the literature looks at important studies that form the basis for the creation of CoBot, a carbon trading assistant driven by RAG architecture and Gemini AI. These contributions demonstrate the continuous advancement of intelligent systems intended to increase the transparency and accessibility of data in markets related to climate change.

CarbonChat: Large Language Model-Based Corporate Carbon Emission Analysis and Climate Knowledge Q&A System, presented in Paper [1], is a retrieval-augmented generation (RAG)-based chatbot designed to answer questions about corporate climate. Using prompt engineering and document chunking for increased accuracy, the model generates context-relevant responses by utilising large language models with embedded domain-specific datasets. Because corporate carbon emission reports make up the majority of the knowledge base, the chatbot can provide highly factual answers to questions about strategy and compliance. The system is robust, but its responsiveness to real-time market changes is limited by its reliance on static reports. The study emphasises the significance of chunking, fine-tuning, and embedding models—concepts that CoBot directly adopts.

A performance-tuned large language model, FAISS vector stores, and LangChain are used in the efficiency-optimized chatbot design presented in Paper [2]. To increase response speed while preserving accuracy, the model makes use of sophisticated chunk sizing, embedding techniques, and modular prompt engineering. Real-time interactions appropriate for narrow-domain applications are made possible by its architecture. The paper is performance-focused, but it ignores deep contextual understanding of regulatory themes. However, its primary contribution—modular chain design and embedding quality—has a direct bearing on how CoBot is built. The technical basis for this project is shaped by the paper's confirmation of LangChain's usefulness in developing lightweight, domain-specific RAG pipelines in sustainability.

A conceptual and economic overview of carbon trading markets can be found in Raymond's 2024 paper [3]. It describes the global evolution of emissions markets, compliance frameworks, credit flows, and cap-and-trade principles. This work offers fundamental insights into the creation, regulation, and trading of carbon credits, despite not being technology-focused.

When creating user queries and retrieval content for CoBot, its thorough analysis of offset systems, allowance auctions, and international cooperation mechanisms is an essential resource. Better prompt engineering and dataset selection for AI models suited to climate finance and trading regulation are supported by the theoretical depth.

Using a RAG framework driven by LangChain, Paper [4] suggests a domain-specific chatbot for university enquiries. The model mimics a real-time academic assistant by fusing vector search with LLM-based answer generation. It illustrates how retrieval chains and FAISS can be used to modify custom chatbots to institutional knowledge sources. Although the system works well for internal use cases, it is not integrated with live datasets and has limited generalisability. However, CoBot's architecture was impacted by the structured pipeline, semantic search usage, and conversational design, particularly its emphasis on retrieval based on PDFs, Google embedding models, and Gemini-based responses customised for carbon trading.

The environmental cost of LLM-powered chatbots is the subject of Paper [5], which measures the energy and carbon footprints of these systems using life-cycle assessment (LCA). The study emphasises how substantial computational resources are used by large generative AI systems during training and inference cycles. The paper highlights the need for modular and energy-efficient AI systems, but it makes no technical recommendations. This drives CoBot's lightweight design decisions, like chunk-based retrieval and the preference for effective Gemini models over bulkier LLM stacks. The study reaffirms the importance of optimisation in AI implementations, particularly for applications with a sustainability focus like carbon market analysis.

The reviewed literature emphasises how carbon market intelligence systems are interdisciplinary, combining real-time data processing, environmental policy awareness, and AI efficiency. Every study offers distinct advantages pertinent to CoBot's design, ranging from energy-conscious deployment models and academic domain adaptations to fundamental RAG-based architectures like CarbonChat. Some works concentrate on conceptual understanding and the regulatory framing of carbon credits and trading systems, while others highlight technical performance and modular design. When combined, they highlight how crucial it is to combine generative capabilities with retrieval accuracy in order to satisfy user-specific information requirements. CoBot has become a useful, AI-powered carbon trading assistant thanks in large part to these insights. By addressing the shortcomings identified in previous studies, CoBot seeks to close current knowledge gaps through scalable system architecture, contextual accuracy, and sustainability focus.

3. OBJECTIVE

3.1 GENERAL OBJECTIVE:

Create and implement an intelligent chatbot system that uses RAG-based architecture and LLMs to provide precise, real-time answers to questions about carbon trading and market intelligence.

3.2 SPECIFIC OBJECTIVES:

- i. Compile and analyse reports and documents related to carbon trading in order to create a unique knowledge base.
- ii. For vector-based document retrieval, use Google Generative AI embeddings and FAISS.
- iii. Use LangChain to integrate the Gemini Pro model to produce precise and contextually aware responses.
- iv. Create an intuitive Streamlit web interface that enables users to pose enquiries about carbon trading.
- v. Make it possible for the system to provide market trends, regulatory insights, and tactical assistance to carbon market players.

4. PROPOSED METHODOLOGY

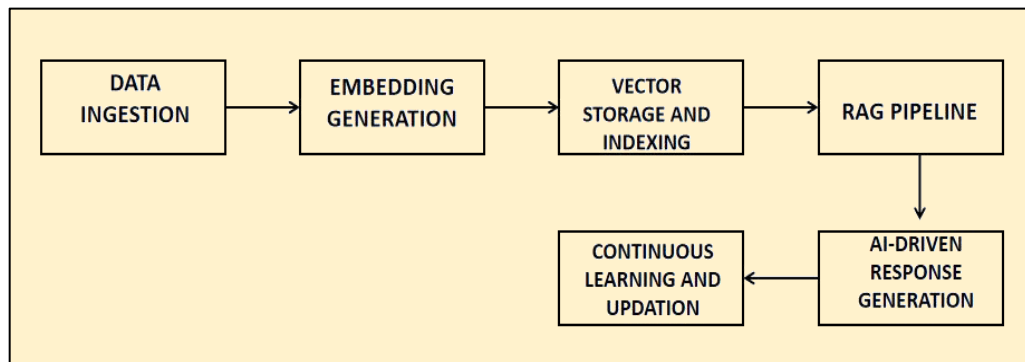


Fig. 4.1 Architecture Diagram

Using RAG architecture, large language models, and vector-based document search, the suggested methodology entails deploying an intelligent chatbot. Real-time document retrieval, semantic understanding, and responsive interaction are guaranteed by the project's three separate phases.

In order to convert unprocessed carbon trading data into interactive, AI-powered solutions, each stage is essential. The objective is to support sustainable decision-making by providing prompt and trustworthy responses to questions regarding carbon markets, trends, and regulatory frameworks.

PHASE 1: INGESTION, PROCESSING, AND EMBEDDING OF DATA

➤ **Gathering Documents and Chunking:**

Documents pertaining to carbon trading, such as reports, policies, pricing information, and regulatory PDFs, are uploaded by users. The documents are read using PyPDF2 and broken into manageable text chunks using LangChain's RecursiveCharacterTextSplitter.

➤ **Embedding Generation (Google Generative AI):** Google Generative AI is used to transform the extracted chunks into high-dimensional vector embeddings (text-embedding-004). In order to facilitate efficient retrieval during queries, these embeddings maintain the semantic relationships between terms.

PHASE 2: QUERY MATCHING, VECTOR INDEXING, AND GENERATION

➤ **Vector Storage (FAISS):**

FAISS is used to index and store all document embeddings, allowing for quick similarity searches across thousands of document chunks.

➤ **Query Embedding & Matching:** The same embedding model is used to transform user questions into embeddings. To find semantically related chunks that are pertinent to the query, the system searches the FAISS store.

➤ **RAG-based Generation (Google Gemini Pro):** Gemini-Pro receives the user's query and the content that has been retrieved via LangChain's chain pipeline. It uses language modelling and prompt templates to produce precise, domain-specific responses.

PHASE 3: FRONTEND INTERFACE, DEPLOYMENT AND INTERACTION

➤ **Web Interface (Streamlit):**

A custom Streamlit UI is built to enable user interaction. Users can post queries and receive chatbot responses in real time. The interface is styled for readability and user comfort.

➤ **Document Update & Re-indexing:**

Users can reload or add more PDFs through the UI. The system reprocesses the files and updates the FAISS index without manual interference.

➤ **Completion & Validation:**

Responses are validated through scenario-based questions to ensure the reliability and contextual understanding of the chatbot. The final application is ready for deployment and extensible with more data.

5. TOOLS AND TECHNIQUES

A carefully selected set of tools and technologies that facilitate smooth information retrieval and response generation are essential to the development of CoBot. In various phases of the pipeline, including document ingestion, text preprocessing, embedding creation, storage, retrieval, and interactive communication, each tool has a specific function. This project creates a strong, scalable, and effective architecture that can respond to intricate domain-specific queries in real time by utilising both well-known libraries and state-of-the-art AI frameworks. The system's overall modularity, interpretability, and user-friendliness are guaranteed by the integration of these tools:

Tool/Library	Purpose
Streamlit	Web Interface
PyPDF2	PDF Reading & Text Extraction
LangChain	RAG Architecture & Pipeline
FAISS	Vector Storage and Similarity Search
Google Generative AI (Embeddings)	Semantic Embedding Generation
Gemini-Pro	Response Generation
dotenv	API Key Handling

Table. 5.1. Toolchain Summary Table

- i. **Streamlit:** Used to create the CoBot chatbot's interactive user interface. It offers a straightforward framework for visualising input and output in real time.
- ii. **PyPDF2:** Manages text extraction from PDF files that have been uploaded. Multi-page carbon trading reports are parsed for additional processing.
- iii. **RecursiveCharacterTextSplitter (LangChain):**To guarantee effective embedding and retrieval, RecursiveCharacterTextSplitter divides lengthy extracted text into digestible portions.
- iv. **GoogleGenerativeAIEmbeddings:** This tool helps with meaningful similarity search by converting text chunks into semantic vector representations using Google's embedding model.

- v. **Facebook AI Similarity Search:** Facebook AI Similarity Search, or FAISS, indexes and stores vectorised document segments for quick access. It makes it possible to match stored knowledge with user queries based on similarity.
- vi. **ChatGoogleGenerativeAI:** Serves as Gemini-Pro's LLM interface, obtaining context and employing sophisticated language modelling to produce accurate responses.
- vii. **PromptTemplate:** Specifies structured prompts that direct the Gemini-Pro model to produce expert-level responses specific to carbon trading.
- viii. **load_qa_chain (LangChain):** Supports the RAG approach by combining language generation and document retrieval into a single pipeline.
- ix. **dotenv:** Makes sure sensitive credentials are safeguarded by safely loading API keys from environment files.
- x. **os (Python standard library):** It is used to control file paths, folder access, and document loading at runtime.

The effective application of CoBot's intelligent document analysis and conversational capabilities is made possible by these tools and libraries taken together. Every element, from data ingestion and semantic embedding to the creation of real-time query responses, is essential to making sure the chatbot runs effectively, precisely, and securely. Utilising cutting-edge technologies like Google Generative AI and FAISS, the system seamlessly integrates generation and retrieval, providing users with accurate and perceptive information about carbon trading.

6. IMPLEMENTATION

6.1. ABOUT THE DATASET:

Source of the Dataset:

1. Source: EY – Carbon Trading: An Emerging Commodity Class (February 2025)

[EY Report on Carbon Trading as an Emerging Commodity](#)

2. Turkish Ministry of Environment – Full Report on Carbon Pricing Developments

[Global State and Trends of Carbon Pricing 2024](#)

3. World Bank – Investment Trends and Market Dynamics in Global Carbon Markets

[World Bank Insights on Carbon Investment Trends](#)

4. European Securities and Markets Authority – Market Supervision & Regulatory Overview

[ESMA Carbon Markets Supervision Report 2024](#)

Dataset Details:

Four carbon trading-related documents (in PDF format) that include pricing models, market trends, policy frameworks, and regulatory insights make up the main dataset. These came from reputable government and business websites. For processing, the documents were uploaded to the "carbontrading" folder path.

Function Used: `get_pdf_text_from_folder(folder_path)`

Using PdfReader, this function reads each PDF file, extracts the text on each page, and then compiles it into a single, sizable raw text block that is then fed into the pipeline.

6.2. PREPROCESSING

To prepare the raw input for semantic search and LLM-based responses, preprocessing was carried out in three crucial steps:

1] **Text Chunking:** RecursiveCharacterTextSplitter divides lengthy texts into digestible 10,000-character segments, preserving context by allowing overlaps.

Function Used: `get_text_chunks(raw_text)`

2] **Vector Embedding Generation:** GoogleGenerativeAIEmbeddings is used to transform each text segment into high-dimensional vector embeddings while preserving semantic meaning.

Function Used: `get_vector_store(text_chunks)`

3/ FAISS indexing: By storing embeddings in an FAISS vector index, an effective similarity search is made possible.

Vector Store Path: *"faiss_index"*

6.3. FUNCTIONAL COMPONENTS AND QUERY PROCESSING

The following steps are taken by the model in response to user queries:

Prompt Design: The system defines the bot as an expert in carbon trading using a prompt template.

Component: *PromptTemplate*

Conversational Chain: Uses *load_qa_chain* to combine user queries and documents into a prompt that is sent to Gemini AI.

Function Used: *Get_conversational_chain()*

User Query Handler: Receives user input through *st.text_input*, uses cosine similarity to retrieve pertinent chunks from the FAISS index, and transmits them to Gemini-Pro along with the query.

Function Used: *User_input(user_question)*

6.4. UI RENDERING AND DEPLOYMENT

The chatbot interface is built using Streamlit, styled with custom CSS. It contains:

- Title and header
- Input box for user query
- Button to refresh document index
- Chat bubble-styled output display
- Footer with project attribution

Main UI Driver: *main()*

Deployed via command: *streamlit run finalcode.py*

6.5. PERFORMANCE AND RESPONSE GENERATION FLOW

The CoBot system follows an end-to-end streamlined flow that emphasizes real-time performance and contextually accurate response generation. When a user inputs a query, it is immediately converted into an embedding using Google's Gemini Embedding model.

This embedding is compared against the FAISS-stored vectors to find the most contextually similar document chunks.

Once relevant chunks are retrieved, they are passed along with the original query into the Gemini-Pro LLM using a RAG (Retrieval-Augmented Generation) chain. This ensures that the LLM not only understands the user's intent but also grounds the response in factual, pre-indexed content. The combination of semantic similarity search and large language model reasoning leads to reliable and specific answers.

Finally, the result is displayed in a visually enriched chat bubble using Streamlit, making the interaction user-friendly. Performance-wise, the design ensures fast retrieval, minimal latency, and consistently relevant outputs, even with a limited number of source PDFs.

Step	Action	Description
1	User uploads carbon trading PDFs	Parses and extracts text
2	Text converted to chunks	Splits into manageable sizes
3	Embeddings generated	Converts text to semantic vectors
4	Vectors indexed	Enables fast similarity searches
5	User enters a query	Accepts natural language questions
6	Relevant vectors retrieved	Finds matching content
7	Gemini-Pro generates answer	Forms final human-like response

Table. 6.1. Workflow Table

7. RESULTS AND DISCUSSIONS

Results

Some of domain-specific carbon trading documents were successfully processed and indexed by the CoBot system, which also made semantic querying possible through the use of Retrieval-Augmented Generation (RAG). Using uploaded sources such as World Bank, ESMA, and EY reports, the application produced accurate responses. With the help of Google Generative AI Embeddings and FAISS, the semantic search made sure that pertinent chunks were retrieved quickly. Domain alignment and contextual accuracy were used to assess the quality of the responses. Real-time responses to the majority of user enquiries, such as those about pricing patterns and market regulations, were precise and supported by references. The user interface of the chatbot, which was constructed with Streamlit, was responsive and easy to use. CoBot showed more flexibility than hardcoded QA systems, particularly when it came to parsing input from multiple documents and providing dynamically generated, document-specific responses.

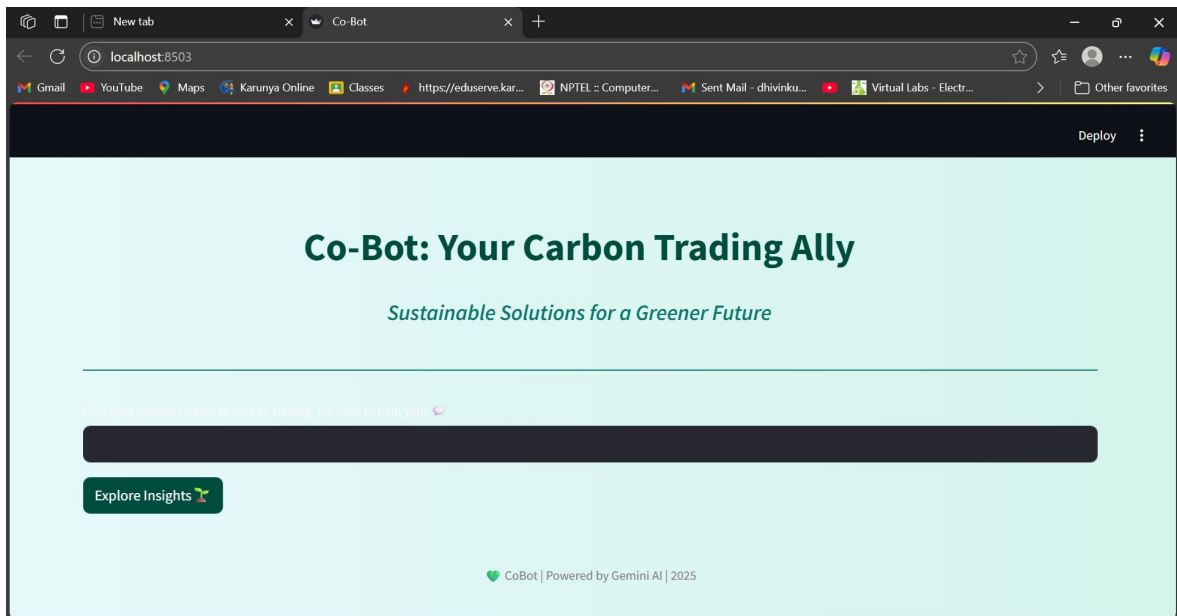


Fig. 7.1. Output of Interface Created

Discussions

The outcomes demonstrate RAG's effectiveness in document-grounded AI applications. CoBot connected static datasets with dynamic conversational capabilities by combining generative reasoning and document embeddings. The quality and coverage of the ingested documents—currently restricted to few PDFs—have a significant impact on the system's performance. Nonetheless, scaling is made easy by the modular design. CoBot offers more in-depth, contextually aware responses than conventional search-based retrieval systems. FAISS maximised retrieval time, while Gemini-Pro guaranteed fluency and relevance in generation. The trade-off between embedding accuracy and retrieval latency is highlighted by the integration. CoBot's ability to support carbon market decisions will be further improved by future dataset expansion and the addition of real-time data sources.

8. CONCLUSION

In this project, CoBot, an AI-powered chatbot system that uses Retrieval-Augmented Generation (RAG) architecture to provide real-time insights and answers about carbon trading, was designed and developed. CoBot provides a dependable and intelligent interface for people, organisations, and policymakers to query market dynamics, regulations, credit types, and other carbon market dimensions by fusing Google Gemini-Pro's language generation capabilities with FAISS-based document retrieval. The system's ability to process domain-specific PDFs, embed textual chunks meaningfully, and generate precise, context-aware responses was demonstrated by the implementation. By enabling multi-document parsing and dynamic interaction, CoBot overcomes drawbacks like limited dataset scope and lack of real-time query handling when compared to existing literature. The current system establishes the groundwork for scalable environmental informatics solutions, despite being limited by the quantity and variety of documents uploaded. Multilingual support, real-time API integration, and adaptive learning for wider policy applications in emissions governance and sustainable finance are possible future improvements.

9. FUTURE ENHANCEMENT

Although CoBot shows that it can use a RAG-powered chatbot framework to respond to real-time carbon trading enquiries, there is still a lot of room for improvement to increase accuracy and operational reach. To improve the dynamic nature of responses, one of the main improvements is broadening the range of input sources, such as live databases, carbon pricing APIs, government policy repositories, and real-time emissions trackers.

By fine-tuning the language model on domain-specific corpora like UNFCCC protocols, trading registries, and verified ESG reports, contextual understanding can be further enhanced. This would improve the chatbot's capacity to manage intricate, multi-intent enquiries involving subtle regulatory differences or comparisons across markets.

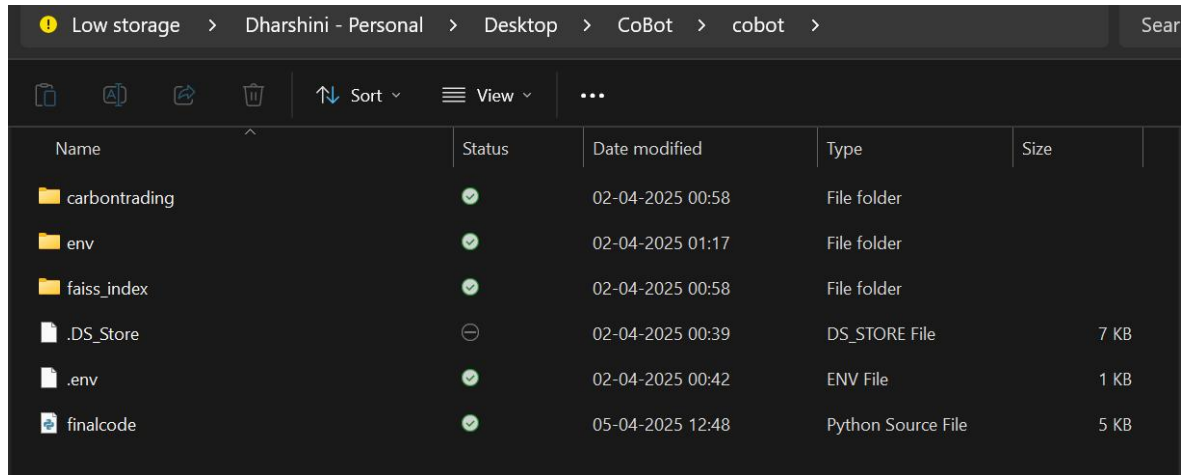
The system may become available to a worldwide audience by incorporating multilingual support, particularly for stakeholders in carbon markets where English is not the primary language. In a similar vein, field officers or policymakers may find voice-based interaction modules that use text-to-speech and speech-to-text technologies easier to use in situations requiring quick consultations.

Technically speaking, system updates could be streamlined by automated document ingestion through user uploads or URL fetch. Over time, real-world applicability may be improved by a mobile version or deployment on embedded systems (like edge devices in regulatory offices). Including feedback learning loops and usage analytics could aid in the model's ongoing adaptation to guarantee scalability and sustainability. CoBot can develop into a comprehensive, real-time decision support tool for stakeholders in the global carbon market with these improvements.

10.APPENDICIES

10.1 FULL CODE AND OUTPUT

PROGRAM FOLDER SCREENSHOT:



FINAL CODE SCREENSHOTS:

```
finalcode.py X
C: > Users > ADMIN > OneDrive > Desktop > CoBot > cobot > finalcode.py > ...

1  import streamlit as st
2  from PyPDF2 import PdfReader
3  from langchain.text_splitter import RecursiveCharacterTextSplitter
4  import os
5  from langchain_google_genai import GoogleGenerativeAIEmbeddings
6  import google.generativeai as genai
7  from langchain_community.vectorstores import FAISS
8  from langchain_google_genai import ChatGoogleGenerativeAI
9  from langchain.chains.question_answering import load_qa_chain
10 from langchain.prompts import PromptTemplate
11 from dotenv import load_dotenv
12
13 # Load environment variables and configure Gemini API
14 load_dotenv()
15 genai.configure(api_key=os.getenv("api_key"))
16
17 # Set up paths
18 PDF_FOLDER = "C:/Users/ADMIN/OneDrive/Desktop/CoBot/cobot/carbontrading"
19 INDEX_PATH = "faiss_index"
20
21 # Streamlit page config
22 st.set_page_config(page_title="Co-Bot", layout="wide")
```



```
finalcode.py X
C: > Users > ADMIN > OneDrive > Desktop > CoBot > cobot > finalcode.py > ...
23
24 # Custom CSS for styling
25 st.markdown("""
26 <style>
27     .stApp {
28         background: linear-gradient(to right, #e6f9f9, #d4f4ec);
29     }
30
31     h1.title {
32         color: #004d40;
33         text-align: center;
34         font-size: 42px;
35         font-weight: bold;
36         padding-top: 20px;
37     }
38
39     h2.header {
40         color: #0f766e;
41         text-align: center;
42         font-size: 24px;
43         font-style: italic;
44     }
45
46     .chat-response {
47         background-color: #004d40;
48         padding: 15px;
49         border-radius: 10px;
50         margin-top: 20px;
51     }
```

finalcode.py X

C: > Users > ADMIN > OneDrive > Desktop > CoBot > cobot > finalcode.py > ...

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

```
.chat-response p {
  color: white;
  font-size: 18px;
}

.footer {
  text-align: center;
  color: gray;
  font-size: 14px;
  margin-top: 40px;
}

input[type="text"] {
  color: #000000 !important;
}

.stTextInput > div > div > input {
  color: black !important;
}

button[kind="primary"] {
  background-color: #004d40 !important;
  color: white !important;
}
```

```
finalcode.py X
C: > Users > ADMIN > OneDrive > Desktop > CoBot > cobot > finalcode.py > ...

78     .stButton>button {
79         background-color: #004d40;
80         color: white;
81         font-weight: bold;
82     }
83
84 </style>
85 """ , unsafe_allow_html=True)
86
87
88 # PDF text extraction
89 def get_pdf_text_from_folder(folder_path):
90     text = ""
91     for file_name in os.listdir(folder_path):
92         if file_name.endswith(".pdf"):
93             pdf_path = os.path.join(folder_path, file_name)
94             pdf_reader = PdfReader(pdf_path)
95             for page in pdf_reader.pages:
96                 text += page.extract_text()
97     return text
98
99 # Chunking long text
100 def get_text_chunks(text):
101     text_splitter = RecursiveCharacterTextSplitter(chunk_size=10000, chunk_overlap=1000)
102     return text_splitter.split_text(text)
103
```

```
finalcode.py X
C: > Users > ADMIN > OneDrive > Desktop > CoBot > cobot > finalcode.py > ...

104 # Create and save FAISS vector index
105 def get_vector_store(text_chunks):
106     embeddings = GoogleGenerativeAIEmbeddings(model="models/text-embedding-004")
107     vector_store = FAISS.from_texts(text_chunks, embedding=embeddings)
108     vector_store.save_local(INDEX_PATH)
109
110 # Build RAG pipeline
111 def get_conversational_chain():
112     prompt_template = """
113     You are a carbon trading expert with extensive knowledge and experience in the field, adept at answering
114     dimensions of carbon trading including but not limited to market dynamics, credit types, transaction p
115     and compliance reporting. Your extensive expertise allows you to provide comprehensive and precise ans
116
117     Context:
118     {context}
119
120     Question:
121     {question}
122
123     Answer:
124     """
125     model = ChatGoogleGenerativeAI(model="gemini-2.0-flash", temperature=0.3)
126     prompt = PromptTemplate(template=prompt_template, input_variables=["context", "question"])
127     return load_qa_chain(model, chain_type="stuff", prompt=prompt)
```

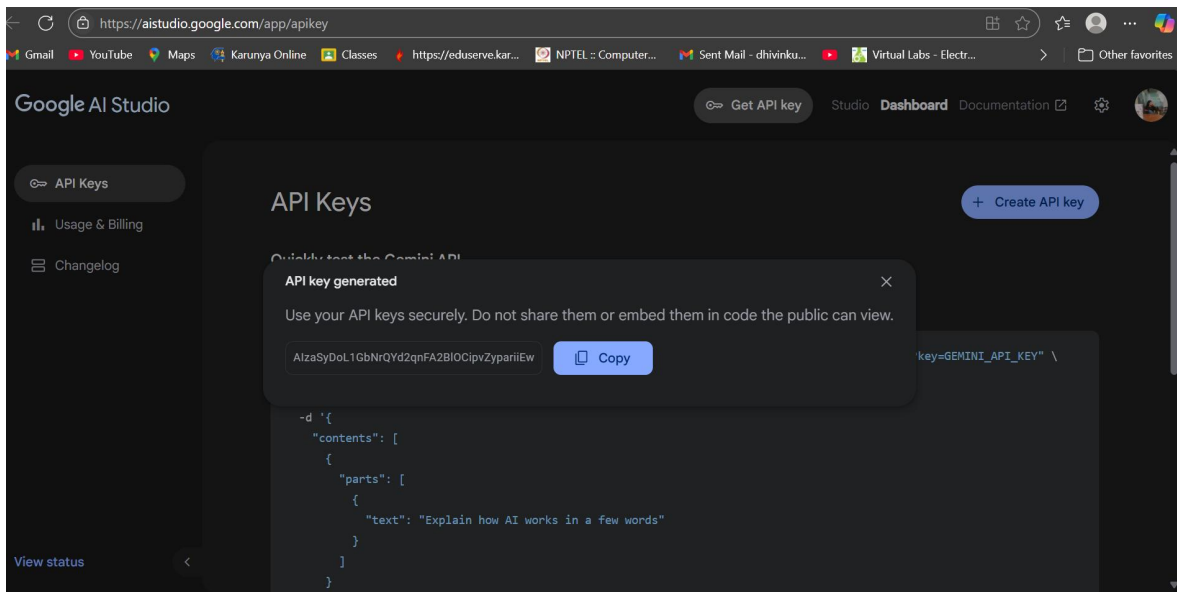
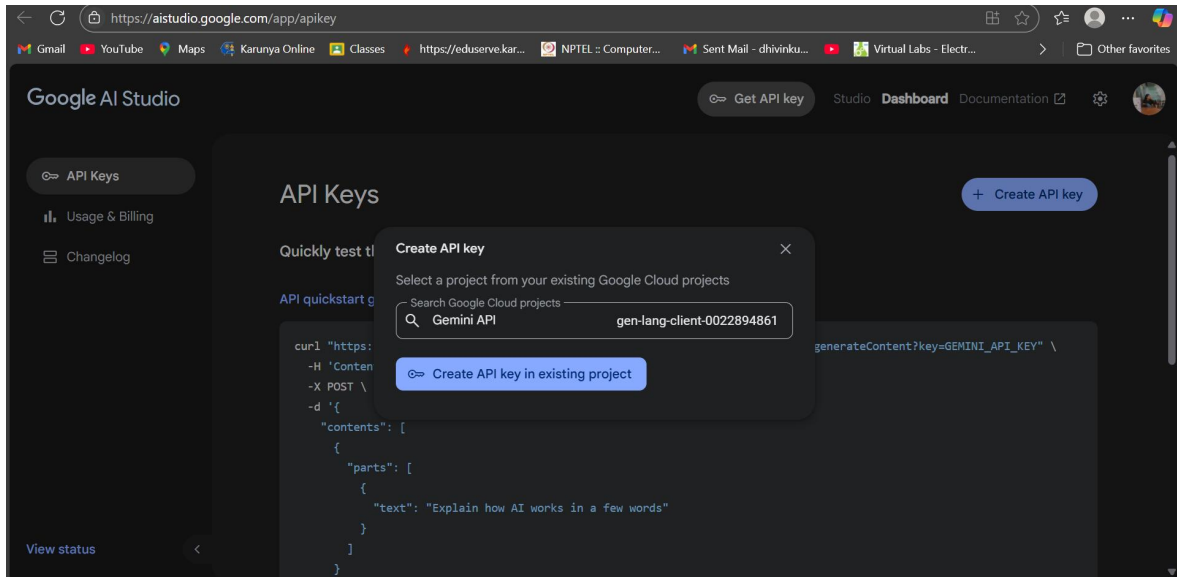
```

finalcode.py X
C: > Users > ADMIN > OneDrive > Desktop > CoBot > cobot > finalcode.py > ...

127 |         return load_qc_chain(model, chain_type="stuff", prompt=prompt)
128
129 | # User input handler
130 | def user_input(user_question):
131 |     embeddings = GoogleGenerativeAIEmbeddings(
132 |         model="models/embedding-001",
133 |         google_api_key=os.getenv("api_key")
134 |     )
135 |     new_db = FAISS.load_local(INDEX_PATH, embeddings, allow_dangerous_deserialization=True)
136 |     docs = new_db.similarity_search(user_question)
137 |     chain = get_conversational_chain()
138 |     response = chain({"input_documents": docs, "question": user_question}, return_only_outputs=True)
139
140 |     st.markdown(f"""
141 |         <div class="chat-response">
142 |             <h4 style='color:white;'>Reply:</h4>
143 |             <p>{response["output_text"]}</p>
144 |         </div>
145 |     """, unsafe_allow_html=True)
146
147 | # Preprocess PDF documents
148 | def process_documents():
149 |     with st.spinner("Processing local PDFs..."):
150 |         raw_text = get_pdf_text_from_folder(PDF_FOLDER)
151 |         text_chunks = get_text_chunks(raw_text)
152 |         get_vector_store(text_chunks)
153 |         st.success("Documents loaded and indexed successfully!")
154
155 | # Main streamlit UI logic
156 | def main():
157 |     st.markdown("<h1 class='title'>Co-Bot: Your Carbon Trading Ally</h1>", unsafe_allow_html=True)
158 |     st.markdown("<h2 class='header'>Sustainable Solutions for a Greener Future</h2>", unsafe_allow_html=True)
159
160 |     if not os.path.exists(INDEX_PATH):
161 |         process_documents()
162
163 |     st.markdown("<hr style='border:1px solid #0f766e;'>", unsafe_allow_html=True)
164
165 |     user_question = st.text_input("Post your queries related to carbon trading, I'm here to help you! 🗨️")
166 |     if user_question:
167 |         user_input(user_question)
168
169 |     if st.button("Explore Insights 📊"):
170 |         process_documents()
171
172 |     st.markdown("""
173 |         <div class='footer'>
174 |             ❤️ CoBot | Powered by Gemini AI | 2025
175 |         </div>
176 |     """, unsafe_allow_html=True)
177
178 | if __name__ == "__main__":
179 |     main()
180

```

TO GENERATE API KEY:



URL TO GENERATE API KEY: <https://aistudio.google.com/app/apikey>

11. REFERENCES

- [1] Cao, Z., Han, M., Wang, J., & Jia, M. (2025). *CarbonChat: Large language model-based corporate carbon emission analysis and climate knowledge Q&A system*. arXiv. <https://doi.org/10.48550/arxiv.2501.02031>
- [2] Vidivelli, S., Manikandan, R., & Dharunbalaji, A. (2024). *Efficiency-driven custom chatbot development: Unleashing LangChain, RAG, and performance-optimized LLM fusion*. CMC—Computers, Materials & Continua.
- [3] Raymond, L. (2024). *Carbon trading*. In Edward Elgar Publishing.
- [4] Mustafa, B. S., & Madhi, Y. E. (2024). *LLM and RAG powered chatbot for the College of Computer Science and Mathematics at the University of Mosul*. *International Research Journal of Innovations in Engineering and Technology*.
- [5] Jiang, P., Sonne, C., Li, W., You, F., & You, S. (2024). *Preventing the immense increase in the life-cycle energy and carbon footprints of LLM-powered intelligent chatbots*. *Engineering*. Elsevier.



12.WORKLOG











International Centre for Education and Research (ICER), VIT-Bangalore

PROJECT STUDENT'S WORKLOG SHEET

Student Name: Dharshini M
 Register Number: 24MSP3070
 Project Title: Cobot - The Carbon Trade Ally
 Domain: AI (Sustainability)
 Tools Learned / used: Retrieval augmented Generation, LLM, Generative AI, Vector Retrieval.

Day	Date	Task	Student's Remarks	Signature of Student with Date	Signature of the Project Mentor
Day 1	2/6/25	Domain Selection Base Paper Selection, Finalising the Title and Problem statement	Selected a project Title	 2/6	

Day 2	3/6/25	Final Abstract (Objective, Dataset, methodology, Tools, Expected Outcome) and Review of Related Literature	Attended Zoroth Review	 3/6	
Day 3	4/6/25	Approach to the Problem write up 2 page (Aim, Research Objective, Research questions, Algorithm, Tools, Dataset, proposed architecture, Expected outcome, References). Perform EDA	Defined objectives-general and specific, selected tools to perform the process	 4/6	
Day 4	4/6/25	Exploratory Data Analysis	Model choosed for the process and data preprocessing is done	 4/6	
Day 5	6/6/25	Implementation	Starting writing and defining functions	 6/6	

Day 6	9/6/25	Implementation	Implementation of Code in Visual Studio.	P.D 9/6	Sg
Day 7	10/6/25	Implementation	Extracting API key for integration of the model	P.D 10/6	Sg
Day 8	11/6/25	Implementation	Elimination of Errors.	P.D 11/6	Sg
Day 9	12/6/25	Second Review	Attended Second Review	P.D 12/6	Sg
Day 10	13/6/25	Results and Discussion	Starting Running Code for obtaining results.	P.D 13/6	Sg
Day 11	16/6/25	Conclusion and future scope	Reached to write conclusion & Scope.	P.D 16/6	Sg

Day 12	17/6/25	Rough Draft of Report submission	Project Report Drafting	P.D 17/6	Sg
Day 13	18/6/25	PPT for Final Review Submission	Completed Third Review	P.D 18/6	Sg 18/6/25
Day 14	19/6/25	PPT for Final Review Submission and Project Report Submission on approval by Project Mentor	Report Drafting	P.D 19/6	Sg
Day 15	20/6/25	Final Review	Completed Report Final Review Completed.	P.D 20/6	Samya 20/6