

MEDIBOT: A LARGE LANGUAGE MODEL BASED HEALTHCARE CHATBOT

Submitted in partial fulfilment of the
requirements for the award of
Bachelor of Technology Degree in Information Technology

By

SAGAR KUMAR(Reg No. 40120076)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY**

(DEEMED TO BE UNIVERSITY)

Category – 1 University by UGC

**Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by
AICTE**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Sagar Kumar** (Reg. No. 40120076) who carried out the Project work entitled "**MediBot: A Large Language Model (LLM) based Healthcare Chatbot**" under my supervision from January 2024 to April 2024.

Internal Guide

Dr. R Subhashini, M.E., Ph.D.

Head of the Department

Dr. R Subhashini, M.E., Ph.D.

Submitted for Viva Voce Examination held on 18/04/2024

Internal Examiner

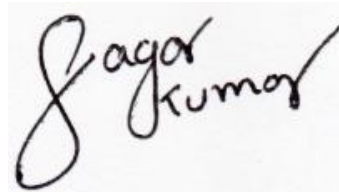
External Examiner

DECLARATION

I, **Sagar Kumar (Reg. No. 40120076)** hereby declare that the Project Report entitled "**MediBot: A Large Language Model (LLM) based Healthcare Chatbot**" is done by me under the guidance of **Dr. R Subhashini, M.E., Ph.D.**, and is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 18/04/2024

PLACE: Chennai

A handwritten signature in black ink, reading 'Sagar Kumar', written on a light-colored background.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T Sasikala, M.E., Ph.D.**, Dean, School of Computing **Dr. R Subhashini, M.E., Ph.D.**, Head of the Department of Information Technology for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R Subhashini, M.E., Ph.D.** for her valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

The internet has revolutionized access to information, but it has also given rise to a daunting problem within the realm of healthcare: the unchecked proliferation of unverified and potentially misleading medical information. This problem is characterized by the overwhelming volume of online health content, much of which lacks credibility, consistency, or relevance. As a result, individuals seeking reliable healthcare knowledge often find themselves navigating a labyrinth of misinformation, leading to confusion, anxiety, and suboptimal health decisions.

To address this pervasive issue, this project introduces a novel and specialized solution—a custom-built medical chatbot designed to serve as a trusted source of organized and up-to-date medical information. In the face of the problem at hand, this innovative chatbot emerges as a beacon of clarity and reliability.

Critical to the success of this solution is the incorporation of clear disclaimers, underscoring the chatbot's role as a supplementary information resource, rather than a replacement for professional medical advice. This approach mitigates the risk of users relying on the chatbot for diagnostic or treatment decisions, thus safeguarding their health and well-being.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	vii
	LIST OF TABLES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
	2.1 Inferences from the Literature Survey	3
	2.2 Existing System and Proposed System	4
	2.2.1 Existing System	4
	2.2.2 Proposed System	4
3	REQUIREMENT ANALYSIS	6
	3.1 Feasibility Study	6
	3.1.1 Data Source	6
	3.1.2 Technical Stack	6
	3.1.3 Natural Language Processing (NLP) Capabilities	7
	3.2 Risk Analysis	7
	3.2.1 Data Quality and Availability	7
	3.2.2 Legal and Ethical Risks	8
	3.2.3 Scalability and Expansion	8
	3.3 Software Requirements Specification Document	9
	3.3.1 Hardware Configuration	9
	3.3.2 Software Configuration	9
	3.3.3 Python Libraries Used	9

4	DESCRIPTION OF PROPOSED SYSTEM	11
4.1	Proposed Methodology	11
4.2	Architecture of Proposed System	13
4.3	Project Management Plan	16
5	METHODOLOGY	18
5.1	Software Implementation	18
5.1.1	Dataset Description	20
5.1.2	Description Of Programming Language and Software	21
5.1.3	Description of Libraries Used	22
5.1.4	Description of ML Algorithm Modules	24
5.2	Algorithm	25
5.2.1	Algorithm for Knowledgebase Creation	25
5.2.2	Algorithm for Chatbot Creation	26
5.3	Environment Setup	28
5.4	Testing Strategy	29
5.3.1	Beta Testing	29
5.3.2	Unit Testing and Integration Testing	29
6	RESULTS AND DISCUSSIONS	30
6.1	Result	31
6.2	Discussions	33
6.3	Performance Evaluation	34
6.4	Extra Observations and Discoveries	36
6.5	Features of the Proposed System	36
7	CONCLUSION	38
7.1	Limitations of the Proposed System	39
7.2	Future Work	39

REFERENCES	41
APPENDICES	43
A Source Code	43
B Screenshots	50
C Research Paper	51

LIST OF FIGURES

FIGURE NUMBER	FIGURE NAME	PAGE NO.
4.1	Sample Query Used for Generating Synthetic Data	12
4.2	System Architecture for Creating a Knowledge Base	13
4.3	Chatbot Implementation Flow Diagram	14
4.4	Top-Level Architecture	15
5.1	Time consumption while creating knowledge	18
5.2	GPU Details while creating the Knowledge Base	19
5.3	GPU Details while hosting the User Interface	19
5.4	URL generation for hosting the User Interface	20
6.1	User Review Representation	34
6.2	Downloaded Chat History File	36
B.1	Final Output with resource utilization (CPU, GPU and Time consumption)	50

LIST OF TABLES

TABLE NUMBER	TABLE DESCRIPTION	PAGE NO.
3.1	Details of all Python libraries used	9
4.1	Week-wise project plan	16
5.1	Types of User Inputs and their Characteristics	20
6.1	Comparison of Responses Generated with Different Tools	31
6.2	Types of Match Conditions and Respective Performance	35

LIST OF ABBREVIATIONS

Sl. No	Abbreviations	Definition
1	AI	Artificial Intelligence
2	CPU	Central Processing unit
3	GPU	Graphics Processing unit
4	RAM	Random Access Memory
5	vRAM	Video Random Access Memory
6	LLM	Large Language Model
7	NLP	Natural Language Processing
8	UI	User Interface
9	TPU	Tensor Processing Unit
10	URL	Uniform Resource Locator
11	CNN	Convolutional Neural Network
12	RNN	Recurrent Neural Network
13	KNN	K-Nearest Neighbors
15	API	Application Programming Interface
16	JSON	JavaScript Object Notation

Chapter 1

INTRODUCTION

In the age of the internet, access to information has never been more ubiquitous. However, this boundless wealth of knowledge is a double-edged sword, especially when it comes to healthcare. While the digital era has democratized information, it has also ushered in a new era of uncertainty, misinformation, and anxiety for individuals seeking reliable medical guidance.

The advent of "Dr. Google" has fundamentally altered how we approach our health concerns. A mere online search for a seemingly innocuous symptom can lead down a rabbit hole of alarmist articles, contradictory advice, and self-diagnostic anxiety. The internet once hailed as a beacon of knowledge, has become a labyrinth where reliable healthcare information often feels elusive.

This perplexing landscape raises a critical question: How can individuals navigate the vast and treacherous terrain of online medical information to make informed decisions about their health? It is within this conundrum that the problem at the heart of this work resides.

This project recognizes the pressing need for a solution—a beacon of clarity, reliability, and organization amidst the chaos of online health information. We propose the creation of a specialized medical chatbot, meticulously designed to serve as a trusted source of medical knowledge, addressing the fundamental issue of unreliable online medical information.

The following sections of this work delve into the intricacies of this innovative solution, detailing the construction of a custom knowledge base, the implementation of a Large Language Model (LLM), and the incorporation of user-friendly features. We explore how this medical chatbot empowers individuals to access accurate, up-to-date medical information, all while emphasizing the chatbot's complementary role to professional healthcare advice.

In a world where navigating the digital sea of healthcare information can be treacherous, this project seeks to provide individuals with a reliable and steadfast companion—a specialized medical chatbot that guides them toward informed decisions about their health, fostering a healthier, more empowered society.

In addition to the challenges posed by the proliferation of online healthcare information, there is a growing concern regarding the impact of misinformation on public health outcomes. With the ease of publishing content on the internet, anyone can present themselves as an authority on medical matters, regardless of their qualifications or the accuracy of their information. This phenomenon has led to the spread of myths, unfounded claims, and pseudoscientific remedies, which can not only confuse individuals but also pose serious risks to their well-being.

Moreover, the advent of social media platforms has amplified the dissemination of health-related information, often in the form of viral posts, memes, and anecdotes. While these platforms have the potential to connect individuals with valuable resources and support networks, they also serve as breeding grounds for sensationalized stories and unverified claims. The viral nature of social media can quickly magnify the impact of misinformation, making it difficult for users to discern fact from fiction.

Furthermore, the personalization algorithms employed by search engines and social media platforms can exacerbate the issue by creating echo chambers where individuals are exposed only to information that aligns with their existing beliefs and biases. This selective exposure can reinforce misinformation and hinder critical thinking, making it even more challenging for users to sift through the vast array of online content to find reliable sources of medical information.

In light of these complexities, there is a clear need for a solution that not only provides accurate and trustworthy medical information but also helps individuals develop the skills necessary to critically evaluate online content. By combining cutting-edge technology with evidence-based practices, our specialized medical chatbot aims to fill this gap, empowering users to navigate the digital landscape of healthcare information with confidence and clarity.

Through educational initiatives, interactive features, and ongoing updates to its knowledge base, the chatbot will not only serve as a valuable resource for immediate health inquiries but also as a tool for promoting health literacy and informed decision-making. By fostering a culture of scepticism and discernment, we believe that our project can contribute to a healthier, more resilient society in the digital age and beyond.

.Chapter 2

LITERATURE SURVEY

With the rise of the digital era, many research institutions and research personnel have proposed different ideas for making a chatbot that can be used for different purposes. Our proposed work takes inspiration from many of such proposed ideas. The following sections include some of the recently proposed ideas that are similar to our work which we draw inspiration from.

2.1. INFERENCES FROM LITERATURE SURVEY

Tatsuya Ito et al. (2019) created a similar chatbot to assess the mental health of individuals. They used Structured Association Technique (SAT) counselling as their base of assessment. A set of predefined questions and quizzes will be shown to the user in an interactive web interface. The response from the user will be recorded to assess the mental state of the individual. Post assessment the user would be given a small course to help them improve their state of mind. They experimented with a group of 27 college students divided into two groups to check the efficiency of the chatbot and their subject's mental state. The experiment was carried out with the approval of the ethics review committee in the Faculty of Library Information and Media Science, University of Tsukuba (Notification No. 29-137).

M. M. Rahman et al. (2019) created a chatbot that could easily classify different diseases based on some set of symptoms entered as input. This chatbot was designed to work with the Bangla language alone. The user would enter some personal details which will be stored by the bot that would later be used to make the responses interactive. Their work is largely dependent on AIML (Artificial Intelligence Mark-up Language) for creating the responses. The chatbot is also dependent on the Named Entity Recognition (NER) algorithm to identify the names of users in the input query. Their work claims to have an accuracy of 98.9% for the classification of diseases but the number of diseases being covered by them is not sufficient.

Lekha Athota et al. (2020) created a chatbot that could work as an intermediary between a patient and the doctor. Initially, the chatbot uses a SQL database to

generate responses but if the requested data is not found then the question is sent to a doctor who would have already registered on the portal. They have used Natural Language Processing techniques like tokenization, and N-grams to extract keywords from the user input. All these keywords have predefined weight which helps in identifying the most suitable response.

Mendapara H et. al. (2021) built an AI Based Healthcare Chatbot System by Using Natural Language Processing. They created a MySQL-based database that had different symptoms and a counter medication for those symptoms. Though the chatbot performed well enough it was solely based on potential matching of results. For a given set of symptoms, the model would give a medication.

Jahnvi Gupta et al. (2021) created a chatbot named Florence similar to Rahman et al. (2019). They added a new feature with which the user can find the nutritional details about any particular food item. Another feature was to check if a particular doctor was available on a particular day or not which was supplied by a dynamic flat log file.

2.2. EXISTING SYSTEM AND PROPOSED SYSTEM

2.2.1. Existing System

Based on an extensive exploration of the contemporary sources it was found that currently, there are not many options that can provide extensive results for a particular type of disease. A majority of the options that do exist are focused on a foreign audience. The available options include normal web searching and Generative AI Chatbots like ChatGPT.

Though the performance of such tools is good it is tough to rely completely on their authenticity. A mere online search for a seemingly innocuous symptom can lead down a rabbit hole of alarmist articles, contradictory advice, and self-diagnostic anxiety.

2.2.2. Proposed System

The system proposed by us aims to revolutionize the accessibility and reliability of healthcare information through the development of a sophisticated chatbot. Central to this endeavour is the creation of a chatbot that can be trained on a custom corpus comprising diverse healthcare and medical-related data. Notably,

this corpus includes not only general human healthcare data but also encompasses comprehensive drug data, ensuring a holistic understanding of medical inquiries. The distinctive advantage of our chatbot lies in its robust knowledge base, sourced from a curated collection of legitimate and authentic data provided by medical practitioners and reputable medical institutes. By leveraging this rich repository of information, our chatbot promises to offer users a trusted source of medical insights, enabling informed decision-making and fostering greater confidence in navigating healthcare queries.

Chapter 3

REQUIREMENT ANALYSIS

3.1. FEASIBILITY STUDY

A large number of efforts need to be put into the technical aspect of this proposed solution. Though there already are many similar works each one of them is unique in its own way. Any previous work similar to our proposed idea had its own challenges. Same with any other solution our proposed work also has some factors that largely influence its viability. Some of such factors are mentioned in the following sections.

3.1.1. Data Sources

The feasibility of our proposed solution hinges significantly on the availability and quality of medical and healthcare data. A critical step in this process involves conducting a comprehensive assessment of potential data sources to populate the chatbot's knowledge base effectively. It is imperative to ascertain the accessibility of authoritative medical databases and sources, as their inclusion can enhance the reliability and accuracy of the chatbot's responses. However, considerations regarding the legalities and suitability of open-domain sources and corpora must also be carefully evaluated to mitigate any potential risks associated with their use. Prioritizing the acquisition of data from reputable and verified sources ensures that the chatbot's knowledge base is built on a foundation of credible information, thereby bolstering its efficacy and trustworthiness in providing accurate healthcare insights to users.

3.1.2. Technical Stack

The technical feasibility of the project is intricately tied to the choice of technology stack, which dictates the tools and frameworks used for chatbot development. In our proposed solution, the selection of the technology stack is pivotal, and we have opted for Python programming language coupled with the Langchain framework. This decision is grounded in the extensive support offered by Python and Langchain for Natural Language Processing (NLP), a critical component of our project. Leveraging these technologies empowers us to effectively parse, understand, and generate medical content, thereby enhancing the chatbot's ability

to provide accurate and contextually relevant responses to user queries. By capitalizing on the robust capabilities of Python and the specialized features of the Langchain framework, we aim to streamline the development process and maximize the efficiency and effectiveness of our chatbot solution.

3.1.3. Natural Language Processing (NLP) Capabilities:

NLP forms the core of a conversational chatbot, involving the understanding and generation of human language, which is crucial for medical and healthcare applications. One of the major considerations is the evaluation of the availability and suitability of the NLP libraries and models for medical domain-specific language processing.

3.2. RISK ANALYSIS

This section talks about different potential risks and uncertainties associated with the development and deployment of our custom knowledge-based chatbot. Understanding these risks is vital for proactive management and mitigation strategies.

3.2.1. Data Quality and Availability

In the realm of risk analysis, the quality and availability of data stand out as critical factors that demand careful consideration. One significant risk pertains to the potential shortcomings of medical and healthcare data sourced from open-domain repositories, which may suffer from incompleteness, obsolescence, or inaccuracies. Such deficiencies pose a tangible threat to the reliability and credibility of the chatbot's responses, potentially undermining user trust and satisfaction.

To mitigate this risk effectively, our strategy revolves around prioritizing the use of authoritative medical databases and peer-reviewed publications. By leveraging these reputable sources, we aim to fortify the integrity and accuracy of the data underpinning the chatbot's knowledge base. Additionally, the rigorous validation processes inherent in peer-reviewed publications offer an additional layer of assurance regarding data quality. Through this proactive approach, we endeavour to minimize the likelihood of encountering data-related issues and uphold the chatbot's reputation as a dependable source of medical information.

3.2.2. Legal and Ethical Risks

Navigating the legal and ethical landscape surrounding medical data entails addressing potential risks that could undermine patient trust and result in copyright violations. Concerns may arise regarding the erosion of trust in healthcare professionals when patients opt for chatbots or unverified medical advice. Furthermore, the use of copyrighted materials, such as books or corpora, without proper authorization poses a risk of legal repercussions.

To mitigate these challenges, our approach involves vigilant monitoring of potential breaches and violations, with a concerted effort to prioritize sources that comply with legal and ethical standards. We aim to minimize reliance on conflicting or questionable sources, thereby reducing the likelihood of encountering legal issues or ethical dilemmas. Additionally, we plan to implement a robust disclaimer emphasizing that our work is intended for educational purposes only and should not be construed as a substitute for professional medical advice. This disclaimer serves to manage expectations, clarify the scope of our project, and underscore our commitment to ethical conduct and transparency. Through these measures, we strive to uphold the highest standards of integrity and responsibility in our handling of medical data while safeguarding patient trust and adherence to legal requirements.

3.2.3. Scalability and Expansion

In considering scalability and expansion, it's crucial to address the dynamic nature of the knowledge base and its ongoing refinement to encompass a wide array of topics. Failure to update the knowledge base can lead to users accessing outdated information, potentially posing risks given the critical nature of healthcare-related inquiries.

To tackle this challenge, our strategy entails implementing a versioning system designed to facilitate seamless updates and expansions to the knowledge base. This system ensures that whenever new information is added, the chatbot is automatically updated while preserving the integrity of previous knowledge as a separate version. By employing this approach, we not only maintain the relevance and accuracy of the chatbot's responses but also provide users with access to the most up-to-date medical information available. This proactive measure not only

enhances the chatbot's effectiveness in delivering reliable healthcare insights but also underscores our commitment to continuous improvement and user satisfaction.

3.3. SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

3.3.1. Hardware Configuration

Central Processing Unit (CPU) : Intel Xeon with 2 vCPUs

Random Access Memory (RAM) : 13 GB

Graphical Processing Unit (GPU) : NVIDIA T4 Tensor Core

Virtual RAM (vRAM) : 15 GB

3.3.2. Software Configuration

Operating System (OS) : Linux 5.15.109+

OS Architecture : x86_64

Programming Language : Python 3.9 or above

Other Tools : Google Colab, Localtunnel 2.0.2, Poppler

3.3.3. Python Libraries Used

Table 3.1 Details of all the Python libraries used

Sl. no.	Name	Version	Purpose
1	InstructorEmbedding	1.0.1	Used to generate the vector embedding
2	langchain	0.0.285	Used to train a LLM based model
3	sentence_transformers	2.2.2	Used to process natural sentences
4	faiss-cpu	1.7.4	Used for similarity searching of vectors

5	streamlit	1.26.0	Used for creating a web app
6	unstructured	0.10.12	Used for processing raw documents
7	pickle	4.0	Used for deserialization and serialization
8	psutil	5.9.8	Used for process monitoring
9	gputil	1.4.0	Used for getting GPU details
10	streamlit_bokeh_events	0.1.2	Used to enhance User Interactions
11	extra-streamlit-components	0.1.71	Used to enhance UI

Chapter 4

DESCRIPTION OF PROPOSED SYSTEM

The idea proposed by us was to make an Artificial Intelligence (AI) Based Chatbot that can be used in the healthcare field. The idea revolves around reacting to a knowledge base that is exclusively dedicated to human healthcare. The idea we adopt to make such an extensive knowledge base requires a thorough evaluation of different data sources and corpuses available in the open domain. The legitimacy of the information given by our chatbot is also something that needs to be considered at each step. This system is not an alternative to professional medical advice and is only a demonstration for educational purposes. Since the work is in its initial stages, we request all users to be cautious of the pieces of advice given by this system. With further refinement of the knowledge base, the reliability of this system can also be improved.

4.1.PROPOSED METHODOLOGY

The data collection process for our proposed system involves gathering information from three distinct sources to ensure comprehensive coverage and accuracy.

- 1) Synthetic Data** – Data created from AI generative tools like ChatGPT, Bing Chat, Jasper AI and Claude. For this type of data, a basic query (like Fig. 4.1) is given to the AI tool. Based on the query the tool generates a response that is manually recorded.
- 2) Manually Collected Data** – These data include different legitimate corpora, articles, journals and books that are available in the open domain. All these data are gathered and processed to remove any irrelevant or unnecessary information.
- 3) Primary Data** – These data are directly obtained from individuals belonging to different age groups and professions using surveys via rolling out Google forms.

Once the data from these sources is collected and stored in a single folder, it undergoes further processing. Each text document is extracted and converted into a vector format, with each vector processed using a pre-trained

embeddingmodel to generate vector embeddings. These embeddings are then indexed and

stored in a vector database, forming the foundation of our knowledge base. To manage vector data efficiently, we utilize the vector-store service from Langchain,

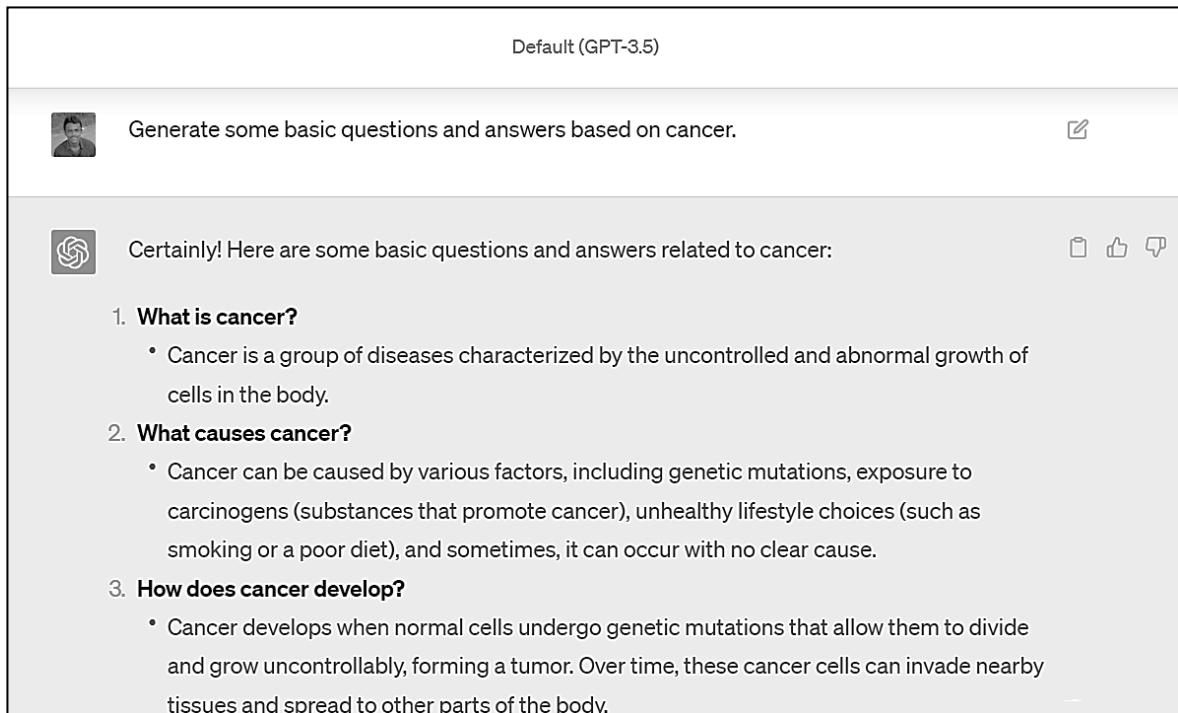


Fig. 4.1 Sample Query used for Generating Synthetic Data

which stores vector data as local files, thus optimizing storage resources.

Subsequently, the implemented chatbot utilizes this knowledge base to respond to user queries. When a user initiates a conversation, their input is converted into a vector using the previously generated embeddings. This vector is then compared against the knowledge base to extract the most relevant data, which is subsequently unpacked and presented to the user. For efficient searching and matching, we employ the FAISS (Facebook AI Similarity Search) algorithm, leveraging GPU acceleration to expedite the process of finding matching vectors based on query relevance.

Overall, this data-driven approach ensures that our chatbot is equipped with a robust and diverse knowledge base, enabling it to provide accurate and relevant responses to user queries in the healthcare domain.

4.2 ARCHITECTURE OF PROPOSED SYSTEM

The architecture of the proposed system is depicted in Fig. 4.2, which illustrates a block diagram representing two main components: the creation of a knowledge base and the implementation of the knowledge base as a chatbot for user interactions.

The process begins with the collection and aggregation of data from various sources. This data undergoes preprocessing to ensure uniformity and relevance.

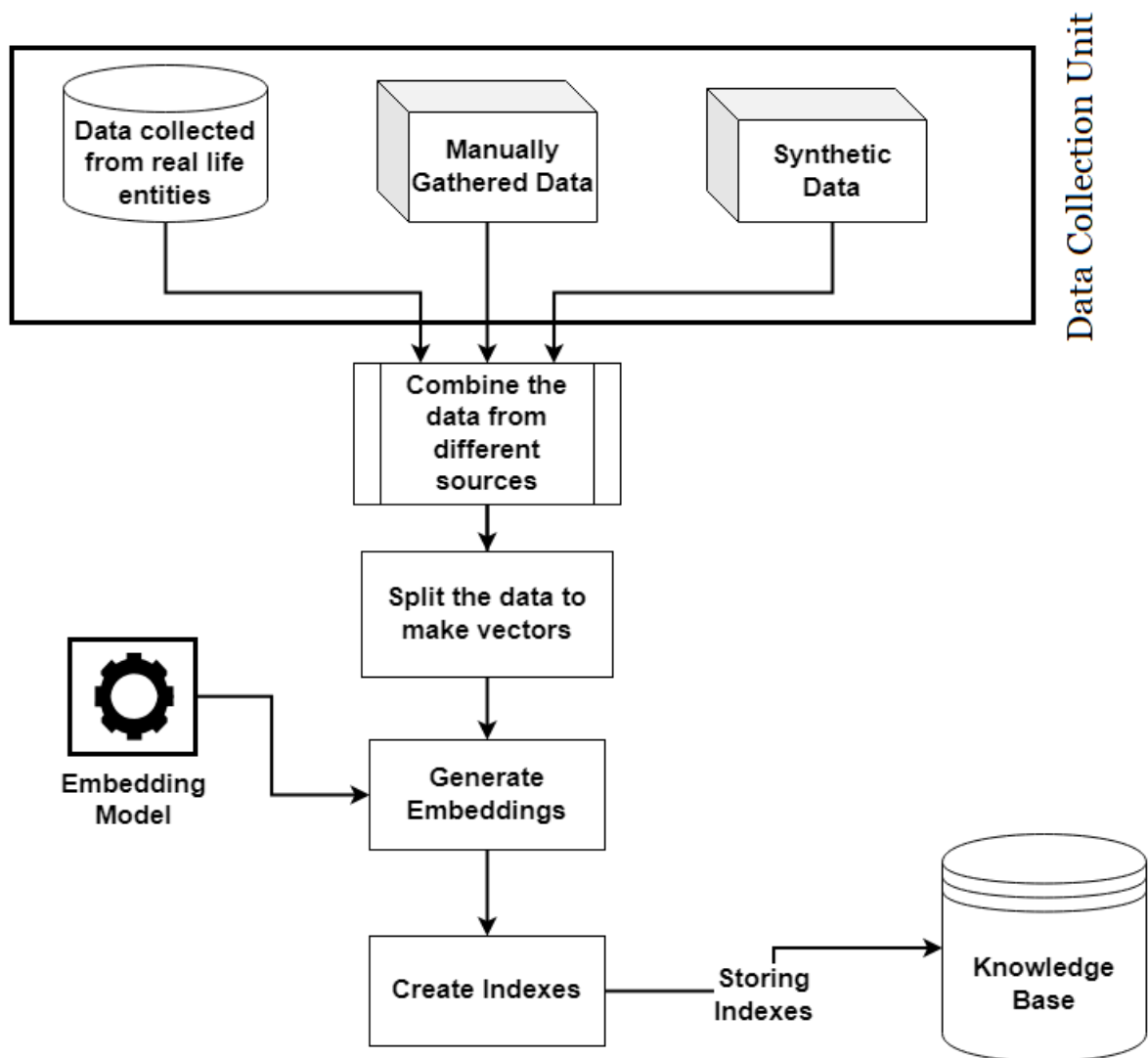


Fig. 4.2 System Architecture for creating a Knowledge Base

Subsequently, it is processed to generate embeddings using a specialized Natural Language Processing (NLP) model known as the InstrucTOR embedding model,

developed by the NLP group of the University of Hong Kong. These embeddings serve as numerical representations of the textual data and are crucial for subsequent operations.

The generated embeddings are then stored to create a comprehensive knowledge base. This knowledge base acts as a repository of information, containing a diverse range of healthcare-related data sourced from multiple channels. It serves as the backbone of the chatbot system, enabling it to access and retrieve relevant information in response to user queries.

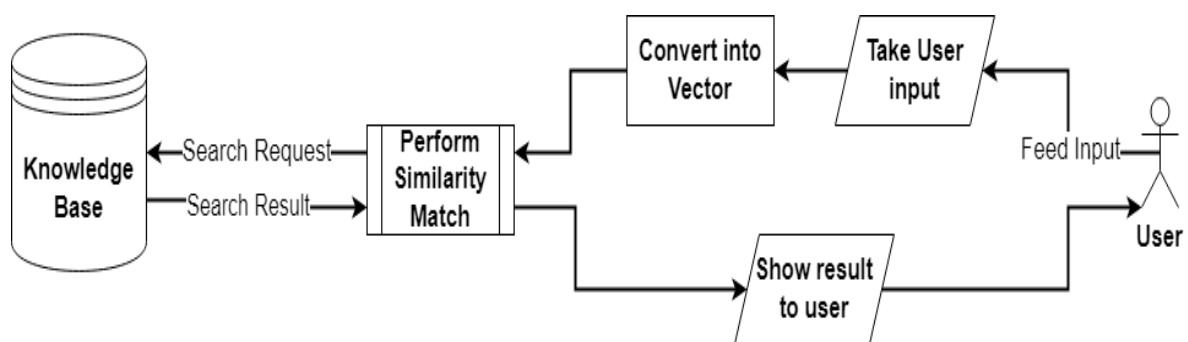


Fig. 4.3 Chatbot Implementation Flow Diagram

In Fig. 4.3, the flow diagram illustrates the interaction between the user and the chatbot system. The process begins with the user inputting a statement or query into the chatbot interface. This input is then processed by the system, which involves converting the text input into a format suitable for comparison and analysis. Subsequently, the processed user input is used to search the knowledge base for the most relevant data or information. This search operation involves comparing the input against the stored data to identify matches or similarities. Once the matching data is found, it is retrieved from the knowledge base

Next, the retrieved data is converted back into a textual response format suitable for presentation to the user. This conversion process involves transforming the structured data into natural language text that is understandable and meaningful to

the user. The text response encapsulates the relevant information extracted from the knowledge base.

Finally, the formatted text response is returned to the user, completing the interaction loop. The user receives the response from the chatbot, which addresses their initial query or statement. This iterative process of user input, data processing, knowledge retrieval, response generation, and user feedback forms the basis of the chatbot's functionality, enabling seamless and effective communication between the user and the system.

Overall, the architecture of the proposed system is designed to seamlessly integrate data collection, processing, storage, and retrieval, culminating in the development of an intelligent chatbot capable of delivering accurate and personalized healthcare information to users.

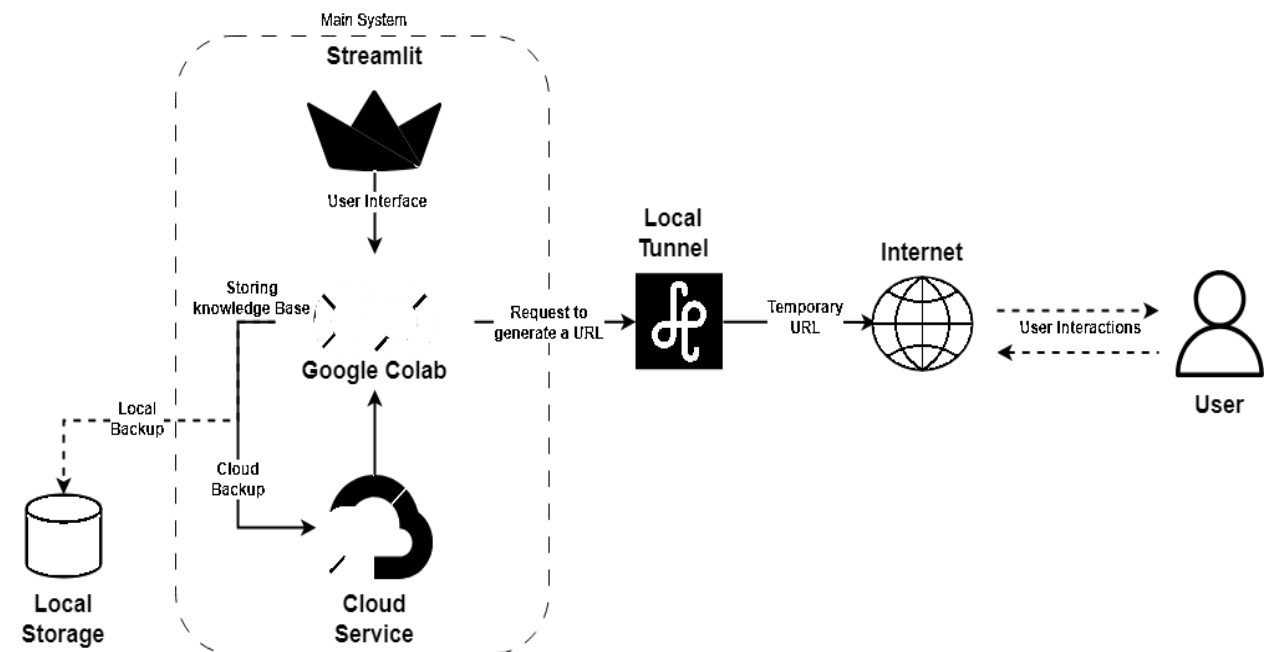


Fig. 4.4 Top-Level Architecture Diagram

Table 4.1 Week-wise project plan

4.3. PROJECT MANAGEMENT PLAN

Month	Weeks	TasksCompleted
November	Week 1 (15-19)	Gathering Resources and Usage Permissions
	Week 2 (20-26)	
	Week 3 (27-03)	
December	Week 3 (04 – 10)	Data Collection for Drugs
	Week 4 (11 – 17)	
	Week 5 (18-24)	
	Week 6 (25-31)	Creation of a new Knowledge Base
January	Week 7 (01-07)	Integration of new Knowledge base with the previous Knowledge Base
	Week 8 (08-14)	
	Week 9 (15-21)	Implementation of User Interface
January	Week 10 (22-28)	Implementation of User Interface
	Week 11 (29-31)	

February	Week 11 (01-04)	Unit Testing and Integration Testing
	Week 12 (05-11)	
	Week 13 (12-18)	Paper Presentation at Conference
	Week 14 (19-25)	Software Enhancements and Performance Evaluation
	Week 15 (25-29)	Internal Beta Testing
March	Week 15 (01-03)	Internal Beta Testing
	Week 16 (04-10)	Documentation
	Week 17 (11-17)	
Project Completed		

Chapter 5

METHODOLOGY

The whole system is implemented in two stages. The first stage is dedicated to creating a knowledge base with all the collected data. The second stage is about the implementation of the Knowledge Base as a chatbot. Both of the operations are done separately through respective sets of codes and environments. Both of the operations need a GPU-based environment.

5.1.SOFTWARE IMPLEMENTATION

The Knowledge Basecreation process is carried out in Google Collab. This

.gitattributes: 100%	1.48k/1.48k [00:00<00:00. 51.5kB/s]
1_Pooling/config.json: 100%	270/270 [00:00<00:00. 4.63kB/s]
2_Dense/config.json: 100%	116/116 [00:00<00:00. 3.37kB/s]
pytorch_model.bin: 100%	3.15M/3.15M [00:00<00:00. 26.0MB/s]
README.md: 100%	66.3k/66.3k [00:00<00:00. 1.76MB/s]
config.json: 100%	1.52k/1.52k [00:00<00:00. 46.4kB/s]
config_sentence_transformers.json: 100%	122/122 [00:00<00:00. 1.54kB/s]
pytorch_model.bin: 100%	4.96G/4.96G [00:44<00:00. 95.1MB/s]
sentence_bert_config.json: 100%	53.0/53.0 [00:00<00:00. 1.79kB/s]
special_tokens_map.json: 100%	2.20k/2.20k [00:00<00:00. 65.6kB/s]
spiece.model: 100%	792k/792k [00:00<00:00. 28.4MB/s]
tokenizer.json: 100%	2.42M/2.42M [00:00<00:00. 6.09MB/s]
tokenizer_config.json: 100%	2.40k/2.40k [00:00<00:00. 171kB/s]
modules.json: 100%	461/461 [00:00<00:00. 29.1kB/s]
load INSTRUCTOR_Transformer	
max_seq_length 512	
Starting the knowledge base creation...	
Present time: 2023-12-28 08:34:08.426746	
Operation completed at: 2023-12-29 01:59:04.785422	
Total Time Consumed: 17:24:56.358676 Hrs	

Fig. 5.1Time consumption while Creating the Knowledgebase

processtakes around 17:25 hours with aGPU Environment. This process consumes around 93% or 14209 out of 15360 Mib of the GPU. During the knowledge base creation,the system temperature reaches to 78°C as the load increases.

This approach to software implementation facilitates the efficient processing of large-scale data and enables the creation of a comprehensive knowledge base essential for the subsequent development and deployment of the chatbot system.

In the second stage, a similar collab environment is required. The User Interface is implemented using streamlit which requires a Python source file to operate and

NVIDIA-SMI 535.104.05			Driver Version: 535.104.05			CUDA Version: 12.2		
GPU	Name		Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp	Perf	Pwr:Usage/Cap		Memory-Usage	GPU-Util	Compute M.	MIG M.
=====								
0	Tesla T4		Off	00000000:00:04.0	Off		0	
N/A	78C	P0	58W / 70W	14209MiB / 15360MiB		93%	Default	N/A

Fig. 5.2 GPU Details while Creating the Knowledgebase

process all operations. The main driver code for the UI is stored as a .py file in the local runtime environment.

The UI consumes approximately 31% of the GPU when hosted through a third-

NVIDIA-SMI 535.104.05			Driver Version: 535.104.05			CUDA Version: 12.2		
GPU	Name		Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp	Perf	Pwr:Usage/Cap		Memory-Usage	GPU-Util	Compute M.	MIG M.
=====								
0	Tesla T4		Off	00000000:00:04.0	Off		0	
N/A	65C	P0	29W / 70W	4887MiB / 15360MiB		0%	Default	N/A

Fig. 5.3 GPU Details while Hosting the User Interface

party service called localtunnel. This service allows temporary access to the UI via a dynamically generated URL. As depicted in Fig 5.4, users can access the UI through this temporary URL, enabling them to interact with the chatbot system seamlessly.

It's worth noting that when the runtime environment stops, the hosted UI also stops, resulting in the temporary URL becoming inaccessible. Therefore, users need to ensure the runtime environment remains active to maintain access to the UI and continue interacting with the chatbot system.

Collecting usage statistics. To deactivate, set `browser.gatherUsageStats` to `False`.

You can now view your Streamlit app in your browser.

Network URL: <http://172.28.0.12:8501>

External URL: <http://34.83.10.45:8501>

npx: installed 22 in 3.249s

your url is: <https://chatty-dancers-dig.loca.lt>

Fig. 5.4 URL Generation for Hosting the User Interface

Any input given by the user can be categorized into 3 types i.e. Complete Input, Partial Input and Incomplete. All of the 3 types of inputs are explained in Table 5.1.

Table 5.1 Types of User Inputs and their characteristics

Input Type	Characteristics
Complete Input	Only the Complete and correct keyword* is present
Partial Input	Keywords are present with other unimportant words
Incomplete Input	No keyword is present or only contains special symbols

**The words that will majorly be used to search the knowledgebase. In this case, it will be the name of a medical condition or drug.*

5.1.1. Dataset Description

The dataset used in the project consists of various types of data collected from individuals and generated synthetically, all stored as PDF files. These PDF files contain information relevant to the healthcare domain, covering a wide range of topics such as medical conditions, treatments, and diagnostic procedures.

During the preprocessing stage, the PDF files are processed sequentially to extract the text content, which is then used to create a collection of vectors representing the textual data. These vectors serve as numerical representations of the textual information, facilitating efficient storage, retrieval, and analysis.

The vector collection is subsequently embedded and stored in Google Drive, providing a centralized and easily accessible repository for the dataset. This storage solution enables seamless integration with the chatbot system, allowing for efficient retrieval of information during user interactions.

It's important to note that while collecting data from individuals, strict measures were taken to ensure the anonymity of responders' identities. This safeguard protects the privacy and confidentiality of individuals contributing to the dataset, aligning with ethical considerations and data protection regulations.

Additionally, the dataset includes manually collected data from reputable sources such as books like "The Merck Manual for Diagnosis" and "Human Diseases," among others. These books provide authoritative and comprehensive information on various medical conditions, enriching the dataset with diverse and reliable content.

Overall, the dataset description highlights the diversity and richness of the data sources utilized in the project, underscoring the importance of robust data collection and preprocessing techniques in developing a high-quality healthcare chatbot system.

5.1.2. Description Of Programming Language and Software

1) Google Colab (Tool)

Google Colab, short for Google Colaboratory, is a free cloud-based platform by Google designed for developing and running machine learning and data analysis projects. It offers access to powerful GPUs and TPUs, a user-friendly interface akin to Jupyter Notebooks, and easy integration with Google Drive for collaborative work. While it has some limitations, it's a valuable tool for data scientists and researchers, especially for projects requiring substantial computational resources.

2) Poppler (Service)

Poppler is an open-source software library used for rendering and processing PDF files. It provides developers with tools to extract text and images from PDFs and to render PDFs for viewing and printing. Poppler is widely used in Linux and open-source software and is essential for working with PDF documents programmatically.

3) Python (Programming Language):

Python is a versatile, high-level programming language known for its simplicity and broad usage. Developed by Guido van Rossum in 1991, Python's readability and extensive library support have made it a favourite among developers for tasks ranging from web development to artificial intelligence. With its strong community and ease of use, Python remains a prominent player in technology and innovation.

4) LocalTunnel (Tool)

LocalTunnel is a service that allows individuals to expose their locally hosted web applications to the internet with ease. It operates by creating a temporary public URL that can be shared with others, providing a way to showcase projects and websites in development on the World Wide Web. This service is particularly valuable for developers and testers who need to share their work with clients, colleagues, or stakeholders without the need for a dedicated web hosting setup. LocalTunnel simplifies the process of making local web applications accessible to a wider audience, facilitating collaboration and feedback collection.

5.1.3. Description of Libraries Used

1) Langchain

Langchain is a Python NLP library that facilitates common language processing tasks and machine learning model building. It contains functions for text cleaning, tokenization, tagging and vectorization. Popular pre-trained models like BERT can be easily loaded and fine-tuned. APIs simplify creating and evaluating deep learning models using CNNs, RNNs and Transformers for tasks such as classification, question answering and summarization. Langchain aims to streamline NLP workflows in Python from text preprocessing to model development and tuning.

2) Streamlit

Streamlit is a Python library that simplifies the development of interactive web applications with minimal code. It's commonly used for creating data dashboards, visualization tools, and simple web applications by data scientists and developers.

3) Unstructured

The unstructured package provides tools for preprocessing and analyzing messy, real-world text data in Python. It contains functions for normalizing text, intelligent tokenization, part-of-speech tagging, and topic modelling on noisy texts. The package also supports text classification with limited or noisy labels. It aims to make NLP more accessible for analysts working with dirty, unstructured data not covered by other libraries. The tools help clean, understand, explore and visualize unstructured text collections.

4) Pickle

The pickle module in Python provides functionalities for the serialization and deserialization of Python objects into a binary format. It supports various data types, including custom classes and built-in structures. Serialized data is in a non-human-readable format specific to Python versions. Caution is advised when deserializing data from untrusted sources to avoid potential security risks. Key functions include `dump()` and `load()` for file-based operations, and `dumps()` and `loads()` for in-memory usage.

5) Psutil

The psutil is a Python cross-platform library providing system utilities for retrieving information on running processes and system utilization, including CPU, memory, disk, and network usage. It simplifies system monitoring and management tasks by offering an easy-to-use interface for querying various system metrics. psutil supports multiple operating systems, including Linux, Windows, and macOS, making it suitable for diverse development environments. Key functionalities include querying process information, monitoring system resource usage, and managing system processes.

6) Gputil

The GPUtil is a Python module designed for querying GPU (Graphics Processing Unit) information, including utilization and memory statistics. It provides a straightforward interface for retrieving GPU-related data, facilitating tasks such as monitoring GPU usage and managing resources in GPU-accelerated applications. GPUtil supports various GPU vendors, including NVIDIA and AMD, making it versatile for different hardware configurations. Developers can utilize GPUtil to optimize GPU utilization, monitor performance, and ensure efficient resource allocation in applications leveraging GPU computing.

7) Streamlit_bokeh_events

The streamlit_bokeh_events is a Python package designed to enhance Streamlit applications by enabling interaction with Bokeh plots. Bokeh is a powerful library for creating interactive visualizations, and streamlit_bokeh_events seamlessly integrates Bokeh plots into Streamlit apps, allowing users to interact with plots and receive event notifications. This package bridges the gap between Streamlit's simplicity for building web applications and Bokeh's capabilities for creating dynamic and interactive plots.

8) Extra-streamlit-components

The extra-streamlit-components is a Python package designed to extend the functionality of Streamlit by providing additional custom components and widgets. Streamlit is a popular framework for building web applications with Python, focusing on simplicity and ease of use. With extra-streamlit-components, developers can enhance their Streamlit apps by incorporating specialized components and widgets that are not available in the standard Streamlit library. These components may include advanced visualization tools, interactive elements, or integrations with external services.

5.1.4. Description of ML Algorithm Modules

1) Faiss-cpu

FAISS is an open-source C++ library for efficient similarity search in large vector collections. It contains algorithms that optimize operations like K-NN search and clustering using techniques like product quantization and inverted files. FAISS supports various vector distances and dimensions. It provides a simple API for

indexing and querying vectors. FAISS is fast, scalable, and supports CPU/GPU. It is widely used for image search, recommendations, and NLP tasks involving large embedding spaces.

2) Instructorembdding

`instructorEmbedding` is a Python library that produces low-dimensional embeddings of instructors. It takes in structured instructor data like courses taught, evaluations, and publications. A neural network model like Word2Vec is then trained to embed each instructor as a dense numeric vector. Similar instructors will have nearby vectors in the embedded space. The embeddings can be used for applications like recommending instructors or predicting teaching fit. The library makes it easy to train, access, and utilize instructor embedding models in Python code.

3) Sentence_transformers

Sentence Transformers is a Python package designed for transforming sentences and text data into numerical representations (embeddings). This package often incorporates pre-trained models for generating state-of-the-art sentence embeddings, which are useful for various NLP tasks like text similarity, clustering, and classification.

5.2.ALGORITHM

5.2.1. Algorithm forKnowledgebase Creation

1) Step 1: Initialization

The process begins by initializing the system, ensuring that all required components are ready for operation.

2) Step 2: Library Import

Following initialization, the algorithm imports the necessary libraries for Natural Language Processing (NLP). These libraries provide essential tools and functionalities for text processing and analysis.

3) Step 3: Model Loading

Next, the trained word embedding model is loaded into memory. This model is crucial for understanding and representing textual data in a numerical format, facilitating various NLP tasks.

4) Step 4: Document Collection

The algorithm proceeds to collect documents relevant to the knowledgebase. These

Documents are typically stored in a designated folder, ensuring organized access to the textual data.

5) Step 5: Text Embedding

Once the documents are collected, the algorithm utilizes the loaded word embedding model to convert the text into numerical vectors. This transformation is essential for enabling computational operations on textual data, such as similarity calculations and clustering.

6) Step 6: Knowledgebase Storage

Following text embedding, the resulting knowledgebase is ready for storage. The algorithm saves the processed data, typically utilizing a cloud-based storage solution for accessibility and scalability.

5.2.2. Algorithm for Chatbot Creation

1) Step 1: Initialization

The chatbot creation process begins with initializing the system, ensuring that all necessary components are ready for operation.

2) Step 2: Cloud Connection

Next, a connection is established between the local environment and the cloud, facilitating access to remote resources and data storage.

3) Step 3: Module Import

Following the cloud connection, the algorithm imports the required connector modules, enabling communication and interaction between the local environment and cloud services.

4) Step 4: Knowledgebase Instance

Subsequently, an instance of the knowledgebase is created, allowing the chatbot to search and retrieve relevant information from the stored data.

5) Step 5: User Input

The user is required to enter a text response which is a question in the input box. The algorithm prompts the user for input, initiating an interaction between the user and the chatbot.

6) Step 6: Text Preprocessing

Upon receiving user input, the algorithm performs text preprocessing tasks such as lemmatization and segmentation, ensuring standardized and structured input data.

7) Step 7: Text Embedding

The preprocessed user input is then converted into a numerical vector representation, enabling computational operations on the text data.

8) Step 8: Searching Model Invocation

Next, the algorithm calls the searching model instance, initiating the search process within the knowledgebase to find a relevant response to the user query.

9) Step 9: Resource Monitoring

Concurrently, the algorithm monitors CPU and GPU usage readings, as well as the current time, in parallel with the search operation.

10) Step 10: Knowledgebase Search

The knowledgebase is searched for a relevant response based on the user input, leveraging the search capabilities of the chatbot's knowledgebase instance.

11) Step 11: Response Conversion

Upon finding a relevant response, the numerical vector representation of the response is converted back into normal text format for presentation to the user.

12) Step 12: Response Display

The chatbot displays the retrieved response on the screen, providing the user with the information or assistance they requested.

13) Step 13: Resource Utilization Calculation

Following the response presentation, the algorithm continues to monitor CPU and GPU usage readings, as well as the current time, in parallel.

14) Step 14: Resource Utilization Reporting

The total amount of resources utilized during the chatbot interaction is calculated by subtracting the initial resource readings from the final readings. The algorithm then prints the resource utilization metrics for analysis.

5.3. ENVIRONMENT SETUP

Setting up the environment for the entire system involves configuring a Google Colab notebook to ensure compatibility and efficient execution. Before running the program, it's essential to install all necessary dependencies and select the appropriate runtime environment, with the T4 GPU being the preferred option for optimal performance.

Once the GPU environment is prepared, the next step is to install dependencies. One crucial requirement is downgrading the Pillow package to a version prior to version 7, ensuring compatibility with certain functionalities of the system. Additionally, installing LocalTunnel using the Node.js package manager is necessary for hosting the user interface and enabling temporary access via dynamically generated URLs.

With these initial setup steps completed, the remaining dependencies mentioned in the previous section (section 4.3) can be installed effortlessly. These dependencies encompass a range of libraries and tools essential for data processing, natural language processing, and machine learning tasks, ensuring the smooth operation of the entire system.

By following these steps, users can effectively set up the environment for the system on a Google Colab notebook, enabling seamless execution and interaction

with the chatbot system. This streamlined setup process simplifies deployment and enhances accessibility, empowering users to leverage the system's capabilities for healthcare information retrieval and user interaction.

5.4. TESTING STRATEGY

For this study, a majority of testing done included Beta testing and Unit & Integration Testing.

5.3.1. Beta Testing

Once a very basic User Interface was ready, multiple questions were collected and fed one by one. For each question, the respective response was recorded. A Google form was used where for each question the respective answer was written. People belonging to different groups were asked to rank the answers based on the question. These responses were collated and presented as a chart as mentioned in the next section (section 6).

5.3.2. Unit Testing and Integration Testing

Inorder to reduce the workload, the unit testing was done manually. Different test cases were written and then manually tested on the final system. Any abnormalities were noted down and worked upon later on. During this testing, the work was compared with the predicted work. In case of deviations, it was also noted down.

Chapter 6

RESULTS AND DISCUSSIONS

6.1. RESULTS

Since the chatbot cannot be evaluated solely based on mathematical metrics we used two different ways to evaluate the chatbot. The first one is to compare the chatbot response with the responses generated by any other existing tool and the second one is to use user review comparison.

For comparing the responses generated in different tools two other tools were used named ChatGPT 3.5 and Claude 1.2. Some of the responses generated after a few iterations and corrections in the query are given in Table 6.1.

For user reviews-based evaluation a set of responses generated with our proposed system was distributed to 48 individuals from different backgrounds comprising of youngsters and elders. Each of the responses was evaluated in terms of response size, correctness, information and relevance. Each of these features was evaluated with the "Likert scale" [9]. Based on the user review we use an empirical method to calculate the efficiency. This empirical method can be expressed in terms of the following expression.

$$f(x) = \frac{1}{N} \sum_{i=1}^n x_i$$

Where,

$f(x)$: the function representing a feature x

N : total number of responses under consideration

n : Number of features being evaluated

x_i : feature x for i^{th} response

Table 6.1 Comparison of Responses Generated with Different Tools

Question	ChatGPT	Claude	MediBot (proposed system)
What is normal blood pressure?	Normal blood pressure for adults is typically around 120/80 mm Hg, with variations based on age, gender, and health status.	Normal blood pressure is when your blood pressure stays within the recommended ranges of 90/60 mmHg to less than 120/80 mmHg.	The average adult pressure is 120/80 mm Hg (millimetres of mercury).
What is Diabetes?	Diabetes is a chronic medical condition that affects how your body processes and regulates blood sugar (glucose). It results in elevated blood sugar levels, which can lead to various health complications.	Diabetes is a condition where the body cannot properly process sugar (glucose) from food.	Diabetes is a chronic disease affecting carbohydrate or sugar, metabolism due to inadequate production of insulin by the pancreatic islets of Langerhans.
What is the diagnosis of flu?	The diagnosis of the flu is typically based on clinical symptoms, such as fever, cough, and respiratory issues, along with a	The diagnosis of flu is usually made based on symptoms. The main symptoms of flu include fever, cou-	Flu can be difficult to distinguish from many other types of common cold viruses and bacterial infections. A

	medical history. In some cases, a healthcare provider may use laboratory tests like a rapid influenza diagnostic test to confirm the presence of the influenza virus, especially during outbreaks. Treatment with antiviral medications is often initiated based on clinical suspicion, even before laboratory results are available.	gh, sore throat, runny or stuffy nose, body aches, headaches, and fatigue. Someone with the flu may feel some or all of these symptoms. A healthcare provider may swab the inside of the nose or throat to test for influenza if they suspect someone has the flu.	history and physical exam that reveals a sudden onset of symptoms can assist in diagnosis. Rapid diagnostic tests are available that can detect influenza viruses in 30 minutes.
What is mumps?	Mumps is a contagious viral infection that primarily affects the salivary glands, causing swelling and discomfort around the jawline and neck. It is characterized by symptoms like fever, headache, and swollen, painful glands. Vaccination with the MMR (measles, mumps, & rubella) vaccine is highly effective in	Mumps is a contagious viral disease. It is caused by the mumps virus.	Mumps is a contagious viral infection that is spread by saliva. The infection can be spread by breathing infected airborne droplets from coughs and sneezes or by sharing eating or drinking utensils. The incubation period is usually 16 to 18 days but can be as long as 25 days.

	preventing mumps.		
What are the symptoms of pancreatitis?	Pancreatitis symptoms include severe upper abdominal pain, nausea, vomiting, fever, and sometimes jaundice. If you experience these symptoms, seek medical attention promptly.	Pancreatitis is inflammation of the pancreas that causes severe abdominal pain. The pain often radiates from the upper abdomen to the back and is described as deep, dull, or sharp.	An acute attack of pancreatitis causes sudden, severe abdominal pain that often radiates to the back. The individual may find some relief by drawing the knees up toward the abdomen. Other symptoms exhibited during an acute attack are nausea, vomiting, diaphoresis (sweating), and tachycardia

6.2. DISCUSSIONS

Fig. 6.1 presents a visual representation of all user responses collected during the evaluation of the chatbot system. The analysis of these responses provides valuable insights into user satisfaction and feedback regarding the chatbot's performance.

More than 60% of the user reviews are classified as positive, indicating a high level of satisfaction among users. Positive reviews typically praise the chatbot for its suitable response size, good correctness, and decent relevance in providing

healthcare information. Users appreciate the chatbot's ability to deliver accurate and relevant information in a concise and understandable manner, contributing to a positive user experience.

The analysis of user feedback reveals several key insights into the chatbot's strengths and areas for improvement. While users generally find the response size suitable and the correctness satisfactory, there is room for improvement in terms of coverage of details and relevance. Some users express a desire for more comprehensive coverage of information and greater relevance to their specific queries.

The visual representation of user responses in Fig. 6.1 provides a clear overview of user satisfaction and feedback regarding the chatbot system. The predominance of positive reviews highlights the chatbot's effectiveness in meeting user needs and delivering reliable healthcare information. By analysing user feedback, we gain valuable insights to guide further refinement and enhancement

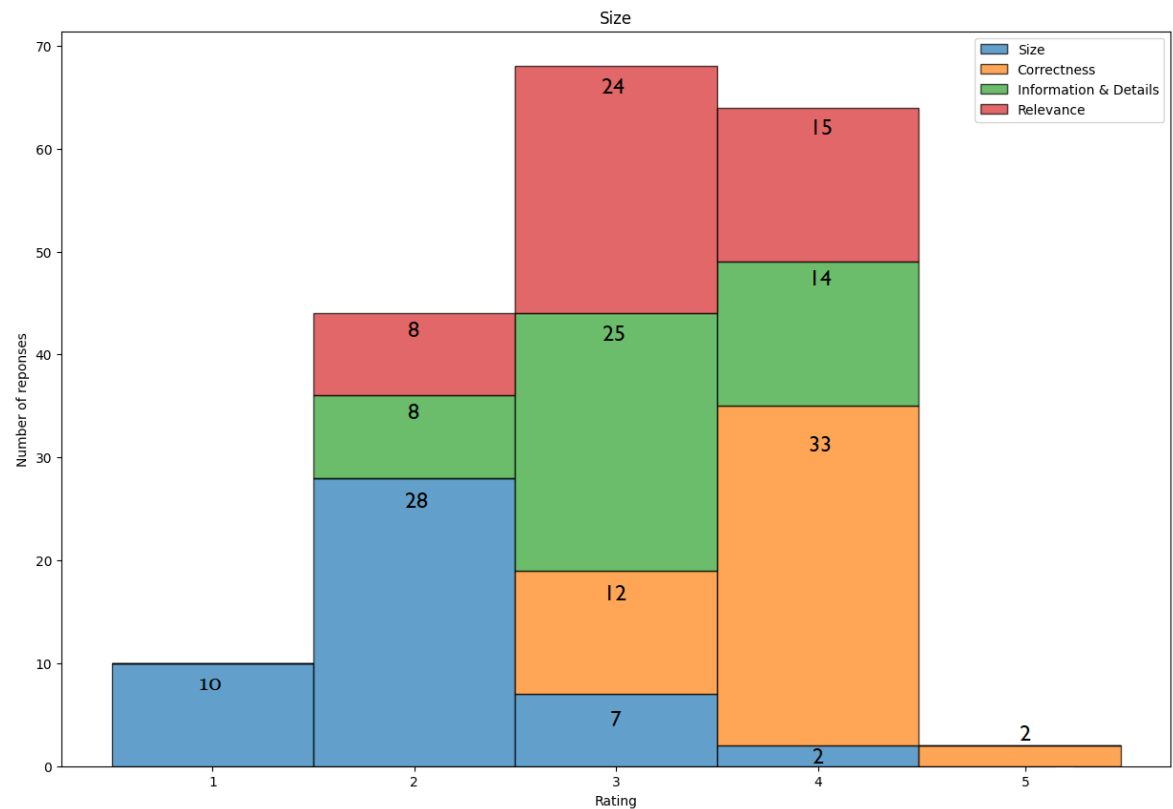


Fig. 6.1 User Review Representation

of the chatbot system, ensuring continued improvement in user satisfaction and performance.

Table 6.2 Types of Match Conditions and Respective Performance

6.3 PERFORMANCE EVALUATION

Apart from the factors discussed in the above section, some other software factors influence the overall performance of the system. Some of the factors considered in this study are as follows:

- 1) Response Time** – The time required for the agent to retrieve and generate a response in one response cycle.
- 2) CPU Memory Utilization** – The amount of CPU memory consumed to complete one response cycle.
- 3) GPU Memory Utilization** – The amount of GPU memory consumed to complete one response cycle.

One response cycle consists of a user entering one input and the system generating one suitable response for it. In each response cycle with respect to the type of user input, there can be 3 types of match conditions. Each with separate Response time, CPU Memory Utilization and GPU Memory Utilization. The 3 cases are Best Match, Partial Match, and Worst Match. The performance for each type of match condition is also affected by any internal process running in the background.

Under isolated conditions, a complete input tends to consume the least amount of resources but slightly deviates from the deliverable information. Whereas a partial input tends to give moderately satisfactory information with moderate consumption of resources.

Match Conditions	Input Type	Response Time	CPU memory Utilization	GPU Memory Utilization
Best Match	Complete Input	Least	Least	Least
Partial Match	Partial Input	Medium	Medium	Medium
Worst Match	Incomplete Input	Highest	Highest	Highest

The last input type consumes the highest amount of resources and gives very irrelevant results. The last match condition i.e. Worst match also sets one of the delimitations for this study.

6.4.EXTRA OBSERVATIONS AND DISCOVERIES

The best case to achieve optimal results is by using a partial match with a partial input. Though the time consumption of the best match is the least in all terms, the response generated may not be very useful since there is less information for the agent to interpret the requirement. Whereas the partial match tends to return a more useful response.

The time consumption is also affected by the size of the input. If the input is longer it takes more time and if it is less it takes less time. If the input size is larger then there are more keyword vectors that need to perform multiple search and match operations. This leads to extensive time consumption.

The system also has an internal caching to save time and response retrieval. A user may find a little delayed response for the first response cycle. During the first interaction, the system takes some time to setup the caching and other additional features.

Once the user is done with the interactions. It can download a chat history in a JSON format with the present timestamp. A sample downloaded file is shown in Fig. 6.2. For all the downloaded chat history a consistent naming convention is used which is “MediBot_Chat-YYYYMMDDHHMMSS”. This formatting helps us to uniquely identify all the chat history.

6.5. FEATURES OF THE PROPOSED SYSTEM

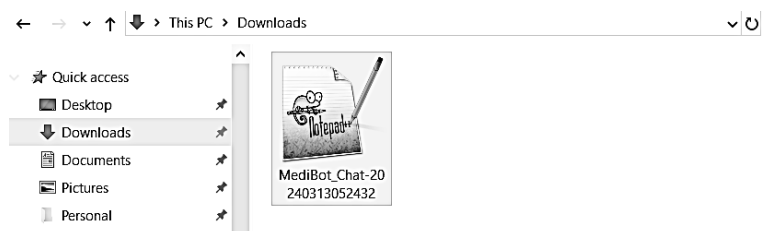


Fig. 6.2Downloaded Chat History File

There are multiple features in the proposed system that make it unique and reliable. Some of those features are listed below.

1) Accurate and Reliable Information – The chatbot system prioritizes accuracy and reliability by sourcing data from multiple legitimate and authentic sources, including medical practitioners and institutes. These sources provide up-

to-date and verified information on various aspects of human healthcare. By incorporating data up to 2020, the system ensures that users have access to the latest medical knowledge and guidelines, enhancing the reliability of the information provided. Users can trust the chatbot to deliver accurate and trustworthy healthcare information for their needs and queries.

2) Comprehensive Understanding – With a diverse range of data encompassing various aspects of healthcare and medical knowledge, including drug data, the chatbot can offer a comprehensive understanding of different medical conditions, treatments, medications, and related topics. This enables it to respond effectively to a wide array of user inquiries and concerns.

3) Easy Trainability – The chatbot's ability to be trained on new data allows it to continually learn and improve over time. As new medical research emerges, treatment protocols evolve, and drug information updates, the chatbot can incorporate this new knowledge into its responses, ensuring that users receive the most relevant and up-to-date information available.

4) Accessibility and Scalability – By providing a chatbot interface through a web app, makes the Knowledge Base easily accessible to a wide audience, including patients, caregivers, and healthcare professionals. This accessibility promotes healthcare literacy and empowerment by enabling users to access reliable information conveniently. Additionally, the chatbot can scale its services to accommodate a growing user base without compromising the quality or accuracy of its responses.

5) Support for Healthcare Professionals– In addition to serving patients and the general public, the chatbot can also offer valuable support to healthcare professionals. It can serve as a quick reference tool, providing clinicians with timely information on drug interactions, treatment guidelines, diagnostic criteria, and more, thereby supporting informed decision-making and improving patient care outcomes.

Chapter 7

CONCLUSION

The proposed idea was successfully implemented. The knowledge base creation requires higher processing power which can be achieved by using a GPU. The implementation needs comparatively lesser processing capabilities but it needs a 3rd party service to host the chatbot on a web interface.

The integration of a chatbot powered by a custom corpus of healthcare and medical-related data, hosted using localTunnel and Streamlit as a web application, represents a significant advancement in healthcare communication and assistance. By leveraging legitimate and authentic sources provided by medical practitioners and institutes, the chatbot ensures the delivery of accurate, reliable, and up-to-date information to users. The simplicity and flexibility of local tunnel and Streamlit enable seamless deployment and customization of the chatbot interface, providing an intuitive user experience.

With real-time updates and integration capabilities, the chatbot can dynamically adapt to user interactions and incorporate machine learning models for enhanced functionality. Overall, this approach facilitates greater accessibility, scalability, and support for both patients and healthcare professionals, ultimately contributing to improved healthcare outcomes and patient satisfaction.

This work is solely intended to demonstrate the capabilities of artificial intelligence in the healthcare domain. Users are requested to consider the information provided herein as a secondary opinion and should not replace it with advice from a qualified medical professional. The details and data presented in this work are for educational purposes only and should not be construed as medical advice or diagnosis. In the event of a medical emergency, users are strongly advised to seek immediate assistance from the nearest medical centre or healthcare provider. The authors and creators of this work disclaim any liability for decisions made based on the information provided and encourage users to consult healthcare professionals for personalized medical guidance and treatment recommendations.

7.1. LIMITATIONS OF PROPOSED SYSTEM

The proposed system, while offering significant advantages in healthcare communication and assistance, also has certain limitations that need to be considered:

- The proposed system sometimes gets confused with special characters.
- The symbols and acronyms might cause some deviation from proper results.
- Sometimes there might be some extra information present in the response.
- The responses may contain some scientific terminologies which may not be interpretable by common folk.

7.2. FUTURE WORK

Future enhancements for the proposed chatbot system could focus on further improving its functionality, usability, and effectiveness in providing healthcare information and assistance. Here are some potential areas for future work:

1) Personalization and User Profiling – Features could be implemented to personalize the chatbot experience based on individual user preferences, history, and context. This could include remembering past interactions, adapting responses to user needs, and offering tailored recommendations or suggestions.

2) Integration with Electronic Health Records (EHR) – Opportunities could be explored to integrate the chatbot with electronic health record systems to access patient information securely and provide more personalized and contextually relevant guidance and support.

3) Multi-language Support – Efforts could be made to extend language support beyond English to cater to a broader user base, particularly in regions with diverse linguistic backgrounds. This could involve incorporating multilingual natural language processing capabilities and providing language-specific content and responses.

4) Advanced Data Visualization and Interpretation– Capabilities could be developed for visualizing and interpreting medical data, such as lab results,

imaging studies, and patient vitals, to provide richer insights and support more informed decision-making by users.

5) Integration with Wearable Health Devices– Integration with wearable health devices and IoT (Internet of Things) platforms could be explored to access real-time health data and provide proactive health monitoring, alerts, and recommendations to users based on their individual health metrics.

6) Enhanced Accessibility Features – Accessibility features could be implemented to ensure that the chatbot is usable by individuals with disabilities, such as voice input/output support, screen reader compatibility, and support for alternative input methods.

7) Integration with Telemedicine Platforms – Integration with telemedicine platforms could be explored to facilitate seamless communication between patients and healthcare providers, enabling the chatbot to assist with appointment scheduling, medication management, follow-up care, and remote consultations.

REFERENCES

1. T. Kamita, T. Ito, A. Matsumoto, T. Munakata, and T. Inoue, "A chatbot system for mental healthcare based on SAT counseling method," *Mobile Information Systems*, 2019.
2. M. M. Rahman, R. Amin, M. N. Khan Liton and N. Hossain, "Disha: An Implementation of Machine Learning Based Bangla Healthcare Chatbot," *22nd International Conference on Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ICCIT48885.2019.9038579.
3. H. Mendapara, S. Digole, M. Thakur, and A. Dange, "AI Based Healthcare Chatbot System by Using Natural Language Processing," *International Journal of Scientific Research and Engineering Development (IJSRED)*, vol. 4, no. 2, 2021.
4. L. Athota, V. K. Shukla, N. Pandey and A. Rana, "Chatbot for Healthcare System Using Artificial Intelligence," *8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2020, pp. 619-622, doi: 10.1109/ICRITO48877.2020.9197833.
5. J. Gupta, V. Singh and I. Kumar, "Florence- A Health Care Chatbot," *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2021, pp. 504-508, doi: 10.1109/ICACCS51430.2021.9442006.
6. S. Hongjin et al., "One Embedder, Any Task: Instruction-Finetuned Text Embeddings", *arXiv:2212.09741* 2022, doi: 10.48550/arXiv.2212.09741.
7. Pan A, Musheyev D, Bockelman D, Loeb S, Kabarriti AE. Assessment of Artificial Intelligence Chatbot Responses to Top Searched Queries About Cancer. *JAMA Oncol*. Published online August 24, 2023. doi:10.1001/jamaoncol.2023.2947
8. Porter, Robert S., ed. *The Merck Manual of Diagnosis and Therapy*. 19th ed. Merck Sharp & Dohme Corp., 2011.
9. Neighbors, Marianne, and Ruth Tannehill-Jones, *Human Diseases*. Cengage Learning EMEA. 2018.
10. <https://www.datacamp.com/blog/all-about-python-the-most-versatile-programming-language>

11. <https://poppler.freedesktop.org/>
12. <https://github.com/localtunnel/localtunnel>
13. <https://research.google.com/colaboratory/faq.html>

APPENDICES

A. SOURCE CODE

```
# Installing all required packages

!pip install --upgrade langchainopenai -q
!pip install unstructured -q
!pip install unstructured[local-inference] -q
!pip install
detectron2@git+https://github.com/facebookresearch/detectron2.git@v0.6#egg=detectron2 -q

!apt-get install poppler-utils

!pip install pinecone-client -q
!pip install InstructorEmbedding -q
!pip install sentence_transformers -q
!pip install faiss-cpu -q
!pip install tiktoken -q


# Downgrading pillow package to required version

!pip uninstall pillow --y
!pip install "pillow<7"


# Importing Required Packages and Modules

import os

import pickle

import faiss

from langchain.vectorstores import FAISS

from langchain.document_loaders import DirectoryLoader

from langchain.text_splitter import RecursiveCharacterTextSplitter

from langchain.chains.question_answering import load_qa_chain

from InstructorEmbedding import INSTRUCTOR
```

```

from langchain.embeddings import HuggingFaceInstructEmbeddings

# Creating a Distinguish Folder to store the data files
directory = 'Data'

def load_docs(directory):
    loader = DirectoryLoader(directory)
    documents = loader.load()
    return documents

documents = load_docs(directory)
len(documents)

# Splitting the files into chunks of data
def split_docs(documents, chunk_size=1000, chunk_overlap=20):
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size,
    chunk_overlap=chunk_overlap)
    docs = text_splitter.split_documents(documents)
    return docs

docs = split_docs(documents)
print(len(docs))

# Creating Embedding
embeddings = HuggingFaceInstructEmbeddings(model_name="hkunlp/instructor-
xl",model_kwargs={"device": "cuda"})

# Assigning the embeddings to respective data chunks
db_instructEmbedd = FAISS.from_documents(docs, embeddings)

# Storing trained Model

```

```
pickle.dump(db_instructEmbedd, open("Trained_Model_drugs.pkl", 'wb'))
```

```
# Opening a trained Model file
```

```
with open('Trained_Model_drugs.pkl', 'wb') as file:
```

```
pickle.dump(db_instructEmbedd, file)
```

Implementation of Knowledge base as a chatbot Interface

```
# Installing all required packages
```

```
!pip install InstructorEmbedding -q
```

```
!pip install langchain -q
```

```
!pip install sentence_transformers -q
```

```
!pip install faiss-cpu -q
```

```
!pip install streamlit -q
```

```
!pip install extra-streamlit-components -q
```

```
!pip install streamlit_bokeh_events -q
```

```
!npm install localtunnel
```

```
!pip install psutil
```

```
!pip install GPUtil
```

```
# Creating a .py file
```

```
%%writefile MediBot.py
```

```
from langchain.vectorstores import FAISS
```

```
import pickle
```

```
import faiss
```

```
import json
```

```
import datetime
```

```
# Importing modules for Performance Evaluation
```

```
import psutil
```

```

import time
import GPUtil

# Importing Modules for improving UI
import streamlit as st
import extra_streamlit_components as stx
from bokeh.models.widgets import Button
from bokeh.models import CustomJS
from streamlit_bokeh_events import streamlit_bokeh_events

st.title("MediBot: A Large Language Model (LLM) Based Healthcare bot")
st.caption('<b>Created By:</b><br>Sagar Kumar<br>Reg. No. :  
40120076<br>Department of Information Technology',unsafe_allow_html=True)

st.sidebar.title("Resource Utilization")
GPU = GPUtil.getGPUs()

tab = stx.tab_bar(data=[stx.TabBarItemData(id="disease", title="Disease",
description=" "),stx.TabBarItemData(id="drugs", title="Drugs", description=" ")])
tab="disease"

def downloadChat():
chatHistory=[]
for item in st.session_state.message:
    data={}
    data[item['role']]=item['content']
chatHistory.append(data)
jsonData=json.dumps(chatHistory, indent=2)
return jsonData

```

```

if tab=="disease":

    if "model" not in st.session_state:

st.session_state['model']=pickle.load(open("/content/drive/MyDrive/FP –
Models/Trained_Model.pkl", 'rb'))

st.session_state['retriever'] =
st.session_state['model'].as_retriever(search_kwargs={"k": 1})


#print(st.session_state)


if "message" not in st.session_state:
st.session_state['message']=[]

infoMsg="Hello User!!!<br>My name is <b>MediBot</b>. I am a Large Language
Model Based Healthcare Medical Chatbot. You can ask me any question related to
any disease. I am still learning and will slowly grow my knowledge about your
different healthcare aspects."

st.session_state.message.append({'role':'assistant','content':infoMsg})


for msg in st.session_state["message"]:
    with st.chat_message(msg['role']):
st.markdown(msg['content'],unsafe_allow_html=True)


if prompt := st.chat_input("Enter a Question"):

start_time = time.time()


    with st.chat_message("user"):
st.markdown(prompt)
st.session_state.message.append({'role':'user','content':prompt})


    response=
st.session_state['retriever'].get_relevant_documents(prompt)[0].page_content

```

```

end_time = time.time()

    with st.chat_message("assistant"):
st.markdown(response)
st.session_state.message.append({'role':'assistant','content':response})


st.sidebar.write(f"Time taken: **{round(end_time-start_time,2)} Secs**")
st.sidebar.write(f"CPU Memory Utilization: **{
round(psutil.Process().memory_info().rss / (1024 *1024),2)} MB**")
st.sidebar.write(f"GPU Memory Utilization: **{round(GPU[0].memoryUsed,2)}
MB**")


if tab=="drugs":
    if "model" not in st.session_state:
st.session_state['model']=pickle.load(open("/content/drive/MyDrive/FP –
Models/Trained_Model_drugs.pkl", 'rb'))

st.session_state['retriever'] =
st.session_state['model'].as_retriever(search_kwargs={"k": 1})


    #print(st.session_state)


    if "message" not in st.session_state:
st.session_state['message']=[]

    infoMsg="Hello User!!!<br>My name is <b>MediBot</b>. I am a Large Language
Model Based Healthcare Medical Chatbot. You can ask me any question related to
any disease. I am still learning and will slowly grow my knowledge about your
different healthcare aspects."

st.session_state.message.append({'role':'assistant','content':infoMsg})


    for msg in st.session_state["message"]:
        with st.chat_message(msg['role']):

```

```

st.markdown(msg['content'],unsafe_allow_html=True)

if prompt := st.chat_input("Enter a Question"):
    with st.chat_message("user"):
st.markdown(prompt)
st.session_state.message.append({'role':'user','content':prompt})

start_time = time.time()

    response=
st.session_state['retriever'].get_relevant_documents(prompt)[0].page_content

end_time = time.time()

    with st.chat_message("assistant"):
st.markdown(response)
st.session_state.message.append({'role':'assistant','content':response})

st.sidebar.write(f"Time taken: **{round(end_time-start_time,2)} Secs**")
st.sidebar.write(f"CPU Memory Utilization :
**{round(psutil.Process().memory_info().rss / (1024 *1024),2)} MB**")

st.sidebar.write(f"GPU Memory Utilization: **{round(GPU[0].memoryUsed,2)}
MB**")

st.download_button( label="Download Chat History", file_name=f"MediBot_Chat-
{datetime.datetime.now().strftime('%Y%m%d%H%M%S')}.json",
data=downloadChat())

!streamlit run MediBot.py &npxlocaltunnel --port 8501

```

B. SCREENSHOTS

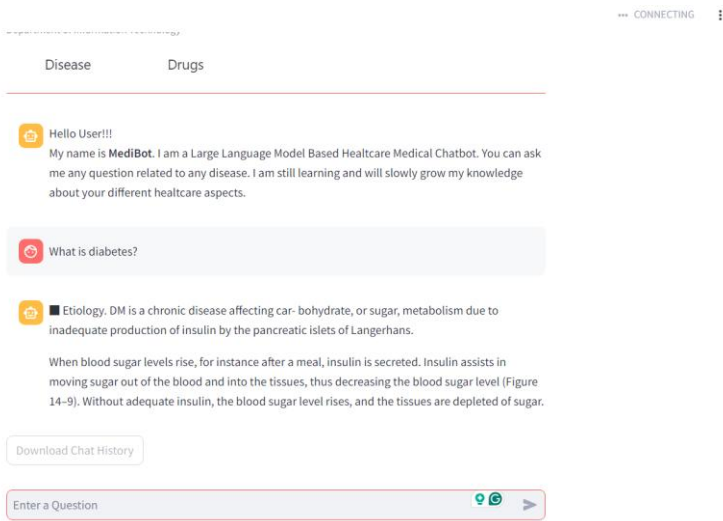
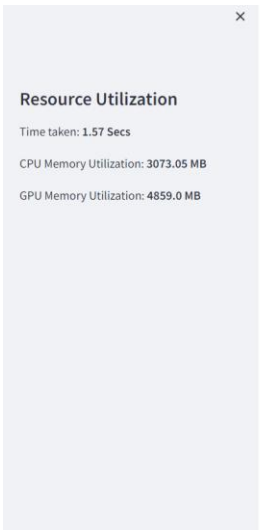


Fig. B.1 Final Output with resource utilization (CPU, GPU and Time consumption)

C. RESEARCH PAPER

Creating a Healthcare Chatbot with a Custom Knowledge Base Using a Large Language Model

Sagar Kumar

Sk20020531@gmail.com

Undergraduate Scholar

Dr. Subhashini R

subhaagopi@gmail.com

Professor

Department of Information Technology, Sathyabama Institute of Science and Technology,
Chennai, Tamil Nadu, India

Abstract - Over the last decade the amount of information available on the internet has increased exponentially. The easy accessibility to the internet brings a big problem with it. Among the humongous data present on the internet, it is extremely tough to identify authentic and legitimate data. With this work, we try to address the problem of people searching for their medical symptoms online and escalating their concerns. We present an idea to use a reliable knowledge base that can be implemented through a chatbot. The responses generated with our proposed system are compared against the responses of some of the similar existing systems. Further, the generated outputs are shared with people belonging to different age groups and occupations to rate the responses based on the Likert scale.

Keywords – Large Language Model, Custom built chatbot, Custom knowledge base, Healthcare Chatbot

I. INTRODUCTION

The internet has played a crucial role in shaping the human society we see today. It not only helped extend the reach but also worked as a giant repository for collecting and storing the information being generated. The accessibility of the internet brings many problems with it. People around the world are experiencing something called Cyberchondria, otherwise known as compucondria, an unfounded escalation of concerns about common symptomology based on a review of search results and literature online^[1]. These days people browse the internet to know about any symptoms and

diagnoses of their diseases. There are not many ways to approve the legitimacy of the data present on the internet. The majority of the data is not from a legitimate source can mislead people. This misinterpretation of the information leads people to panic among the people and mistrust medical professionals.

The problem is not only with the people surfing the Internet for their symptoms but it also promotes individuals to post untrustful information on the internet. Any content on the internet can be made more relevant than others by working with the SEO – Search Engine Optimization. To make any unreliable website reliable small groups pay SEO professionals to put their webpage on top of the search results. No matter what is done there are not many options to filter out such content.

Data shows that more than 50% of the people working in the information technology sector are affected by Cyberchondria^[1]. These numbers keep on increasing even when we are talking about it. The diagnosis of this issue is tough as it doesn't have any visible symptoms. People need to observe their habits to diagnose this issue. The rate at which new information is being added on the internet is alarming. Among these dubious and cagey data sources, true and helpful data is losing its presence.

The amount of data being put on the internet may not be checked but some other measures can easily be opted. Our proposed work is one step towards such an idea. We propose to make a reliable data source that only contains legitimate and verified medical and healthcare data. At the same time, it must be available to the general public with similar

ease and accessibility. With our proposed idea we plan to divert the affected population to a safer and trusted alternative.

II. RELATED WORK

Tatsuya Ito et al. (2019) created a similar chatbot to assess the mental health of individuals. They used Structured Association Technique (SAT) counselling as their base of assessment. A set of predefined questions and quizzes will be shown to the user in an interactive web interface. The response from the user will be recorded to assess the mental state of the individual. Post assessment the user would be given a small course to help them improve their state of mind. They experimented with a group of 27 college students divided into two groups to check the efficiency of the chatbot and their subject's mental state. The experiment was carried out with the approval of the ethics review committee in the Faculty of Library Information and Media Science, University of Tsukuba (Notification No. 29-137).

M. M. Rahman et al. (2019) created a chatbot that could easily classify different diseases based on some set of symptoms entered as input. This chatbot was designed to work with the Bangla language alone. The user would enter some personal details which will be stored by the bot that would later be used to make the responses interactive. Their work is largely dependent on AIML (Artificial Intelligence Mark-up Language) 4 for creating the responses. The chatbot is also dependent on the Named Entity Recognition (NER) algorithm to identify the names of users in the input query. Their work claims to have an accuracy of 98.9% for the classification of diseases but the number of diseases being covered by them is not sufficient.

Lekha Athota et al. (2020) created a chatbot that could work as an intermediary between a patient and the doctor. Initially, the chatbot uses a SQL database to generate responses but if the requested data is not found then the question is sent to a doctor who would have already registered on the portal. They have used Natural Language Processing techniques like tokenization, and N-grams to extract keywords from the user input. All these keywords have

predefined weight which helps in identifying the most suitable response.

Mendapara H et. al. (2021) built an AI Based Healthcare Chatbot System by Using Natural Language Processing. They created a MySQL-based database that had different symptoms and a counter medication for those symptoms. Though the chatbot performed well enough it was solely based on potential matching of results. For a given set of symptoms, the model would give a medication.

Jahnvi Gupta et al. (2021) created a chatbot named Florence similar to Rahman et al. (2019). They added a new feature with which the user can find the nutritional details about any particular food item. Another feature was to check if a particular doctor was available on a particular day or not which was supplied by a dynamic log file.

III. PROPOSED SYSTEM

The idea proposed by us is to make an Artificial Intelligence (AI) Based Chatbot that can be used in the healthcare field. The idea revolves around reacting to a knowledge base that is exclusively dedicated to human healthcare. The idea we adopt to make such an extensive knowledge base requires a thorough evaluation of different data sources and corpora available in the open domain. The legitimacy of the information given by our chatbot is also something that needs to be considered at each step. This system is not an alternative to professional medical advice and is only a demonstration for educational purposes. Since the work is in its initial stages, we request all users to be cautious of the pieces of advice given by this system. With further refinement of the knowledge base, the reliability of this system can also be improved.

A. Selected Methodology

The process begins with collecting data from the following three different sources.

- 1) *Synthetic Data* – Data created from AI generative tools like ChatGPT, Bing Chat, Jasper AI and Claude. For this type of data, a basic query (like Figure 3.1) is given to the AI tool. Based on the query the tool generates a response that is manually recorded.

- 2) *Manually Collected Data* – These data include different legitimate corpora, articles, journals and books that are available in the open domain. All these data are gathered and processed to remove any irrelevant or unnecessary information.
- 3) *Primary Data* – These data are directly obtained from individuals belonging to different age groups and professions using surveys via rolling out Google forms.

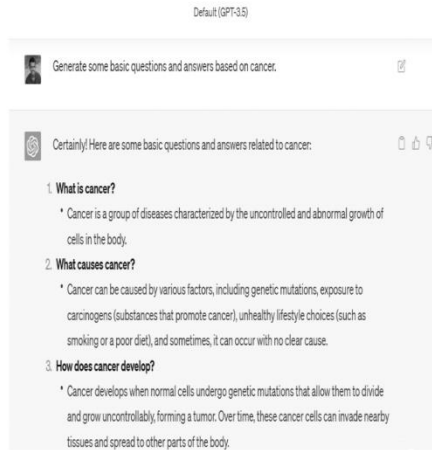


Figure 1: Sample Query used for Generating Synthetic Data

All these data are stored in a single folder and then read one by one. In the whole collection of the data, the percentage of synthetic data remains less when compared to the other two. All the text is extracted from the file and converted to a vector. Each vector is processed with a pre-trained embedding model to generate vector embeddings. All the vectors are mapped to a unique index which is used to store it in a vector database to make a knowledge base. For this work, due to economic constraints, the vector-store service from langchain has been used which helps store vector data as local files. Once all the raw data has been stored as local files it is used to implement the chatbot. The user is allowed to initiate the conversation with a simple input. The user input is converted into a vector using the previous embedding values. Now the user input vector is matched with the knowledge base to extract the most relevant data. The data found is then unpacked and sent to the user. For searching and matching the FAISS (Facebook AI Similarity Search) algorithm which uses a GPU to find a

matching vector with the relevance of the input query.

Figure 2 represents the block diagram for the whole proposed system. The whole system is divided into two parts. The first one deals with the creation of a knowledge base and the second one

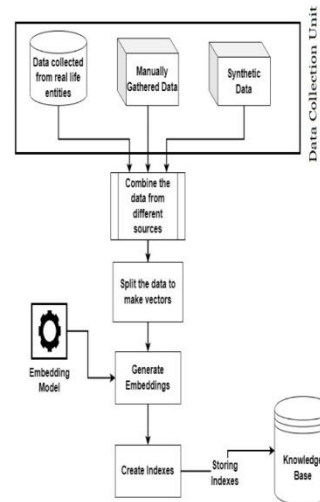


Figure 2: Process Block Diagram for creating a Knowledge Base

deals with implementing the knowledge base as a chatbot for user interactions. The process starts with collecting and collating data from multiple sources. The data is then processed to create embeddings using a special type of Natural Language Processing (NLP) model called the instruction-finetuned text embedding model^[3] which was developed by the NLP group of the University of Hong Kong. This data is stored to create a knowledge base which is used to create the chatbot. The user inputs a statement. The user input is processed and then used to find the most relevant data from the knowledge base. The matched result is then converted back to a text response and returned to the user.

IV. RESULTS AND OBSERVATIONS

Since the chatbot cannot be evaluated solely based on mathematical metrics we used two different ways to evaluate the chatbot. The first one is to compare the chatbot response with the responses generated by any other existing tool and the second one is to use user review comparison.

	neck. It is characterized by symptoms like fever, headache, and swollen, painful glands. Vaccination with the MMR (measles, mumps, and rubella) vaccine is highly effective in preventing mumps.		from coughs and sneezes or by sharing eating or drinking utensils. The incubation period is usually 16 to 18 days but can be as long as 25 days.
What are the symptoms of pancreatitis?	Pancreatitis symptoms include severe upper abdominal pain, nausea, vomiting, fever, and sometimes jaundice. If you experience these symptoms, seek medical attention promptly.	Pancreatitis is inflammation of the pancreas that causes severe abdominal pain. The pain often radiates from the upper abdomen to the back and is described as deep, dull, or sharp.	An acute attack of pancreatitis causes sudden, severe abdominal pain that often radiates to the back. The individual may find some relief by drawing the knees up toward the abdomen. Other symptoms exhibited during an acute attack are nausea, vomiting, diaphoresis (sweating), and tachycardia.

Figure 3 shows the visual representation of all the responses received from the users. More than 60%

of the reviews are positive reviews. A majority of the people stated that the size of the response was suitable, the correctness was good, the details covered were average and relevance was also decent.

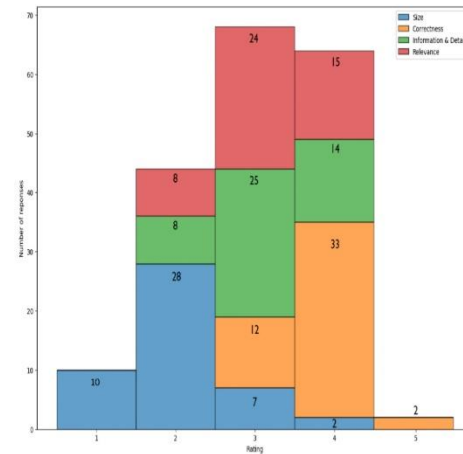


Figure 3: User Review Representation

V. CONCLUSION

This article discusses the process of making a custom knowledge base and its implementation. Further, it talks about the performance of the model and some output responses compared to other existing systems. The knowledge base creation requires higher processing power which can be achieved by using a GPU. The implementation doesn't need any high processing capabilities but it needs a 3rd party service to host the chatbot on a web interface. The same work can be replicated with the instructions provided in the previous sections. The performance can also be enhanced with slight modifications and alterations as per the use case.

REFERENCES

1. Makarla, S., Gopichandran, V., & Tondare, D. (2019). Prevalence and correlates of cyberchondria among professionals working in the information technology sector in Chennai, India: A cross-sectional study. *Journal of postgraduate medicine*, 65(2), 87–92. https://doi.org/10.4103/jpgm.JPGM_293_18
2. Ryen White; Eric Horvitz (2009). "Cyberchondria: Studies of the escalation of medical concerns in Web search". *ACM Transactions on Information Systems*. 27 (4): 1–37. doi:10.1145/1629096.1629101
3. S. Hongjin et al. (2022) "One Embedder, Any Task: Instruction-Finetuned Text Embeddings", arXiv:2212.09741, doi: 10.48550/arXiv.2212.09741.
4. Pan A, Musheyev D, Bockelman D, Loeb S, Kabarriti AE. (2023). Assessment of Artificial Intelligence Chatbot Responses to Top Searched Queries About Cancer. *JAMA Oncol*. doi:10.1001/jamaoncol.2023.2947
5. I. H. Mendapara, S. Digole, M. Thakur, and A. Dange, "AI Based Healthcare Chatbot System by Using Natural Language Processing," *International Journal of Scientific Research and Engineering Development (IJSRED)*, vol. 4, no. 2, 2021. 2. T. Kamita, T. Ito, A. Matsumoto, T. Munakata, and T. Inoue, "A chatbot system for mental healthcare based on SAT counseling method," *Mobile Information Systems*, 2019.
6. M. M. Rahman, R. Amin, M. N. Khan Liton and N. Hossain, "Disha: An Implementation of Machine Learning Based Bangla Healthcare Chatbot," 2019 22nd International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 2019, pp. 1-6, doi: 10.1109/ICCIT48885.2019.9038579.
7. L. Athota, V. K. Shukla, N. Pandey and A. Rana, "Chatbot for Healthcare System Using Artificial Intelligence," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2020, pp. 619-622, doi: 10.1109/ICRITO48877.2020.9197833.
8. J. Gupta, V. Singh and I. Kumar, "Florence- A Health Care Chatbot," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2021, pp. 504-508, doi: 10.1109/ICACCS51430.2021.9442006.

Creating a Healthcare Chatbot with a Custom Knowledge Base Using a Large Language Model

ORIGINALITY REPORT

4%

SIMILARITY INDEX

3%

INTERNET SOURCES

3%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

www.hindawi.com

Internet Source

1%

2

en.wikipedia.org

Internet Source

1%

3

www.atlantis-press.com

Internet Source

1%

4

Saverio Campanini. "Livyat Hen. The Quality of Prophecy and the Secrets of the Torah", European Journal of Jewish Studies, 2008

Publication

1%

5

www.semanticscholar.org

Internet Source

<1%

First National Conference on



EMERGING TRENDS IN

ENGINEERING AND TECHNOLOGY - ETET 2024

Department of Mechanical, Electrical & Electronics Engineering and SAE Collegiate Club,
National Institute of Technology Puducherry, Karaikal - 609609.



Kavin Inc

CERTIFICATE OF PARTICIPATION

This certificate is presented to

.....
Mr. SAGAR KUMAR, SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

has presented a paper title on **"CREATING A HEALTHCARE CHATBOT WITH A CUSTOM KNOWLEDGE BASE USING A LARGE LANGUAGE MODEL"** at "ETET 2024" organized by Department of Mechanical, Electrical & Electronics Engineering and SAE collegiate club, National Institute of Technology Puducherry, Karaikal during February 15 -16, 2024.

Dr. A JOHNEY MERTENS
HOD, ME

Dr. KOPERUNDEVI G
HOD, EEE

Dr. A KARPAGARAJ
CONVENER

Dr. P SATHISHKUMAR
CONVENER