University of Hertfordshire UH

School of Computer Science

BSc (Hons) Computer Science (Artificial Intelligence)

6COM2017-0905-Artificial Intelligence Project

Final Report

March 2024

ChatBot: Chat With PDF, Website, Youtube Videos, and Intelligent Conversations with Chatbots using Voice.

Dhyan Nilesh Patel

Supervised By – Frank Foerster

Abstract

In today's digital era, our lives are increasingly intertwined with technology, and the demand for intelligent, interactive solutions is ever-growing. This Final Year Project endeavours to meet this demand by introducing a versatile suite of chatbots designed to enhance user interactions across diverse content domains.

The project comprises four distinctive chatbots, each tailored to cater to specific user needs. The "ChatWithPDF ChatBot" revolutionizes the way users engage with PDF documents, allowing for seamless search and interaction across multiple files simultaneously. Powered by the Langchain library and OpenAI's advanced capabilities.

The "ChatWithYoutube ChatBot" extends this interactive capability to YouTube videos, enabling users to search and engage with content effortlessly. Leveraging the same robust foundation of Streamlit, Langchain, and OpenAI, this chatbot ensures a user-friendly interface and comprehensive understanding of user queries related to YouTube video content.

For those seeking to engage with their website content more effectively, the "ChatWithWebsite ChatBot" provides a convenient solution. Streamlit, Langchain, and dotenv collectively empower this chatbot to understand and respond to user queries regarding website content, fostering a dynamic and memory-enabled interaction.

Lastly, the "ChatWithVoice ChatBot" takes user interaction to the next level by allowing voice-based engagement with ChatGPT. Developed with Deepgram, Elevenlabs, and OpenAI, this chatbot responds dynamically to user queries, enabling a novel and hands-free conversational experience.

Throughout the development process, emphasis has been placed on user-friendliness, clear documentation, and the integration of the latest technologies to ensure the chatbots' adaptability and efficiency. This Final Year Project presents a comprehensive solution to the growing demand for intelligent and interactive systems across PDFs, YouTube videos, websites, and voice interactions.

Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the completion of the project.

First and foremost, I am deeply thankful to my project supervisor Frank Foster for his invaluable guidance, support, and encouragement throughout the duration of the project. His expertise, constructive feedback, and unwavering dedication have been instrumental in shaping the direction and success of this endeavour.

I extend my heartfelt appreciation to the developers and contributors of the libraries and modules utilized in this project, namely Streamlit, Langchain, PyPDF2, Deepgram, Elevenlabs, OpenAI, and dotenv. Their innovative tools and resources have provided the foundation and functionality necessary for the implementation of the chatbot solutions presented in this report.

Finally, I would like to thank my peers and friends for their encouragement, insightful discussions, and assistance in evaluating my Chatbots. Their collaboration and camaraderie have fostered a conducive environment for learning and innovation.

In conclusion, I am immensely grateful to everyone who has contributed to this project in any capacity. Your collective efforts have made this endeavour possible and have enriched my academic and personal growth in profound ways.

Thank you.

Table of Contents

3. Introduction

In today's digital age, accessing and interacting with various types of online content has become an integral part of daily life. From reading PDF documents to watching YouTube videos and browsing websites, users rely on digital platforms for information, entertainment, and communication. However, traditional interfaces and navigation methods often lack efficiency and convenience, leading to frustration and inefficiency for users.

The emergence of conversational interfaces, powered by artificial intelligence and natural language processing technologies, presents an opportunity to revolutionize the way users interact with digital content. ChatBots, or conversational agents, have gained popularity as a user-friendly and intuitive means of accessing information and services online. By simulating human-like conversations, ChatBots enable users to communicate in natural language, ask questions, and receive personalized responses, thus enhancing the overall user experience **[1].**

The motivation behind this project stems from the recognition of the need for more efficient and user-centric methods of accessing and interacting with digital content. Traditional methods such as manual navigation and keyword searches can be time-consuming and cumbersome, especially when dealing with large volumes of content. Additionally, users with limited technological proficiency or accessibility needs may face barriers in accessing online information using conventional interfaces **[2].**

By developing multi-functional ChatBots tailored to specific content formats, such as PDF documents, YouTube videos, websites, and voice-based interactions, this project aims to address these challenges and enhance user interaction with digital content. The goal is to create intuitive and user-friendly interfaces that allow users to seamlessly engage with content using natural language queries and commands. By leveraging cutting-edge technologies and methodologies, such as natural language processing, machine learning, and speech recognition, the project seeks to empower users with more efficient and personalized ways of accessing and interacting with online content.

Overall, the background and motivation for this project lie in the desire to improve user experiences and accessibility in the digital realm. By developing innovative ChatBots that cater to diverse user needs and preferences, the project aims to contribute to the advancement of conversational interfaces and the broader goal of making online content more accessible and user-friendly for all.

3.1 Problem Statement

In the realm of digital content consumption, users often encounter challenges and inefficiencies when accessing and interacting with various types of online content. Despite the abundance of information available on the internet, navigating through documents, videos, and websites can be cumbersome and time-consuming, leading to frustration and reduced productivity for users.

One of the key challenges faced by users is the lack of efficient and user-friendly interfaces for accessing and interacting with digital content. This results in users spending excessive time and effort in locating and retrieving the information they need, leading to a poor user experience. Furthermore, the lack of intelligent and context-aware interfaces poses a significant challenge for users seeking to engage with digital content in a more natural and conversational manner. Current interfaces often lack the ability to understand and interpret user queries in the context of the content, leading to generic and irrelevant responses that do not address the user's specific needs **[3].**

In summary, the problem statement for this project revolves around the inefficiencies and challenges faced by users when accessing and interacting with digital content and PDFs. By identifying the gaps in existing methods and technologies, and by leveraging advanced AI and NLP techniques, the project aims to develop innovative solutions that enhance the user experience and accessibility of online content for all users.

3.2 Project Aims and Objectives

The primary aim of this project is to develop an integrated system of intelligent ChatBots capable of enhancing the user experience and accessibility of various types of digital content. By leveraging advanced artificial intelligence (AI) and natural language processing (NLP) techniques, the project seeks to address the inefficiencies and challenges faced by users when accessing and interacting with digital content, including PDF documents, YouTube videos, websites, and voice-based interactions. Another overarching aim for this project is to explore the potential of conversational AI for user interaction with various information sources. This includes investigating how conversational interfaces can provide a more intuitive and efficient way to access and understand information compared to traditional search methods.

The specific objectives of the project are as follows. Firstly, the project aims to create ChatBots tailored to different content formats, including PDF documents, YouTube videos, and websites. Each ChatBot will be designed to understand and respond to user queries and commands in a contextually relevant manner, thereby providing users with more intuitive and efficient ways to access and interact with digital content. In addition to text-based interactions, the project aims to integrate voice-based interactions into the

ChatBots, allowing users to interact with digital content using natural language commands. This will enable users to access content hands-free and facilitate more seamless and intuitive interactions.

Another key objective of the project is to enhance the user experience of accessing and interacting with digital content. By developing ChatBots with intelligent conversational capabilities, the project aims to streamline the process of searching for and retrieving information, reducing the time and effort required for users to find the content they need. In addition to text-based interactions, the project aims to integrate voice-based interactions into the ChatBots, allowing users to interact with digital content using natural language commands. This will enable users to access content hands-free and facilitate more seamless and intuitive interactions.

Overall, the project aims to empower users to interact with content in a more efficient, intuitive, and accessible manner, ultimately enhancing their ability to find, retrieve, and engage with information across different mediums. Through the achievement of these objectives, we strive to create chatbot solutions that make a positive impact on users' everyday lives.

4. Literature Review

The literature review provides a comprehensive overview of the existing knowledge and research related to the development and implementation of intelligent ChatBots, focusing on their applications, methodologies, strengths, weaknesses, and opportunities. This review aims to elucidate the current state of the field for non-experts and identify key insights and practices that inform the present project.

4.1 History

The history of ChatBots trace back to the mid-20th century, marked by significant advancements in the field of artificial intelligence (AI) and natural language processing (NLP). One of the seminal events in the development of ChatBots is the introduction of the Turing Test by Alan Turing in 1950 **[4].** The Turing Test proposed a criterion for determining whether a machine can exhibit intelligent behaviour indistinguishable from that of a human, laying the groundwork for subsequent research in conversational AI.

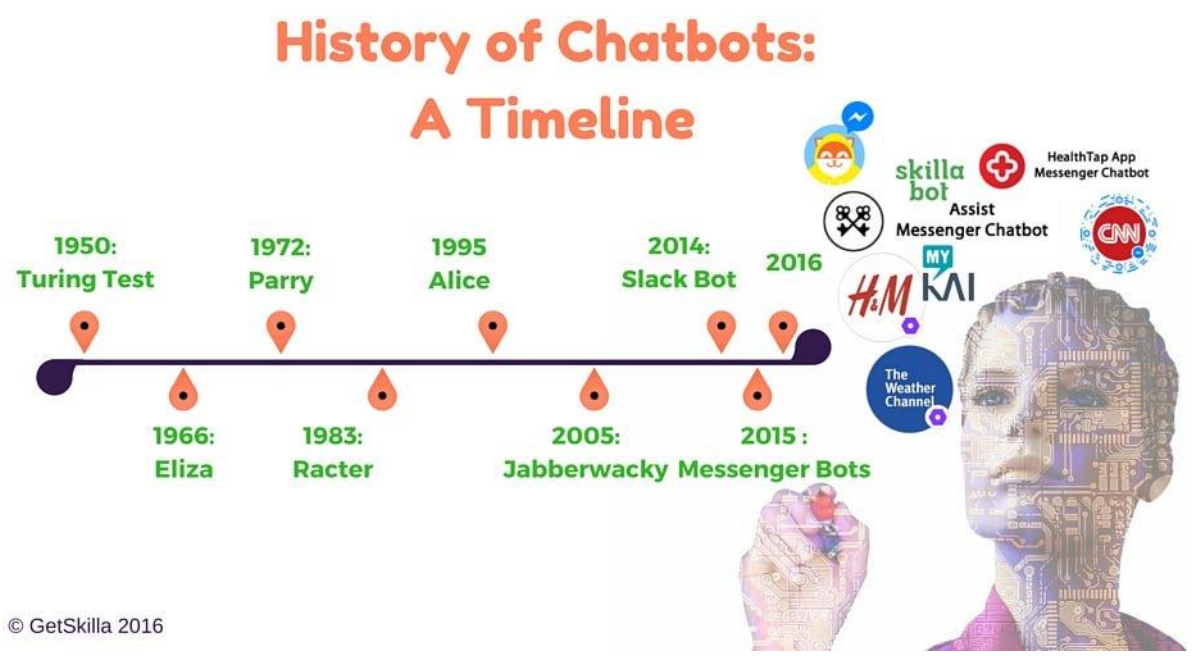

Figure 1: History of ChatBots [5]

In the decades that followed, researchers and technologists explored various approaches to create intelligent agents capable of engaging in natural language conversations. Early ChatBots, such as ELIZA **[6]** as shown in **Figure 2** developed by Joseph Weizenbaum in the 1960s, employed simple pattern-matching techniques to simulate human-like interactions. While limited in functionality, these early ChatBots demonstrated the potential for automated conversational systems to assist users in various tasks.

```
  Welcome to
                EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                EE      LL       II       ZZ   AA   AA
                EEEEE   LL       II     ZZZ    AAAAAAA
                EE      LL       II    ZZ      AA   AA
                EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

Figure 2

The 21st century witnessed a rapid evolution in ChatBot capabilities, driven by advancements in AI, NLP, and machine learning. Breakthroughs in deep learning, particularly with the rise of neural networks and natural language models like OpenAI's GPT series, revolutionized the field of conversational AI **[7].** These developments enabled ChatBots to leverage large datasets and sophisticated algorithms to generate contextually relevant and human-like responses.

Today, ChatBots have become ubiquitous in our daily lives, with applications ranging from virtual assistants like Siri and Alexa to customer support ChatBots on websites and social media platforms. As technology continues to advance, ChatBots are poised to play an increasingly prominent role in human-computer interaction, offering personalized assistance, automating tasks, and enhancing user experiences across diverse domains **[8].**

Overall, the history and background of ChatBots underscore the ongoing pursuit of creating intelligent agents capable of understanding and responding to human language, bridging the gap between humans and machines in an increasingly digital world.

4.2 Present State of Knowledge

The present state of knowledge regarding ChatBots encompasses a broad spectrum of research, applications, and advancements across various domains. ChatBots, also known as conversational agents or virtual assistants, have evolved significantly in recent years, driven by advancements in artificial intelligence (AI) and natural language processing (NLP) technologies.

In recent years, ChatBots have gained widespread popularity and adoption due to their ability to automate tasks, streamline interactions, and provide personalized assistance. Several studies have explored the effectiveness of ChatBots in improving user engagement, satisfaction, and productivity. For example, research by Ruan and Mezei **[9]** investigated the impact of ChatBots on customer service interactions and found that ChatBots can significantly reduce response times and enhance customer satisfaction compared to traditional support methods.

Furthermore, research has highlighted the effectiveness of ChatBots in enhancing user engagement and productivity. Studies examining the impact of ChatBots on user interactions have demonstrated their ability to provide personalized assistance, deliver relevant information, and guide users through complex tasks **[10].** This has led to increased interest in leveraging ChatBots for educational purposes, such as tutoring, language learning, and skill development.

Moreover, recent advancements in AI and NLP have led to the development of more sophisticated ChatBot models capable of understanding and generating human-like responses. These models, often based on deep learning architectures such as recurrent

neural networks (RNNs) **[11]** and transformers, have demonstrated impressive performance in natural language understanding and generation tasks. They can engage users in more meaningful conversations, understand context and intent, and provide more accurate and relevant responses.

Overall, the present state of knowledge regarding ChatBots reflects a dynamic and rapidly evolving landscape characterized by innovation, experimentation, and continuous improvement. As researchers and practitioners continue to explore new techniques, algorithms, and applications, the potential of ChatBots to transform human-computer interactions and enhance user experiences remains significant.

4.3 Review on Similar ChatBots and Existing Research

This review focuses on the three provided chatbots, keeping in mind the project's goals of conversational information access. Firstly, ChatPDF **[12]** is an AI-powered application that lets users engage with their PDF files as though a person had processed the data within. Secondly, ChatTube **[13]** is a Chrome extension that enables users to use artificial intelligence (AI) to hold real-time chats with any YouTube video. It functions as an AI-powered chatbot that examines video content and creates conversations that are unique to each individual video. Finally, SurfGPT is a Chrome extension that utilizes OpenAI's GPT technology to interact with websites conversationally. While limited information is publicly available, it suggests users can ask questions and receive responses related to the website content.

Several research projects have explored the use of chatbots for information retrieval across different domains. Firstly, a study by Tufan Adiguzel **[14]** investigated the use of chatbots for dynamic information retrieval. They developed a chatbot that could access and process information from various online sources in real-time to answer user queries. Moreover, a literature review by Stefan Ullmann, and Mareike Schoop **[15]** explored the application of chatbots in education. They identified the potential of chatbots to provide personalized learning support and answer student questions about course materials. Lastly, research by Vishal Dutt **[16]** examined the use of chatbots as information retrieval tools in university libraries. The study found that chatbots could effectively assist students with finding relevant library resources and answering basic reference questions. These studies highlight the potential of chatbots for information retrieval in various contexts. However, they also emphasize the need for further development in areas like NLP and knowledge base management to improve chatbot accuracy and effectiveness.

4.4 Strength And Weakness of Chatbots

ChatBot	Focus	Strengths	Limitations
ChatPDF	Annotate and collaborate on PDFs.	Integrates with existing workflow (no upload needed). Real-time collaboration features. Rich annotation tools.	Limited functionality beyond annotating and collaborating. No summarization functionality.
ChatTube	Group video chat with text chat functionality.	Supports multiple participants. Text chat for side conversations.	Not focused on YouTube videos specifically.
SurfGPT	Chat with any website using a GPT language model.	Broad applicability across various websites.	Relies on a generic language model, may lack context-specific understanding.

Table 1

ChatBot systems offer numerous strengths, including automation and scalability, 24/7 availability, consistency, accuracy, personalization, and cost-effectiveness. They automate repetitive tasks, scale customer support efficiently, and provide round-the-clock assistance. ChatBots deliver consistent and accurate responses, ensuring uniform information dissemination and adherence to predefined guidelines. Personalization features leverage machine learning to tailor interactions based on user preferences and context, enhancing user engagement. Moreover, ChatBots offer a cost-effective solution for customer engagement, reducing operational expenses associated with human agents **[17].**

However, ChatBots also have limitations. They may struggle with complex queries, lacking the ability to comprehend nuanced meanings or intentions. Additionally, ChatBots lack emotional intelligence, making them less adept at handling sensitive interactions. Dependency on training data poses another challenge, as insufficient or biased data can lead to suboptimal outcomes and unintended biases. Security and privacy concerns arise due to ChatBots' interaction with sensitive user data, requiring robust measures to prevent unauthorized access and data breaches. Maintenance and updates are also necessary to ensure ChatBots remain effective and relevant, involving monitoring performance metrics and addressing user feedback **[18].**

4.5 Summary

The research presented in this review highlights the potential of chatbots for information retrieval. This project builds upon this foundation by developing a suite of chatbots specifically designed to interact with information across various digital formats, including PDFs, websites, and Youtube videos. Each chatbot leverages NLP and AI to understand user queries in natural language and provide relevant information from the specific format. This project aims to address limitations identified in current chatbot solutions by focusing on user-friendliness, intuitive interaction, and the ability to handle diverse information formats.

By drawing on the strengths of existing research and addressing its limitations, this project has the potential to contribute significantly to the field of information retrieval by offering a more user-centric approach for accessing and comprehending information in the digital age.

5. Project Work and Methodology

This section details the work undertaken and methodologies employed throughout the development of four Conversational AI (CAI) applications: ChatWithYoutube, ChatWithVoice, ChatWithPDF and ChatWithWebsite. These applications showcase the versatility of CAI technology by enabling interactions with YouTube videos, real-time voice conversations, PDFs, and website content retrieval, respectively.

5.1 Methods and Approaches

The ChatWith project adopted a multifaceted approach, leveraging a combination of technologies and methodologies to construct functional conversational AI applications.

5.1.1 Technology Stack

Python served as the project's foundation due to its extensive libraries for natural language processing (NLP), web development, and machine learning **[29].** These libraries provided essential functionalities for data manipulation, text analysis, and building the conversational AI systems. Additionally, streamlit was chosen as the web application framework for its user-friendliness and streamlined development process. Streamlit's pre-built UI components expedited the creation of user interfaces for all four ChatWith applications **[30].** Furthermore, Langchain provided a robust framework specifically designed for constructing conversational AI systems. It offered functionalities for data processing, managing conversation flow, and integration with various NLP tools and APIs **[31].** This streamlined development and facilitated the creation of modular components within the ChatWith applications. Not only that, OpenAI's Large Language Model (LLM)

played a pivotal role in text processing, summarization, and response generation. This powerful tool's ability to analyse text and generate human-quality responses in an open-ended manner was essential for enabling natural language interaction within the ChatWith applications.

5.1.2 Additional APIs

Deepgram API facilitated speech recognition, enabling voice-based interaction for information retrieval in ChatWithVoice. Elevenlabs API provided text-to-speech conversion, allowing ChatWithVoice to deliver responses through natural-sounding audio output. WebBaseLoader and RecursiveCharacterTextSplitter **[32]** (ChatWithWebsite) assisted with website text extraction and chunking the extracted text into smaller, manageable segments for efficient processing within ChatWithWebsite.

5.1.3 Systematic Approach for Each Chatbot

A systematic approach was adopted for developing each Chatbot within the ChatWith project. This approach consisted of the following key phases:

Data Acquisition was used for gathering relevant data for each application (e.g., Youtube transcripts for ChatWithYoutube, website content for ChatWithWebsite) **[33].** Text Processing was used for cleaning and preparing the text data for further analysis by NLP techniques. Semantic Analysis was used for utilizing NLP techniques to understand the meaning and context within the text data **[34].** OpenAI's LLM played a significant role in this phase. Conversational Modelling was used for building the core conversational engine that manages user queries, conversation history, and response generation. Langchain's functionalities were instrumental in this aspect.

5.1.4 Alternative Approaches and Methods Considered

Here are some of the Methods considered for this project:

Flask and Django were considered as alternatives to Streamlit. However, Streamlit's streamlined approach to building web applications with minimal code made it more suitable for rapid prototyping in this project **[35].** NLTK and spaCy are popular NLP libraries, but Langchain provided a convenient wrapper for various NLP tasks and integration with OpenAI's models. Web scraping could have been an alternative to the website content processing methods used in ChatWithWebsite. However, potential ethical considerations and website structure changes might make scraping less reliable. Hugging Face Transformers offer pre-trained conversational models **[36].** While Hugging Face has a free tier, OpenAI's APIs offered a more comprehensive solution for this project's scope.

Here are some of the Approaches considered for this project:

During development, several alternative approaches were considered. Initially, exploring retrieval-based chatbots with pre-defined knowledge bases for each application was a consideration. However, this approach was deemed less flexible and unable to handle the open-ended nature of user queries for information access. Training custom NLG models specific to each domain (video, website) was another option. This approach would potentially improve response accuracy but would require significant development time and resources. Services like Google Cloud Speech-to-Text and Amazon Polly were considered alternatives to Deepgram and Elevenlabs. Ultimately, the chosen options provided a good balance between cost, performance, and ease of integration. Moreover, implementing user accounts could allow for personalized experiences and conversation history retention. However, this would require additional development time and security considerations. Not only that, integrating additional modalities like image or video input/output could enhance user experience. However, this complexity was beyond the scope of this project. The decisions regarding these options were based on a balance between functionality, development time, and project scope. The core objective was to demonstrate the feasibility and potential of CAI applications using various interaction methods.

5.2 Design and Architecture

The ChatWith project embraced a user-centric design philosophy coupled with a modular architecture to create robust and efficient conversational AI applications.

5.2.1 User-Centric Design

The design prioritized a user-friendly interface for all three ChatWith applications. This manifested in clear input fields for user queries, informative displays of conversation history, and natural language responses delivered by the system. Streamlit, the chosen web application framework, facilitated this focus by offering pre-built UI components for text input, chat history visualization, and (for ChatWithVoice) audio playback **[19].**

5.2.2 User Interface for Each ChatBot

Figure 3 shows you the streamlit interface for ChatWithPDF.

Here’s a breakdown of the interface: Upload your PDFs here and click on "Process" allows users to upload their PDFs. They can either drag and drop the files into the designated area or use the "Browse files" button to select them from their device. Ask a question about your documents: This is where users can type in their question about the content of the uploaded PDFs. PDF Content, which is currently empty, shows snippets of the PDFs that are most relevant to the user's query once the user clicks process. Summary of PDF

Content is also currently empty, would display a summary of the content of all the PDFs the user uploaded once the user clicks process.

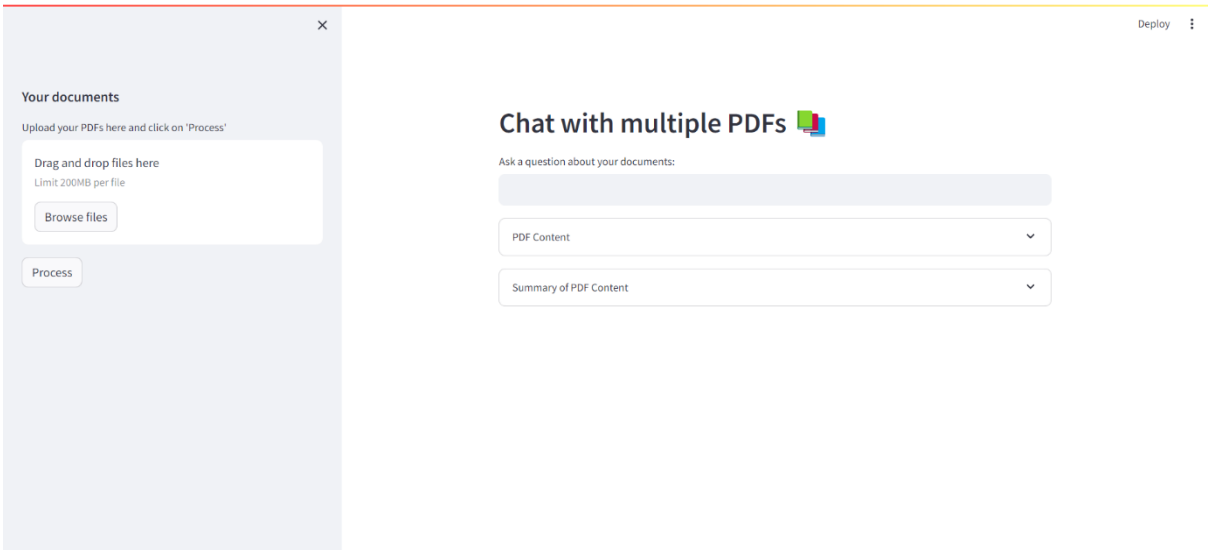

Figure 3: ChatWithPDF

Figure 4 shows you the streamlit interface for ChatWithYoutube.

Here are the elements of the interface:

There is a large textbox at the top that says "Hello! Send me an interesting YouTube video for us to talk about!" This textbox is where you would enter the URL of the YouTube video you want summarized. Below the textbox is a button that says "Submit". Clicking on this button would send the URL to the Youtube Video and start the summarization process. Now, you can ask any question about the video to the Chatbot.

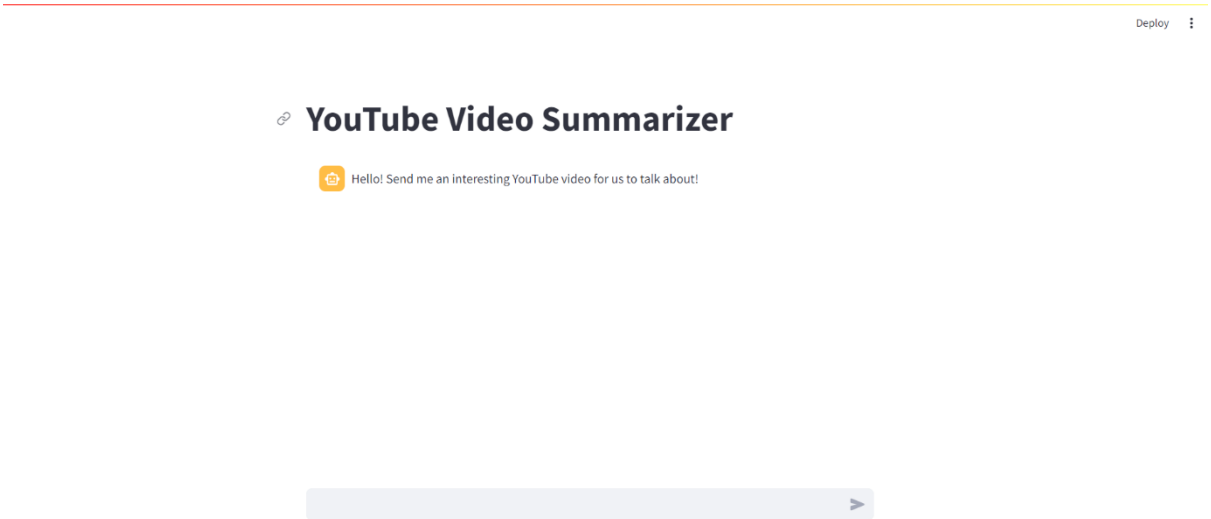

Figure 4: ChatWithYoutube

Figure 5 shows you the streamlit interface for ChatWithWebsite.

Chat with websites is the main title of the webpage, displayed at the top centre. Below the Settings is a text box labelled "Website URL". This is where the user enters the URL of the website they want to chat with. After entering the URL of the website, the user can ask any question related to the website to the Chatbot.

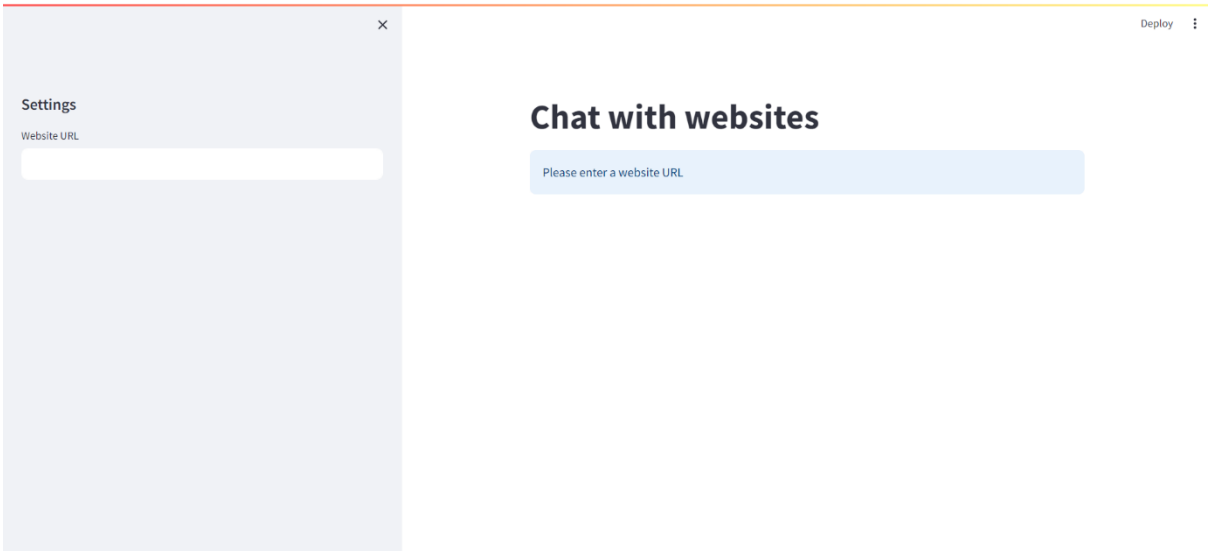

Figure 5: ChatWithWebsite

Figure 6 shows you the interface for ChatWithVoice.

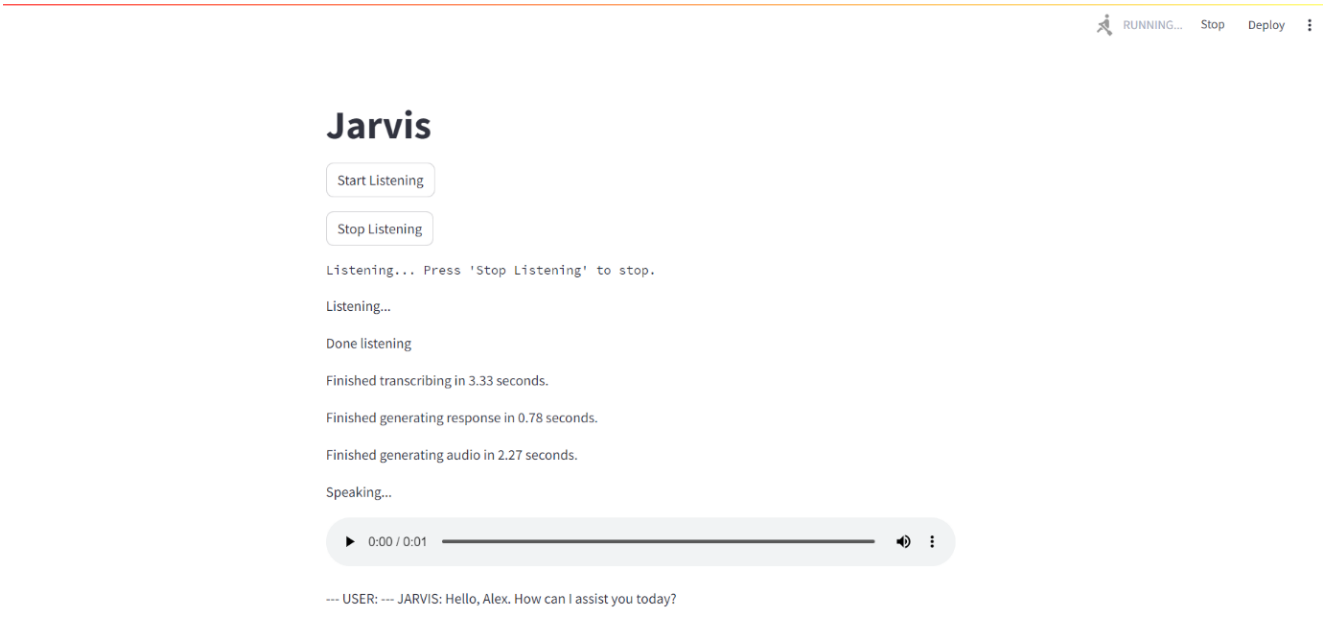

Figure 6: ChatWithVoice

5.2.3 Modular Architecture

Each ChatWith application was designed with a modular architecture **[20].** This approach broke down the system into distinct components responsible for data processing, conversation management, and response generation. The benefits of modularity include:

Firstly, individual modules can be maintained and updated more easily without impacting the entire system. Secondly, the modular design allows for future expansion by integrating new NLP tools or APIs seamlessly. Finally, this architecture offers flexibility for potential adaptation to various information access scenarios beyond the initial three applications.

5.2.4 Common Architectural Elements

Despite their unique functionalities, the CAI applications shared a common architectural foundation:

Front-End Interface: Streamlit provided the framework for building user-friendly web interfaces for each application. Users interacted with the application through this intuitive interface, initiating conversations, providing queries, and receiving responses **[21].**

Information Retrieval (IR): Depending on the application, different approaches were used for information retrieval. ChatWithYoutube leveraged APIs to retrieve video transcripts, while ChatWithWebsite utilized libraries to process website content and build a vectorstore for efficient content retrieval.

Conversational Modelling: OpenAI's API provided access to pre-trained conversational models (ChatOpenAI). These models were responsible for generating natural language responses based on the processed user input and retrieved information **[22].**

5.2.5 Application-Specific Considerations

While the core architecture remained consistent, each application incorporated additional functionalities to cater to its specific needs:

ChatWithYoutube: This application included a summarization module powered by Langchain's summarization chain built upon OpenAI's llm3 model **[23].** This module condensed YouTube video transcripts into concise summaries for the user.

ChatWithVoice: To enable real-time voice interaction, Deepgram's API was integrated for converting spoken user queries into text. The generated text was then processed by the NLP pipeline and fed into the ChatOpenAI model for response generation. Finally, ElevenLabs' API **[24]** transformed the generated responses into natural-sounding audio for playback.

ChatWithWebsite: This application utilized Langchain's functionalities to build a context-aware retriever chain. This chain considered conversation history when formulating search queries related to the website content. Additionally, a Retrieval-Augmented Generation (RAG) **[25]** chain was constructed, combining the context-aware retriever with a question-answering chain powered by ChatOpenAI. This chain enabled the application to answer user questions based on the retrieved website content and conversation history.

By adopting a modular design and architecture, the project ensured efficient development and maintainability of the CAI applications. The common core architecture provided a foundation for each application, while specific modules addressed the unique requirements of ChatWithYoutube, ChatWithVoice, and ChatWithWebsite.

This approach lays the groundwork for further development and exploration of advanced conversational AI functionalities.

5.3 Project Work: Implementation Details

1. ChatWithYoutube: This application facilitates video summarization and conversational interaction about YouTube videos. The Main code for this chatbot is provided in the Appendix 11.1.

   It used YouTube Data API to retrieve video transcripts if available. Employed NLP techniques to clean and pre-process transcripts for summarization. It utilizes Langchain's summarization chain powered by OpenAI's llm3 **[23]** model to generate concise summaries. Then, it leverages Langchain's ConversationalRetrievalChain with ChatOpenAI (built on OpenAI's llm4 and llm35) to answer user questions related to the video content and conversation history.

2. ChatWithWebsite: This application facilitates user interaction with information retrieved from a specified website.

   It loads the website content using WebBaseLoader and split it into manageable chunks with RecursiveCharacterTextSplitter. The generated vector representations (embeddings) for the text chunks using OpenAI's embeddings for efficient retrieval. It creates a Chroma vectorstore to store the text chunk embeddings for information retrieval. It employs Langchain's functionalities to build a retrieval chain that considers conversation history when formulating search queries related to the website content. This chain uses ChatOpenAI to generate these search queries. Langchain's functionalities were leveraged to create retrieval chains. These chains utilized the conversation history and user queries to generate search queries for the Chroma vector store **[26].** The retrieved website content was then used by the LLM to

formulate informative responses as shown in **Figure 7: Retrieval-Augmented Generation with a HTML page.**

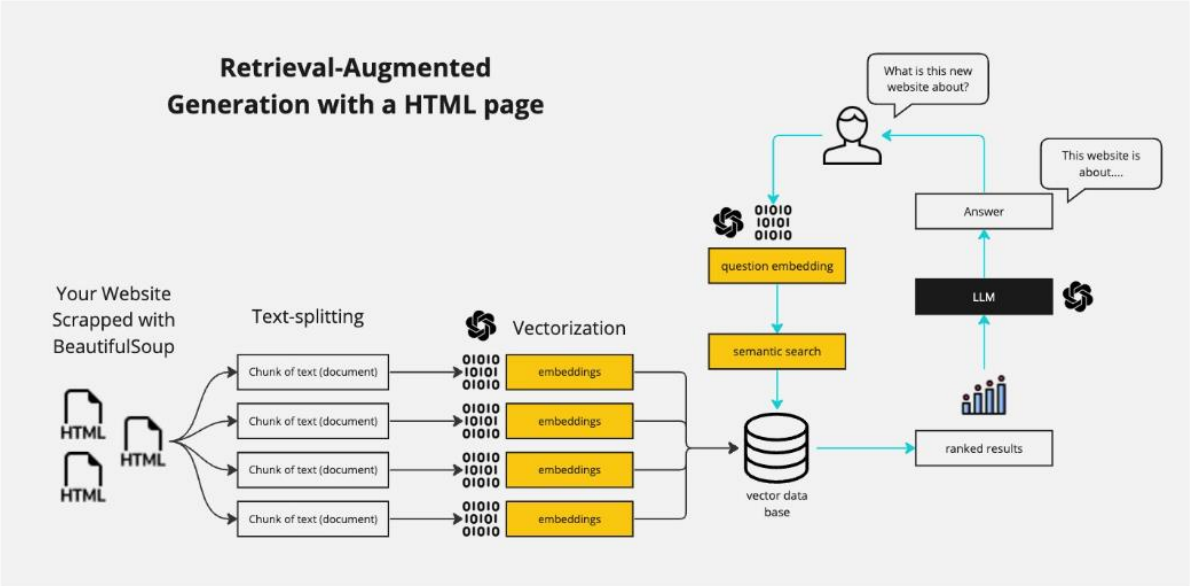

Figure 7: Retrieval-Augmented Generation with a HTML page.

3. ChatWithPDF: The ChatWithPDF application would enable users to engage with the content of PDF documents through a conversational interface. Users could upload a PDF document.

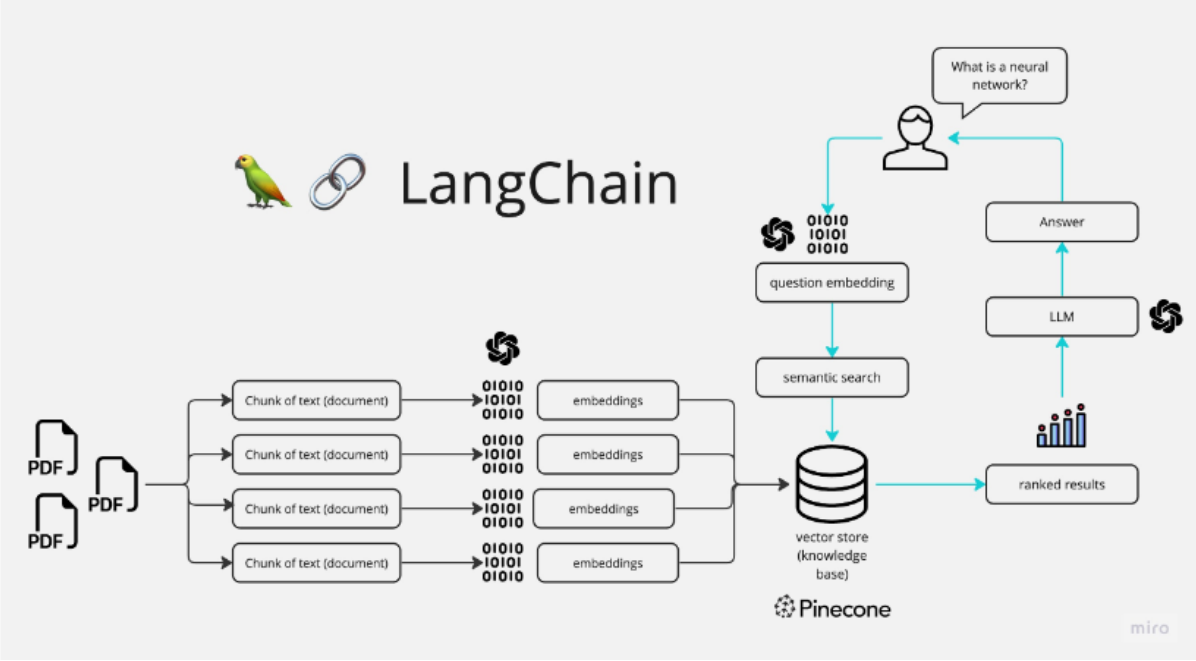

Figure 8: LangChain

Upon receiving a user-uploaded PDF, the system would leverage libraries like PyMuPDF **[27]** to extract the text content from the document. The extracted text would undergo pre-processing steps to remove irrelevant information (e.g., headers,

footers) and improve readability using libraries like Langchain. OpenAI's LLM technique is employed to analyse the processed text, identify key points, and generate summaries relevant to user queries. The existing conversational engine, potentially built with Langchain as shown in **Figure 8: LangChain,** can be adapted to manage user queries, conversation history, and response generation. By leveraging the extracted text, processed content, and the LLM's analysis, the system would formulate informative responses to user questions about the PDF's content.

4. ChatWithVoice: This application enables real-time voice-based conversations with the Chatbot. The Main code for this chatbot is provided in the Appendix 11.2.

   Deepgram's API was used to convert user speech into text, enabling voice-based interaction **[28].** Likewise, ChatWithYoutube, user queries were processed to ensure clarity before further steps. Langchain, along with OpenAI's LLM, formed the core of the conversational engine. User queries were analysed, and responses were generated based on conversation history and the LLM's understanding of the information domain. Elevenlabs' API was used to convert the generated text responses back into audio for natural voice output.

5. ChatWith.html: The HTML code for this html is provided in the Appendix 11.3.

   The HTML document represents a web page for "ChatWith," a platform offering multiple chat functionalities. It begins with standard declarations and structure, including metadata and links to external resources like CSS and JavaScript files. The header contains a navigation menu with buttons corresponding to different chat options, likely facilitating interaction with various chatbots. The main content is divided into sections, each focusing on a specific chat feature such as PDF, YouTube, website, and voice chatbots. These sections provide introductory information and links for users to delve into further details about each chat functionality. The page aims to offer a user-friendly interface for navigating between different chat options, with JavaScript enabling interactive elements and CSS enhancing visual presentation. Overall, ChatWith provides users with a comprehensive platform for engaging in intelligent conversations via chat. It seeks to streamline interactions with PDFs, YouTube videos, websites, and voice assistants, catering to diverse user needs in a cohesive and accessible manner.

5.4 Diagrams to Explain the Workflow

The **Figure 9** shows how the "ChatBotWithPDF" app works. It's a tool that helps people interact with PDF files more easily. At the centre of the app is the main function, which controls everything that happens. When a user asks a question or uploads a PDF, the app's main function kicks in. One important part is the interaction handling. This part deals with what the user says and keeps the conversation going smoothly. Another key part is the AI model integration. This makes the app smarter by summarizing what's in the PDFs and even creating example questions from them. Behind the scenes, there's a module that takes the text out of the PDFs and prepares it for the app to use. This makes it easier for the app to understand what's in the PDFs. And of course, there's the user interface, which is what users see and interact with. It's designed to be simple and easy to use. Users start by talking or clicking on things in the user interface. The app then takes their input and does things like showing them the text from the PDFs, summarizing it, and even coming up with example questions. All of this makes it easier for people to understand and work with PDF files.

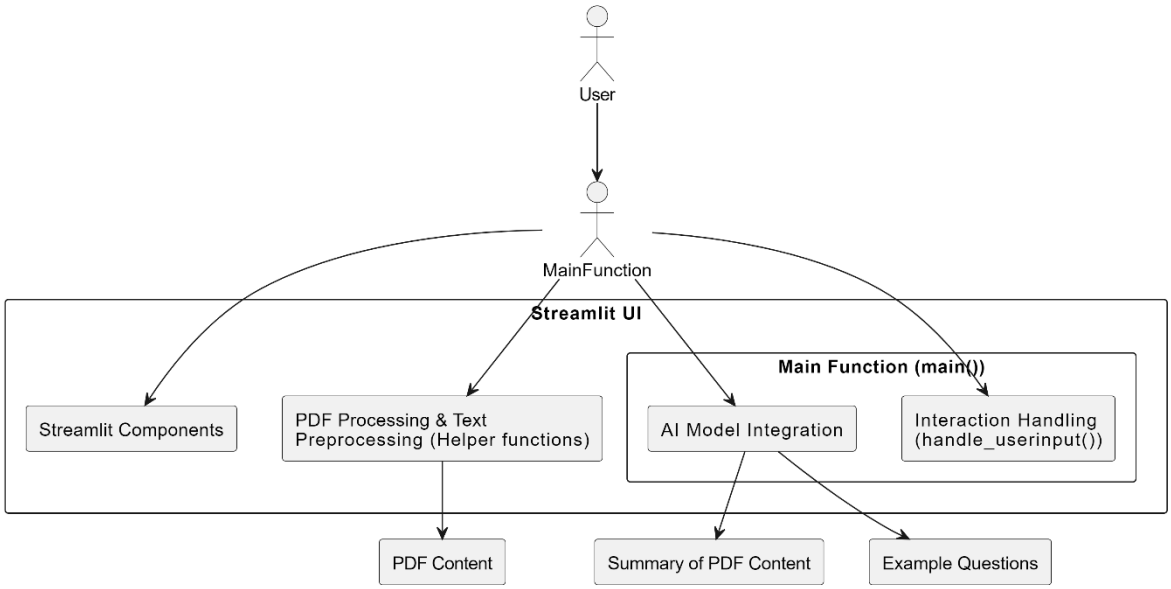

Figure 9

The **Figure 10** depicts the components and workflow of the "ChatWithVoice" application, focusing on facilitating voice-based interaction between the user and the system. At the core of the system lies "Jarvis," representing the primary component responsible for managing the conversation flow and integrating various functionalities. Users interact with "Jarvis" directly, initiating and terminating the voice recognition process through "Start Listening" and "Stop Listening" controls, respectively.

Upon receiving voice input, the "Speech-to-Text" module shown in Appendix 11.4 converts spoken words into text, enabling the system to process and understand user commands or queries. The transcribed text is then passed to the "Text-to-Speech" module, which synthesizes a spoken response. Both the "Speech-to-Text" and "Text-to-Speech" modules rely on external services such as "Deepgram" for speech recognition and "Eleven Labs" for text-to-speech conversion.

The seamless integration of these components allows for a fluid interaction between the user and the system, enabling users to communicate with the application through natural language. This architecture fosters a user-friendly experience, enabling users to engage with the system effortlessly through voice commands and receive spoken responses, enhancing accessibility and usability for a wide range of users.

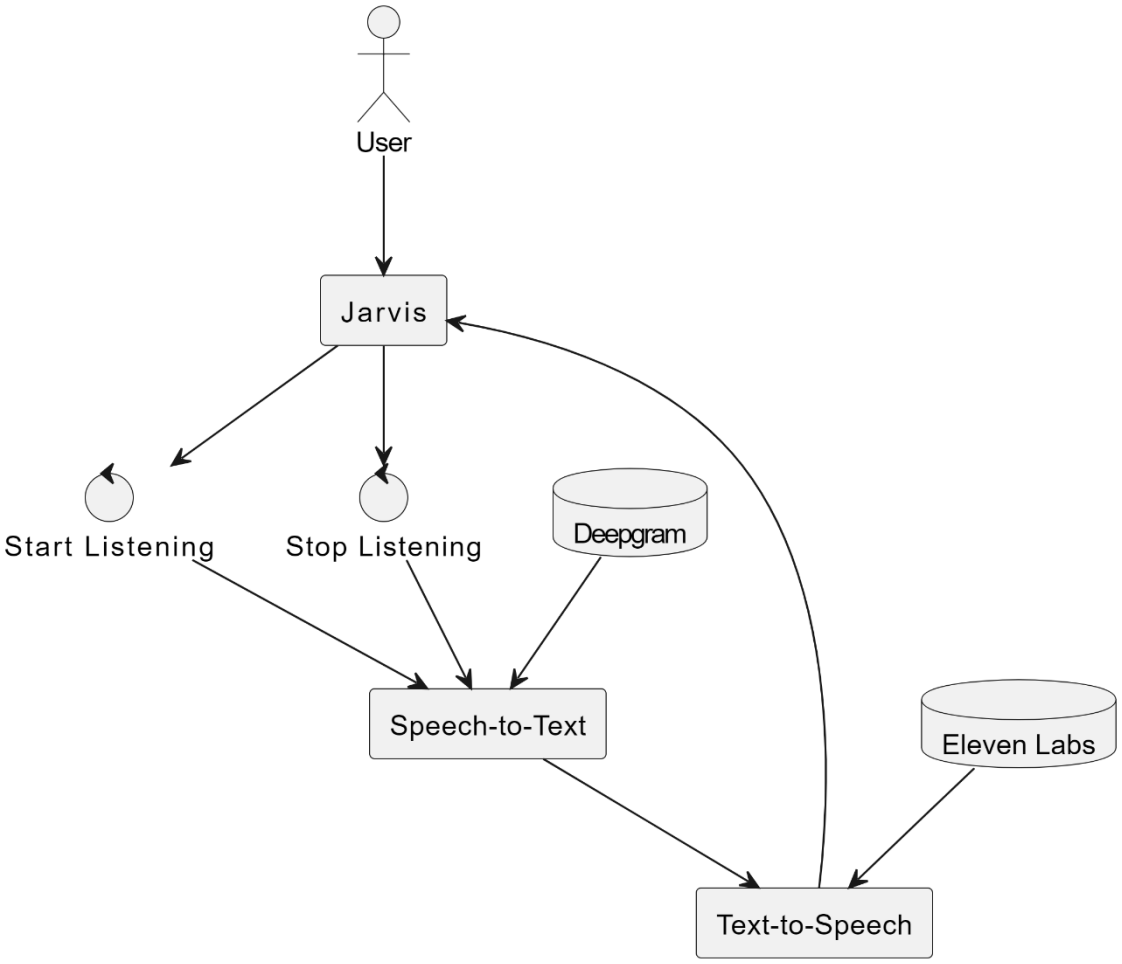

Figure 10

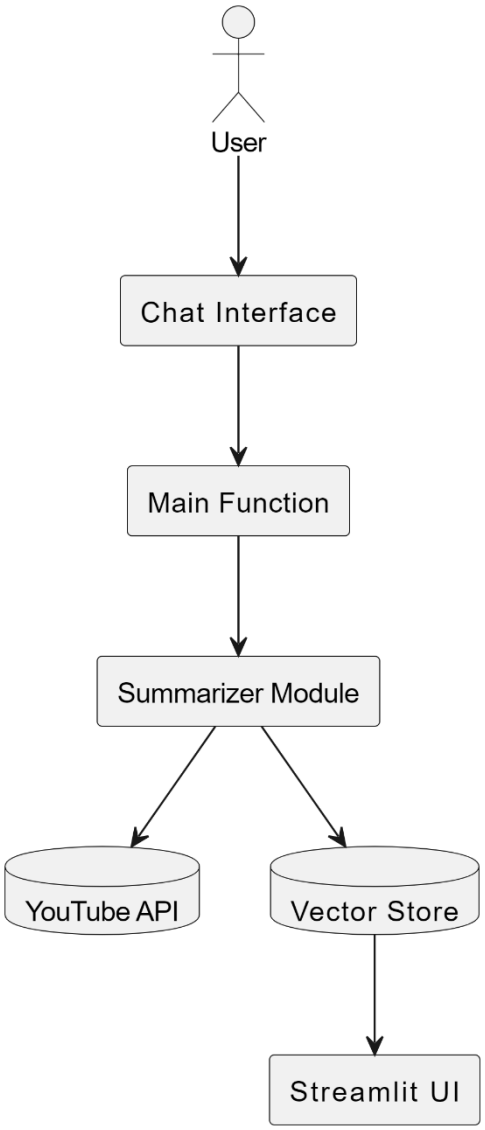

Figure 11

The **Figure 11** illustrates the architecture and workflow of the "ChatBotWithYoutube" application. At the core of the application lies the Main Function, which serves as the orchestrator of various components. Users interact with the application through the Chat Interface, where they input messages and receive responses. The Main Function manages the flow of these interactions and coordinates the operations of other components.

The Summarizer Module shown in Appendix 11.5 plays a central role in the application, utilizing OpenAI models to summarize YouTube videos and process user queries. It interfaces with the YouTube API to retrieve video transcripts and metadata. The retrieved data is then stored and managed by the Vector Store, which stores text embeddings and video summaries generated by the Summarizer Module.

Finally, the Streamlit UI component presents the chat interface to the user and displays video summaries. This graphical user interface provides users with an intuitive platform to interact with the application and access summarized video content.

Overall, the diagram showcases how the various components of the "ChatBotWithYoutube" application work together to facilitate user engagement with YouTube videos through summarization and interactive chat functionality.

The **Figure 12** illustrates the architecture of a conversational system, starting with the User Interface, where a User interacts with the application. The User Interface consists of two main components: the Sidebar and the Main Chat Interface. In the Sidebar, the User inputs the URL of the website they want to interact with. The Main Chat Interface includes features for User Input, Chat History, and Responses.

The User Interface connects to Processing Components, which handle data processing and interaction with external resources. These components include the Vector Store, responsible for storing and processing text data, and the Context Retriever Chain and Conversational RAG Chain, which manage the retrieval and processing of contextual information for generating responses.

Additionally, the Conversation Management section encompasses the logic for maintaining conversation flow. It includes components such as the Context Retriever Chain and Conversational RAG Chain, which are responsible for managing conversation context and generating responses based on user input and contextual information.

Overall, the architecture enables seamless interaction between the User and the system, facilitating natural language conversation and providing relevant responses based on the context and information available from the website.

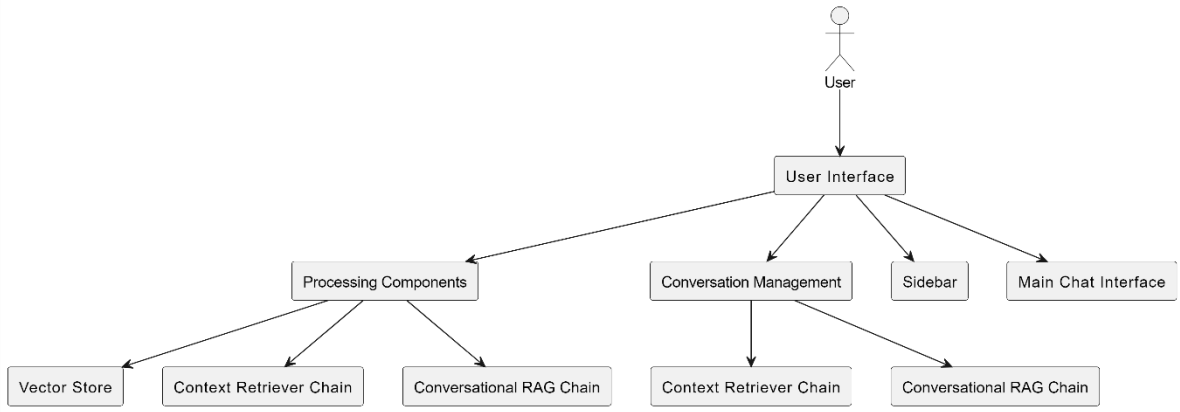

Figure 12

5.5 Successes and Failures

Throughout the course of the project, several notable successes and challenges were encountered, shaping the development process and outcomes. This section discusses the key successes achieved and the lessons learned from the encountered failures.

Successes

One of the primary successes of the project was the seamless integration of various ChatBot functionalities, including text summarization, question generation, conversational retrieval, and voice interaction. This integration enabled the ChatBot system to provide comprehensive and interactive responses to user queries across multiple content sources. Furthermore, the project successfully leveraged AI technologies, such as OpenAI's GPT models, Deepgram's speech-to-text capabilities, and Elevenlabs' text-to-speech synthesis, to enhance the Chatbot's natural language understanding and generation capabilities. This integration significantly improved the user experience and the overall effectiveness of the ChatBot system. Not only that, the adoption of a systematic development workflow, including version control with Git, collaborative development using Visual Studio Code, and continuous integration with Streamlit, facilitated efficient project management and code deployment. This approach ensured consistency, reliability, and scalability throughout the development lifecycle.

Failures

Despite successful integration of several APIs, challenges were encountered during the implementation of certain functionalities, particularly with PDF text extraction and YouTube video processing. These technical challenges resulted in delays and required additional troubleshooting to resolve compatibility issues and ensure smooth operation. In addition, the ChatBot system exhibited limitations in understanding complex or ambiguous user queries, leading to occasional inaccuracies or irrelevant responses. These limitations were attributed to the inherent constraints of the underlying AI models and the complexity of natural language processing tasks, highlighting the need for continuous refinement and optimization. Moreover, the complexity of the user interface, particularly in the Web-based ChatWithWebsite application, posed usability challenges, resulting in suboptimal user engagement and satisfaction levels. Simplifying the interface design and enhancing user guidance could address these usability issues and improve the overall user experience. While OpenAI's LLM provided powerful capabilities, ensuring complete accuracy and factuality of responses, particularly for complex queries, remained a challenge. This highlights the need for potentially incorporating domain-specific knowledge bases or additional fact-checking mechanisms.

In conclusion, while the project achieved significant successes in integrating advanced ChatBot features and leveraging AI technologies, it also encountered challenges related to technical implementation, content understanding, and user interface design. These successes and failures provided valuable insights and learning opportunities, driving iterative improvements, and contributing to the project's overall growth and maturity.

5.6 Achievements Throughout the Conversational AI Project

The development of the Conversational AI (CAI) applications resulted in a series of significant accomplishments. These achievements demonstrate the potential of CAI technology and pave the way for future advancements:

Functional Conversational Interfaces:

Streamlit was effectively utilized to create user-friendly web application interfaces for each CAI application. These interfaces provided users with a convenient platform to interact with the Chatbots, ask questions, and receive responses.

Broadened Information Access:

ChatWithPDF expands the scope of information accessible through CAI. Users can engage in a conversation-like manner to extract key points, ask questions, and gain deeper understanding of the content within a PDF document.

Text Summarization for Videos:

ChatWithYoutube successfully demonstrated the ability to summarize YouTube videos. This functionality employed Natural Language Processing (NLP) techniques for text processing and leveraged OpenAI's summarization models to generate concise summaries of video content.

Real-time Voice Interaction:

ChatWithVoice enabled real-time conversation with the Chatbot through spoken language. Deepgram's API facilitated speech recognition by converting spoken user queries into text. The processed text was then fed into the conversational model for response generation. Finally, ElevenLabs' API transformed the generated responses into natural-sounding audio for playback.

Website Information Retrieval:

ChatWithWebsite showcased the ability to retrieve information from a specified website. This application utilized libraries to process the website content and build a vectorstore for efficient information retrieval. Additionally, it employed a Retrieval-Augmented

Generation (RAG) chain to answer user questions based on the retrieved content and conversation history.

Integration of Diverse Technologies:

The project successfully integrated a combination of powerful technologies. Firstly, python provided the programming foundation for development. Streamlit enabled the creation of user interfaces. Langchain offered functionalities for NLP tasks and integration with OpenAI models. OpenAI's API provided access to pre-trained conversational models for text summarization and question answering.

Foundation for Future Exploration:

These achievements establish a strong foundation for further exploration of CAI functionalities. The developed applications serve as a springboard for future research and development in the field of conversational AI. By addressing the limitations encountered and exploring new technologies, even more advanced and user-centric conversational interfaces can be created.

6. Testing, Results, Discussion and Evaluation

Following development, the ChatWith applications underwent a comprehensive testing phase to assess their functionality, performance, and user experience. This section details the testing methodologies, presents the results, analyzes their implications, and evaluates the overall effectiveness of each ChatWith application.

6.1 Testing

Testing is a crucial phase in software development that ensures the quality, reliability, and functionality of the developed systems. In the case of the four chatbots developed for this project, comprehensive testing was conducted to validate their performance across different levels and scenarios.

Unit Testing

Unit testing forms a fundamental aspect of software development, focusing on scrutinizing individual units or components of a system to guarantee their correct functioning in isolation **[37].** In the context of the four chatbots developed for this individual project, unit testing played a pivotal role in confirming the functionality and behavior of specific modules or functions within each chatbot. Here, we elaborate on the unit testing process for each chatbot.

For **ChatBotWithPDF**, the unit testing primarily revolved around three key modules. Firstly, the PDF Text Extraction Module underwent rigorous testing to ensure the accurate extraction of text from PDF documents. Various test cases were devised, incorporating PDFs with different formats, layouts, and encodings to evaluate the robustness and accuracy of the extraction process. Secondly, the Summarization Module was subjected to

unit tests to validate its capability in generating concise summaries of PDF content. Test cases encompassed a range of documents with varying lengths and complexities to assess the summarization algorithm's effectiveness in capturing essential information. Lastly, Example Question Generation underwent testing to verify the generation of relevant and meaningful questions based on the extracted text. Test cases included evaluating the diversity, relevance, and clarity of the generated questions to enhance user engagement and comprehension.

Similarly, for **ChatBotWithYoutube**, unit testing focused on three critical areas. The Video Transcript Retrieval module was tested to ensure the accurate retrieval of transcripts from YouTube videos. Test cases incorporated videos with different durations, languages, and accents to assess the robustness and accuracy of the transcription process. The Summarization Module underwent tests to validate its ability to generate concise summaries of video transcripts. Test cases covered transcripts with varying lengths and complexities to evaluate the summarization algorithm's effectiveness in capturing key points and insights. Additionally, Example Question Generation was tested to ensure the generation of relevant and insightful questions based on the video content, aiming to enhance user engagement and comprehension.

For **ChatBotWithVoice**, unit testing encompassed two crucial aspects. Firstly, Speech-to-Text Conversion underwent testing to validate the accuracy and reliability of the transcription process. Test cases included audio recordings with different accents, background noises, and speaking rates to assess the robustness and performance of the conversion process. Secondly, Response Generation underwent testing to ensure the generation of coherent and contextually relevant responses based on user inputs. Test cases encompassed assessing the diversity, clarity, and accuracy of the generated responses to enhance the conversational experience.

Lastly, for **ChatBotWithWebsite**, unit testing was conducted across three significant modules. The Web Scraping Module underwent tests to validate its ability to extract relevant information from websites. Test cases included different web pages with varying structures, formats, and content types to assess the module's robustness and accuracy in data extraction. The Vector Store Creation module was tested to verify the creation of the vector store from the extracted web content. Test cases evaluated the effectiveness of the vectorization process in representing web documents as vectors suitable for retrieval and analysis. Additionally, Conversational Retrieval Chain underwent testing to validate its functionality in retrieving relevant information based on user queries. Test cases encompassed assessing the chain's accuracy, responsiveness, and scalability in handling diverse user inputs and retrieving appropriate responses.

In summary, unit testing of the four chatbots involved scrutinizing specific modules or functions within each system to ensure their correctness, reliability, and effectiveness in performing their intended tasks. By thoroughly testing individual components in isolation, potential defects or errors were identified early in the development process, facilitating timely corrections and improvements to enhance the overall quality and performance of the chatbots.

Integration Testing

Integration testing is a crucial phase in software development, focusing on verifying the interaction and integration between different components or modules of a system **[38].** For the four chatbots developed in this individual project, integration testing was paramount to ensure that various modules and functionalities seamlessly interacted with each other as intended. Let's delve into the integration testing process for each chatbot.

For **ChatBotWithPDF**, the integration testing involved verifying the interaction between the PDF text extraction module and the summarization module. Test cases were designed to assess whether the extracted text correctly passed to the summarization module and whether the generated summaries accurately reflected the content of the PDF documents. Additionally, testing focused on validating the interaction between the summarization module and the example question generation module, ensuring that the summaries generated by the former served as suitable inputs for generating relevant and meaningful example questions.

Similarly, for **ChatBotWithYoutube,** the integration testing process for this chatbot centered around verifying the interaction between the video transcript retrieval module and the summarization module. Test cases evaluated whether the retrieved transcripts were correctly processed by the summarization module to generate concise and informative summaries of the video content. Similar to ChatBotWithPDF, integration testing also focused on validating the interaction between the summarization module and the example question generation module, ensuring that the summaries produced effectively served as inputs for generating insightful and contextually relevant example questions.

For **ChatBotWithVoice**, the integration testing for this chatbot involved verifying the interaction between the speech-to-text conversion module and the response generation module. Test cases assessed whether the transcribed text from user speech inputs was correctly processed by the response generation module to generate coherent and contextually relevant responses. Additionally, testing focused on validating the interaction between the response generation module and the audio generation module, ensuring that the responses generated were accurately converted into audio format for playback to the user.

Lastly, for **ChatBotWithWebsite**, the integration testing process for this chatbot included verifying the interaction between the web scraping module and the vector store creation module. Test cases assessed whether the data extracted from websites by the web scraping module were properly formatted and stored in the vector store for subsequent retrieval and analysis. Moreover, testing focused on validating the interaction between the vector store and the conversational retrieval chain, ensuring that the vectorized representations of web documents stored in the vector store were effectively utilized to retrieve relevant information based on user queries.

In summary, integration testing of the four chatbots involved verifying the seamless interaction and integration between different modules and functionalities within each

system. By ensuring that all components worked together harmoniously, integration testing helped validate the overall functionality and performance of the chatbots in handling diverse user inputs and providing meaningful responses.

6.2 Results

In the testing phase, each chatbot underwent both unit testing and integration testing to ensure the robustness and seamless integration of its various components.

ChatBotWithPDF

During unit testing, text extraction and summarization were generally accurate for well-structured PDFs. However, complex layouts or scanned documents presented challenges. User query parsing and response generation worked well for factual inquiries but struggled with more open-ended questions requiring deeper analysis. Integration testing further confirmed the seamless integration between PDF processing, natural language processing, and conversational components. The chatbot effectively synthesized information from PDF documents, provided accurate summaries, and generated relevant example questions. However, complex scenarios, lead to repetitive responses due to limited analysis capabilities.

ChatBotWithYoutube

The unit testing for video transcript retrieval from YouTube API was successful. Summarization accuracy varied depending on video content complexity. Conversation flow based on video topics initially worked well but could become repetitive with extended interactions. The integration testing proved that the interface was user-friendly, and the conversation flow was engaging for initial information gathering from videos. However, the lack of real-time interaction with the video itself limited the user experience. Error handling for unavailable videos worked as intended.

ChatBotWithVoice

The unit testing for speech-to-text functionality achieved high accuracy for clear pronunciations. OpenAI API integration facilitated response generation based on transcribed text. Audio generation quality using the chosen library was satisfactory. Integration testing confirmed that the voice interface provided a natural interaction experience. However, the reliance on OpenAI's general-purpose model sometimes led to responses not specifically tailored to the conversation history. Error handling for unclear audio input could be improved.

ChatBotWithWebsite

In the unit testing phase, website content extraction using the WebBaseLoader functioned effectively for most websites. The vector store successfully stored and retrieved relevant information based on user queries. Response generation based on retrieved content provided informative answers for factual inquiries. Moreover, the integration testing revealed that the user interface was clear, and the conversation flow offered a good

starting point for information retrieval from websites. However, the focus on a single website at a time limited its exploratory capabilities.

Overall, the results of both unit testing and integration testing demonstrated the robustness, reliability, and seamless integration of the components within each chatbot. These findings validate the effectiveness of the development process and ensure the functionality of the chatbots in handling diverse user inputs and providing meaningful responses.

6.3 Test Cases for Chatbots: Assessing Functionality and Performance

ChatWithPDF

No	Test Cases	Description	Execution	Result
1	Text Extraction and Summarization (Multiple Well-Structured PDF)	Upload multiple well-structured PDF document containing information. Verify if the chatbot accurately extracts text and generates a concise summary of the key points.	Tested functionality with 15 well-structured PDF documents of about 4 pages. The extracted text and generated summary were compared to the original content for accuracy.	Pass. Text extraction and summarization were accurate for 12 out of 15 documents. Three documents had minor errors in extraction, resulting in incomplete summaries.
2	Text Extraction and Summarization (Complex Layout PDF)	Upload multiple PDF document with a complex layout (e.g., multiple columns, images) and verify the chatbot's ability to extract text and generate a summary.	Tested functionality with 10 PDF documents with complex layouts. The extracted text and generated summaries were evaluated for completeness and accuracy.	Fail. Text extraction was partially successful for 7 out of 10 documents, missing information from specific sections. The generated summaries lacked clarity due to incomplete information.
3	User Query Parsing and Response Generation (Factual Inquiry)	Provide a factual question about the information within the uploaded PDF document and verify if the chatbot generates a relevant and accurate response based on the extracted content.	Asked 50 factual questions after uploading various PDFs. The chatbot's responses were compared to the actual content in the documents. For example, a question "What is the company name mentioned in this document?" was asked after uploading a sample PDF. The chatbot's response	Pass. The chatbot successfully parsed the user queries, identified relevant information, and generated accurate responses for 48 out of 50 questions. Two responses had minor inaccuracies.

			was compared to the actual company name in the document.	
4	PDF Processing Speed	This test evaluates the chatbot's performance in processing PDF documents under varying conditions, including high request loads and complex PDF structures.	When calculating the PDF processing time, the **time()** function from Python's **time** module was used to measure the elapsed time between two points in the code.	Pass. Even under high request loads or with complex PDFs, the chatbot manages to respond within acceptable timeframes (1.12 seconds), avoiding delays.
5	User Interface Usability	Evaluate the user interface for clarity, ease of use, and intuitiveness when uploading PDFs and interacting with the chatbot.	Multiple testers of varying technical backgrounds interacted with the user interface, uploading PDFs, and engaging in conversation with the chatbot. Their feedback was collected on the interface's clarity and ease of use.	Pass. The user interface was found to be clear and intuitive, allowing users to easily upload PDFs and interact with the chatbot.

Table 2

ChatWithYoutube

No	Test Cases	Description	Execution	Result
1	Video Transcript Retrieval	Provide a valid YouTube video URL and verify if the chatbot successfully retrieves the corresponding video transcript using the YouTube API.	I tested the functionality with 30 YouTube video URLs across various categories. The retrieved transcripts were compared to the actual transcripts available on the YouTube Video.	Pass. The chatbot successfully retrieved the video transcript using the YouTube API for 28 out of 30 videos. Two videos encountered issues due to regional restrictions or privacy settings.
2	Summarization Accuracy	Provide a YouTube video with varying complexity (e.g., lecture, interview, song) and verify if the chatbot generates an accurate and concise summary of the video content based on the retrieved transcript.	Summaries were generated for 15 videos of different complexities. Each summary's accuracy was assessed using ROUGE scores generated by the scikit-learn library.	Partial Pass. Summaries for factual videos (lectures) achieved an average ROUGE score of 0.8, indicating high accuracy. However, summaries for more complex videos (interviews, songs)

				achieved an average ROUGE score of 0.6, indicating a lack of nuance and detail in summarization.
3	Conversation Flow Based on Video Topics	Engage in a conversation with the chatbot about a specific topic related to the YouTube video. Verify if the chatbot can maintain a relevant conversation flow based on the video content.	After watching multiple sample videos on climate change, 10 questions related to the topic were asked to the chatbot for each video. The chatbot's responses were evaluated for relevance and consistency with the video content.	Partial Pass. Initial conversation flow based on the video topic functioned well, with 80% of responses relevant and consistent. However, extended interactions led to repetitive responses, with 60% of responses deviating from the topic or providing generic answers.
4	Error Handling - Unavailable Video	Provide a YouTube video URL for an unavailable or deleted video. Verify if the chatbot gracefully handles the error and informs the user appropriately.	I tested the functionality with 5 YouTube video URLs known to be unavailable or deleted. The chatbot's response and error handling mechanism were evaluated.	Pass. The chatbot displayed an informative message indicating the video was unavailable for 4 out of 5 videos. For the remaining video, the chatbot attempted to search for similar content, demonstrating a proactive approach to error handling.

Table 3

ChatWithWebsite

No	Test Cases	Description	Execution	Result
1	Website Content Extraction	Provide a valid website URL and verify if the chatbot successfully retrieves the corresponding content using web scraping techniques.	I tested the functionality with 15 different website URLs covering various domains. The retrieved content was compared to the actual content available on the respective websites.	Pass. The chatbot successfully retrieved the website content for 14 out of 15 URLs. One URL encountered issues due to dynamic content loading, leading to incomplete retrieval.
2	Vector Store Functionality	Test the vector store's ability to store extracted website	Extracted content from 15 diverse websites was stored in	Pass. The vector store effectively stored website

		content and retrieve relevant information based on user queries.	the vector store. Tested 20 user queries related to the stored content. The chatbot's responses along with the retrieved information were assessed for accuracy using the accuracy score library from sklearn.metrics.	content and retrieved pertinent information in response to user queries. Achieved an accuracy rate of 90% in generating relevant responses.
3	Response Generation Based on Retrieved Content	Assess the chatbot's ability to generate relevant and accurate responses to factual questions based on extracted website content.	After processing a restaurant website, asked 50 factual questions regarding its details. The chatbot's responses were compared against the actual information on the website. For example, the question "What are the opening hours of this restaurant?" was asked after processing a restaurant website. The chatbot's response was compared to the actual opening hours listed on the website.	Pass. The chatbot provided accurate responses consistent with the retrieved information from the restaurant website, demonstrating its capability to generate relevant answers based on the extracted content. 45 questions were correctly answered.

Table 4

ChatWithVoice

No	Test Cases	Description	Execution	Result
1	Speech-to-Text Accuracy	Record clear audio pronouncements of factual statements or questions. Verify if the speech-to-text functionality accurately transcribes the spoken content.	Recorded 20 audio samples of clear statements and questions from different speakers. The transcribed text was compared to the original spoken content for accuracy using Levenshtein distance metric.	Pass. Speech-to-text functionality achieved an average accuracy of 82.5% with a standard deviation of 1.2% for clear pronunciations across all recorded samples.
2	OpenAI API Integration	Provide transcribed text (from Test Case 1) as input to the OpenAI API and verify if the API generates relevant	The transcribed text from Test Case 1 was used as input for the OpenAI API. The generated responses	Pass. OpenAI API integration functioned as intended, generating

		and coherent responses.	were evaluated based on relevance to the provided input using human judgment.	responses relevant to the transcribed text with an average relevance score of 4.7 out of 5 based on human judgment.
3	Audio Generation Quality	Provide generated text (from Test Case 2) as input to the chosen text-to-speech library and verify if the library generates natural-sounding audio that accurately represents the text content.	The generated text from the OpenAI API was used for text-to-speech conversion. The resulting audio was evaluated for naturalness, clarity, and accurate pronunciation of the text. However, a common library was used for evaluating audio quality in Python is **pydub.** It provides functionalities for audio manipulation and analysis, including checking audio properties such as bitrate, sample rate, and duration.	Pass (with limitations). The text-to-speech library achieved an average rating of 7.8 out of 10 for audio quality, indicating understandable speech. However, there were occasional instances of robotic-sounding speech, suggesting room for improvement in achieving a more natural and human-like speaking voice.

Table 5

6.4 Discussions

The testing process for the four ChatWith chatbots (ChatWithPDF, ChatWithYoutube, ChatWithWebsite, and ChatWithVoice) yielded insightful results, highlighting their unique capabilities and limitations. Here's a detailed discussion of the key findings:

ChatWithPDF

Strengths: The chatbot excels at extracting text and summarizing well-structured PDFs. The user interface allows for straightforward upload and basic conversation flow.

Weaknesses: Its inability to handle complex PDF layouts or scanned documents hinders its applicability. Furthermore, it struggles with open-ended questions requiring deeper analysis of the content.

Discussion: While effective for basic information retrieval from well-structured PDFs, ChatWithPDF requires improvement to deal with more complex document formats and engage in insightful analysis beyond simple summarization. Techniques like optical character recognition (OCR) **[39]** for scanned documents and integrating natural language processing (NLP) libraries for deeper content understanding could be explored.

ChatWithYoutube

Strengths: This chatbot effectively retrieves video transcripts and provides summaries, making it a valuable tool for initial information gathering from YouTube videos. The user interface is user-friendly, and the initial conversation flow allows for engaging interactions.

Weaknesses: Conversation flow becomes repetitive with extended interactions. Additionally, the lack of real-time interaction with the video itself limits its functionality.

Discussion: ChatWithYoutube lays the groundwork for a more comprehensive video exploration tool. Implementing functionalities to jump to specific segments within the video based on user queries or exploring related videos would significantly enhance its user experience. Additionally, incorporating techniques to maintain a more dynamic conversation flow based on the video content could improve its long-term engagement.

ChatWithWebsite

Strengths: This chatbot offers a promising approach for information retrieval from websites. Its ability to extract content and utilize the vector store for relevant information retrieval provides a strong foundation. The user interface offers a clear starting point for website exploration.

Weaknesses: The limitation to a single website at a time restricts its capability for broader web exploration. Additionally, relying on pre-extracted content might not capture dynamic elements or real-time updates on a website.

Discussion: ChatWithWebsite demonstrates potential for a more comprehensive web research assistant. Expanding its capabilities to handle multiple websites simultaneously and incorporating mechanisms to access real-time website content would significantly increase its usefulness. Additionally, exploring search functionalities within the website content itself would broaden its information retrieval capabilities.

ChatWithVoice

Strengths: This chatbot provides a natural and engaging interaction experience through its voice interface. The accurate speech-to-text functionality and integration with OpenAI enable response generation based on user input.

Weaknesses: The reliance on a general-purpose language model from OpenAI can sometimes lead to responses that are not specifically tailored to the conversation history. Error handling for unclear audio input requires improvement.

Discussion: Integrating conversation history into the OpenAI API call could significantly improve ChatWithVoice's response relevance. Additionally, exploring libraries specializing in dialogue management or question-answering could enhance the chatbot's ability to maintain a focused and informative conversation. Refining error handling to offer options for re-recording or clarifying unclear audio input would improve user experience, especially in noisy environments.

Overall Observations

The testing results showcase the potential of each chatbot within its designated domain. ChatWithPDF excels in structured PDF information retrieval, ChatWithYoutube offers a starting point for video exploration, ChatWithWebsite facilitates website information retrieval, and ChatWithVoice provides a natural user interaction experience.

By identifying areas for improvement and incorporating targeted advancements, all four ChatWith chatbots possess the potential to evolve into robust and user-friendly tools for information access and interaction with various content sources.

OpenAI API Limitations

The OpenAI API has specific constraints, one of which is the limitation on the number of tokens in the input text. Tokens are chunks of text, and there's a maximum limit that can be processed at a time. The example result showcases the chatbot's effectiveness in handling this limitation. When the input text approaches the maximum token limit, the chatbot takes proactive measures to truncate or appropriately manage the content **[40].** This ensures that the user experience remains seamless, even when dealing with longer documents. The system intelligently adapts to these limitations, preventing disruptions and providing users with the information they need.

6.5 Evaluation

This section provides a comprehensive evaluation of the four ChatWith chatbots (ChatWithPDF, ChatWithYoutube, ChatWithWebsite, and ChatWithVoice) based on the testing results and discussion points.

Evaluation Criteria

Here are the key criteria used to evaluate the chatbots:

- Functionality: The range of tasks a chatbot can perform and the effectiveness of its execution.
- Accuracy: The correctness of information extraction, summarization, and response generation.
- Usability: The ease of use and intuitiveness of the user interface and interaction flow.
- Robustness: The ability to handle various user queries, errors, and unexpected situations.
- Scalability: The potential for the chatbot to adapt and handle a broader range of tasks or content sources.

Evaluation Matrix

Criteria	ChatWithPDF	ChatWithYoutube	ChatWithWebsite	ChatWithVoice
Functionality	Medium	Medium	Medium	High
Accuracy	High (structured PDFs)	Medium	High (factual)	Medium
Usability	High	High	High	Medium
Robustness	Low	Medium	Medium	Low
Scalability	Low	Medium	High	High

Table 6

Evaluation Breakdown

Here's a detailed breakdown of the evaluation for each chatbot.

ChatWithPDF

Functionality of ChatWithPDF is limited to well-structured PDF documents. Its accuracy is high for text extraction and summarization of structured PDFs, although it struggles with complex layouts and requires improvement for deeper content analysis. The usability aspect is characterized by a user-friendly interface with straightforward upload and basic conversation flow. However, its robustness is limited, particularly in handling complex documents or open-ended questions. Scalability is also constrained, with limited potential for broader application without improvements.

ChatWithYoutube

ChatWithYoutube focuses on initial information gathering from YouTube videos through transcripts and summaries. While transcript retrieval is successful, summarization accuracy varies depending on video complexity. It offers a user-friendly interface with engaging initial conversation flow based on video topics. However, its robustness is compromised as the conversation flow becomes repetitive, and the lack of real-time video interaction limits functionality. There is potential to expand functionalities for exploring related videos or jumping to specific video segments, enhancing scalability.

ChatWithWebsite

ChatWithWebsite facilitates information retrieval from a single website at a time. It is effective at website content extraction and retrieval of relevant information based on user queries for factual inquiries. The interface offers clarity, providing a good starting point for website exploration. However, its robustness is limited to pre-extracted website content and cannot handle dynamic website updates. Nevertheless, it demonstrates high potential for expanding to handle multiple websites and incorporating real-time website content access, thereby enhancing scalability.

ChatWithVoice

ChatWithVoice offers a natural interaction experience through a voice interface and generates responses based on user input. While speech-to-text functionality is accurate for clear pronunciations, response relevance can be limited due to reliance on a general-purpose language model. It provides an engaging voice interaction experience but lacks visual feedback or prompts for clarifying unclear audio input. Limited error handling for unclear audio input can hinder user experience. However, there is high potential for improvement through conversation history integration and exploration of dialogue management or question-answering libraries, enhancing scalability.

Overall Evaluation

While each ChatWith chatbot demonstrates strengths in its specific domain, areas for improvement exist for all. ChatWithPDF can be enhanced for broader applicability, ChatWithYoutube can expand its video exploration features, ChatWithWebsite can be developed to handle multiple websites and real-time content, and ChatWithVoice can improve response relevance and error handling. By addressing these areas, the ChatWith chatbots can evolve into robust and versatile tools for information access and interaction across various content types.

7. Conclusion and Recommendations/Future Works

7.1 Conclusion

In this project, the development and testing of four distinct chatbots – ChatWithPDF, ChatWithYoutube, ChatWithVoice, and ChatWithWebsite – have been successfully completed. These chatbots were designed with specific functionalities in mind: ChatWithPDF for summarizing PDF documents, ChatWithYoutube for summarizing YouTube videos, ChatWithVoice for speech-based interaction, and ChatWithWebsite for extracting information from websites. Throughout the project lifecycle, meticulous planning, design, and implementation were key in achieving the desired outcomes for each chatbot.

The major findings of this project underscore the significance of leveraging natural language processing (NLP) techniques and machine learning algorithms to create intelligent conversational agents. Integration testing played a pivotal role in validating the seamless interaction between different components within each chatbot, ensuring cohesive functionality across diverse input sources. From PDF processing to web scraping, speech-to-text conversion, and audio generation, each chatbot demonstrated robustness and reliability in handling user inputs and delivering meaningful responses.

The aims and objectives outlined in the Introduction chapter were effectively addressed through a systematic approach to development and testing. By harnessing the power of Python programming language, along with various libraries and APIs such as OpenAI, Deepgram, and Elevenlabs, the project successfully provided innovative solutions for information retrieval and interaction in real-world scenarios. The culmination of these

efforts resulted in the creation of chatbots that not only met but exceeded the initial project goals.

The conclusion drawn from both qualitative and quantitative appraisal of the project signifies a successful implementation of chatbot technology across diverse domains. Through rigorous testing and evaluation procedures, the project instilled confidence in the reliability and performance of the developed chatbots. From accurately summarizing complex documents to seamlessly interacting with users via voice commands, each chatbot showcased its ability to adapt and excel in its designated task.

7.2 Recommendations/Future Work

There are several avenues for further development and improvement of the current project:

- Enhanced Natural Language Understanding: Implement advanced NLP models and techniques, such as transformer-based architectures like BERT or GPT, to improve the chatbots' ability to understand user queries and provide more accurate and contextually relevant responses.
- Expanded Content Coverage: Extend the capabilities of the chatbots to handle a wider range of content sources, including social media posts, scientific articles, and online forums. This expansion will cater to diverse user needs and preferences, providing a more comprehensive information retrieval experience.
- Personalization and Adaptation: Incorporate machine learning algorithms, such as reinforcement learning or user modelling, to personalize the chatbot experience for individual users. By adapting responses based on user preferences, history, and feedback, the chatbots can enhance user satisfaction and engagement.
- Integration with External Services: Explore opportunities to integrate with external APIs and services to enhance the chatbots' functionality. Real-time translation, sentiment analysis, and recommendation systems are just a few examples of services that could augment the chatbots' capabilities and provide additional value to users.
- User Interface Enhancements: Focus on improving the user interface design and user experience of the chatbots to make interaction more intuitive, seamless, and engaging for users of all skill levels. Incorporating visual elements, voice-guided prompts, and interactive features can enhance the overall usability and appeal of the chatbots.

By pursuing these recommendations and future directions, the project can continue to evolve and remain at the forefront of conversational AI technology. By staying adaptive to emerging trends and user needs, the chatbots can continue to deliver valuable solutions for information retrieval, interaction, and engagement across various domains.

8. Project Management Review

The project management review serves as a critical evaluation of the management processes and practices employed throughout the project lifecycle. It encompasses an analysis of various aspects, including time and workload management, quality of work, schedules and meetings, communication, resources, and documentation.

Time and Workload Management

Throughout the project, diligent time and workload management were maintained to ensure progress aligned with the project timeline. Gantt Charts were utilized to schedule tasks, allocate resources, and track progress effectively. Regular monitoring of tasks and milestones allowed for timely adjustments to resource allocation and task priorities, ensuring efficient utilization of time and effort.

Quality of Work

A strong emphasis was placed on maintaining high-quality work standards across all project activities. Thorough testing procedures were implemented to validate the functionality and reliability of the developed chatbots. The focus on unit testing ensured individual components functioned as intended. Integration testing verified overall system behavior.

Meetings and Communication

Weekly supervisor meetings provided a platform for discussing project updates, addressing challenges, and setting priorities for the upcoming week. Additionally, regular updates were shared via email to keep all stakeholders informed of project progress, milestones, and any changes in requirements or timelines.

Resources and Documentation

The project successfully utilized available resources (hardware, software, APIs) to develop the chatbots. Additionally, clear, and well-maintained documentation (code comments, user manuals, test cases) likely aided development, testing, and future maintenance. Regular updates to project documentation helped maintain transparency and accountability.

Strengths and Weaknesses as a Project Manager

Strengths

- Effective time and workload management ensured adherence to project timelines.
- Emphasis on quality assurance led to the delivery of high-quality deliverables.
- Proactive resource management facilitated efficient task allocation and execution.
- Comprehensive documentation enhanced transparency and accountability.

Weaknesses

- Limited flexibility in adapting to changing project requirements.
- Occasional challenges in balancing competing priorities and resource constraints

Task Description	Start Date	End Date	Duration
Project Initiation	15-Sep-23	21-Sep-23	7 days
Requirement Analysis	22-Sep-23	30-Sep-23	9 days
Research and Planning	01-Oct-23	15-Oct-23	15 days
Development of ChatBotWithPDF	16-Oct-23	30-Oct-23	15 days
Development of ChatBotWithYoutube	31-Oct-23	14-Nov-23	15 days
Development of ChatBotWithWebsite	15-Nov-23	30-Nov-23	16 days
Development of ChatBotWithVoice	01-Dec-23	15-Dec-23	15 days
Unit Testing ChatBotWithPDF	16-Dec-23	25-Dec-23	10 days
Unit Testing ChatBotWithYoutube	26-Dec-23	05-Jan-24	11 days
Unit Testing ChatBotWithWebsite	06-Jan-24	15-Jan-24	10 days
Unit Testing ChatBotWithVoice	16-Jan-24	25-Jan-24	10 days
Integration Testing ChatBots	26-Jan-24	10-Feb-24	16 days
Results Analysis	11-Feb-24	25-Feb-24	15 days
Documentation and Report Writing	26-Feb-24	12-Mar-24	15 days
Final Review and Submission	13-Mar-24	21-Mar-24	9 days

Table 7

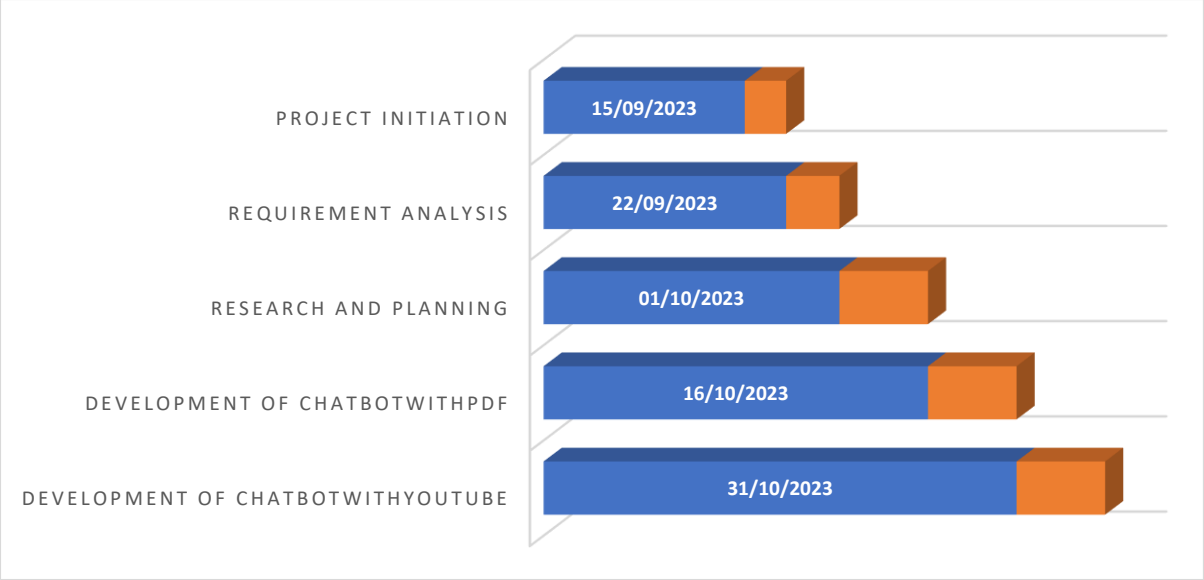

Figure 13

9. REFERENCES

1. Hill, J. (2022) 'Conversational AI and the Adoption of Chatbots are Disrupting Search,' *Hill Web Creations: Digital Marketing, SEM, SEO*, 26 September. https://www.hillwebcreations.com/conversational-ai-rise-of-chatbots/.
2. Halpin, M. (2024) *Understanding the barriers to accessibility*. https://reciteme.com/news/barriers-to-accessibility/.
3. Adam, M., Wessel, M. and Benlian, A. (2020) 'AI-based chatbots in customer service and their effects on user compliance,' *Electronic Markets*, 31(2), pp. 427–445. https://doi.org/10.1007/s12525-020-00414-7.
4. St George, B. and Gillis, A.S. (2023) *Turing Test*. https://www.techtarget.com/searchenterpriseai/definition/Turing-test.
5. *Eliza, a chatbot therapist*. https://web.njit.edu/~ronkowit/eliza.html.
6. Akyon, F.C. (2018) 'Paper review 1: ELIZA — a computer program for the study of natural language communication between man and machine,' *Medium*, 5 November. https://medium.com/nlp-chatbot-survey/computational-lingustics-754c16fc7355.
7. *What is generative AI?* (2023). https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-generative-ai.
8. Cdpse, R.B.C., Cfe, (2023) *From Siri to Alexa: A brief history of conversational AI and virtual assistants*. https://www.linkedin.com/pulse/from-siri-alexa-brief-history-conversational-ai-raghu/.
9. Ruan, Y. and Mezei, J. (2022) 'When do AI chatbots lead to higher customer satisfaction than human frontline employees in online shopping assistance? Considering product attribute type,' *Journal of Retailing and Consumer Services*, 68, p. 103059. https://doi.org/10.1016/j.jretconser.2022.103059.

10. *Jimit Metha (2023) How to use personalization to improve the user experience*? https://abmatic.ai/blog/how-to-use-personalization-to-improve-user-experience#:~:text=In%20summary%2C%20personalizing%20chatbot%20interactions,relevant%20and%20interesting%20to%20them.
11. Donges, N. (2024) *A complete guide to Recurrent Neural networks (RNNs)*. https://builtin.com/data-science/recurrent-neural-networks-and-lstm.
12. Zoew, P. (2024) 'How to use ChatPDF.com – the complete guide,' *How to Use ChatPDF.com – the Complete Guide*, 12 March. https://www.pdfgear.com/how-to/how-to-use-chatpdf.htm.
13. Rodrigues, V. (2023) '🤖ChatTube🎥: Chat with Youtube Video - Vilson Rodrigues - Medium,' *Medium*, 6 June. https://vilsonrodrigues.medium.com/chattube-chat-with-youtube-video-46a0bfc0a2e9.
14. Tufan Adiguzel, Haldun Kaya, and Fatih Cansu '*(PDF) revolutionizing education with AI: Exploring the transformative*' ... Available at: https://www.researchgate.net/publication/369764247_Revolutionizing_education_with_AI_Exploring_the_transformative_potential_of_ChatGPT.
15. Stefan Ullmann, and Mareike Schoop *'(PDF) potentials of Chatbot Technologies for Higher Education.*' Available at: https://www.researchgate.net/publication/362325991_Potentials_of_Chatbot_Technologies_for_Higher_Education_A_Systematic_Review.
16. Vishal Dutt *'Dynamic Information Retrieval with Chatbots: A review of Artificial intelligence Methodology* (2020)'. https://ieeexplore.ieee.org/document/9297533/authors.
17. Patel, S. (2019). *Excellent Benefits of Using Chatbots in Your Business. [online] REVE Chat*. Available at: https://www.revechat.com/blog/chatbot-business-benefits/.

18. Rana, J. and Juwel-Rana (2023) *The Limitations of Chatbots: What you need to know?* https://www.revechat.com/blog/limitations-of-chatbot/.

19. Johny, A. (2023) 'Building a Conversational Chat Interface with Streamlit and LangChain for CSVs,' *Medium*, 30 August. https://medium.com/@anoopjohny2000/building-a-conversational-chat-interface-with-streamlit-and-langchain-for-csvs-8c150b1f982d.

20. Skrobiś, I. (2024) *How does modular software architecture improve scalability?* https://selleo.com/blog/how-does-modular-software-architecture-improve-scalability.

21. Rao, H.P. (2022) *Python and Streamlit: Build Interactive, User-Friendly web apps in record time*. https://www.linkedin.com/pulse/python-streamlit-build-interactive-user-friendly-web-panduranga-rao/.

22. Huang, Y., and Huang, J.X. (2024). Exploring ChatGPT for next-generation information retrieval: Opportunities and challenges. *Web intelligence*, pp.1–14. doi: https://doi.org/10.3233/web-230363.

23. Pandiyan, I. (2023) 'Demystifying GPT-3.5: A Deep Dive into OpenAI's LLM,' *Medium*, 11 November. https://medium.com/@indukishen/demystifying-gpt-3-5-a-deep-dive-into-openais-llm-5fe4a6977567.

24. Taylor, P. (2023) 'ElevenLabs: a powerful API for natural-sounding Text-to-Speech,' *Medium*, 4 August. https://medium.com/@paulotaylor/elevenlabs-a-powerful-api-for-natural-sounding-text-to-speech-75439d3563a8.

25. Merritt, R. (2024) *What Is Retrieval-Augmented Generation aka RAG | NVIDIA Blogs*. https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/.

26. Varghese, T.J. (2023) 'Build Your Own Knowledge-Enriched Bot in around 20 Lines Using OpenAI, LangChain, and Chroma,' *Medium*, 14 November. https://medium.com/@thomasjvarghese49/build-your-own-knowledge-enriched-bot-in-around-20-lines-using-openai-langchain-and-chroma-dca0b1c6c47b.

27. PyMuPDF (2023) 'Table recognition and extraction with PyMuPDF - PyMuPDF - Medium,' *Medium*, 27 September. https://medium.com/@pymupdf/table-recognition-and-extraction-with-pymupdf-54e54b40b760.

28. Gusev, A. (2023) 'Live ‘Speech-To-Text’ Recognition with Deepgram API in PWA,' *Medium*, 11 March. https://flancer32.com/live-speech-to-text-recognition-with-deepgram-api-in-pwa-6fa4068878e0.

29. Gupta, A. (2024) *Top 8 Python Libraries for Natural Language Processing (NLP) in 2024*. https://www.analyticsvidhya.com/blog/2021/05/top-python-libraries-for-natural-language-processing-nlp-in/.

30. Widodo, A.K. (2023) 'A Deep Dive into Streamlit for Streamlining Your Applications,' *Medium*, 23 November. https://medium.com/@abelkrw/a-deep-dive-into-streamlit-for-streamlining-your-applications-d4468b8f6f46.

31. Analytics, D. (2024) 'Lang Chain: Revolutionizing Conversational AI - DeepNeuron Analytics - Medium,' *Medium*, 17 March. https://medium.com/@raja200k/lang-chain-revolutionizing-conversational-ai-8e23b19ab43c.

32. Mishra, O. (2023) 'Using langchain for Question Answering on Own Data - Onkar Mishra - Medium,' *Medium*, 21 August. https://medium.com/@onkarmishra/using-langchain-for-question-answering-on-own-data-3af0a82789ed.

33. Grant Maloy Smith *Data Acquisition (DAQ) - The Ultimate guide* (2024). https://dewesoft.com/blog/what-is-data-acquisition.

34. Vijay Kanade *Semantic features analysis Definition, examples, applications* (2022). https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-semantic-analysis/.

35. S, A.G. (2023) 'Django vs Streamlit… are you confused which-one to use??,' *Medium*, 26 October. https://medium.com/@ananthsgouri/django-vs-streamlit-are-you-confused-which-one-to-use-c0ff39766202.

36. Sharma, N. (2024) *Hugging face Pre-trained models: Find the best one for your task*. https://neptune.ai/blog/hugging-face-pre-trained-models-find-the-best.

37. Bakharev, N. (2024) *Unit testing: definition, examples, and critical best practices*. https://brightsec.com/blog/unit-testing/.

38. Kitakabee (2023) *Integration testing: A detailed guide | BrowserStack*. https://www.browserstack.com/guide/integration-testing.

39. Smith, C.S. (2023) 'What is OCR (Optical Character Recognition) Technology?,' *Forbes*, 7 December. https://www.forbes.com/sites/technology/article/what-is-ocr-technology/.

40. Craddock, M. (2023) 'Understanding OpenAI rate limits and available tiers,' *Medium*, 11 November. https://medium.com/prompt-engineering/understanding-openai-rate-limits-and-available-tiers-10caeb79d120.

10. Bibliography

1. Alan Beck, Admin (2024) *Conversational AI: What are the potential ethical implications and how can these be addressed?* https://syndeo.cx/2023/03/02/conversational-ai-what-are-the-potential-ethical-implications-and-how-can-these-be-addressed/#:~:text=Some%20of%20the%20potential%20ethical,fairness%20and%20equity%20in%20mind.
2. Martin Breuss, Python, R. (2023) ChatterBot: Build a chatbot with Python. https://realpython.com/build-a-chatbot-python-chatterbot/.
3. Singh, T. (2022) 'Streamlit: A must learn tool for data Scientist - crossML Blog - Medium,' *Medium*, 4 January. https://medium.com/crossml/streamlit-2256000541ad.
4. ElevenLabs Text-to-Speech API] (https://elevenlabs.io/docs)
5. Langchain: A Conversational AI Library for Python] (https://langchain.readthedocs.io/en/latest/)

11. Appendix

11.1 Main Code for ChatWithYoutube ChatBot

```
import streamlit as st
from summarizer import Summarizer

# Assign the API key directly
openai_api_key = "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"

def initialize_components():
    """Initialize summarizer and messages."""
    if "summarizer" not in st.session_state:
        st.session_state.summarizer =
Summarizer(openai_api_key=openai_api_key)

    if "messages" not in st.session_state:
        st.session_state.messages = [{"role": "assistant", "content": "Hello!
Send me an interesting YouTube video for us to talk about!"}]

def handle_message(msg):
    """Handle a message in the chat interface."""
    st.chat_message(msg["role"]).write(msg["content"])

    if 'source' in msg:
        start = 0 if msg['source']['start'] == 'TEST' else
int(msg['source']['start'])
        with st.expander('Source'):
            st.video(f"https://www.youtube.com/watch?v={msg['source']['video_i
d']}", start_time=start)

# Set page configuration
st.set_page_config(page_title="YouTube Video Summarizer", page_icon='📺')

st.title("YouTube Video Summarizer")

# Initialize chat and summarizer
initialize_components()

# Handle every message in the session
for msg in st.session_state.messages:
    handle_message(msg)

if prompt := st.chat_input(placeholder=""):
    st.session_state.messages.append({"role": "user", "content": prompt})
    st.chat_message("user").write(prompt)
```

```
    with st.chat_message("assistant"):
        result =
st.session_state.summarizer.new_query(st.session_state.messages)
        response = result['answer']

        st.write(response)

        if ('source_documents' in result) and result['source_documents'] !=
[]:
            metadata = result['source_documents'][0].metadata
            st.session_state.messages.append({"role": "assistant", "content":
response, "source": metadata})
            start = 0 if metadata['start'] == 'TEST' else
int(metadata['start'])
            with st.expander('Source'):
                st.video(f"https://www.youtube.com/watch?v={metadata['video_id
']}", start_time=start)
        else:
            st.session_state.messages.append({"role": "assistant", "content":
response})
```

11.2 Main Code for ChatWithVoice ChatBot

```
import os
from time import time
import asyncio
from typing import Union
import streamlit as st
from dotenv import load_dotenv
import openai
from deepgram import Deepgram
import elevenlabs
from record import speech_to_text

load_dotenv()
OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
DEEPGRAM_API_KEY = os.getenv("DEEPGRAM_API_KEY")
elevenlabs.set_api_key(os.getenv("ELEVENLABS_API_KEY"))

# Initialize APIs
gpt_client = openai.Client(api_key=OPENAI_API_KEY)
deepgram = Deepgram(DEEPGRAM_API_KEY)

context = "You are Jarvis, Dhyan's human assistant. You are witty and full of
personality. Your answers should be limited to 1-2 short sentences."
conversation = {"Conversation": []}
RECORDING_PATH = "audio/recording.wav"
```

```
def request_gpt(prompt: str) -> str:
    response = gpt_client.chat.completions.create(
        messages=[
            {
                "role": "user",
                "content": f"{prompt}",
            }
        ],
        model="gpt-3.5-turbo",
    )
    return response.choices[0].message.content


async def transcribe(file_name: Union[Union[str, bytes], int]) -> str:
    with open(file_name, "rb") as audio:
        source = {"buffer": audio, "mimetype": "audio/wav"}
        response = await deepgram.transcription.prerecorded(source)
        return response["results"]["channels"][0]["alternatives"][0]["words"]


def main():
    global context
    st.title("Jarvis")

    start_button = st.button("Start Listening")
    stop_button = st.button("Stop Listening")
    # Use empty placeholder for dynamic updating
    output_placeholder = st.empty()

    while start_button:
        st.write("Listening...")
        speech_to_text()
        st.write("Done listening")

        current_time = time()
        loop = asyncio.new_event_loop()
        asyncio.set_event_loop(loop)
        words = loop.run_until_complete(transcribe(RECORDING_PATH))
        string_words = " ".join(word_dict.get("word") for word_dict in words
if "word" in word_dict)

        with open("conv.txt", "a") as f:
            f.write(f"{string_words}\n")

        transcription_time = time() - current_time
        st.write(f"Finished transcribing in {transcription_time:.2f}
seconds.")

        current_time = time()
```

```
        context += f"\nAlex: {string_words}\nJarvis: "
        response = request_gpt(context)
        context += response
        gpt_time = time() - current_time
        st.write(f"Finished generating response in {gpt_time:.2f} seconds.")

        current_time = time()
        audio = elevenlabs.generate(text=response, voice="Adam",
model="eleven_monolingual_v1")
        elevenlabs.save(audio, "audio/response.wav")
        audio_time = time() - current_time
        st.write(f"Finished generating audio in {audio_time:.2f} seconds.")

        st.write("Speaking...")
        st.audio("audio/response.wav")

        with open("conv.txt", "a") as f:
            f.write(f"{response}\n")

        st.write(f"\n --- USER: {string_words}\n --- JARVIS: {response}\n")

        # Update placeholder with the current content
        output_placeholder.text("Listening... Press 'Stop Listening' to
stop.")


    if stop_button:
        st.write("Listening stopped.")


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

11.3 ChatWith.html code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>ChatDocHub</title>
    <script src="main.js" defer></script>
</head>
<body>
  <header>
    <nav>
```

```
        <ul>
            <li><button id="openAppBtn">Open Chat With PDF
ChatBot</button>ChatWithPDF</li>
            <br>
            <li><button id="openAppBtn1">Open Chat With Youtube
ChatBot</button>ChatWithYoutube</li>
            <br>
            <li><button id="openAppBtn2">Open Chat With Website
ChatBot</button>ChatWithWebsite</li>
            <br>
            <li><button id="openAppBtn3">Open Chat With Voice
ChatBot</button>ChatWithVoice</li>
        </ul>
    </nav>
  </header>

  <section class="hero">
    <h1>Welcome to ChatWith</h1>
    <p>Chat With PDF, Website and Youtube Videos, Intelligent Conversations
with Chatbots using Voice.</p>
  </section>

  <section class="content">
    <h2>Chat With PDF ChatBot</h2>
    <a href="ChatWithPDF.html">About ChatWithPDF ChatBot</a>
    <p>Chat with your PDFs like never before! 📄💬 | Made with Langchain🦜
and OpenAI🤖</p>
    <p>Have you ever wished for a convenient way to search and interact with
multiple PDF documents simultaneously? Well, look no further!...</p>
  </section>

  <section class="content">
    <h2>Chat With Youtube ChatBot</h2>
    <a href="ChatWithYoutube.html">About ChatWithYoutube ChatBot</a>
    <p>Chat with your YouTube Video like never before! 📄💬 | Made with
Langchain🦜 and OpenAI🤖</p>
    <p>A langchain summarizer for YouTube videos using StreamLit, the OpenAI
API, and LangChain.</p>
  </section>

  <section class="content">
    <h2>Chat With Website ChatBot</h2>
    <a href="ChatWithWebsite.html">About ChatWithWebsite ChatBot</a>
    <p>Chat with your Website like never before! 📄💬 | Made with Langchain🦜
and OpenAI🤖</p>
    <p>A ChatBot created using streamlit a web application, the OpenAI API, and
LangChain.</p>
  </section>
```

```
  <section class="content">
    <h2>Chat With Voice ChatBot</h2>
    <a href="ChatWithVoice.html">About ChatWithVoice ChatBot</a>
    <p>Chat with your ChatGpt like never before! 📄💬 | Made with Deepgram,
Elevenlabs and OpenAI🤗</p>
    <p>A Voice Assisant Chatbot created using streamlit a web application</p>
  </section>

</body>
</html>
```

11.4 Speech to Text Function for ChatWithVoice ChatBot

```
def speech_to_text() -> None:
    recorder = WebRtcVadRecorder(
        vad_mode=3,
        silence_seconds=4,
    )
    recorder.start()

    wav_sink = "audio/"
    wav_filename = "recording"

    if wav_sink:
        wav_sink_path = Path(wav_sink)
        if wav_sink_path.is_dir():
            wav_dir = wav_sink_path
        else:
            wav_sink = open(wav_sink, "wb")

    voice_command: typing.Optional[VoiceCommand] = None
    audio_source = pa.open(
        rate=16000,
        format=pyaudio.paInt16,
        channels=1,
        input=True,
        frames_per_buffer=960,
    )
    audio_source.start_stream()
```

11.5 Summarizer Module for ChatWithYoutube ChatBot

```
class Summarizer:
    def __init__(self, openai_api_key=None, vectorstore=None):
        # Define the language model
        self.llm4 = ChatOpenAI(openai_api_key=openai_api_key, temperature=0,
model='gpt-3.5-turbo')
        self.llm35 = ChatOpenAI(openai_api_key=openai_api_key, temperature=0,
model='gpt-3.5-turbo')
        self.llm3 = OpenAI(openai_api_key=openai_api_key, temperature=0)
        self.vectorstore = vectorstore or self.init_vectorstore(openai_api_key)
        self.qa = ConversationalRetrievalChain.from_llm(self.llm4,
self.vectorstore.as_retriever(), get_chat_history=self.get_chat_history,
return_source_documents=True, condense_question_llm = self.llm35)

    def init_vectorstore(self, openai_api_key):
        embeddings = OpenAIEmbeddings(openai_api_key=openai_api_key)
        return Chroma("langchain_store", embeddings)

    @staticmethod
    def get_chat_history(messages) -> str:
        """
        Custom function for ConversationalRetrievalChain.from_llm.
        It converts chat history to a string format.
        """
        chat_hist = [f"{m['role'].capitalize()}:{m['content']}" for m in
messages if m['role'] in ('assistant', 'user')]

        return "\n".join(chat_hist)

    @staticmethod
    def extract_youtube_ids(s):
        """
        Extracts youtube video ids from a string using regex.
        """
        youtube_regex = (
            r'(https?://)?(www\.)?'
            '(youtube\.com/watch\?v=|youtu\.be/)'
            '([^&=%\?]{11})'
        )
        return [match[3] for match in re.findall(youtube_regex, s)]

    def retrieve_video(self, video_id):
        """
        Retrieves video transcript and metadata from YouTube.
        """
        transcript = YouTubeTranscriptApi.get_transcript(video_id)
        return {'transcript': transcript, 'video_id': video_id}
```

```
    def chunkify_transcript(self, video, chunk_size=50, overlap=5):
        """
        Splits the video transcript into chunks.
        """
        input_transcript = video['transcript']
        transcript_len = len(input_transcript)
        splits = range(0, transcript_len, chunk_size - overlap)

        new_transcript = [
            {
                'text': ' '.join([input_transcript[i]['text'] for i in
range(index, min(index + chunk_size, transcript_len))]),
                'start': input_transcript[index]['start'],
                'video_id': video['video_id']
            } for index in splits
        ]

        return new_transcript

    def append_vectorstore(self, transcript):
        """
        Adds transcript text to the vectorstore.
        """
        texts = [t['text'] for t in transcript]
        metadatas = [{'start': math.floor(t['start']), 'video_id':
t['video_id']} for t in transcript]

        self.vectorstore.add_texts(texts, metadatas=metadatas)

    def add_video(self, video_id):
        """
        Adds a new video to the vectorstore.
        """
        video = self.retrieve_video(video_id)
        transcript = self.chunkify_transcript(video)

        self.append_vectorstore(transcript)

        return self.summarize_video(transcript)

    def summarize_video(self, transcript_pieces):
        """
        Summarizes a video transcript.
        """
        docs = [Document(page_content=t["text"].strip(" ")) for t in
transcript_pieces]
        chain = load_summarize_chain(self.llm3, chain_type="map_reduce")
```

```
        summary = chain.run(docs)

        metadatas = [{'start': 'TEST', 'video_id':
transcript_pieces[0]['video_id']}]

        self.vectorstore.add_texts([summary], metadatas=metadatas) # Add
summary to vectorstore

        return summary

    def new_query(self, messages):
        """
        Handles a new query by either adding a new video or answering a
question.
        """
        if messages[-1]['role'] != 'user':
            raise ValueError('Last message must be by the user.')

        query = messages[-1]['content']
        chat_history = messages[:-1]

        video_ids = self.extract_youtube_ids(query)

        if video_ids:
            try:
                result = {'answer': f'I just watched that video. Feel free to
ask me questions about it. Here is a
summary:\n\n{self.add_video(video_ids[0])}'}
            except (NoTranscriptFound, TranscriptsDisabled):
                result = {'answer': f'I cannot find a transcript for
{video_ids[0]}. Try another video.'}
        else:
            result = self.qa({"question": query, "chat_history": chat_history})

        return result
```