Implementation of RAG based Question-Answering Application

Khushi Suryavanshi
Department of computer Engineering
RAIT,Dr D Y Patil deemed to be University
Navi Mumbai,India
suryawanshikhushi26@gmail.com

Nupur Thikekar
Department of computer Engineering
RAIT,Dr D Y Patil deemed to be University
Navi Mumbai,India
nupurthikekar@gmail.com

Raj Pawar
Department of computer Engineering
RAIT,Dr D Y Patil deemed to be University
Navi Mumbai,India
rajveerpawar023@gmail.com

Shweta Ashtekar
Department of computer Engineering
RAIT,Dr D Y Patil deemed to be University
Navi Mumbai,India
shweta.ashtekar@rait.ac.in

2025 International Conference on Data Science and Business Systems (ICDSBS) | 979-8-3315-8560-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDSBS63635.2025.11031968

***Abstract*—This project proposes the use of Streamlit, Python, LangChain, Chroma, and Google's Gemini Model to develop a Retrieval-Augmented Generation (RAG) based question-answering chat bot.With the use of cutting-edge language models and RAG architecture, this chatbot can ac curately answer user questions and retrieve pertinent document portions. After loading docu ments, the system uses the Gemini model to create succinct responses after processing them through a vector storage for similarity-based retrieval. For those looking for exact information from large text sources, this solution offers an effective, interactive experience with Streamlit as the user interface. To provide a strong, contextually aware question-answering tool, the project uses Google's Gemini Model for intelligent replies, Chroma for vector storage, and LangChain for document management.**

***Index Terms*—Retrieval-Augmented Generation (RAG), Gemini Model, LangChain, ChromaDB ,Streamlit, Google Generative AI, Context-Aware Question Answering, Semantic Embeddings, Vector Database, Natural Language Processing (NLP), Document Analysis, Research Paper Summarization, Chatbot Interface, Technical Content Understanding, PDF Document Querying**

I. Introduction

Imagine asophisticated AI-powered chat interface where users can freely inquire about complex computer vision concepts without getting lost in technical jargon. This is where Our RAG-based document analysis technology comes in quite handy in this situation. Built with a combination of state-of-the-art technologies, such as the LangChain framework and Google's Gemini model, it fosters an easy-to-use and thorough research paper discovery experience, However, the system does more than just query-response exchanges, though. A user friendly interface is developed using Streamlit, ChromaDB, and the LangChain framework, providing researchers and developers with a seamless experience and succinct, clear responses. Additionally, integration with Google's embedding and language models enables efficient doc ument processing and real-time answer production. The user interface's responsiveness and accessibility are guaranteed by the popular Python web framework Streamlit, which easily ad justs to different use cases and research requirements. The increasing complexity of research articles necessitates more intelligent access to informa tion. By automating the extraction and comprehension of technical content, our system—which is based on RAG architecture and contemporary AI technologies—creates a more inclusive and accessible learning environment for all. It makes research more effective and interesting by relieving researchers and students of the need to manually analyze papers.

II. Related Works and Baseline Models

[1] This paper proposes improvements to the standard Retrieval-Augmented Generation (RAG) framework by introducing graph-based retrieval methods. The authors emphasize that traditional retrieval approaches often fail to capture complex semantic relationships. To resolve this, they incorporate robust graph structures and neighborhood propagation techniques, which enhance the relevancy of retrieved documents. This enriched retrieval mechanism ultimately boosts the performance of downstream tasks like question answering. The study contributes significantly by integrating structured knowledge representations into the RAG pipeline, improving contextual understanding and retrieval precision.

[2] OpenRAG introduces a flexible, open-source architecture for Retrieval-Augmented Generation tailored to personalized learning applications. The paper emphasizes modular design, allowing integration with various data sources and large language models (LLMs). Through experiments in educational settings, the architecture shows improved adaptability to learner preferences, enabling more accurate and context-aware content generation. By supporting scalable personalization, OpenRAG demonstrates potential for enhancing e-learning systems, thus contributing to the

broader adoption of RAG in educational technology

[3] This study introduces a novel Context Awareness Gate (CAG) designed to enhance the filtering and selection process within Retrieval-Augmented Generation systems. The gate selectively allows only the most relevant retrieved content to influence the generative output, addressing the problem of noise from irrelevant documents. The experiments show that incorporating CAG leads to better factual consistency and reduces hallucinations in the output of LLMs. This approach marks an advancement in context-sensitive generation, especially in complex question-answering tasks

[4] This paper explores the integration of Chain-of-Thought (CoT) prompting with Retrieval-Augmented Generation, focusing on the improvement of reasoning tasks. Through detailed empirical evaluations, the authors demonstrate that CoT-guided prompts allow LLMs to better utilize retrieved knowledge in a step-by-step manner. The results show enhanced performance across logical reasoning and factual QA benchmarks. The study validates that combining CoT with RAG leads to more interpretable and accurate outputs, offering insights into better prompt engineering strategies for LLMs

[5] This paper proposes a novel retrieval strategy aimed at improving the efficiency and effectiveness of the RAG pipeline. The authors introduce a multi-stage filtering method that narrows down search results using semantic similarity and relevance feedback loops. They demonstrate that this technique outperforms conventional dense retrieval in terms of precision and response quality. By refining the retrieval phase, the study significantly contributes to optimizing the end-to-end RAG workflow.

[6] In this work, structured knowledge from knowledge graphs is incorporated into a hybrid architecture that improves RAG. The model obtains contextually aligned documents by connecting query items to graph nodes, which are subsequently fed into a generative LLM. Particularly for domain-specific queries, the approach enhances the interpretability and factual foundation of responses. The study successfully pushes the limits of LLM-based question answering systems by combining neural and symbolic techniques.

[7] LocatingGPT presents a multi-modal RAG system for document retrieval that combines visual and textual data. In order to align embeddings from various modalities and enable more precise content localization, the authors use contrastive learning approaches. The
framework improves the model's comprehension and production of well-founded answers from mixed-media sources. Applications in scientific literature, medical imaging, and other fields needing multi-modal cognition will be most impacted by this invention.

[8] The authors use RAG in the biomedical field to determine the causal links between occurrences in this paper. Their proposal is to reduce misinformation by using fine-grained denoising techniques to sanitize the retrieved documents prior to generation. A component of their optimization approach is multi-task learning, which strikes a compromise between generating fluency and retrieval accuracy. Improved performance on biomedical datasets indicates that the method is a useful addition to automated information extraction in the life sciences and healthcare.

[9] This research focuses on simplifying the embedding process in RAG to reduce computational overhead without compromising performance. The authors design a lightweight vectorization strategy and apply it to a chatbot model. The results show that their method retains accuracy while significantly lowering memory and processing time. The study emphasizes practical deployment of RAG-based systems in low-resource environments, providing a foundation for scalable real-time QA systems.

[10] Despite extraction challenges, this paper likely presents an overview of integrating large language models with RAG frameworks. The emphasis appears to be on architectural cohesion between retrieval and generation modules. Techniques discussed may include embedding alignment, query reformulation, and document reranking. The study underlines the importance of seamless integration for achieving high-quality responses in LLM-powered applications like chatbots and virtual assistants.

[11] A user friendly interface is developed using Streamlit, ChromaDB, and the LangChain framework, providing researchers and developers with a seamless experience and succinct, clear responses. Google Gemini Enable to provide real-time information speciFIc to WPI Additionally, integration with Google's embedding and language models enables efficient document processing and real-time answer production. The user interface's responsiveness and accessibility are guaranteed by the popular Python web framework Streamlit, which easily adjusts to different use cases and research requirements.The paper mainly focusses To create a question-answering system where technical research concepts are made approachable and everyone gets a chance to understand complex papers.

[12] Google's Gemini Model for generating embeddings and intelligent answers, and PyPDFLoader for precise text extraction from documents. The application is deployed on Firebase Cloud Platform,which supports secure database management and authentication. By integrating these advanced technologies, the system provides accurate and context-aware answers to user queries,enabling seamless interaction with complex technical documents. Canopy is an open-source framework built on top of Pinecone that allows users to

streamline the process of building a RAG application .Canopy is compatible with all OpenAI LLM which provides great exibility to implement a chat engine. Canopy a full RAG work with three main components:knowledge base,context engine, and chat engine. The user only needs to provide the text data and the knowledge base chunks and calculate the embeddings to upsert the data to the vector database.

[13] Many existing tools for analyzing research papers, like regular PDF readers and simple search features, are not very effective for fully understanding the content. Their widespread utilizationacross diverse sectors highlights their capability to enhance services in customer support, education,and healthcare, among others. This technological leap demonstrates not only the versatility of LLMs in addressing complex informational and computational tasks, but also their role in streamlining processes and improving information accessibility and e ciency in many industries.They lack interactive question nasking features and don't offer a deeper understanding of the material.

[14] To deliver accurate contextually aware responses in real time, the system leverages the efficiency of retrieved generation (RAG). RAG systems analyze the source document by dividing it into smaller, manageable segments and creating semantic embeddings for precise information RETRIEVAL. THE system then combines the relevant context with large language models to produce well-informed responses

[15] removing the need for researchers and students to manually analyze papers, it increases the effectiveness and interest of study. This work's primary goal is to create an advanced chat interface driven by AI that allows people to freely ask questions about intricate computer vision ideas without becoming bogged down in technical jargon. This is the point at which RAG-based document analysis technology is quite helpful. Constructed using a blend of cutting-edge technologies, including Google's Gemini model and the Lang Chain framework.

III. Proposed Methodology

The system design for the RAG (Retrieval-Augmented Generation) application utilizing the Gemini Model is based on a cloud-driven architecture, prioritizing scalability and performance. The frontend, built with Streamlit, delivers an intuitive and interactive user interface, while the backend leverages the LangChain framework to streamline the retrieval and response generation processes. Core features include ChromaDB for efficient vector storage and document retrieval, Google's Gemini Model for generating embeddings and intelligent answers, and PyPDFLoader for precise text extraction from documents. The application is deployed on Firebase Cloud Platform,which supports secure database management and authentication. By integrating these advanced technologies, the system provides accurate and context-aware answers to user queries,enabling seamless interaction with complex technical documents. Canopy is an open-source framework built on top of Pinecone that allows users to streamline the process of building a RAG application. Canopy is compatible with all OpenAI LLM which provides great exibility to implement a chat engine. Canopy o ers a full RAGwork with three main components:knowledgebase,context engine, and chat engine . The user only needs to provide the text data and the knowledge base chunks and calculate the embeddings to upsert the data to the vector database. A Tools Methods The paper uses following tools as well as methods as

- Frontend: : Streamlit is used to create the interactive chat interface as a Python-based web application.
- Backend: LangChain framework for orchestrating the RAG pipeline and managing document processing.
- Document Processing: PyPDFLoader is used for text extraction and PDF loading.Content is divided by RecursiveCharacterTextSplitter.
- Vector Store Integration: ChromaDB is used to store and retrieve document segments.Semantic embeddings are produced by Google's embedding model.
- LLM Integration: generates responses using Google's Gemini 1.5 Pro model. Controls the token and temperature settings for the best possible replies.
- Context Management: A retrieval chain is used to provide context-sensitive answers utilizes the "stuff" method to combine document fragments. system prompts ensure technical accuracy in responses.
- Response Generation: Combines retrieved context with user queries to produce concise,accurate answers while maintaining alignment with the source document.
- Query Processing: Understand the natural language queries, performs context-aware search, and effectively formats and presents responses.

IV. Experiments

A. Development Environment

Code Editor the suggested IDEs for development are PyCharm or Visual Studio Code. The coding and debugging experience is improved by these IDEs' strong support for Python development,which includes features like an integrated terminal, Git version management, and broad extension support for Python debugging. In order to use the Gemini API, you must also have a Google Cloud Platform (GCP) account. Throughout the project, GCP offers tools for regulating quotas, tracking consumption, and managing API keys, guaranteeing safe and effective access to the Gemini model and other services. Javascript Libraries We used npm to install the libraries.We used firebase and its related libraries to make the website.

B. API and Cloud Services

Access to the Gemini API and embedding models is made possible by the Google AI Platform,which also offers secure API authentication. For effective data retrieval, ChromaDB facilitates vector store configuration, persistence setup, and local or cloud-based installation.Without disclosing private information, environment variables like GOOGLE API KEY and others guarantee safe, adaptable setting.

C. Frontend and Backend Development:

The system is built using Streamlit for the frontend and LangChain for backend orchestration. Streamlit enables rapid deployment of a responsive web interface with real-time user input and dynamic query handling. On the backend, LangChain manages the entire RAG pipeline by coordinating document processing, semantic retrieval, prompt engineering, and response generation, ensuring seamless interaction between the user interface, vector store, and language model.

D. Document Ingestion and Processing:

Documents are uploaded in PDF format and parsed using PyPDFLoader, which extracts raw text from structured pages. The content is then segmented using RecursiveCharacter-TextSplitter, which divides the text into overlapping chunks to preserve contextual continuity during retrieval.

E. Semantic Embedding and Vector Store Integration:

Each document chunk is transformed into high-dimensional vectors using Google's Generative AI Embeddings (models/embedding-001). These vectors are stored in ChromaDB, a high-performance vector database that supports similarity search. ChromaDB enables fast and efficient retrieval of semantically relevant document fragments.

F. Language Model Integration:

The system uses Google's Gemini 1.5 Pro model as the core LLM for generating natural language responses. The model parameters such as temperature and token limits are fine-tuned to balance coherence, creativity, and factual accuracy.

G. Context Management and Response Generation:

The system employs a retrieval chain to combine semantically relevant chunks from the vector store with user queries. Using the "stuff" method, document fragments are merged into a coherent input for the language model. A system prompt ensures domain specificity, technical tone, and concise output. The Gemini 1.5 Pro model then generates accurate, context-aware responses grounded in the source content, with a focus on minimizing redundancy and preserving factual correctness.

H. Query Processing and Output Handling:

User queries are interpreted using natural language parsing and tokenization. The system performs contextual similarity search to identify the most relevant information, which is then presented through a clean and responsive chat interface. Error-handling routines are in place to address ambiguous or low-confidence queries, enhancing the reliability of the system.

I. Security, Integration, and Scalability:

API access to Google's Gemini model and ChromaDB is secured through environment variables managed by the dotenv package, preventing key exposure. The architecture supports modular expansion, enabling future integration of multi-document support, real-time data updates, diverse file format ingestion, and deployment via cloud platforms such as Firebase or Google Cloud Platform. Robust exception handling and request validation mechanisms further improve system stability and scalability.

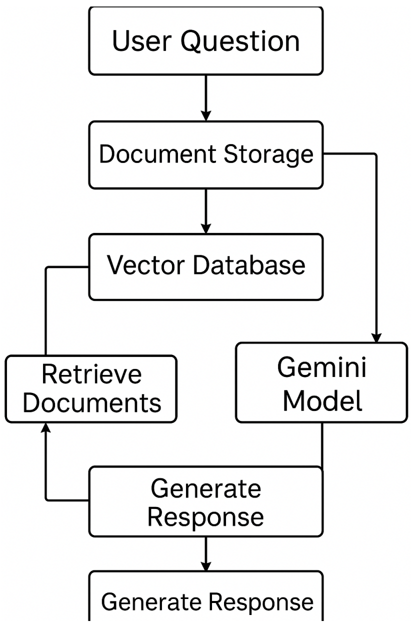

Fig. 1. Working Flow Diagram

V. Results

With the help of Streamlit, LangChain, Chroma, and Google's Gemini Model, the RAG-based question-answering chatbot offers a workable way to make difficult research articles easier to follow. The chatbot provides users with succinct, contextually relevant responses by combining document retrieval with sophisticated language models, eliminating the need for laborious manual searches. With the help of this approach, users may engage with complex technical knowledge in a natural way, making research easier and more efficient. The application overcomes major drawbacks in conventional QA systems by combining document chunking, semantic embedding, and rapid engineering to guarantee that responses stay accurate and consistent with the source content. This method increases user trust because the answers are always based on the original document, in addition to improving answer accuracy. In conclusion, the chatbot is capable of answering queries based on the PDF documents loaded by retrieving the relevant chunks and passing on to Gemini. Hence, generating contextually correct answers which are grounded in retrieved documents.

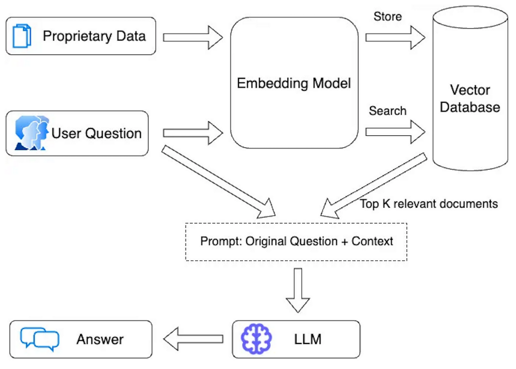

Fig. 2. Architectural Diagram

VI. Conclusion

In this project, we have successfully designed and implemented a Retrieval-Augmented Generation (RAG)-based question-answering chatbot utilizing advanced technologies such as Google's Gemini 1.5 Pro, LangChain, ChromaDB, and Streamlit. The system demonstrates the seamless integration of modern language models with semantic vector databases, delivering accurate and contextually grounded responses from complex documents.

The proposed architecture addresses critical limitations of traditional PDF readers and standalone language models by combining the power of large language models with real-time document retrieval. Unlike conventional approaches that struggle with long-context management and factual grounding, our system retrieves document fragments relevant to a user's query and generates answers that are both concise and technically accurate. This enhances the overall comprehension of technical content and reduces the cognitive load on users when engaging with dense research materials.

Our use of LangChain provides a modular and scalable pipeline for chaining retrieval and generation tasks, while ChromaDB offers high-performance vector storage and fast similarity search capabilities. Additionally, Streamlit enables a user-friendly, interactive interface that supports real-time query input and dynamic answer display. Together, these components form a comprehensive, end-to-end solution for document-based conversational AI applications.

The implementation validates the potential of RAG systems in academic and research settings, particularly for users seeking fast and precise insights from structured documents. Moreover, our system's architecture allows for future expansion, including multi-document support, fine-tuning on domain-specific corpora, integration with various file formats (e.g., DOCX, TXT, HTML), and deployment via cloud platforms for wider accessibility.

In summary, this project contributes a practical and scalable approach to intelligent document analysis, bridging the gap between complex textual data and natural language understanding. It lays a strong foundation for the future development of AI-driven knowledge assistants, educational tools, and enterprise document processing systems that demand real-time, accurate, and context-sensitive information retrieval.

References

[1] M. Rani, B. K. Mishra, D. Thakker and M. N. Khan, "To Enhance Graph-Based Retrieval-Augmented Generation (RAG) with Robust Retrieval Techniques," 2024 18th International Conference on Open Source Systems and Technologies (ICOSST), Lahore, Pakistan, 2024, pp. 1-6, doi: 10.1109/ICOSST64562.2024.10871140. keywords: Accuracy;Retrieval augmented generation;Pipelines;Knowledge based systems;Training data;Knowledge graphs;Cognition;Vectors;Real-time systems;Diabetes;Retrieval-augmented generation;Large language model;Knowledge graph;Diabetes;Healthcare;Graph-based Retrieval-augmented generation,

[2] R. Shan, "OpenRAG: Open-source Retrieval-Augmented Generation Architecture for Personalized Learning," 2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC), Xiamen, China, 2024, pp. 212-216, doi: 10.1109/ICAIRC64177.2024.10900069. keywords: Retrieval augmented generation;Pipelines;Systems architecture;Learning (artificial intelligence);Computer architecture;User interfaces;User experience;Robots;Optimization;Indexing;Retrieval-Augmented Generation (RAG);open source;architecture;system design;vector database;orchestration;information retrieval;Natural Language Processing (NLP);large language models (LLMs);personalized learning,

[3] M. H. Heydari, A. Hemmat, E. Naman and A. Fatemi, "Context Awareness Gate for Retrieval Augmented Generation," 2024 15th International Conference on Information and Knowledge Technology (IKT), Isfahan, Iran, Islamic Republic of, 2024, pp. 260-264, doi: 10.1109/IKT65497.2024.10892659. keywords: Accuracy;Statistical analysis;Large language models;Retrieval augmented generation;Pipelines;Context awareness;Logic gates;Question answering (information retrieval);Vectors;Context modeling;Retrieval-Augmented Generation;Hallucination;Large Language Models;Open Domain Question Answering,

[4] Y. Zhao, H. Cao, X. Zhao and Z. Ou, "An Empirical Study of Retrieval Augmented Generation with Chain-of-Thought," 2024 IEEE 14th International Symposium on Chinese Spoken Language Processing (ISCSLP), Beijing, China, 2024, pp. 436-440, doi: 10.1109/ISCSLP63861.2024.10800207. keywords: Training;Retrieval augmented generation;Noise;Performance gain;Chatbots;Information retrieval;Cognition;Data mining;Faces;generative dialogue model;large language model;chain-of-thought;retrieval augmented generation,

[5] S. Mengmeng, L. Zhibin, W. Qingwei, H. Man and X. Feiyang, "An Effective Retrieval Method to Improve RAG Performance," 2024 7th International Conference on Data Science and Information Technology (DSIT), Nanjing, China, 2024, pp. 1-5, doi: 10.1109/DSIT61374.2024.10881380. keywords: Accuracy;Scalability;Large language models;Retrieval augmented generation;Performance gain;Information filters;Hybrid power systems;Multilingual;Resource management;Information technology;large language model;retrieval augmented generation;recursive retrieval,

[6] S. Gijre, R. Agrawal, P. Laddha and G. Keswani, "Enhancing Question-Answering with Knowledge Graph Retrieval and Generation using LLMs," 2024 International Conference on Artificial Intelligence and Quantum Computation-Based Sensor Application (ICAIQSA), Nagpur, India, 2024, pp. 1-6, doi: 10.1109/ICAIQSA64000.2024.10882212. keywords: Text analysis;Quantum computing;Large language models;Soft sensors;Retrieval augmented generation;Knowledge graphs;Information retrieval;Vectors;Maintenance;Standards;Knowledge Graph;Large Language Models;LangChain;Neo4j;Natural Language Processing;Retrieval Augmented Generation,

[7] Z. Chen et al., "LocatingGPT: A multi-modal document retrieval method based on retrieval-augmented generation," 2024 IEEE 9th International Conference on Data Science in Cyberspace (DSC), Jinan, China, 2024, pp. 232-239, doi: 10.1109/DSC63484.2024.00038. keywords: Location awareness;Accuracy;Retrieval augmented generation;Information retrieval;Cognition;Robustness;Question answering (information retrieval);Multilingual;Information technology;Indexing;Document localization;Multi-modal document retrieval;Multi-step reasoning;Cross-language information retrieval;Retrieval-Augmented Generation,

[8] J. Zhao, L. Li, W. Ning, J. Hao, Y. Fei and J. Huang, ”Multiple Optimization with Retrieval-Augmented Generation and Fine-Grained Denoising for Biomedical Event Causal Relation Extraction,” 2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Lisbon, Portugal, 2024, pp. 4040-4043, doi: 10.1109/BIBM62325.2024.10822677. keywords: Filtering;Noise reduction;Retrieval augmented generation;Semantics;Syntactics;Data augmentation;Data models;Indexes;Data mining;Optimization;biomedical event causal relation extraction;targeted selection of external knowledge;retrieval-augmented generation;fine-grained data denoising,

[9] M. R. Putri, A. Y. Husodo and B. Irmawati, ”Simplification of Embedding Process in Retrieval Augmented Generation for Optimizing Question Answering Chatbot Model,” 2024 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), Mataram, Indonesia, 2024, pp. 665-670, doi: 10.1109/COMNETSAT63286.2024.10862926. keywords: Measurement;Training;Costs;Satellites;Scalability;Large language models;Retrieval augmented generation;Chatbots;Question answering (information retrieval);Indexing;Chatbot;Embedding;Large Language Model;Prompt;Retrieval Augmented Generation,

[10] B. Tural, Z. Örpek and Z. Destan, ”Retrieval-Augmented Generation (RAG) and LLM Integration,” 2024 8th International Symposium on Innovative Approaches in Smart Technologies (ISAS), İstanbul, Turkiye, 2024, pp. 1-5, doi: 10.1109/ISAS64331.2024.10845308. keywords: Accuracy;Databases;Large language models;Retrieval augmented generation;Semantics;Information retrieval;Natural language processing;Complexity theory;Context modeling;Retrieval-Augmented Generation;Large Language Models;Information Retrieval;Natural Language Processing,

[11] L.Huangetal.,“ASurveyonHallucination in Large Language Models: Principles, Taxonomy,Challenges, and Open Questions.” 2023.

[12] G.Cullen,“Introducing Canopy: Aneasy, free, and exible RAG framework powered by Pinecone — Pinecone.” Accessed: Feb. 26, 2024. [Online]. Available:https://www.pinecone.io/blog/canopy-rag-framework

[13] M.U.Hadietal., “A Survey on Large Language Models: Applications, Challenges, Limitations,and Practical Usage,” preprint, Jul. 2023. doi: 10.36227/techrxiv.23589741.v1.

[14] E. AdamopoulouandL.Moussiades, “Chatbots: History, technology, and applications,”Mach. Learn. Appl., vol. 2, p. 100006, Dec. 2020, doi: 10.1016/j.mlwa.2020.100006.

[15] S. K. Singh, S. Kumar, and P. S. Mehra, “Chat GPTGoogleBardAI:AReview,”in 2023 International Conference on IoT, Communication and Automation Technology (ICICAT),Gorakhpur, India: IEEE, Jun. 2023, pp. 1–6. doi: 10.1109/ICICAT57735.2023.10263706.