॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS INSTITUTE OF TECHNOLOGY, NAGPUR
Symbiosis International (Deemed University)
(Established under section 3 of the UGC Act, 1956)
Re-accredited by NAAC with 'A++' Grade | Awarded Category – I by UGC
Founder: Prof. Dr. S. B. Mujumdar, M. Sc., Ph. D. (Awarded Padma Bhushan and Padma Shri by President of India)

ONE MONTH INTERNSHIP REPORT

Submitted
By

Mr. Mukund Kuthe

Semester: 5th

Section: B

PRN: 23070521082

at

Riverstream Consultancy Services

JULY 2025

Mouza - Wathoda, Nagpur 440008, Maharashtra, India. | Tel: +91-712-6192370 / +91-712-6192368
Email: info@sitnagpur.siu.edu.in | Website: www.sitnagpur.edu.in

Declaration

I hereby declare that the Industry Internship Report submitted herein has been carried out by me in Riverstream Consultancy Services as part of the One-Month Internship.

This report is a true and accurate representation of the tasks performed, knowledge gained, and experiences acquired during the internship period. It reflects my individual effort, understanding, and involvement in the assigned activities.

I confirm that the contents of this report are original and authentic, and no part of it has been copied or reproduced from any other report or document without proper citation.

Any content sourced from references, manuals, websites, or other external materials is duly acknowledged in the report.

Name of Student: Mukund Kuthe
PRN: 23070521082
Date: 18-07-2025

Signature:

Verified by Class In-charge

Name of Class In-charge: **Prof. Soubhagya R Mallick**

Signature:

Riverstream
CONSULTANCY SERVICES

PN 36 A, Shilpa Society
Narendra Nagar,
Nagpur - 440015
PH: 9209254402 | 8484834402

25th Jun, 2025

To Whom It May Concern

This is to certify that Mukund Kuthe has successfully completed internship during the month of June, 2025.

During the period of his internship, he was found attentive, diligent and hardworking.

We wish him every success in his life and career.

Sincerely,

Manish Sayre
CEO
Riverstream Consultancy Services
Nagpur, India
manish@riverstreamweb.com

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to **Riverstream Consultancy Services** for providing me with the opportunity to complete my internship in **Artificial Intelligence** and gain valuable industry experience. This internship has been a highly enriching journey, helping me develop both technical and professional skills.

I am deeply thankful to my supervisor, **Sushant Choudhari,** for his constant guidance, insightful feedback, and encouragement throughout the internship. Their expertise and support played a crucial role in enhancing my learning experience. I also extend my appreciation to my colleagues and team members at **Riverstream Consultancy Services** for their cooperation, assistance, and for making my internship a pleasant and productive experience.

I am immensely grateful to my academic institution, Symbiosis Institute of Technology, Nagpur for facilitating this internship opportunity. I extend my heartfelt appreciation to **Dr. Nitin Rakesh**, Director of Symbiosis Institute of Technology, Nagpur, for his continuous support and encouragement in providing students with industry exposure. I would also like to extend my deepest gratitude to **Dr. Sagarkumar S. Badhiye**, Head of the Department Computer Science and Engineering, Nagpur, for his constant support, encouragement, and invaluable guidance throughout my learning experience.

Thank you all for your invaluable contributions to my learning and professional growth.

Name of Student: Mukund Kuthe

PRN: 23070521082

Signature:

Abstract

The goal of the internship project was a design and realization of a smart medical assistant under the name of MediBot, the personal assistant who can provide context-sensitive and timely responses, based on an inquiry of a particular document in the field of medicine. This project was facilitated by the increased demand in fast and reliable access to healthcare expertise. The unique combination of up-to-date natural language processing (NLP) solutions and user-friendly interface will create a mediation between the end-user and credible medical information.

Medical Chatbot runs on RAG that is combined with Large Language Models (LLMs). Instead of relying on a pretrained language model exclusively, it smartly retrieves the most important pieces of information in a given medical PDF then creates a response. Equipped with LLaMA 3.3, such a configuration will make the answers quite specific and based on the original information base.

The chatbot was developed in Python and was published on Streamlit, within which a lightweight yet interactive frontend could be achieved. PyPDF2 is used to parse and extract text of the medical document and it is embedded as a vector using Sentence Transformers. The embeddings are indexed and stored in a FAISS vector database that could quickly find the most relevant chunks of documents in a query. User questions are each compared with this index to generate content which is further given to the language model which produces the answer.

In order to enhance the user experience, such features as multi-session chat history, contextual awareness, and the possibility to show what the definite answer is the reply to were added. This increases trust as well as usability but has the benefit of that all responses are traceable to their source in the document.

The process of developing a medical Q&A assistant began clearly defining the critical requirements of a medical Q & A assistant: namely, unequaled precision, traceability, and user friendliness. The system was customised to be able to process medical information through superior embedding techniques and document cutting down. The prompt engineering also played a significant role of shaping the model not to exceed the boundaries of the healthcare sector and minimize the possibility to produce misinformed or false answers. An additional benefit came with integrating Groq LLaMA 3.3 that enhanced the efficiency and accuracy of the assistant.

On the whole, the project was an experiential exercise to do in web development, machine learning, and applied AI. The end product is a solid working prototype with subsequent future developments in higher levels of document-driven chatbots, particularly in a highly exacting field on accuracy such as healthcare.

Index

List of Tables

List of Figures

List of Symbols

Symbol	Description
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
FAISS	Facebook AI Similarity Search (Vector DB)
PDF	Portable Document Format
UI	User Interface
API	Application Programming Interface

Introduction

Artificial intelligence (AI) usage in healthcare increases exponentially especially in the field of diagnostics, communication with a patient, and access to knowledge in recent years. Advancement of large language model (LLM) has opened a new chapter of applications that can understand and react to human questions. Nevertheless, in the sensitive areas such as in medicine, mere answers provided by these models must be precise and specific as well as factual. To get to such a level of reliability would not be simply understanding the general language. Retrieval-Augmented Generation (RAG) is one example because it significantly improves over traditional retrieval tools and generative AI.

The project, with the name of MediBot- Medical Chatbot using RAG and LLM was a way to create a smart chatbot that could answer any medical-related questions with the use of a document as the source. Having completed the project as an internship at Riverstream Consultancy Services. The use of LLM and document embeddings is used in this assistant to read and interpret information in medical PDFs (manual, clinical guidelines, textbook-based notes), and answer the user query accurately.

To do this, the project utilized HuggingFace Transformer as a text embedding generator, LangChain as a structure of the application elements, FAISS (Facebook AI Similarity Search) as a data search tool, and LLaMA 3.3 model via Groq, which allows generating a detailed and context-specific response. The backend process was well organized in such a way to make it efficient and precise in the system. First the uploaded document (usually in PDF file) is divided into small, manageable text fragments. These pieces are transformed into their vectors by means of HuggingFace embeddings and indexed of FAISS. Whenever there is a question by a user, the system finds the most applicable chunks in the database and sends it to LLM so it can respond knowledgeably and reliably as befits the medical situation.

It was all done in an area-friendly and maintainable manner, which ensures that the chatbot is easily extensible or given modifications to use fields beyond healthcare. The project architecture is divided as displayed in Figures 1, into two distinct but closely connected workflows in order to enhance legibility and bring down the level of the speed of understanding.

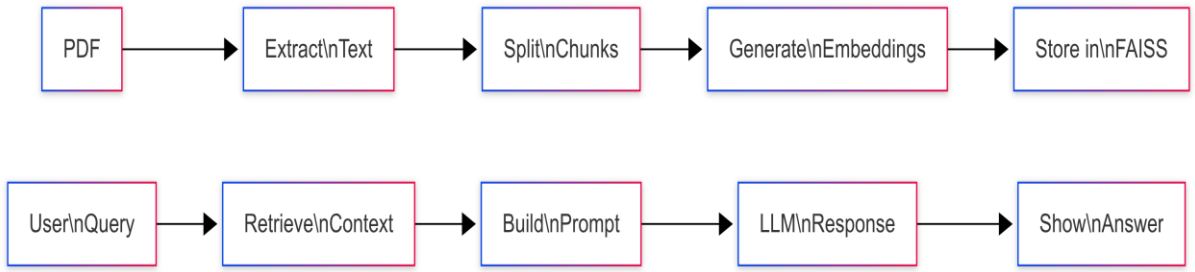


Fig. 1: Document & Query Processing Workflow

Objectives of the Internship

The internship's main goals were:

1. To create a conversational medical assistant that can respond to inquiries using a given PDF document.

2. To investigate and use contemporary AI tools such as HuggingFace embeddings, Groq LLM, and LangChain for Retrieval-Augmented Generation (RAG).

3. To use Streamlit to implement the solution in an easy-to-use interface.

4. To comprehend the entire process of web AI applications, including web deployment, model integration, and data preprocessing.

5. To acquire soft skills in a work setting, such as debugging, problem-solving, and framework adaptation.

Company Profile

Riverstream Consultancy Services is an IT-consulting and software company based in Nagpur; it was founded in 2011. The company has many services offered, including the desktop services, IT advisory, long-term maintenance and web and mobile application development. Riverstream, having a team of well-educated specialists, focuses on the delivery of safe, scalable, and high-quality digital services to clients representing various sectors.

Riverstream, the company developing over 25 applications and offering its services to clients both on the domestic and foreign market, is also more than familiar with client-centric methodology and having based much of its work on focusing on efficiency and simplicity. The company, because of its reliability and innovativeness, has been in line with the current trends in the industry.

Riverstream
CONSULTANCY SERVICES

Fig. 2: Logo of Riverstream Consultancy Services

Internship Activities

At Riverstream Consultancy Services during my internship, I had the chance to participate in all the phases of the development of MediBot, a medical chatbot that would require the use of Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG). The growth process needed a healthy dosing of artificial intelligence technologies and full-stack development.

The first step was based on obtaining the profound knowledge of the RAG architecture, and the manner it works within the area of the natural language processing. I examined the role of RAG in supplementing the traditional LLMs by increasing contextual accuracy and retrieving information.The raw text received was subsequently cleaned up and preprocessed, in preparation of higher order tasks.

In order to make sure that the chatbot gave correct and appropriate responses, I used custom prompt template that was specifically designed to serve as medical question. After having prepared the prompts, as well as the sections with the relevant documents, I incorporated those into the workflow. The Meditbot ui was developed with the help of Streamlit library.

During the last step, I performed a great number of tests when different kinds of user entries were used to make sure that the system could provide relatively accurate and useful responses. With the continuous optimization and feedback the usability and performance of the chatbot have both increased considerably. This repetitive process developed a solid starting point of scaling up with a domain-specific AI assistant targeting specifically to the field of medicine.

Skills Learned

Skill Area	Description
Retrieval-Augmented Generation (RAG)	Learned the architecture and implementation of RAG systems for enhancing LLM-based Q&A capabilities.
Python Programming	Improved proficiency in Python through building data pipelines, APIs, and Streamlit-based UI.
PDF Data Extraction	Used PyPDF2 to extract and preprocess unstructured text data from PDF files.
Embeddings and Vectorization	Applied HuggingFace Sentence Transformers to convert text into embeddings for semantic retrieval.
FAISS Vector Store	Built and queried a FAISS vector database for efficient similarity search on document chunks.
LLM Integration	Integrated and interacted with the Groq-hosted LLaMA 3.3 model to generate human-like answers.
Streamlit Development	Built an interactive and responsive chatbot interface with document and chat history support.

Table 1: Skills Learned

Challenges Faced

The conceptual and technical issues faced in carrying out the MediBot application were a few in number. The acquisition of a clear picture of Retrieval-Augmented Generation (RAG) framework and the ability to use it successfully was one of the main barriers. Because RAG combines both the ideas of traditional information retrieval with the power of the reasoning of large language models a unified and modular architecture needed some planning.

The other significant issue was related to data mining on medical PDF documents. Such reports may include complicated pages, embedded tables and technical medical terminologies. Working with such unstructured data with the help of such tools as PyPDF2 required the application of sophisticated text preprocessing techniques that allowed to guarantee that the processed content could be conveniently chunked and properly embedded.

Also, the use of large language models deployed on off-site platforms, such as Groq, created problems of latency as well as managing and handling of errors. The process of repetitive testing was crucial to cope with the restriction of a specific number of tokens, stable connection to the API and, overall, the ability of the model to provide effective and relevant answers.

On a front-end end, creating Streamlit web UI was challenging in its own right, especially since usability must be expected in addition to performance. The integration of document source visibility, chat history that does not get lost once a window is closed, and previewings were some of the features that had to be carefully approached without making the application sluggish.Nonetheless, the issues were resolved methodically, step by step, with debugging, experimenting, and learning new lessons, resulting in the construction of a stable and functional prototype.

Contribution to the Organization

In my internship at Riverstream Consultancy Services, I was one of the participants in the design and development of a functional prototype of the domain-oriented, document-driven medical chatbot called MediBot. Not only was this project consistent with the overall mission of the company to incorporate AI to real life problems, but it also preconditioned further changes in the area of smart document processing.

One of the main tasks that I did was the overall structure of the application flow assuming the extraction of information, further processing, and generation of relevant answers using the RAG (Retrieval-Augmented Generation) framework. I made sure that the system could be used in a different domain, e. g. academic research or legal analysis of documents pretty easily.

It was a practical development experience that helped improve the proprioceptive knowledge of the company in the uses of AI as well as demonstrated the strength with which modular AI tools can be developed by utilizing the open-sourced libraries and external APIs. MediBot prototype has become a good piece of reference, and now it offers new horizons in healthcare technology and smart document automation in client-suited projects.

Learning Outcomes

```python
import os
import streamlit as st
from PyPDF2 import PdfReader
from langchain.prompts import PromptTemplate
from langchain.chains import RetrievalQA
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_community.vectorstores import FAISS
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_groq import ChatGroq

from secret_api_keys import groq_api_key
os.environ['GROQ_API_KEY'] = groq_api_key
```

Fig. 3: Importing Libraries and API Setup for MediBot

```python
DB_FAISS_PATH = "vectorstore/db_faiss"
PDF_PATH = "Medical.pdf"

@st.cache_resource
def vector():
    embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
    index_file = os.path.join(DB_FAISS_PATH, "index.faiss")
    pkl_file = os.path.join(DB_FAISS_PATH, "index.pkl")

    if os.path.exists(index_file) and os.path.exists(pkl_file):
        return FAISS.load_local(DB_FAISS_PATH, embeddings, allow_dangerous_deserialization=True)
    else:
        pdf_reader = PdfReader(PDF_PATH)
        docs = []
        splitter = RecursiveCharacterTextSplitter(chunk_size=1500, chunk_overlap=150)
        for i, page in enumerate(pdf_reader.pages):
            page_text = page.extract_text()
            if page_text:
                chunks = splitter.split_text(page_text)
                for chunk in chunks:
                    docs.append({
                        "page_content": chunk,
                        "metadata": {"page_number": i + 1}
                    })
        texts = [doc["page_content"] for doc in docs]
        metadatas = [doc["metadata"] for doc in docs]
        vectorstore = FAISS.from_texts(texts, embeddings, metadatas=metadatas)
        vectorstore.save_local(DB_FAISS_PATH)
        return vectorstore

```

Fig. 4: PDF Text Extraction Using PyPDF2

```python
def load_llm():
    return ChatGroq(
        model="llama-3.3-70b-versatile",
        temperature=0.5,
    )

def context_history(chat):
    history = chat[-3:]
    context = ""
    for i in history:
        context += f"User: {i['question']}\nAssistant: {i['answer']}\n"
    return context

```

Fig. 5: Functions used for loading the language model and constructing the chat history

```python
def main():
    if "page" not in st.session_state:
        st.session_state.page = "home"

    page = st.session_state.page

    if page == "home":
        col1, col2 = st.columns(2)
        with col1:
            st.markdown("<h1>Welcome to your Medical Chatbot</h1>", unsafe_allow_html=True)
            st.markdown("<h6>A Project by Mukund Kuthe</h6>", unsafe_allow_html=True)
            if st.button("Let's Chat"):
                st.session_state.page = "chat"
                st.rerun()
        with col2:
            st.image("img.png", width=400)

    elif page == "chat":
        st.title("MediBot - Medical Chatbot")

        if "chat_history" not in st.session_state:
            st.session_state.chat_history = []

        if "saved_chats" not in st.session_state:
            st.session_state.saved_chats = []

        if "selected_chat_index" not in st.session_state:
            st.session_state.selected_chat_index = None

```

Fig. 6: Streamlit main function

```
    with st.sidebar:
        st.markdown("## Chat Options")
        if st.button("Start New Chat"):
            if st.session_state.chat_history:
                st.session_state.saved_chats.append(st.session_state.chat_history.copy())
            st.session_state.chat_history = []
            st.session_state.selected_chat_index = None

        if st.session_state.saved_chats:
            chat_options = [f"Chat {i+1} ({len(chat)} messages)" for i, chat in enumerate(st.session_state.saved_chats)]
            selected_index = st.selectbox("Select a chat to view", options=list(range(len(chat_options))), format_func=lambda x: chat_options[x])
            if st.button("View Selected Chat"):
                st.session_state.selected_chat_index = selected_index

```

Fig. 7: Sidebar chat options

```
    if st.session_state.selected_chat_index is not None:
        active_chat = st.session_state.saved_chats[st.session_state.selected_chat_index]
    else:
        active_chat = st.session_state.chat_history

    for message in active_chat:
        with st.chat_message("user"):
            st.markdown(message["question"])
        with st.chat_message("assistant"):
            st.markdown(message["answer"])

    vector_store = vector()
    llm = load_llm()

```

Fig. 8: Chat History

```
        prompt_template = PromptTemplate(
            input_variables=["context", "question"],
            template="""You are a medical assistant. Answer the question based on the context provided. If the question
is not from the context reply as \"Out of domain question.\".

Context:
{context}

Question:
{question}

Answer:"""
        )

        qa_chain = RetrievalQA.from_chain_type(
            llm=llm,
            retriever=vector_store.as_retriever(search_kwargs={"k": 3}),
            chain_type="stuff",
            chain_type_kwargs={"prompt": prompt_template},
            return_source_documents=True
        )

```

Fig. 9: Prompt Template

```
        question = st.chat_input("Enter your medical question:")
        if question:
            with st.chat_message("user"):
                st.markdown(question)

            with st.spinner("Getting answer..."):
                ques = f"{context_history(active_chat)} User: {question}"
                result = qa_chain.invoke({"query": ques})
                answer = result["result"]
                sources = result["source_documents"]

            with st.chat_message("assistant"):
                st.markdown(answer)

            active_chat.append({
                "question": question,
                "answer": answer,
            })

            if answer.strip() != "Out of domain question.":
                with st.sidebar:
                    for i, doc in enumerate(sources):
                        page = doc.metadata.get("page_number", "Unknown")
                        content_words = doc.page_content.split()
                        preview = " ".join(content_words[:40]) + ("..." if len(content_words) > 40 else "")
                        st.markdown(f"**Sources {i+1} (Page {page})**")
                        st.write(preview)
            else:
                with st.sidebar:
                    st.markdown("**No sources found.**")

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

Fig. 10: User input and source display

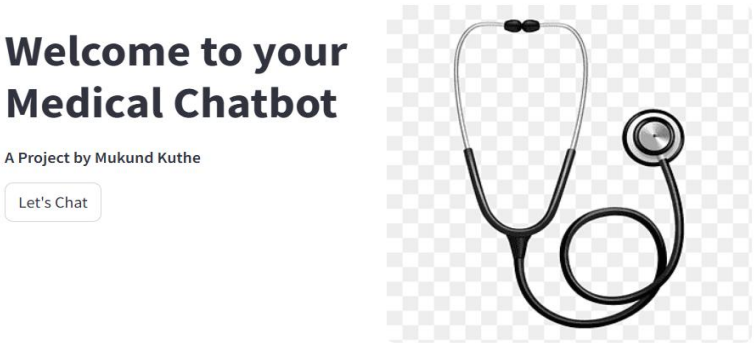


Fig. 11: Medibot welcome screen

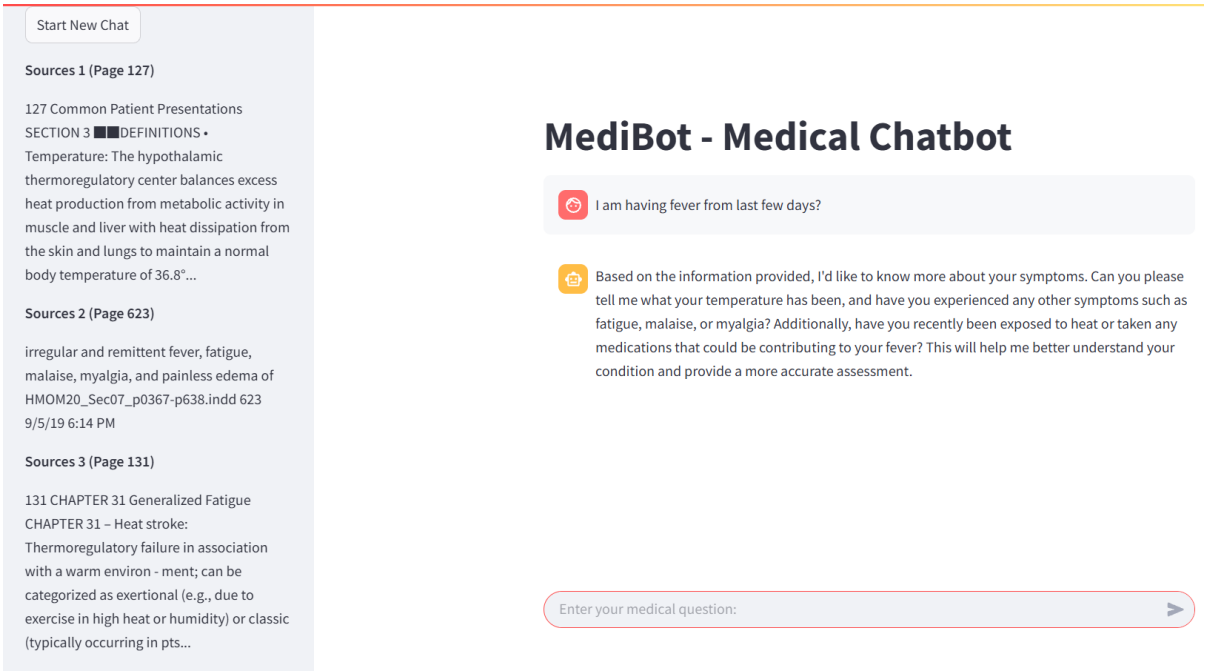


Fig. 12: Medibot Interaction with source

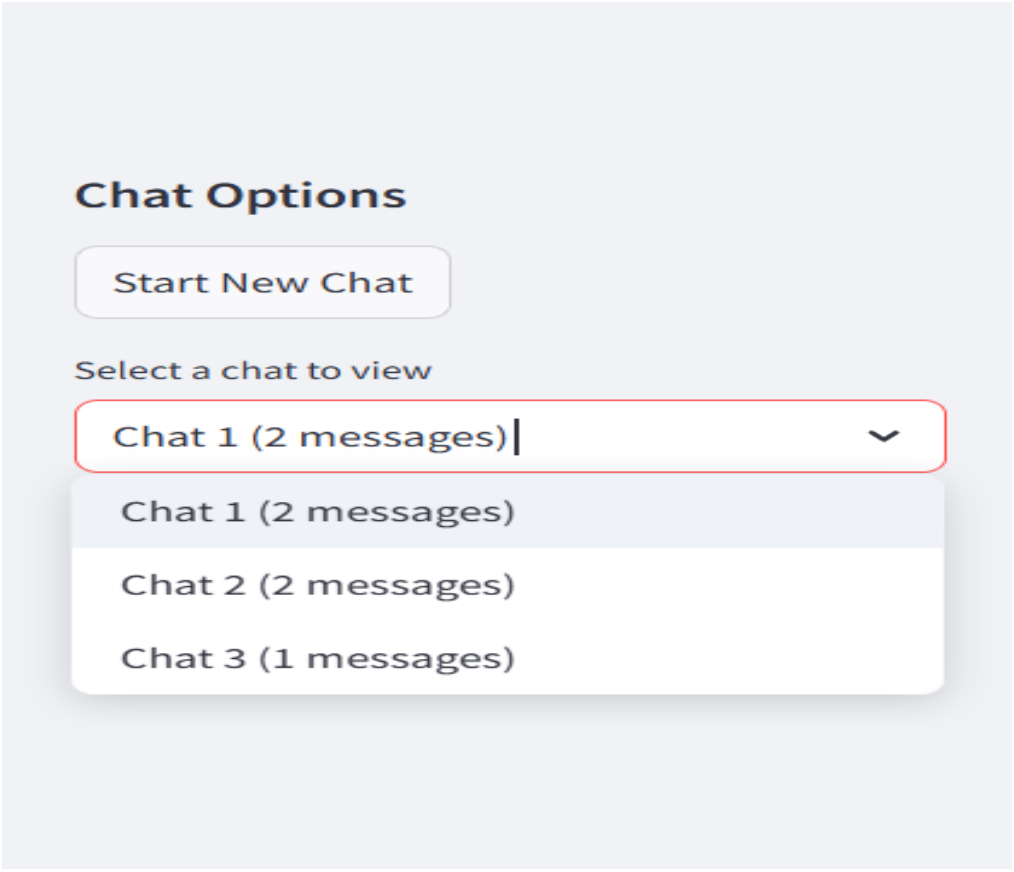


Fig. 13: Chat History Selection

Requirement	Description
langchain	Framework for building LLM-powered applications with chains and agents
langchain-huggingface	Integration to use HuggingFace embedding models with LangChain
langchain_community	Community-contributed tools for extended LangChain support
PyPDF2	Python library to extract text and metadata from PDF documents
faiss-cpu	Facebook AI Similarity Search for efficient vector similarity queries (CPU version)
sentence-transformers	Library for generating dense sentence embeddings using transformer models
streamlit	Python framework for creating interactive web apps
langchain-groq	LangChain integration for Groq-hosted LLaMA LLM

Table 2**:** Software Requirements Used in MediBot Development

Conclusion

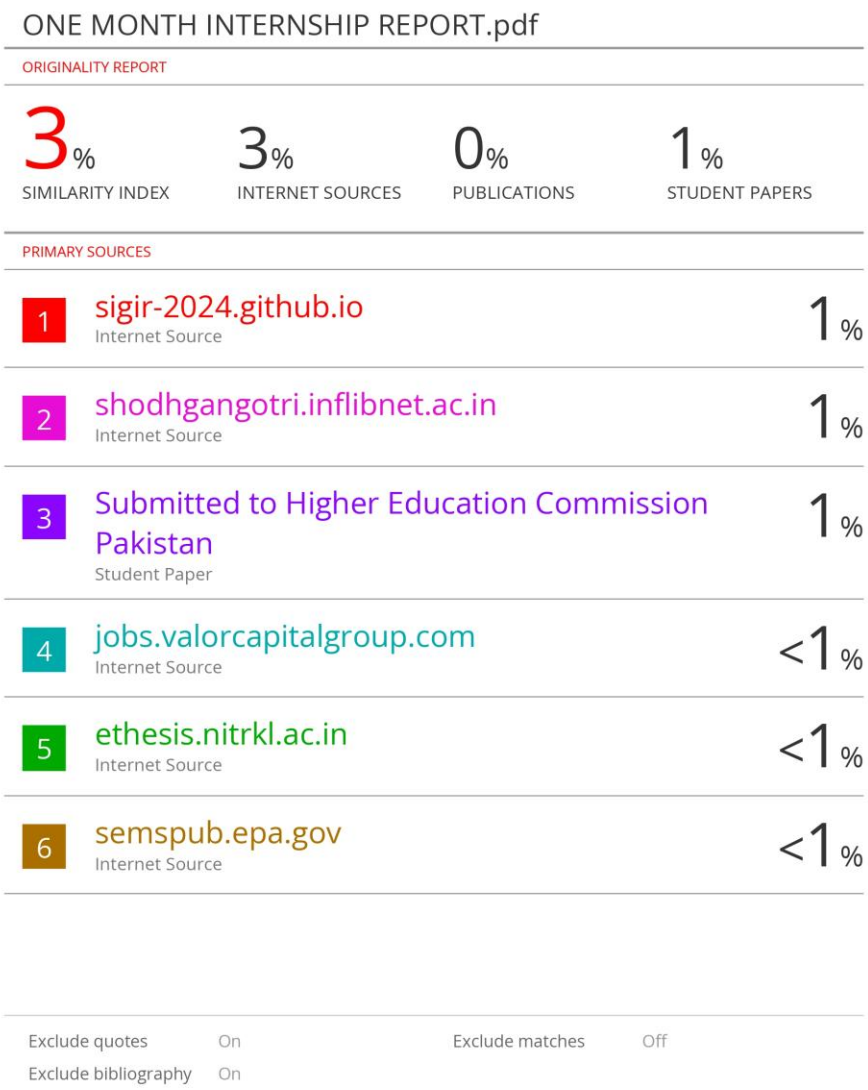
The Riverstream Consultancy Services internship experience was a good experience because it helped in putting the theoretical knowledge to practice. Developing an original idea of the MediBot medical chatbot using the Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs), I had a chance to work with them and be more familiar with how to incorporate powerful language models such as LLAMA 3.3, which Groq is hosting, and how RAG operates.

The presented project enabled me to learn more about such fundamental notions as prompt engineering and vector databases as well as document-based question-answering systems, which are essential concepts modern technological priorities like artificial intelligence fueling the healthcare industry and other crucial spheres.The development pipeline between the PDF ingestion and text chunking and embedding creation stages of the chatbot was well-structured, which largely contributed to my learning experience and mastery of developing scalable, modular, and maintainable AI systems. During this process, focus was on usability, accuracy, and performance to make sure that the prototype could be used with the real world expectations.

Not limiting to technical development, I was able to improve my communication, improved the problem solving capabilities, and learned more about agile development. On the whole, this experience strengthened my technological background and exposed me to more complex issues in the area of AI, either in future academic work or the business market.

Appendix

Plagiarism Report**:**



ONE MONTH INTERNSHIP REPORT.pdf

ORIGINALITY REPORT

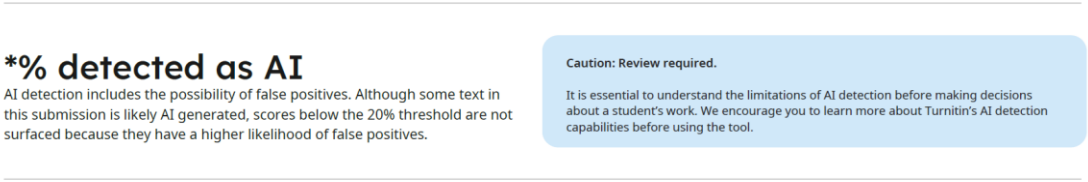
3%	3%	0%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	sigir-2024.github.io Internet Source	1%
2	shodhgangotri.inflibnet.ac.in Internet Source	1%
3	Submitted to Higher Education Commission Pakistan Student Paper	1%
4	jobs.valorcapitalgroup.com Internet Source	<1%
5	ethesis.nitrkl.ac.in Internet Source	<1%
6	semspub.epa.gov Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

AI Report:



References

1. LangChain Documentation. https://python.langchain.com/docs/introduction
2. Groq API Reference. https://console.groq.com/docs/overview
3. Hugging Face Sentence Transformers. https://www.sbert.net
4. PyPDF2 – PDF Parsing Library. https://pypi.org/project/PyPDF2
5. Streamlit Documentation. https://docs.streamlit.io
6. LLaMA 3.3 – Meta AI Language Model. https://ai.meta.com/llama
7. OpenAI. Retrieval-Augmented Generation (RAG) – Concepts & Architecture. https://openai.com/research