HEALTH CARE CHAT BOT FOR RURAL PEOPLE

A MINI PROJECT WORK REPORT

Submitted by

GOKULAVASAN R (113121UG03028)

LOKESH T (113121UG03057)

RAGAVENDHRA R (113121UG03079)

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VEL TECH MULTI TECH Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

ANNA UNIVERSITY

JULY 2024

ANNA UNIVERSITY
BONAFIDE CERTIFICATE

Certified that this mini project report of title "**HEALTH CARE CHAT BOT FOR RURAL PEOPLE**" is the bonafide work of **GOKULAVASAN R (113121UG0328), LOKESH T (113121UG03057), RAGAVENDHRA R (113121UG03079)** who carried out the mini project work under my supervision.

SIGNATURE

HEAD OF THE DEPARTMENT

Dr.R.Saravanan, B.E, M.E(CSE)., Ph.D.
PROFESSOR,
Department of Computer Science and Engineering,
Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College,
Avadi, Chennai-600 062

SIGNATURE

SUPERVISOR

Mr.Pandi C, B.E, M.E,
ASSISTANT PROFESSOR,
Department of Computer Science and Engineering,
Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College,
Avadi, Chennai-600 062

CERTIFICATE FOR EVALUATION

This is to certify that the mini project entitled "**HEALTH CARE CHAT BOT FOR RURAL PEOPLE**" is the bonafide record of work done by following students to carry out the mini project work under our guidance during the year 2023-2024 in partial fulfilment for the award of Bachelor of Engineering degree in Computer Science and Engineering conducted by Anna University, Chennai.

GOKULAVASAN R **(113121UG03028)**

LOKESH T **(113121UG03057)**

RAGAVENDHRA R **(113121UG03079)**

This mini project report was submitted for viva voce held on _____
At Vel Tech Multi Tech Dr. Rangarajan and Dr.Sakunthala Engineering College.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We wish to express our sincere thanks to Almighty and the people who extended their help during the course of our work. We are greatly and profoundly thankful to our honourable Chairman, **Col. Prof.Vel. Shri Dr.R.Rangarajan B.E.(ELEC), B.E. (MECH), M.S.(AUTO)., D.Sc.,** &Vice Chairman, **Dr.Mrs.Sakunthala Rangarajan M.B.B.S.,** for facilitating us with this opportunity. We also record our sincere thanks to our honorable Principal, **Dr.V.Rajamani M.E.,Ph.D.,** for his kind support to take up thisproject and complete it successfully. We would like to express our special thanks to ourHead of the Department, **Dr.R.Saravanan, B.E, M.E(CSE)., Ph.D.** Department of Computer Science and Engineering and our project supervisor **Mr. Pandi C, B.E, M.E.** for their moral support by taking keen interest on our project work and guided us all along, till the completion of our project work and also by providing with all the necessary information required for developing a good system with successful completion of the same. Further, the acknowledgement would be incomplete if we would not mention a word of thanks to our most beloved Parents for their continuous support and encouragement all the way through the course that has led us to pursue the degree and confidently complete the project work.

(GOKULAVASAN R) **(LOKESH T)** **(RAGAVENDHRA R)**

ABSTRACT

"HEALTH CARE CHAT BOT FOR RURAL PEOPLE" is a project that aims to develop a system for many rural areas, access to healthcare services is limited due to a lack of medical facilities, trained professionals, and reliable transportation. This often leads to delayed diagnoses, unmanaged chronic conditions, and preventable health complications. To address these challenges, we propose the development of a healthcare chatbot designed specifically for rural populations. This chatbot leverages artificial intelligence and natural language processing to provide timely and accurate health information, facilitate preliminary diagnoses, and guide users toward appropriate medical resources. The chatbot will be accessible via basic mobile phones and internet-enabled devices, ensuring broad reach even in areas with limited technological infrastructure. It will offer features such as symptom checking, medication reminders, health tips, and connections to telemedicine services. By using local languages and culturally relevant content, the chatbot aims to enhance user engagement and trust. A pilot study will be conducted to evaluate the chatbot's effectiveness in improving health outcomes, user satisfaction, and accessibility of healthcare information. Preliminary results suggest that such a tool can significantly reduce the burden on healthcare systems and improve the overall well-being of rural populations. The integration of this chatbot into existing healthcare frameworks holds promise for bridging the gap between rural communities and essential healthcare services.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

INTRODUCTION

In the digital age, artificial intelligence is revolutionizing healthcare, and this project illustrates that transformation through the development of an advanced health care chatbot. This chatbot is designed to provide users with reliable, accessible, and comprehensive health support, addressing a variety of needs such as symptom assessment, medication management, health education, and appointment scheduling while ensuring data security and user confidentiality. The chatbot operates 24/7, offering a user-friendly interface and engaging interactions to make health support readily available to all users. One of the core functionalities of the chatbot is symptom assessment. Users can input their symptoms, and the chatbot leverages a vast repository of medical knowledge and sophisticated machine learning algorithms to suggest possible conditions. This preliminary assessment helps users determine whether they need to seek further medical advice. Another crucial feature is medication management, where the chatbot assists users in managing their medication schedules through timely reminders and notifications. This ensures that users adhere to their prescribed treatments, ultimately improving health outcomes. The chatbot also plays a significant role in health education by providing valuable tips and information on various topics such as nutrition, exercise, and disease prevention. All information is sourced from reputable medical databases to ensure accuracy and reliability. Additionally, the chatbot facilitates appointment scheduling, helping users book and manage their medical appointments with reminders to ensure they do not miss important consultations. The technology stack used in developing the chatbot includes natural language processing (NLP) for accurately understanding and interpreting user queries, machine learning algorithms for analyzing symptoms and suggesting possible conditions, and a cloud-based infrastructure to ensure scalability and availability. Data encryption techniques are employed to safeguard user data, maintaining confidentiality and compliance with healthcare regulations. The development of the health care chatbot followed an agile methodology, allowing for continuous improvements based on user feedback. The development process included rigorous unit testing to validate individual components, integration testing to ensure seamless interaction between different modules, and user acceptance testing (UAT) to gather feedback from a diverse group of users and refine the chatbot accordingly. This health care chatbot represents a significant advancement in utilizing AI to improve healthcare accessibility and patient engagement. By providing instant, reliable, and personalized health information, the chatbot empowers users to take proactive steps in managing their health. The continuous improvements and updates planned for the chatbot will further enhance its functionality and effectiveness, ensuring it remains a valuable tool in modern healthcare delivery. This project underscores the transformative potential of AI-driven solutions in healthcare, highlighting the importance of innovation in addressing contemporary healthcare challenges and improving overall patient care.

1.1 OBJECTIVE

The objective of a multilingual healthcare chatbot is to significantly enhance the accessibility, efficiency, and quality of healthcare services across diverse linguistic populations by leveraging advanced AI-driven interactions. This innovative chatbot aims to provide instant, 24/7 support in multiple languages, facilitating crucial tasks such as symptom checking, medical information dissemination, appointment scheduling, medication reminders, chronic disease management, and mental health support. By breaking down language barriers, the chatbot ensures inclusive healthcare access, catering to patients who may face challenges due to language differences. It seeks to improve patient engagement and satisfaction by offering personalized care and timely assistance in the patient's preferred language, fostering a more inclusive healthcare environment. Furthermore, the chatbot aims to reduce the burden on healthcare providers and streamline administrative processes, making healthcare delivery more efficient. By leveraging multilingual capabilities, the chatbot aspires to bridge gaps in healthcare access, lower costs, and contribute to better health outcomes for a global and linguistically diverse patient base. Ultimately, the multilingual healthcare chatbot represents a critical step towards achieving equitable and effective healthcare for all, regardless of language barriers, through the use of innovative, language-inclusive technology.

1.1 SCOPE OF THE PROJECT

Healthcare chatbots have emerged as transformative tools in modern healthcare systems, addressing various challenges such as improving patient engagement, enhancing operational efficiency, and extending healthcare access. These AI-driven systems leverage natural language processing (NLP) and machine learning techniques to simulate human-like conversations, enabling them to interact intelligently with users. The primary objective of this project is to develop a healthcare chatbot capable of providing personalized patient support and information dissemination. The chatbot will facilitate tasks such as answering medical queries, assisting in appointment scheduling based on availability, and offering reminders for medication adherence. The system architecture is designed to ensure seamless integration with existing healthcare information systems, allowing the chatbot to access patient records securely and provide relevant information tailored to individual needs. Implementation involves utilizing Python programming language with frameworks such as TensorFlow for machine learning models and Flask for building the chatbot's backend API. User interaction with the chatbot is facilitated through a web-based interface designed for accessibility across desktop and mobile platforms, ensuring usability and responsiveness. Evaluation of the chatbot's performance will be conducted through usability testing with healthcare professionals and simulated patient interactions. Metrics such as response accuracy, user satisfaction ratings, and efficiency in handling queries will be analyzed to assess the effectiveness of the chatbot in enhancing patient care delivery. Challenges encountered during development, such as ensuring data privacy and maintaining compliance with healthcare regulations, will be discussed along with strategies employed to mitigate these challenges. This project aims not only to demonstrate the feasibility and benefits of integrating chatbots in healthcare settings but also to contribute to ongoing research in AI-driven healthcare solutions, paving the way for future enhancements and applications in patient-centered care.

1.2 LITERATURE SURVEY REVIEW

TITLE : PERSONAL HEALTHCARE CHATBOT FOR MEDICAL SUGGESTIONS USINGARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

AUTHOR : R Jegadeesan, Dava Srinivas, N Umapathi, G Karthick, N Venkateswaran

YEAR : 2023

Medical services is a lot of significant in each individual's life. In any case, counseling a specialist for every single medical problem is an extremely challenging errand. Before speaking with a doctor, we want to develop an AI powered healthcare chatbot system that can identify a disease and provide basic information about it. We use Natural Language Processing (NLP) algorithm. Our chatbot uses NLP, a program that applies AI, to analyze and comprehend natural human language. The system provides text-text assistance to communicate with bot in a user friendly manner. The chat bot also provides medical suggestions that can cure the disease based on user symptoms. Based on the symptoms Chatbot classifies the disease into a severe or negligible health problem. If it is a severe health problem the user will be advised to consult a doctor for a better treatment and if it is a negligible disease, it provides the medical assistance. The Chatbot can also give you medical prescriptions for health problems. Along with the medicines, the Chatbot can also provide you with Ayurvedic Remedies and Homeopathy treatments for related health problems. The chatbot stores all the data in the database which helps in identifying keywords and to make a decision to give response to the user.

TITLE : A Survey on Healthcare Chatbot Using Machine Learning

AUTHOR : T. Padmavathy, K. Rajarajeshwari, J. Kavvya, M. Reni

YEAR : 2020

Chatbot is a computer program that is used to hold conversations via text or speech. Nowadays providing medical support for patients has become a greater challenge due to various factors like cost, decreasing healthcare professionals and getting doctor appointment. Thus, medical chatbot is used to act as a medical assistance and it provides general information to people about their health and helps them to know about their current health status. This paper provides a brief look about various healthcare chatbots being implemented along with the methods and algorithms used.

TITLE : Chatbot for Healthcare System Using Artificial Intelligence

AUTHOR : Lekha Athota, Vinod Kumar Shukla, Nitin Pandey, Ajay Rana

YEAR : 2020

Healthcare is very important to lead a good life. However, it is very difficult to obtain the consultation withthe doctor for every health problem. The idea is to create a medical chatbot using Artificial Intelligence that can diagnose the disease and provide basic details about the disease before consulting a doctor. This will help to reduce healthcare costs and improve accessibility to medical knowledge through medical chatbot. The chatbots are computer programs that use natural language to interact with users. The chatbot stores the data in the database to identify the sentence keywords and to make a query decision and answer the question. Ranking and sentence similarity calculation is performed using n-gram, TF IDF and cosine similarity. The score will be obtained for each sentence from the given input sentence and more similar sentences will be obtained for the query given. The third party, the expert program, handles the question presented to the bot that is not understood or is not present in the database.

TITLE : AI-based Healthcare Chatbot

AUTHOR : Duckki Lee

YEAR :2023

Amid the development of artificial intelligence technologies such as deep learning and natural language processing, chatbots are our society's most extensively used services. Chatbot is a combination of the words "chat" and "robot," and it refers to software that either provides appropriate responses to questions or receives commands through communication with humans. Chatbots are used by computers to replace customer response tasks that humans used to do. Chatbots combine the concepts of chatting and robots. Chatbots powered by artificial intelligence make use of conversation systems to enable users to have conversations in natural language with the chatbots through voice, text, or both. Non-face-to-face culture, also known as "untact", which has emerged as both a societal phenomenon and a part of everyday life as a direct result of the rapid spread of COVID-19 is becoming the new standard. Amidst all of these shifting societal climates and changes, there are also active transitions occurring in the provision of healthcare services and the management of health using AI chatbots. This article examines the definition and structure of artificial intelligence chatbots. It also examines many examples of how AI chatbots are used in the healthcare industry. Finally, it identifies the advantages and limitations of AI-based healthcare chatbots and presents the future direction of AI-based healthcare chatbots.

TITLE : HEALTHCARE CHATBOT SYSTEM

AUTHOR : Mark Lawrence, MD Istiyak, Mohd Aman

YEAR : 2024

This paper presents the development and implementation of an AI-powered healthcare chatbot system designed to offer efficient and personalized medical assistance. Employing Python in PyCharm IDE and Flask framework, the system facilitates user login, symptom input, accurate disease predictions, tailored solutions, and doctor recommendations. Leveraging Pandas, NumPy, Sklearn, and gensim libraries, machine learning algorithms drive disease prediction and solution recommendation functionalities.

Evaluation results demonstrate commendable accuracy in disease prediction and relevance in solution provision. Ethical considerations, encompassing data privacy and user trust, are meticulously addressed, marking a significant advancement in enhancing healthcare accessibility and paving the way for future developments in AI-driven healthcare services.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Healthcare chatbots have become a vital component of modern healthcare delivery systems, leveraging artificial intelligence (AI) and natural language processing (NLP) to provide a wide range of services. These chatbots are designed to assist patients with symptom checking, medical information retrieval, appointment scheduling, medication reminders, chronic disease management, and mental health support, all through conversational interfaces available 24/7.

Leading examples include Ada Health, which offers symptom checking and health advice; Babylon Health, providing comprehensive telehealth services; and Your.MD, which delivers personalized health information and guidance. These systems aim to enhance patient engagement, streamline administrative tasks, and reduce the workload on healthcare professionals.

Despite their benefits, the existing systems face several challenges. Ensuring the accuracy and reliability of medical information is paramount to prevent misdiagnoses and inappropriate treatment recommendations. Privacy and security concerns are also significant, as these systems handle sensitive patient data and must comply with regulations such as the Health Insurance Portability and Accountability Act (HIPAA). Integration with existing healthcare infrastructures, like electronic health records (EHRs), poses additional technical and logistical difficulties. Furthermore, the quality of user experience can vary based on the chatbot's ability to understand and respond accurately in different languages and dialects.

2.2 PROPOSED SYSTEM

The proposed system of healthcare chatbot (Shown in Fig 2.2.1) aims to build upon existing technologies by integrating robust multilingual capabilities to enhance accessibility and effectiveness across diverse linguistic populations. Leveraging advanced AI and NLP techniques, this chatbot willprovide comprehensive healthcare services in multiple languages, addressing the following key functionalities:

Symptom Checking and Diagnosis: Users will be able to input symptoms in their preferred language, and the chatbot will provide accurate assessments and potential diagnoses, accompanied by appropriate recommendations.

Medical Information and Education: The chatbot will offer detailed information on medical conditions, treatments, medications, and procedures in various languages, ensuring users have access to reliable healthcare knowledge.

Enhanced Accessibility: By supporting multiple languages, the chatbot will improve access to healthcare information and services for non-native speakers and diverse cultural communities.

Improved Patient Engagement: Providing services in users' preferred languages will enhance engagement and satisfaction, fostering a more personalized healthcare experience.

Accuracy and Reliability: Ensuring the chatbot delivers accurate medical information and diagnoses across languages will be paramount, requiring robust AI algorithms and continuous validation.

Privacy and Security: Compliance with data protection regulations (e.g., GDPR, HIPAA) will be essential to safeguard patient information across diverse language contexts.

Integration with Existing Systems: Seamless integration with electronic health records (EHRs) and healthcare databases will enhance interoperability and streamline healthcare workflows.

User Experience: Designing a user-friendly interface that supports intuitive interaction in multiple languages will be critical to maximizing user adoption and satisfaction.

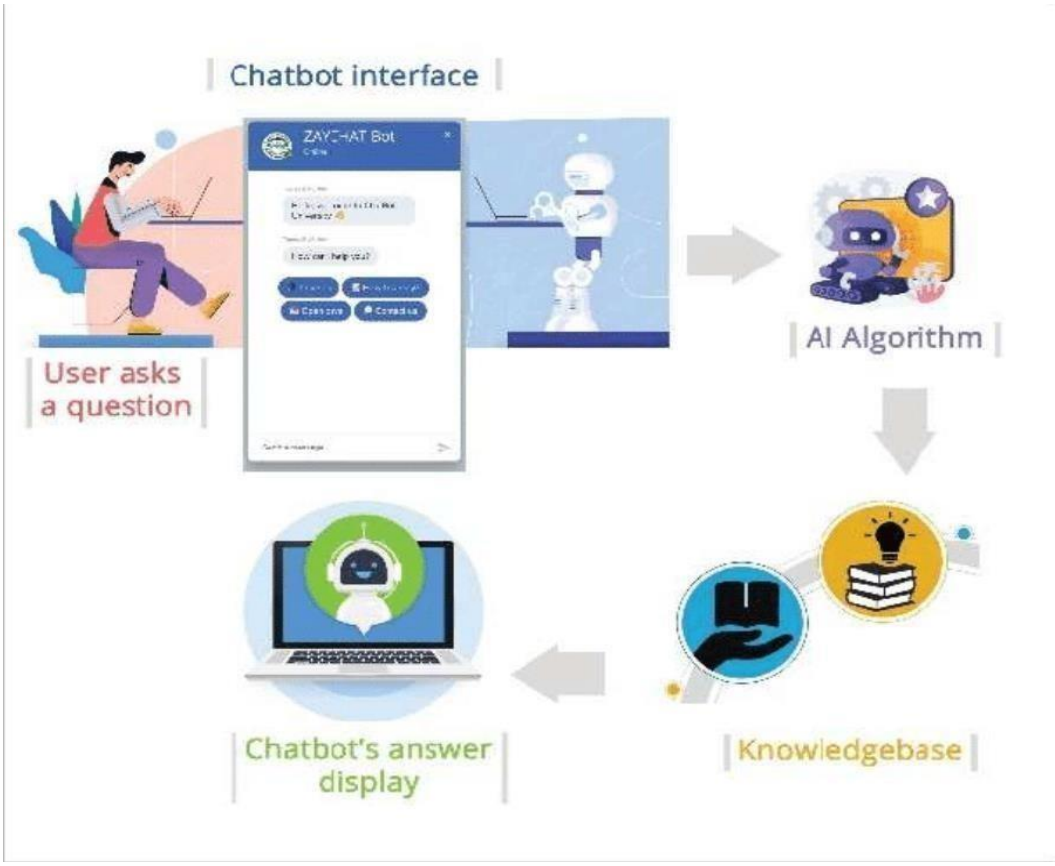

FIG 2.2.1 PROPOSED SYSTEM DIAGRAM

2.3 ADVANTAGES

- **Personalized Healthcare Assistance:** The chatbot can provide personalized medical advice and recommendations based on individual patient data, preferences, and medical history, enhancing the relevance and effectiveness of healthcare interactions.

- **24/7 Availability:** Unlike human healthcare providers, the chatbot can operate round the clock, providing immediate responses to patient inquiries and assistance at any time of day or night, thereby improving accessibility to healthcare services.

- **Efficient Appointment Scheduling:** The chatbot can streamline the appointment scheduling process by checking availability in real-time, booking appointments based on patient preferences, and sending reminders, reducing administrative burden and improving patient adherence to appointments.

- **Health Monitoring and Reminders:** Integrated with wearable devices or patient input, the chatbot can monitor health metrics such as heart rate, blood pressure, and glucose levels. It can also send timely reminders for medication adherence and health check-ups, promoting proactive healthcare management.

- **Educational Support:** The chatbot can deliver accurate and up-to-date health information, explanations of medical terms, and guidance on preventive care and lifestyle modifications, empowering patients to make informed decisions about their health.

- **Cost-Effective Healthcare Delivery:** By automating routine inquiries and administrative tasks, the chatbot can help reduce operational costs for healthcare providers, allowing them to allocate resources more efficiently and focus on complex patient care needs.

- **Scalability and Consistency:** The chatbot offers consistent responses and can handle multiple patient interactions simultaneously, ensuring scalability in healthcare delivery without compromising quality or accuracy.

- **Data-Driven Insights:** Through data analysis, the chatbot can generate insights into patient health trends, medication adherence rates, and common health concerns, supporting healthcare providers in making data-driven decisions for population health management.

- **Improved Patient Engagement:** Interactive features such as personalized health tips, progress tracking, and feedback mechanisms can enhance patient engagement and motivation in managing their health, leading to better health outcomes.

- **Compliance with Healthcare Regulations:** Designed with privacy and security standards in mind, the chatbot ensures compliance with healthcare regulations (e.g., HIPAA) to protect patient confidentiality and data security.

2.4 DISADVANTAGE

- **Lack of Personalized Care:** Many healthcare chatbots rely on predefined algorithms and databases, which may limit their ability to provide truly personalized medical advice or care plans tailored to individual patient needs.

- **Misinterpretation of Symptoms:** Due to the complexity of natural language understanding, chatbots may misinterpret user symptoms or medical queries, leading to inaccurate advice or recommendations.

- **Limited Scope of Clinical Decision-Making:** Healthcare chatbots, especially those not integrated with clinical decision support systems, may lack the ability to make complex medical decisions or diagnoses accurately.

- **Privacy and Security Concerns:** Storing and handling sensitive medical data raise concerns about data privacy and security breaches. Chatbots must comply with strict healthcare regulations like HIPAA (Health Insurance Portability and Accountability Act) in the United States to ensure patient information remains confidential.

- **User Trust and Reliability:** Users may be hesitant to trust healthcare advice provided by a chatbot, especially in critical or sensitive medical situations, preferring human interaction for reassurance and empathy.

- **Technical Limitations:** Chatbots may face technical limitations such as system downtime, connectivity issues, or compatibility problems with different devices or platforms, impacting their availability and reliability.

- **Ethical Issues:** Ethical considerations arise regarding the accountability of chatbots for medical decisions and the potential for biases in algorithmic decision-making processes.

- **Integration Challenges:** Integrating chatbots with existing healthcare IT systems and workflows can be complex and costly, requiring interoperability standards and coordination with healthcare providers.

- **Training and Maintenance:** Continuous training and updating of chatbot algorithms and databases are necessary to keep up with medical advancements and ensure accurate information dissemination.

- **Patient Resistance and Adaptation:** Some patients may be resistant to adopting new technologies like chatbots, preferring traditional methods of healthcare delivery, which can hinder widespread acceptance and usage.

CHAPTER 3
SYSTEM SPECIFICATION

3.1 SOFTWARE REQUIREMENT SPECIFICATION

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description as functional representation of system behavior, an indication of performance requirements and design constraints, appropriate validation criteria.

3.2 SYSTEM REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

Processor : Intel Core i3
Memory: 2GB RAM
Resolution : 1024*768 minimum display resolution
Connectivity : WIFI , Ethernet

3.2.2 SOFTWARE REQUIREMENTS

Software Tool : VSCode: Intel i3 9100f
Operating System : Windows 7 , linux
Packages Used : Tkinter Gui, Python, Numpy, Pandas, Machine Learning, Translation

CHAPTER 4
SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM

In fact, the adaptation process has to be automatic and dynamic to free the users with disabilities from the UI change control.

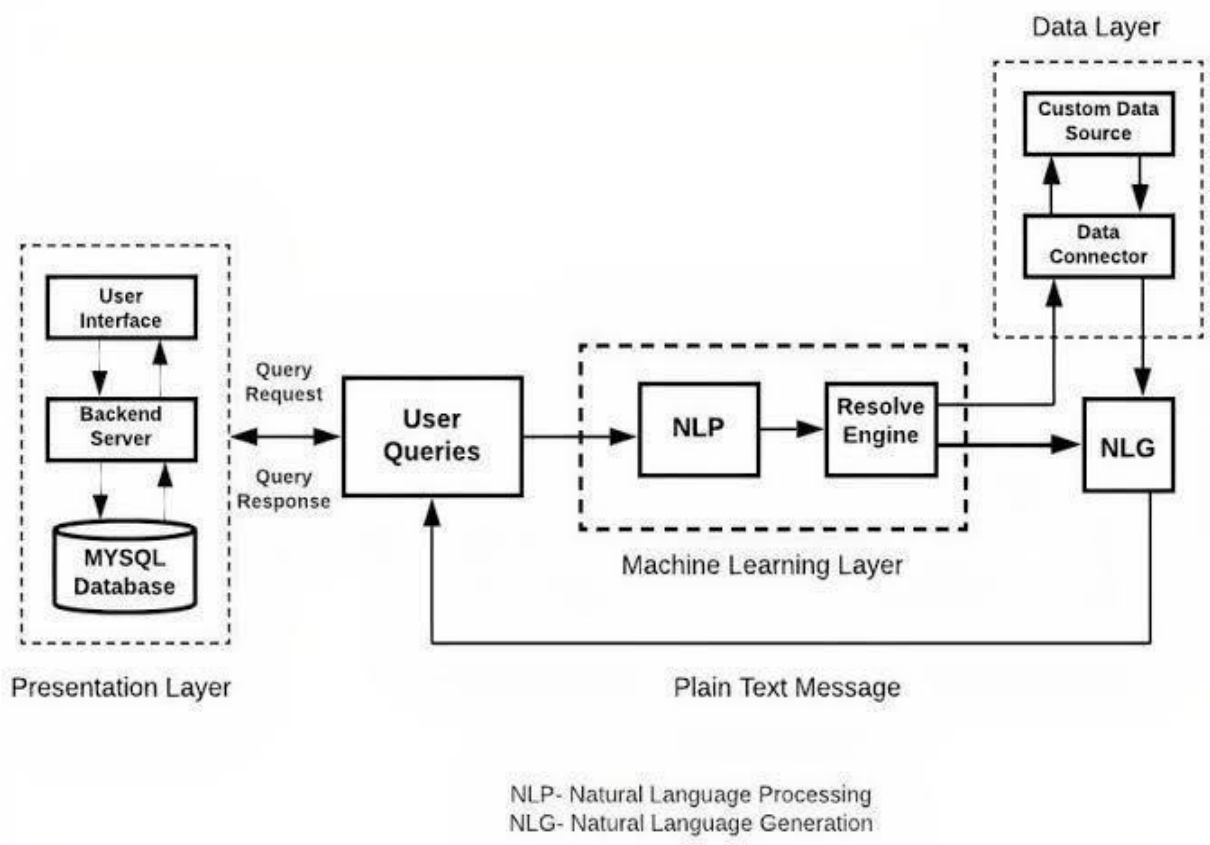

FIG 4.1 ARCHITECTURE DIAGRAM

The architecture of the healthcare chatbot (Shown in Fig 4.1) includes several key layers: the **User Interface** layer, featuring web and mobile apps alongside chat interfaces like WhatsApp and Messenger for user interaction; the **Chatbot Engine** layer, which handles Natural Language Processing (NLP), Dialogue Management, and Response Generation; the **Backend Services** layer, encompassing an API Gateway, User Management, a Medical Database, a Scheduling System, and secure Electronic Health Records (EHR/EMR); the **External Integrations** layer, which connects to third-party APIs and healthcare provider systems; the **Analytics and Monitoring** layer, responsible for logging, monitoring, and analytics dashboards; and the **Security** layer, ensuring data encryption and compliance with healthcare regulations such as HIPAA and GDPR. This architecture is designed to be robust, scalable, and secure, facilitating efficient and compliant healthcare interactions.

4.2 USECASE DIAGRAM

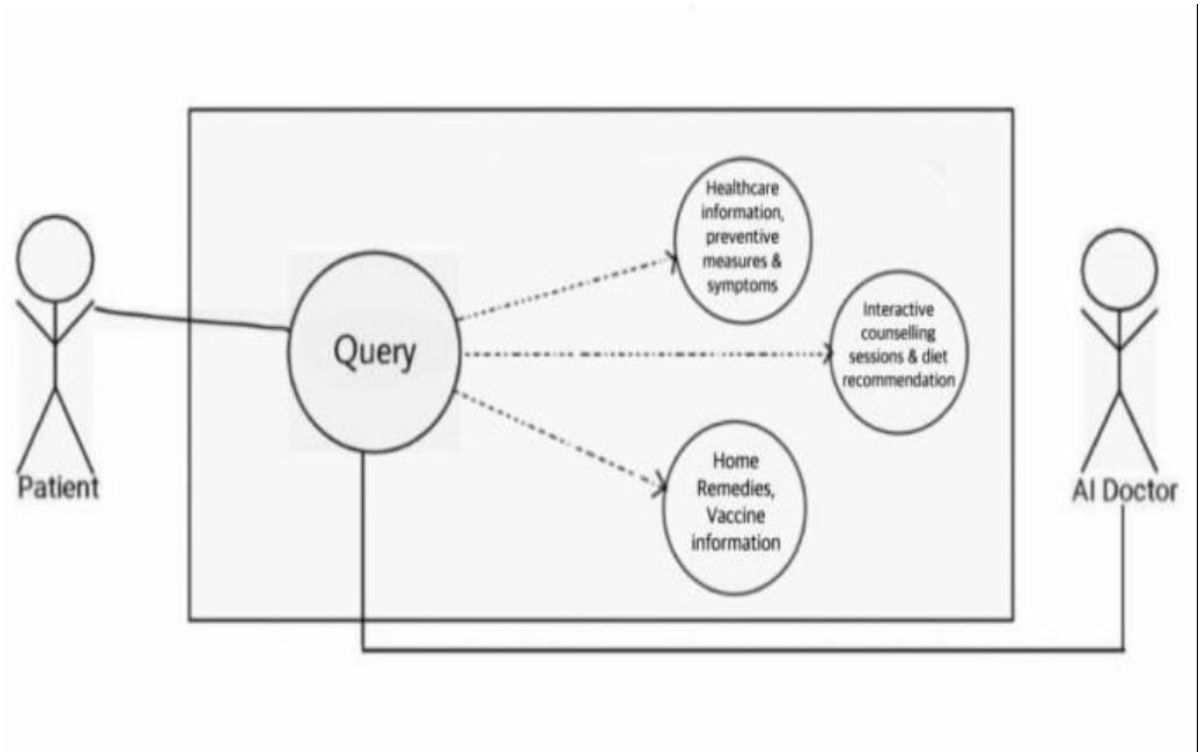

FIG 4.2 USE CASE DIAGRAM

The use case diagram for the healthcare chatbot (Shown in Fig 4.2) outlines key interactions between users and the system. Primary actors include **Patients**, **Healthcare Providers**, and **Administrators**. **Patients** can interact with the chatbot for tasks such as **Symptom Checking**, **Appointment Scheduling**, **Medication Reminders**, **Accessing Medical Records**, and **Health Advice**. **Healthcare Providers** can use the chatbot for **Appointment Management**, **Patient Monitoring**, and **Consultation Requests**. **Administrators** are responsible for **User Management**, **System Monitoring**, and **Content Updates**. This diagram highlights the comprehensive functionalities provided by the healthcare chatbot, ensuring a streamlined and efficient healthcare experience for all users.

4.3 SEQUENCE DIAGRAM

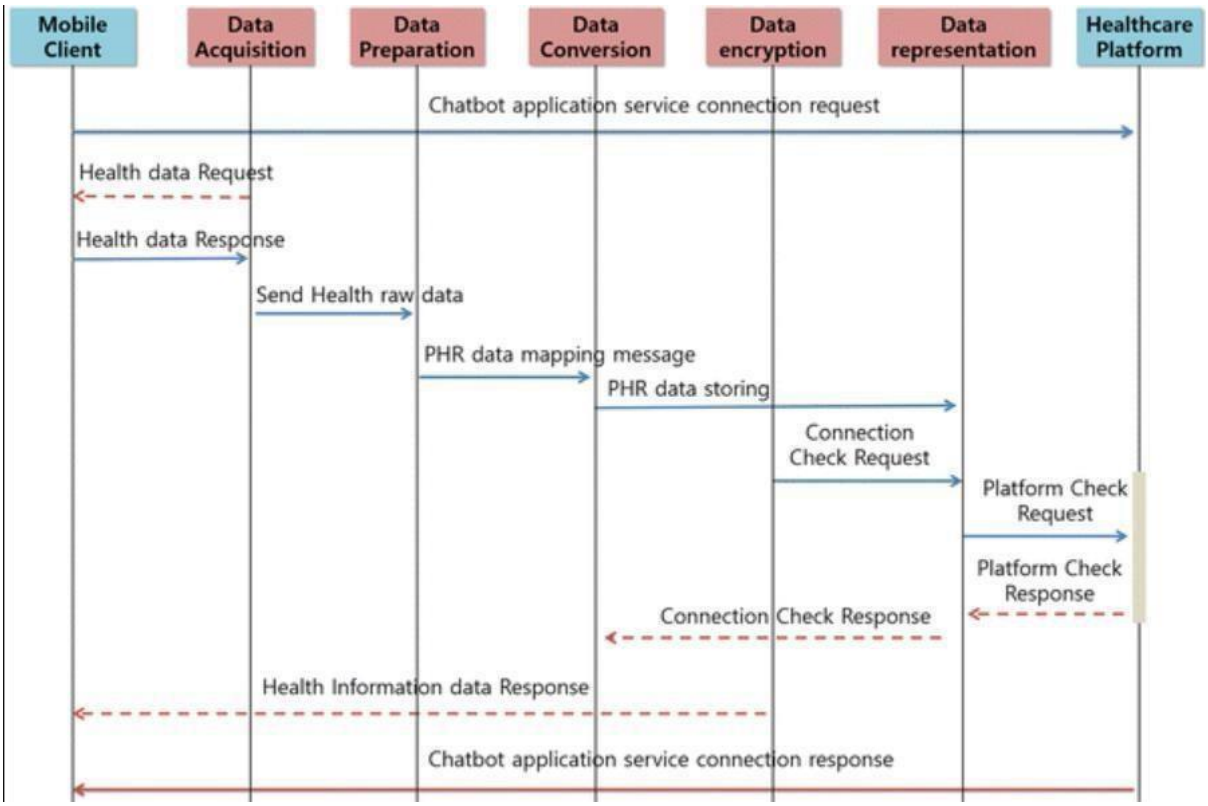

FIG 4.3 SEQUENCE DIAGRAM

The sequence diagram for the healthcare chatbot (Shown in Fig 4.3) illustrates the interaction flow between the user and the system. When a **Patient** initiates a conversation, the **Chat Interface** captures the input and sends it to the **Chatbot Engine**. The engine processes the input using **Natural Language Processing (NLP)** to understand the query and passes it to the **Dialogue Manager** to determine the appropriate response. If the request involves accessing medical data, the Dialogue Manager interacts with the **Backend Services**, including the **Medical Database** and **Patient Records**. For tasks like **Appointment Scheduling**, the Dialogue Manager communicates with the **Scheduling System**. Once the required information is retrieved, the response is generated and sent back through the Chatbot Engine to the Chat Interface, providing the **Patient** with the desired information or action. This sequence ensures a smooth and efficient interaction, covering typical use cases like symptom checking, scheduling appointments, and accessing medical records.

4.4 ARCHITECTURAL FLOW DIAGRAM

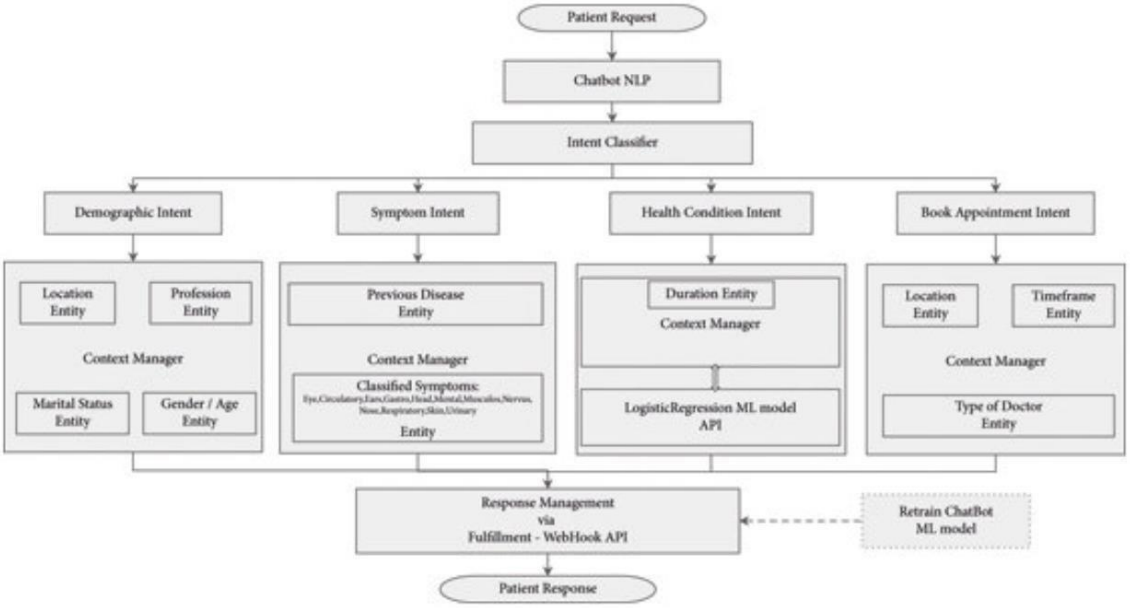

FIG 4.4 ARCHITECTURAL FLOW DIAGRAM

The architectural flow diagram for the healthcare chatbot (Shown in Fig 4.4)demonstrates the data flow and interactions between its components. When a **Patient** interacts with the chatbot via the **User Interface** (web, mobile app, or chat platform), the input is sent to the **Chatbot Engine**, which comprises **Natural Language Processing (NLP)** and **Dialogue Management**. The NLP module processes the input to understand the user query, while the Dialogue Manager decides the next steps. If the query requires accessing medical data or scheduling, the Dialogue Manager communicates with **Backend Services** such as the **Medical Database**, **Patient Records**, and **Scheduling System** via the **API Gateway**. Additionally, the system integrates with external services through the **External Integrations** layer, connecting to third-party APIs and healthcare provider systems. Throughout this process, the **Analytics and Monitoring** layer tracks interactions and system performance, while the **Security** layer ensures data privacy and compliance with regulations like HIPAA and GDPR. The processed response is then sent back through the Chatbot Engine to the User Interface, providing the patient with the necessary information or action.

CHAPTER 5
MODULES

5.1 MODULE LIST

1. Tkinter GUI Building
2. Health Care Model Training
3. Model Integration With GUI
4. Translation Model Integeration

5.1.1 TKINTER GUI BUILDING

In the healthcare chatbot project, Tkinter is utilized for building a graphical user interface (GUI) (Shown in Fig 5.1.1) that enhances user interaction and accessibility. Tkinter, as Python's standard GUI library, allows for the creation of intuitive and responsive desktop applications. It provides tools to design user-friendly interfaces with elements such as text boxes for inputting symptoms or queries, buttons for submitting requests or navigating through options, and display areas for presenting chatbot responses or medical information. This GUI facilitates seamless communication between patients and the chatbot, supporting functionalities like symptom checking, appointment scheduling, and accessing medical records. Tkinter's straightforward implementation and versatility contribute to an efficient and user-centric healthcare chatbot experience, ensuring ease of use and accessibility for all users.

```
from tkinter import *
import time
import tkinter.messagebox
```

```
class ChatInterface(Frame):

    tabnine: test | explain | document | ask
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
        self.tl_bg = "#EEEEEE"
        self.tl_bg2 = "#EEEEEE"
        self.tl_fg = "#000000"
        self.font = "Verdana 10"
        self.current_language = "en"

        menu = Menu(self.master)
        self.master.config(menu=menu, bd=5)
        # Menu bar

        # File
        file = Menu(menu, tearoff=0)
        menu.add_cascade(label="File", menu=file)
        # file.add_command(label="Save Chat Log", command=self.save_chat)
        file.add_command(label="Clear Chat", command=self.clear_chat)
        #  file.add_separator()
        file.add_command(label="Exit", command=self.chatexit)

        # Options
        options = Menu(menu, tearoff=0)
        menu.add_cascade(label="Options", menu=options)

        # font
        font = Menu(options, tearoff=0)
        options.add_cascade(label="Font", menu=font)
        font.add_command(label="Default", command=self.font_change_default)
```

```
font.add_command(label="Times", command=self.font_change_times)
font.add_command(label="System", command=self.font_change_system)
font.add_command(label="Helvetica", command=self.font_change_helvetica)
font.add_command(label="Fixedsys", command=self.font_change_fixedsys)

# color theme
color_theme = Menu(options, tearoff=0)
options.add_cascade(label="Color Theme", menu=color_theme)
color_theme.add_command(label="Default", command=self.color_theme_default)
# color_theme.add_command(label="Night",command=self.)
color_theme.add_command(label="Grey", command=self.color_theme_grey)
color_theme.add_command(label="Blue", command=self.color_theme_dark_blue)

color_theme.add_command(label="Torque", command=self.color_theme_turquoise)
color_theme.add_command(label="Hacker", command=self.color_theme_hacker)
# color_theme.add_command(label='Mkbhd',command=self.MKBHD)

language = Menu(options, tearoff=0)
options.add_cascade(label="Language", menu=l
language.add_command(label="Tamil", command=
language.add_command(label="Hindi", command=self.change_language_to_hindi)
language.add_command(label="Malayalam", command=self.change_language_to_malayalam)
language.add_command(label="Telugu", command=self.change_language_to_telugu)
language.add_command(label="English", command=self.change_language_to_eng)

help_option = Menu(menu, tearoff=0)
menu.add_cascade(label="Help", menu=help_option)
# help_option.add_command(label="Features", command=self.features_msg)
help_option.add_command(label="About MedBot", command=self.msg)
help_option.add_command(label="Developers", command=self.about)
```

```
(parameter) self: Self@ChatInterface
```

```
# contains messages
self.text_box = Text(self.text_frame, yscrollcommand=self.text_box_scrollbar.set, state=DISABLED,
                     bd=1, padx=6, pady=6, spacing3=8, wrap=WORD, bg=None, font="Verdana 10", relief=GROOVE,
                     width=10, height=1)
self.text_box.pack(expand=True, fill=BOTH)
self.text_box_scrollbar.config(command=self.text_box.yview)

# frame containing user entry field
self.entry_frame = Frame(self.master, bd=1)
self.entry_frame.pack(side=LEFT, fill=BOTH, expand=True)

# entry field
self.entry_field = Entry(self.entry_frame, bd=1, justify=LEFT)
self.entry_field.pack(fill=X, padx=6, pady=6, ipady=3)
# self.users_message = self.entry_field.get()

# frame containing send button and emoji button
self.send_button_frame = Frame(self.master, bd=0)
self.send_button_frame.pack(fill=BOTH)

# send button
self.send_button = Button(self.send_button_frame, text="Send", width=5, relief=GROOVE, bg='white',
                          bd=1, command=lambda: self.send_message_insert(None), activebackground="#FFFFFF",
                          activeforeground="#000000")
self.send_button.pack(side=LEFT, ipady=8, expand=True)
self.master.bind("<Return>", self.send_message_insert)
self.last_sent_label(date="No messages sent.")      Amber Kakkar, 3 years ago • Added GUI
```

```
root = Tk()

a = ChatInterface(root)
root.geometry(window_size)
root.title("MedBot")
root.iconbitmap('MedBot.jpg')
root.mainloop()
```

FIG 5.1.1 TKINTER GUI BUILDING

5.1.2 HEALTH CARE MODEL TRAINING

The healthcare chatbot's model training involves collecting and preprocessing a diverse dataset of medical dialogues and queries. Using deep learning architectures like BERT or LSTM, the model learns to understand and respond to user intents such as symptom assessment and treatment recommendations(Shown in Fig 5.1.2). Supervised learning techniques enable the model to predict responses based on labeled examples. Validation ensures accuracy and performance optimization, refining the model's ability to handle medical nuances. This process equips the chatbot to provide reliable and personalized healthcare advice, enhancing user engagement and satisfaction.

```
import random      Amber Kakkar, 3 years ago • First Commit
import json
import pickle
import numpy as np
import pandas as pd

import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import SGD

lemmatizer=WordNetLemmatizer()

with open('intents.json') as json_file:
    intents = json.load(json_file)

#print(intents)

words=[]
classes=[]
documents=[]
ignore_letters=['?','!','.',',']

for intent in intents['intents']:
  for pattern in intent['patterns']:
    word_list=nltk.word_tokenize(pattern)
    words.extend(word_list)
    documents.append((word_list,intent['tag']))
```

```
    word_list=nltk.word_tokenize(pattern)
    words.extend(word_list)
    documents.append((word_list,intent['tag']))      Amber Kakkar, 3 years ago • First Commit
    if intent['tag'] not in classes:
      classes.append(intent['tag'])


words =[lemmatizer.lemmatize(word) for word in words if word not in ignore_letters]
words = sorted(set(words))
classes=sorted(set(classes))
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))



training=[]
output_empty=[0]*len(classes)

for document in documents:
  bag=[]
  word_patterns=document[0]
  words = [lemmatizer.lemmatize(word) for word in words if word and word not in ignore_letters]
  for word in words:
    bag.append(1) if word in word_patterns else bag.append(0)

  output_row=list(output_empty)
  output_row[classes.index(document[1])]=1
  training.append([bag,output_row])

random.shuffle(training)
training=np.array(training)
```

```
train_x=list(training[:,0])
train_y=list(training[:,1])
model=Sequential()
model.add(Dense(128,input_shape=(len(train_x[0]),),activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]),activation='softmax'))

sgd=SGD(learning_rate=0.01,decay=1e-6,momentum=0.9,nesterov=True)
model.compile(loss='categorical_crossentropy',optimizer=sgd,metrics=['accuracy'])
hist = model.fit(np.array(train_x),np.array(train_y),epochs=200,batch_size=5,verbose=1)
model.save('chatbotmodel.h5', hist)
print('Training Done')
```

FIG 5.1.2 HEALTH CARE MODEL TRAINING

5.1.3 MODEL INTEGRATION WITH GUI

Integrating the healthcare model with the Tkinter GUI involves creating a cohesive interface where (Shown in Fig 5.1.3) users can input medical queries and receive responses seamlessly. User interactions, such as entering symptoms or requesting medical information, trigger the model to process and generate relevant outputs. These outputs, such as diagnosis suggestions or treatment advice, are then displayed within the GUI interface components, ensuring clarity and ease of use. This integration enhances the chatbot's functionality, enabling efficient communication and interaction between patients and healthcare information, fostering a user-friendly and effective healthcare experience.

```
# -*- coding: utf-8 -*-      Amber Kakkar, 3 years ago • First Commit
"""chatbot.py

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1feCkdIYr8e1V5KIzxxooJr5o2bRK4R4p
"""

import nltk
import random
import numpy as np
import json
import pickle
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.models import load_model
lemmatizer=WordNetLemmatizer()

with open('intents.json') as json_file:
    intents = json.load(json_file)

words=pickle.load(open('words.pkl','rb'))
classes=pickle.load(open('classes.pkl','rb'))
model=load_model('chatbotmodel.h5')

tabnine: test | explain | document | ask
def clean_up_sentence(sentence):
  sentence_words=nltk.word_tokenize(sentence)
  sentence_words=[lemmatizer.lemmatize(word) for word in sentence_words]
  return sentence_words
```

```python
def bag_of_words(sentence):
  sentence_words=clean_up_sentence(sentence)
  bag=[0]*len(words)
  for w in sentence_words:
    for i,word in enumerate(words):
      if word == w:
        bag[i]=1
  return np.array(bag)

tabnine: test | explain | document | ask
def predict_class(sentence):
  bow=bag_of_words(sentence)
  res=model.predict(np.array([bow]))[0]
  ERROR_THRESHOLD=0.25
  results=[[i,r] for i,r in enumerate(res) if r> ERROR_THRESHOLD]

  results.sort(key=lambda x:x[1],reverse=True)
  return_list=[]
  for r in results:
    return_list.append({'intent': classes[r[0]],'probability':str(r[1])})
  return return_list

tabnine: test | explain | document | ask
def get_response(intents_list,intents_json):
  tag=intents_list[0]['intent']
  list_of_intents=intents_json['intents']
  for i in list_of_intents:
    if i['tag']==tag:
      result=random.choice(i['responses'])
      break
  return result
```

```python
print("GO! BOT IS RUNNING")

# while True:
#   message=input("")
#   ints=predict_class(message)
#   res=get_response(ints,intents)
#   print(res)
```

FIG 5.1.3 MODEL INTEGRATION WITH GUI

5.1.4 TRANSLATION MODEL INTEGRATION

Integrating a translation model into the healthcare chatbot allows for seamless communication (Show in Fig 5.1.4) with users across different languages. Using advanced machine translation techniques or APIs like Google Translate, user inputs in various languages are translated into the chatbot's primary language for processing. This integration ensures that the chatbot can understand and respond to queries effectively, regardless of the user's language preference. Responses generated by the chatbot can also be translated back into the user's language before being presented, facilitating clear and accurate communication. By incorporatingtranslation capabilities, the chatbot expands its accessibility and usability, enhancing engagement and usability for a diverse patient population.

```
tamil_translator = Translator(to_lang="ta")
hindi_translator = Translator(to_lang="hi")
malayalam_translator = Translator(to_lang="ml")
telugu_translator = Translator(to_lang="te")
```

```
def send_message_insert(self, message):
    user_input = self.entry_field.get()
    pr1 = user_input
    self.text_box.configure(state=NORMAL)
    if self.current_language == "ta":
        translated = tamil_translator.translate(pr1)
        self.text_box.insert(END, "Human : " + translated + "\n")
    elif self.current_language == "hi":
        translated = hindi_translator.translate(pr1)
        self.text_box.insert(END, "Human : " + translated + "\n")
    elif self.current_language == "ml":
        translated = malayalam_translator.translate(pr1)
        self.text_box.insert(END, "Human : " + translated + "\n")
    elif self.current_language == "te":
        translated = telugu_translator.translate(pr1)
        self.text_box.insert(END, "Human : " + translated + "\n")
    else:
        self.text_box.insert(END, "Human : " + pr1 + "\n")
    self.text_box.configure(state=DISABLED)
    self.text_box.see(END)
    ob = chat(user_input)
    pr = ob
    self.text_box.configure(state=NORMAL)
    if self.current_language == "ta":
        translated = tamil_translator.translate(pr)
        self.text_box.insert(END, "MedBot : " + translated + "\n")
    elif self.current_language == "hi":
```

```
        if self.current_language == "ta":
            translated = tamil_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        elif self.current_language == "hi":
            translated = hindi_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        elif self.current_language == "ml":
            translated = malayalam_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        elif self.current_language == "te":
            translated = telugu_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        else:
            self.text_box.insert(END, "MedBot : " + pr + "\n")
        self.text_box.configure(state=DISABLED)
        self.text_box.see(END)
        self.last_sent_label(str(time.strftime("Last message sent: " + '%B %d, %Y' + ' at ' + '%I:%M %p')))
        self.entry_field.delete(0, END)
```

```
    def change_language_to_tamil(self):
        self.current_language = "ta"
    tabnine: test | explain | document | ask
    def change_language_to_hindi(self):
        self.current_language = "hi"
    tabnine: test | explain | document | ask
    def change_language_to_malayalam(self):
        self.current_language = "ml"
    tabnine: test | explain | document | ask
    def change_language_to_telugu(self):
        self.current_language = "te"
    tabnine: test | explain | document | ask
    def change_language_to_eng(self):
        self.current_language = "en"

tabnine: test | explain | document | ask
def chat(msg):
    ints = predict_class(msg)
    return get_response(ints, intents)      You, 4 days ago • Uncommitted changes
```

FIG 5.1.4 TRANSLATIN MODEL INTEGRATION

CHAPTER 6
ALGORITHM USED

6.1 Natural Language Processing

Natural Language Processing (NLP) is a critical field of artificial intelligence focused on enabling computers to understand, interpret, and generate human language. It involves tasks such as tokenization, part-of-speech tagging, named entity recognition, syntax analysis, semantic understanding, text classification, machine translation, question answering, and text generation. NLP employs statistical methods, deep learning, and rule-based systems, utilizing techniques like word embeddings and attention mechanisms. Applications span virtual assistants, sentiment analysis, healthcare, finance, and more, yet challenges include ambiguity, data scarcity, bias, and handling multilingualism. Future directions aim for multimodal integration, explainable AI, and continual learning to enhance NLP's capabilities and applications across diverse domains.

6.2 Stochastic Gradient Descent Algorithm

Stochastic Gradient Descent (SGD) is a popular optimization algorithm in machine learning used to minimize the loss function during model training. Unlike batch gradient descent, which computes gradients using the entire dataset, SGD updates model parameters iteratively based on smaller subsets (mini-batches) or individual data points. This approach makes SGD faster and more scalable for large datasets, as it processes data in smaller chunks. During each iteration, SGD randomly selects mini-batches, computes gradients for the loss function with respect to the current parameters, and updates the parameters in the opposite direction of the gradient to minimize the loss. The learning rate determines the size of these updates, influencing the speed and stability of convergence. SGD's stochastic nature introduces noise in gradient estimation, which can help the algorithm escape local minima and improve generalization on unseen data. However, this randomness can also lead to fluctuations in training, requiring careful tuning of the learning rate and monitoring of training progress. Variants like mini-batch SGD and momentum SGD further refine SGD's performance by balancing efficiency and stability in gradient descent. Overall, SGD is foundational in training neural networks and other machine learning models, offering efficiency and scalability essential for modern data-driven applications.

CHAPTER 7
CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The development and implementation of multilingual healthcare chatbots represent a significant advancement in the quest to make healthcare more accessible, efficient, and inclusive. By leveraging artificial intelligence and natural language processing, these chatbots can provide critical healthcare services across various languages, ensuring that linguistic barriers do not impede access to quality healthcare.

The functionalities of these chatbots, including symptom checking, medical information dissemination, appointment scheduling, medication reminders, chronic disease management, and mental health support, offer comprehensive support to patients worldwide. The integration of multilingual capabilities further enhances these functionalities, making healthcare information and services available to a broader and more diverse population.

The benefits of multilingual healthcare chatbots are manifold. They offer 24/7 access to healthcare information, reduce the burden on healthcare providers by automating routine tasks, and improve patient engagement by providing personalized and timely care. Moreover, they have the potential to significantly lower healthcare costs by optimizing resource allocation and reducing unnecessary visits to healthcare facilities.

However, the deployment of these chatbots is not without challenges. Ensuring the accuracy and reliability of medical information, maintaining patient privacy and data security, integrating with existing healthcare systems, and providing high-quality language support are critical areas that require ongoing attention and improvement. Addressing these challenges is essential to build trust and ensure the effective and safe use of healthcare chatbots.

In conclusion, multilingual healthcare chatbots hold the promise of transforming healthcare delivery by making it more accessible and tailored to individual needs, regardless of language barriers. As technology continues to evolve, ongoing research and development are crucial to overcoming existing challenges and enhancing the capabilities of these chatbots. By doing so, healthcare systems can move closer to achieving the goal of equitable and effective healthcare for all, leveraging innovative, language-inclusive technologies to bridge gaps and improve health outcomes globally.

7.2 FUTURE ENHANCEMENT

Future enhancements for multilingual healthcare chatbots focus on expanding their capabilities to provide even more comprehensive, personalized, and efficient healthcare services. Advanced AI and machine learning algorithms will improve the accuracy of diagnostics and personalize care based on a user's medical history and lifestyle. Enhanced natural language processing will support more languages and dialects, understand context better, and handle cultural nuances. Integrating voice recognition will allow voice-activated interactions, making the service accessible to those with literacy challenges or disabilities. Seamless integration with telehealth services will enable virtual consultations and remote

patient monitoring. Additionally, robust data security measures will be implemented to protect sensitive information, and interoperability with electronic health records will ensure real-time data exchange and comprehensive patient profiles. Ethical considerations and bias mitigation in AI will be prioritized to ensure equitable healthcare recommendations. These advancements will make multilingual healthcare chatbots indispensable tools in delivering inclusive, reliable, and high-quality healthcare globally.

APPENDICES

APPENDIX-1

SCREENSHOTS

MAIN PAGE:

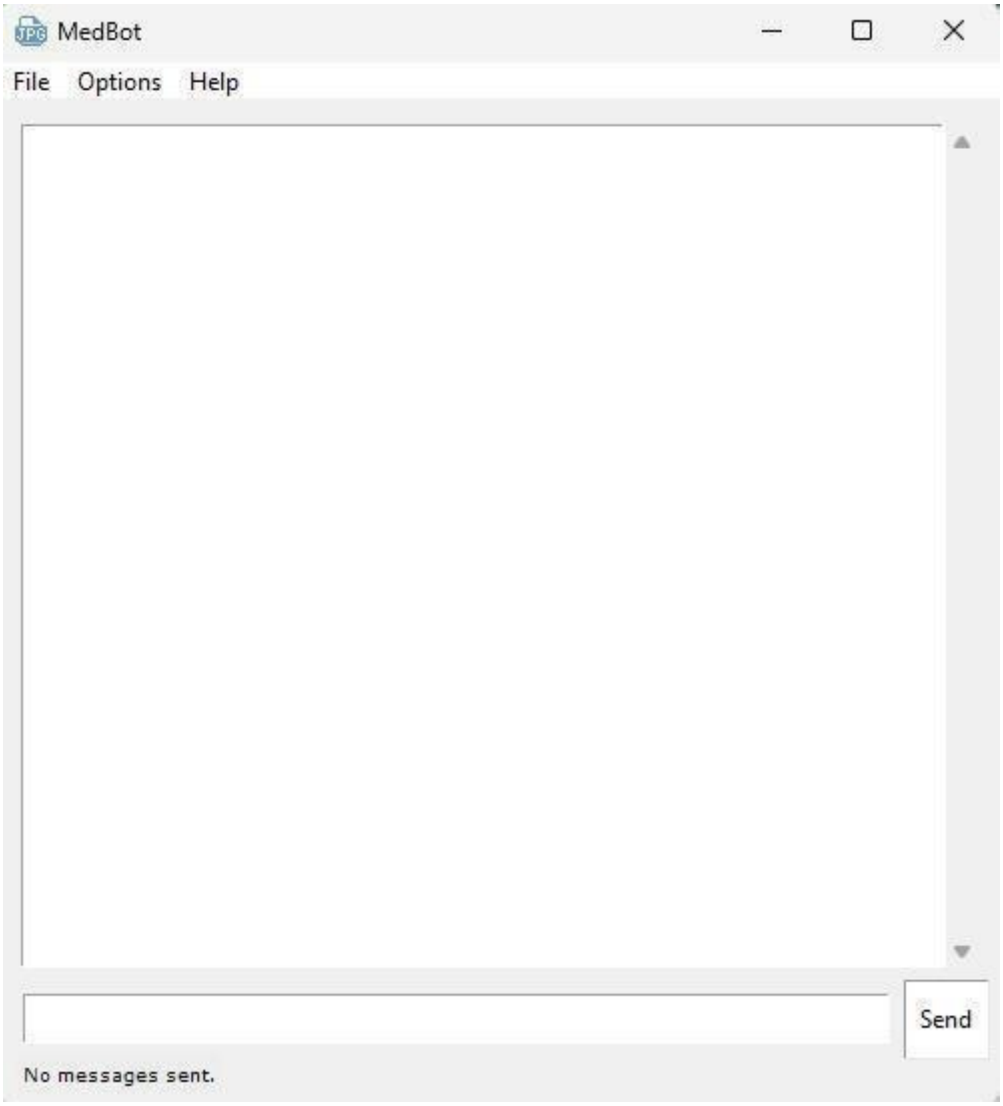

FIG7.2.1 MAIN PAGE

LANGUAGE INFORMATION:

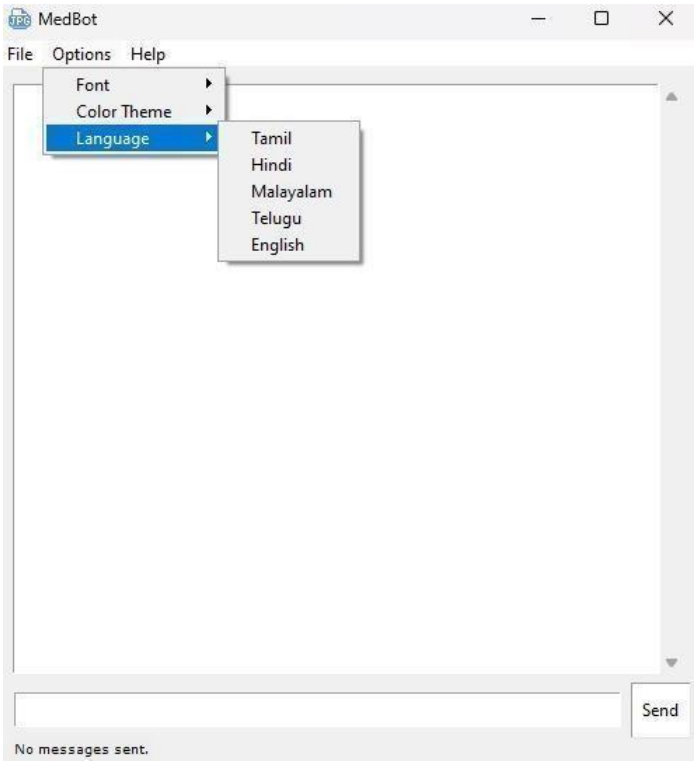

FIG 7.2.2 LANGUAGE INFORMATION

GENERAL CHAT:

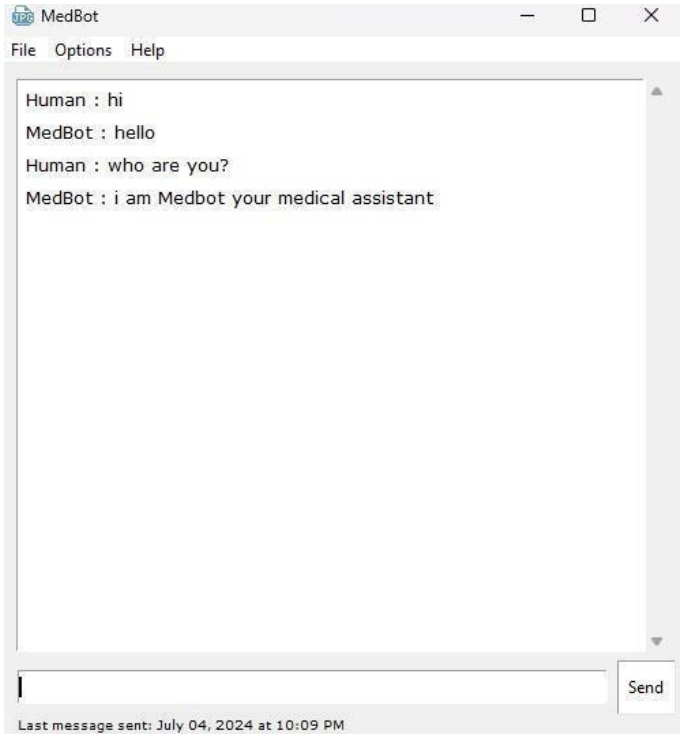

FIG 7.2.3 GENERAL CHAT

QUERIES IN ENGLISH:

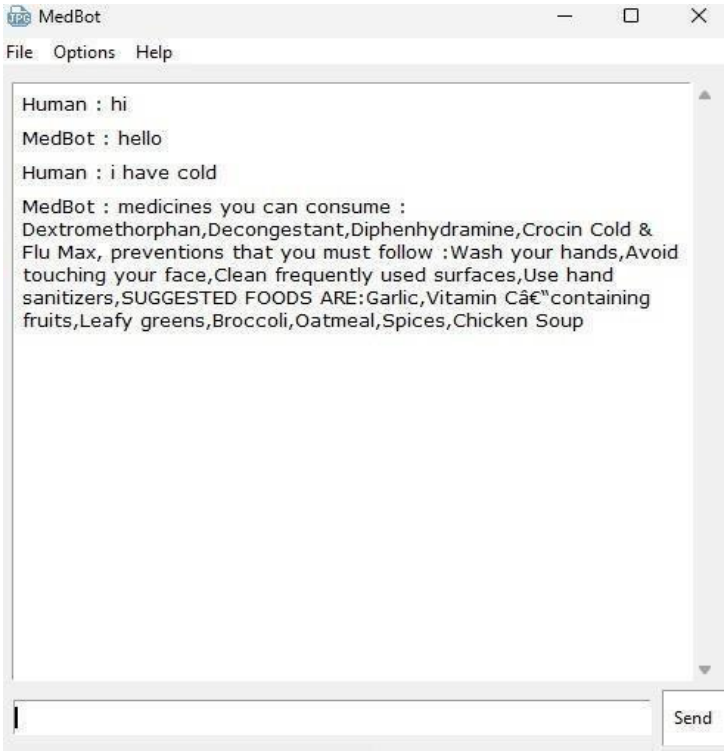

FIG 7.2.4 QUERIES IN ENGLISH

QUERIES IN TAMIL :

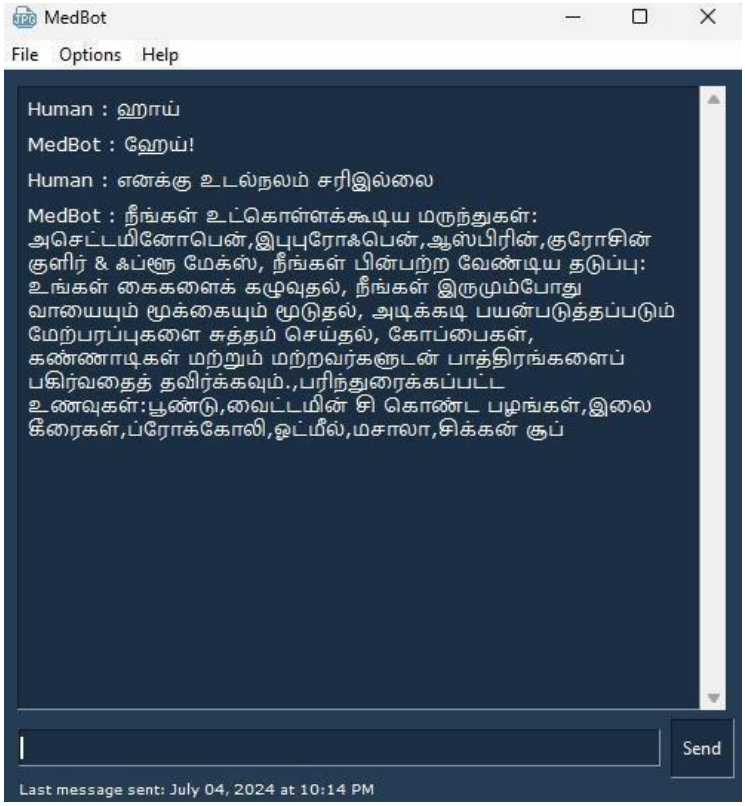

FIG 7.2.5 QUERIES IN TAMIL

QUERIES IN HINDI:

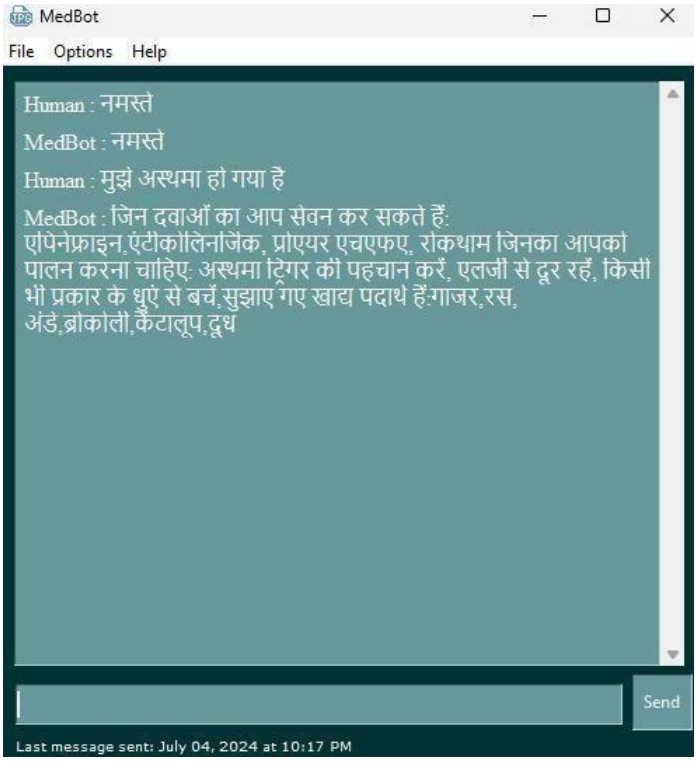

FIG 7.2.6 QUERIES IN HINDI

QUERIES IN MALAYALAM:

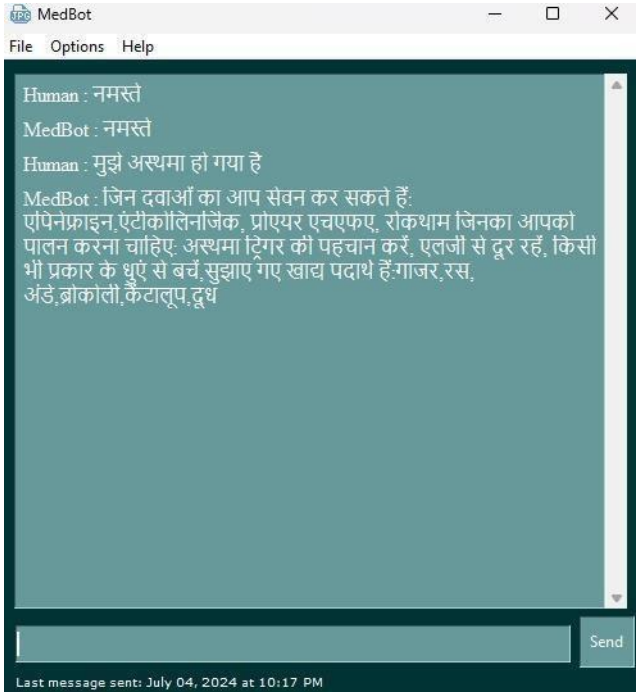

FIG 7.2.7 QUERIES IN MALAYALAM

QUERIES IN TELUGU:

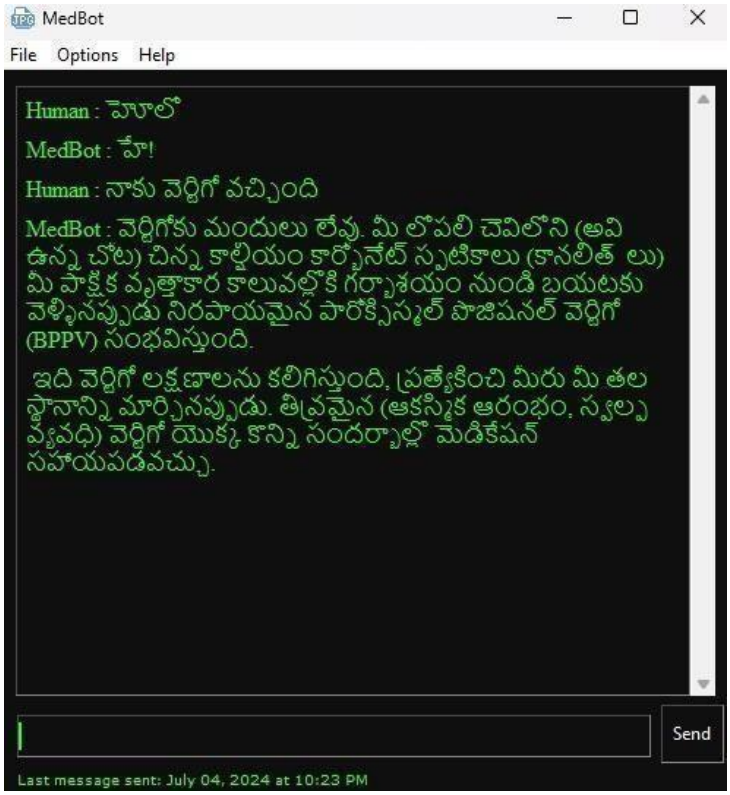

FIG 7.2.8 QUERIES IN TELUGU

BACKGROUND PROCESSING:

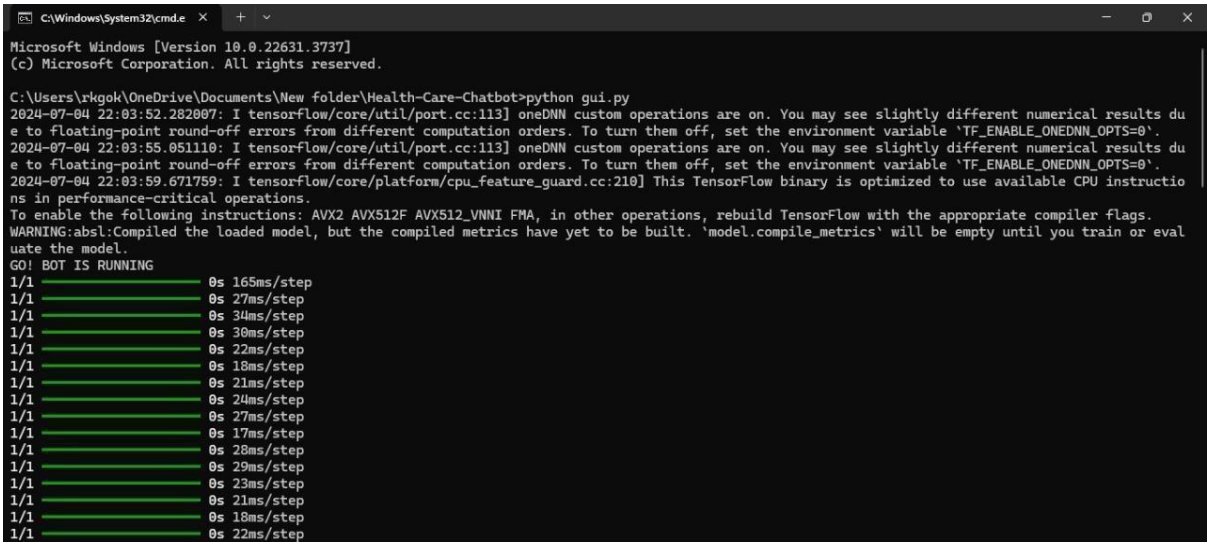

FIG 7.2.9 BACKGROUND PROCESSING

APPENDIX-2 IMPLEMENTATION CODE:

GUI.PY

```
from tkinter import *
import time
import  tkinter.messagebox
from translate import Translator
from chatbot_py import predict_class, get_response, intents
saved_username = ["You"]
window_size = "500x500"

tamil_translator = Translator(to_lang="ta")
hindi_translator = Translator(to_lang="hi")
malayalam_translator = Translator(to_lang="ml")
telugu_translator = Translator(to_lang="te")


class ChatInterface(Frame):

    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master
        self.tl_bg = "#EEEEEE"
        self.tl_bg2 = "#EEEEEE"
        self.tl_fg = "#000000"
        self.font = "Verdana 10"
        self.current_language = "en"

        menu = Menu(self.master)
        self.master.config(menu=menu, bd=5)
        # Menu bar

        # File
        file = Menu(menu, tearoff=0)
        menu.add_cascade(label="File", menu=file)
        # file.add_command(label="Save Chat Log", command=self.save_chat)
        file.add_command(label="Clear Chat", command=self.clear_chat)
        #  file.add_separator()
        file.add_command(label="Exit", command=self.chatexit)
```

```
# Options
options = Menu(menu, tearoff=0)
menu.add_cascade(label="Options", menu=options)

# font
font = Menu(options, tearoff=0)
options.add_cascade(label="Font", menu=font)
font.add_command(label="Default", command=self.font_change_default)
font.add_command(label="Times", command=self.font_change_times)
font.add_command(label="System", command=self.font_change_system)
font.add_command(label="Helvetica", command=self.font_change_helvetica)
font.add_command(label="Fixedsys", command=self.font_change_fixedsys)

# color theme
color_theme = Menu(options, tearoff=0)
options.add_cascade(label="Color Theme", menu=color_theme)
color_theme.add_command(label="Default", command=self.color_theme_default)
# color_theme.add_command(label="Night",command=self.)
color_theme.add_command(label="Grey", command=self.color_theme_grey)
color_theme.add_command(label="Blue", command=self.color_theme_dark_blue)

color_theme.add_command(label="Torque", command=self.color_theme_turquoise)
color_theme.add_command(label="Hacker", command=self.color_theme_hacker)
# color_theme.add_command(label='Mkbhd',command=self.MKBHD)

language = Menu(options, tearoff=0)
options.add_cascade(label="Language", menu=language)
language.add_command(label="Tamil", command=self.change_language_to_tamil)
language.add_command(label="Hindi", command=self.change_language_to_hindi)
language.add_command(label="Malayalam",
command=self.change_language_to_malayalam)
language.add_command(label="Telugu", command=self.change_language_to_telugu)
language.add_command(label="English", command=self.change_language_to_eng)

help_option = Menu(menu, tearoff=0)
menu.add_cascade(label="Help", menu=help_option)
# help_option.add_command(label="Features", command=self.features_msg)
help_option.add_command(label="About MedBot", command=self.msg)
help_option.add_command(label="Developers", command=self.about)

self.text_frame = Frame(self.master, bd=6)
self.text_frame.pack(expand=True, fill=BOTH)

# scrollbar for text box
```

```
        self.text_box_scrollbar = Scrollbar(self.text_frame, bd=0)
        self.text_box_scrollbar.pack(fill=Y, side=RIGHT)

        # contains messages
        self.text_box = Text(self.text_frame, yscrollcommand=self.text_box_scrollbar.set,
state=DISABLED,
                             bd=1, padx=6, pady=6, spacing3=8, wrap=WORD, bg=None,
font="Verdana 10", relief=GROOVE,
                             width=10, height=1)
        self.text_box.pack(expand=True, fill=BOTH)
        self.text_box_scrollbar.config(command=self.text_box.yview)

        # frame containing user entry field
        self.entry_frame = Frame(self.master, bd=1)
        self.entry_frame.pack(side=LEFT, fill=BOTH, expand=True)

        # entry field
        self.entry_field = Entry(self.entry_frame, bd=1, justify=LEFT)
        self.entry_field.pack(fill=X, padx=6, pady=6, ipady=3)
        # self.users_message = self.entry_field.get()

        # frame containing send button and emoji button
        self.send_button_frame = Frame(self.master, bd=0)
        self.send_button_frame.pack(fill=BOTH)

        # send button
        self.send_button = Button(self.send_button_frame, text="Send", width=5,
relief=GROOVE, bg='white',
                                  bd=1, command=lambda: self.send_message_insert(None),
activebackground="#FFFFFF",
                                  activeforeground="#000000")
        self.send_button.pack(side=LEFT, ipady=8, expand=True)
        self.master.bind("<Return>", self.send_message_insert)
        self.last_sent_label(date="No messages sent.")

    def last_sent_label(self, date):

        try:
            self.sent_label.destroy()
        except AttributeError:
            pass

        self.sent_label = Label(self.entry_frame, font="Verdana 7", text=date, bg=self.tl_bg2,
fg=self.tl_fg)
```

```
        self.sent_label.pack(side=LEFT, fill=BOTH, padx=3)

    def clear_chat(self):
        self.text_box.config(state=NORMAL)
        self.last_sent_label(date="No messages sent.")
        self.text_box.delete(1.0, END)
        self.text_box.delete(1.0, END)
        self.text_box.config(state=DISABLED)

    def chatexit(self):
        exit()

    def msg(self):
        tkinter.messagebox.showinfo("MedBot v1.0",
                                    'MedBot is a chatbot for answering health related queries\nIt is based
on retrival-based NLP using pythons NLTK tool-kit module\nGUI is based on Tkinter\nIt can
answer questions regarding users health status')

    def about(self):
        tkinter.messagebox.showinfo("MedBot Developers",
                                    "1.Ragavendhra \n2.Gokulavasan\n3.Lokesh")

    def send_message_insert(self, message):
        user_input  = self.entry_field.get()
        pr1 = user_input
        self.text_box.configure(state=NORMAL)
        if self.current_language == "ta":
            translated = tamil_translator.translate(pr1)
            self.text_box.insert(END, "Human : " + translated + "\n")
        elif self.current_language == "hi":
            translated = hindi_translator.translate(pr1)
            self.text_box.insert(END, "Human : " + translated + "\n")
        elif self.current_language == "ml":
            translated = malayalam_translator.translate(pr1)
            self.text_box.insert(END, "Human : " + translated + "\n")
        elif self.current_language == "te":
            translated = telugu_translator.translate(pr1)
            self.text_box.insert(END, "Human : " + translated + "\n")
        else:
            self.text_box.insert(END, "Human : " + pr1 + "\n")
        self.text_box.configure(state=DISABLED)
        self.text_box.see(END)
        ob = chat(user_input)
        pr = ob
```

```
        self.text_box.configure(state=NORMAL)
        if self.current_language == "ta":
            translated = tamil_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        elif self.current_language == "hi":
            translated = hindi_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        elif self.current_language == "ml":
            translated = malayalam_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        elif self.current_language == "te":
            translated = telugu_translator.translate(pr)
            self.text_box.insert(END, "MedBot : " + translated + "\n")
        else:
            self.text_box.insert(END, "MedBot : " + pr + "\n")
        self.text_box.configure(state=DISABLED)
        self.text_box.see(END)
        self.last_sent_label(str(time.strftime("Last message sent: " + '%B %d, %Y' + ' at ' +
'%I:%M %p')))
        self.entry_field.delete(0, END)

    def font_change_default(self):
        self.text_box.config(font="Verdana 10")
        self.entry_field.config(font="Verdana 10")
        self.font = "Verdana 10"

    def font_change_times(self):
        self.text_box.config(font="Times")
        self.entry_field.config(font="Times")
        self.font = "Times"

    def font_change_system(self):
        self.text_box.config(font="System")
        self.entry_field.config(font="System")
        self.font = "System"

    def font_change_helvetica(self):
        self.text_box.config(font="helvetica 10")
        self.entry_field.config(font="helvetica 10")
        self.font = "helvetica 10"

    def font_change_fixedsys(self):
        self.text_box.config(font="fixedsys")
        self.entry_field.config(font="fixedsys")
```

```
    self.font = "fixedsys"

def color_theme_default(self):
    self.master.config(bg="#EEEEEE")
    self.text_frame.config(bg="#EEEEEE")
    self.entry_frame.config(bg="#EEEEEE")
    self.text_box.config(bg="#FFFFFF", fg="#000000")
    self.entry_field.config(bg="#FFFFFF", fg="#000000", insertbackground="#000000")
    self.send_button_frame.config(bg="#EEEEEE")
    self.send_button.config(bg="#FFFFFF", fg="#000000", activebackground="#FFFFFF",
    activeforeground="#000000")
    self.sent_label.config(bg="#EEEEEE", fg="#000000")

    self.tl_bg = "#FFFFFF"
    self.tl_bg2 = "#EEEEEE"
    self.tl_fg = "#000000"

# Dark
def color_theme_dark(self):
    self.master.config(bg="#2a2b2d")
    self.text_frame.config(bg="#2a2b2d")
    self.text_box.config(bg="#212121", fg="#FFFFFF")
    self.entry_frame.config(bg="#2a2b2d")
    self.entry_field.config(bg="#212121", fg="#FFFFFF", insertbackground="#FFFFFF")
    self.send_button_frame.config(bg="#2a2b2d")
    self.send_button.config(bg="#212121", fg="#FFFFFF", activebackground="#212121",
    activeforeground="#FFFFFF")
    self.sent_label.config(bg="#2a2b2d", fg="#FFFFFF")

    self.tl_bg = "#212121"
    self.tl_bg2 = "#2a2b2d"
    self.tl_fg = "#FFFFFF"

# Grey
def color_theme_grey(self):
    self.master.config(bg="#444444")
    self.text_frame.config(bg="#444444")
    self.text_box.config(bg="#4f4f4f", fg="#ffffff")
    self.entry_frame.config(bg="#444444")
    self.entry_field.config(bg="#4f4f4f", fg="#ffffff", insertbackground="#ffffff")
    self.send_button_frame.config(bg="#444444")
    self.send_button.config(bg="#4f4f4f", fg="#ffffff", activebackground="#4f4f4f",
    activeforeground="#ffffff")
    self.sent_label.config(bg="#444444", fg="#ffffff")
```

```
    self.tl_bg = "#4f4f4f"
    self.tl_bg2 = "#444444"
    self.tl_fg = "#ffffff"

def color_theme_turquoise(self):
    self.master.config(bg="#003333")
    self.text_frame.config(bg="#003333")
    self.text_box.config(bg="#669999", fg="#FFFFFF")
    self.entry_frame.config(bg="#003333")
    self.entry_field.config(bg="#669999", fg="#FFFFFF", insertbackground="#FFFFFF")
    self.send_button_frame.config(bg="#003333")
    self.send_button.config(bg="#669999", fg="#FFFFFF", activebackground="#669999",
    activeforeground="#FFFFFF")
    self.sent_label.config(bg="#003333", fg="#FFFFFF")

    self.tl_bg = "#669999"
    self.tl_bg2 = "#003333"
    self.tl_fg = "#FFFFFF"

    # Blue

def color_theme_dark_blue(self):
    self.master.config(bg="#263b54")
    self.text_frame.config(bg="#263b54")
    self.text_box.config(bg="#1c2e44", fg="#FFFFFF")
    self.entry_frame.config(bg="#263b54")
    self.entry_field.config(bg="#1c2e44", fg="#FFFFFF", insertbackground="#FFFFFF")
    self.send_button_frame.config(bg="#263b54")
    self.send_button.config(bg="#1c2e44", fg="#FFFFFF", activebackground="#1c2e44",
    activeforeground="#FFFFFF")
    self.sent_label.config(bg="#263b54", fg="#FFFFFF")

    self.tl_bg = "#1c2e44"
    self.tl_bg2 = "#263b54"
    self.tl_fg = "#FFFFFF"

# Torque
def color_theme_turquoise(self):
    self.master.config(bg="#003333")
    self.text_frame.config(bg="#003333")
    self.text_box.config(bg="#669999", fg="#FFFFFF")
    self.entry_frame.config(bg="#003333")
    self.entry_field.config(bg="#669999", fg="#FFFFFF", insertbackground="#FFFFFF")
```

```
        self.send_button_frame.config(bg="#003333")
        self.send_button.config(bg="#669999", fg="#FFFFFF", activebackground="#669999",
        activeforeground="#FFFFFF")
        self.sent_label.config(bg="#003333", fg="#FFFFFF")

        self.tl_bg = "#669999"
        self.tl_bg2 = "#003333"
        self.tl_fg = "#FFFFFF"

    # Hacker
    def color_theme_hacker(self):
        self.master.config(bg="#0F0F0F")
        self.text_frame.config(bg="#0F0F0F")
        self.entry_frame.config(bg="#0F0F0F")
        self.text_box.config(bg="#0F0F0F", fg="#33FF33")
        self.entry_field.config(bg="#0F0F0F", fg="#33FF33", insertbackground="#33FF33")
        self.send_button_frame.config(bg="#0F0F0F")
        self.send_button.config(bg="#0F0F0F", fg="#FFFFFF", activebackground="#0F0F0F",
       activeforeground="#FFFFFF")
        self.sent_label.config(bg="#0F0F0F", fg="#33FF33")

        self.tl_bg = "#0F0F0F"
        self.tl_bg2 = "#0F0F0F"
        self.tl_fg = "#33FF33"

    # Default font and color theme
    def default_format(self):
        self.font_change_default()
        self.color_theme_default()

    def change_language_to_tamil(self):
        self.current_language = "ta"
    def change_language_to_hindi(self):
        self.current_language = "hi"
    def change_language_to_malayalam(self):
        self.current_language = "ml"
    def change_language_to_telugu(self):
        self.current_language = "te"
    def change_language_to_eng(self):
        self.current_language = "en"

def chat(msg):
    ints = predict_class(msg)
    return get_response(ints, intents)
```

```
root = Tk()

a = ChatInterface(root)
root.geometry(window_size)
root.title("MedBot")
root.iconbitmap('MedBot.jpg')
root.mainloop()
```

CHATBOT.PY

```
# -*- coding: utf-8 -*-
"""chatbot.py

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1feCkdIYr8e1V5KIzxxooJr5o2bRK4R4p
"""

import nltk
import random
import numpy as np
import json
import pickle
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.models import load_model
lemmatizer=WordNetLemmatizer()

with open('intents.json') as json_file:
    intents = json.load(json_file)

words=pickle.load(open('words.pkl','rb'))
classes=pickle.load(open('classes.pkl','rb'))
model=load_model('chatbotmodel.h5')

def clean_up_sentence(sentence):
  sentence_words=nltk.word_tokenize(sentence)
  sentence_words=[lemmatizer.lemmatize(word) for word in sentence_words]
  return sentence_words

def bag_of_words(sentence):
  sentence_words=clean_up_sentence(sentence)
```

```
  bag=[0]*len(words)
  for w in sentence_words:
    for i,word in enumerate(words):
      if word == w:
        bag[i]=1
  return np.array(bag)

def predict_class(sentence):
  bow=bag_of_words(sentence)
  res=model.predict(np.array([bow]))[0]
  ERROR_THRESHOLD=0.25
  results=[[i,r] for i,r in enumerate(res) if r> ERROR_THRESHOLD]

  results.sort(key=lambda x:x[1],reverse=True)
  return_list=[]
  for r in results:
    return_list.append({'intent': classes[r[0]],'probability':str(r[1])})
  return return_list

def get_response(intents_list,intents_json):
  tag=intents_list[0]['intent']
  list_of_intents=intents_json['intents']
  for i in list_of_intents:
    if i['tag']==tag:
      result=random.choice(i['responses'])
      break
  return result


print("GO! BOT IS RUNNING")

# while True:
#  message=input("")
#    ints=predict_class(message)
# res=get_response(ints,intents)
# print(res)
```

TRAINING.PY

```
import random
import json
import pickle
import numpy as np
import pandas as pd

import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import SGD

lemmatizer=WordNetLemmatizer()

with open('intents.json') as json_file:
    intents = json.load(json_file)

#print(intents)

words=[]
classes=[]
documents=[]
ignore_letters=['?','!','.',',']

for intent in intents['intents']:
  for pattern in intent['patterns']:
    word_list=nltk.word_tokenize(pattern)
    words.extend(word_list)
    documents.append((word_list,intent['tag']))
    if intent['tag'] not in classes:
    classes.append(intent['tag'])


words =[lemmatizer.lemmatize(word) for word in words if word not in ignore_letters]
words = sorted(set(words))
classes=sorted(set(classes))
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
```

```
training=[]
output_empty=[0]*len(classes)

for document in documents:
  bag=[]
  word_patterns=document[0]
  words = [lemmatizer.lemmatize(word) for word in words if word and word not in
ignore_letters]
  for word in words:
    bag.append(1) if word in word_patterns else bag.append(0)

  output_row=list(output_empty)
  output_row[classes.index(document[1])]=1
  training.append([bag,output_row])

random.shuffle(training)
training=np.array(training)

train_x=list(training[:,0])
train_y=list(training[:,1])
model=Sequential()
model.add(Dense(128,input_shape=(len(train_x[0]),),activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]),activation='softmax'))

sgd=SGD(learning_rate=0.01,decay=1e-6,momentum=0.9,nesterov=True)
model.compile(loss='categorical_crossentropy',optimizer=sgd,metrics=['accuracy'])
hist = model.fit(np.array(train_x),np.array(train_y),epochs=200,batch_size=5,verbose=1)
model.save('chatbotmodel.h5', hist)
print('Training Done')
```

INTENTS.JSON


```
{"intents":
[
        {
         "tag":"greetings",
         "patterns": ["hello","hey","hi","good day","greetings","what's up?","how is it going"],
         "responses":["hello","hey!","what can i do for you?"]
        },
        {
         "tag":"goodbye",
         "patterns": ["cya","see you later","goodbye","have a good day","bye","cao","see ya"],
         "responses":["have a nice day","goodbye"]
        },
        {
         "tag":"age",
         "patterns": ["how old","how old are you?","what is your age","how old are you","age?"],
         "responses":["I get reborn after every compilation","hey!","my owners are averagely 20 years!"]
        },
        {
         "tag":"name",
         "patterns": ["what is your name","what should i call you","what's your name?","who are you?","can you tell me your name"],
         "responses":["you can call me Medbot!","i am Medbot!","i am Medbot your medical assistant"]
        },
        {
         "tag":"common cold symptoms",
         "patterns": ["Runny or stuffy nose",
         "Sore throat",
         "Cough",
         "Congestion",
         "Slight body aches or a mild headache",
         "Sneezing",
         "Low-grade fever",
         "Generally feeling unwell (malaise)"],
         "responses":["It seems that you are suffering from common cold"]
        },
        {
         "tag":"fever symptoms",
```

```
 "patterns":["Sweating",
 "Chills and shivering",
 "Headache",
 "Muscle aches",
 "Loss of appetite",
 "Irritability",
 "Dehydration",
 "General weakness"],
 "responses":["It seems that you are suffering from fever"]
},
{
 "tag":"Diabetes symptoms",
 "patterns":["increased hunger",
 "increased thirst",
 "weight loss",
 "frequent urination",
 "blurry vision",
 "extreme fatigue"],
 "responses":["It seems that you are suffering from Diabetes"]
},
{
 "tag":"Depression symptoms",
 "patterns":["Hopeless outlook",
 "Lost interest",
 "Increased fatigue",
 "sleep problem",
 "Anxiety",
 "change in weight",
 "Looking at death"],
 "responses":["It seem that you are suffering from depression"]
},
{
 "tag":"Asthma symptoms",
 "patterns":["coughing",
 "tightness in the chest",
 "shortness of breath",
 "difficulty talking",
 "panic",
 "fatigue"],
 "responses":["It seem that you are suffering from Asthma"]
},
{
 "tag":"gastric symptoms ",
 "patterns":["Passing gas",
```

```
 "Burping",
 "Pain/Cramps",
 "A knotted feeling in the abdomen",
 "Stomach ache (Abdominal Pain)",
 "Indigestion" ],
 "responses":["It seem that you are suffering from gastric symptoms"]
},
{
   "tag":"dysentery symptoms",
   "patterns":["Nausea",
   "Vomiting",
   "Diarrhea",
   "Abdominal pain",
   "Dehydration"],
   "responses":["It seem that you are suffering from dysentery"]
},
{
   "tag":"piles symptoms",
   "patterns":["Anal itching",
   "Swelling",
   "Anal pain",
   "Bleeding",
   "Lumps near the anus",
   "Slimy anal discharge",
   "Sore skin on the anus"],
   "responses":["It seem that you are suffering from piles "]

},
{
"tag":"indigestion symptoms",
"patterns":["Bloating.",
"Gas",
"Heartburn",
"Acid reflux",
"Nausea",
"Burping"],
"responses":["It seem that you are suffering from indigestion "]
},
{
   "tag":"pink eyes symptoms",
   "patterns":["Redness in the white of your eye",
   "Dry eyes or watery eyes (epiphora)",
   "Itchy or irritated eyes",
   "Burning eyes",
```

```
        "Blurred vision",
        "Light sensitivity",
        "Swollen eyelids",
        "Eye pain or discomfort",
        "Foreign object sensation" ],
        "responses":["It seem that you are suffering from pink eyes"]

    },
    {
        "tag":"vertigo symptoms",
        "patterns":["Nausea and vomiting.",
        "Dizziness.",
        "Balance issues.",
        "Hearing loss in one or both ears.",
        "Tinnitus (ringing in your ears).",
        "Headaches.",
        "Motion sickness.",
        "A feeling of fullness in your ear.",
        "Nystagmus"],
        "responses":["It seem that you are suffering from vertigo"]
    },
    {
        "tag":"migraine symptoms",
        "patterns":["Trouble sleeping",
        "Mood changes.",
        "Difficulty concentrating.",
        "Fatigue",
        "Ringing in your ears",
        "Vision changes",
        "Difficulty speaking or concentrating"],
        "responses":["It seem that you are suffering from migraine"]
    },
    {
     "tag":"common cold prevention",
     "patterns":["What medicines can I buy to help me with my common cold?","tell me some prevention method from common cold","What should I eat or drink if i am suffering from common cold?","How can I keep from getting a cold or the flu?"],

     "responses":["medicines you can consume : Dextromethorphan,Decongestant,Diphenhydramine,Crocin Cold & Flu Max, preventions that you must follow :Wash your hands,Avoid touching your face,Clean frequently used surfaces,Use hand sanitizers,SUGGESTED FOODS ARE:Garlic,Vitamin C–containing fruits,Leafy greens,Broccoli,Oatmeal,Spices,Chicken Soup"]
    },
```

```
{
 "tag":"fever prevention",
 "patterns":["What medicines can I buy to help me with my fever?","tell me some prevention method from fever","What should I eat or drink if i am suffering from fever?","How can I keep from getting a fever?"],
 "responses":["medicines you can consume : acetaminophen ,ibuprofen,aspirin,Crocin Cold & Flu Max, prevention that you must follow :Wash your hands,Cover your mouth when you cough and your nose when you sneeze,Clean frequently used surfaces,Avoid sharing cups, glasses, and eating utensils with other people.,SUGGESTED FOODS ARE:Garlic,Vitamin C–containing fruits,Leafy greens,Broccoli,Oatmeal,Spices,Chicken Soup"]
},
{
 "tag":"diabetes prevention",
 "patterns":["What medicines can I buy to help me with my diabetes?","tell me some prevention method from diabetes","What should I eat or drink if i am suffering from diabetes?","How can I keep from getting diabetes?"],
 "responses":["medicines you can consume : Insulin ,Amylinomimetic drug,Dipeptidyl peptidase-4 (DPP-4) inhibitor, prevention that you must follow :Cut Sugar and Refined Carbs From Your Diet,Work Out Regularly,Drink Water as Your Primary Beverage,Lose Weight If You’re Overweight or Obese,Quit Smoking, Follow a Very-Low-Carb Diet,Watch Portion Sizes,SUGGESTED FOODS ARE:Leafy greens,Avocados,Eggs"]
},
{
 "tag":"depression prevention",
 "patterns":["What medicines can I buy to help me with my depression?","tell me some prevention method from depression","What should I eat or drink if i am suffering from depression?","How can I keep from getting depression?"],
 "responses":["medicines you can consume : brexpiprazole, quetiapine,olanzapine, prevention that you must follow :Exercise regularly,Cut back on social media time,Drink Water as Your Primary Beverage,Build strong relationships,Minimize your daily choices, Follow a Very-Low-Carb Diet,SUGGESTED FOODS ARE:Get Enough Vitamin D,Include Omega-3 Fatty Acids,Beans and legumes"]
},
{
 "tag":"asthma prevention",
 "patterns":["What medicines can I buy to help me with my asthma?","tell me some prevention method from asthma","What should I eat or drink if i am suffering from asthma?","How can I keep from getting asthma?"],
 "responses":["medicines you can consume : epinephrine,anticholinergic,Proair HFA, prevention that you must follow : Identify Asthma Triggers, Stay Away From Allergens,Avoid Smoke of Any Type,SUGGESTED FOODS ARE:carrots,juice,eggs,broccoli,cantaloupe,milk"]
},
```

{
"tag":"gastric prevention",
"patterns":["What medicines can I buy to help me with my gastric?","tell me some prevention method from gastric","What should I eat or drink if i am suffering from gastric?","How can I keep from getting gastric?"],
"responses":[" Some medications, including nonsteroidal anti-inflammatory drugs (NSAIDs), aspirin, antibiotics, and oral contraceptives, can irritate the stomach lining and increase the risk of gastric problems or ulcers."]
},
{
"tag":"dysentery prevention",
"patterns":["What medicines can I buy to help me with my dysentery?","tell me some prevention method from dysentery","What should I eat or drink if i am suffering from dysentery?","How can I keep from getting dysentery?"],
"responses":["If dysentery is caused by bacterial infection, antibiotics such as azithromycin, ciprofloxacin, or trimethoprim/sulfamethoxazole may be prescribed to target the specific bacteria responsible. Rehydration is crucial in managing dysentery, especially if diarrhoea is severe and there is a risk of dehydration. Drinking plenty of fluids, such as water, oral rehydration solutions (ORS), clear broths, and electrolyte-rich drinks, can help replace lost fluids and electrolytes."]

},
{
"tag":"piles prevention",
"patterns":["What medicines can I buy to help me with my piles?","tell me some prevention method from piles","What should I eat or drink if i am suffering from piles?","How can I keep from getting piles?","I have piles","piles"],
"responses":["Hydrocortisone for piles and itchy bottom Brand names: Anusol, Anugesic, Germaloids, Perinal, Proctosedyl, Uniroid, Xyloproct.. Use pain killers and stool softeners as prescribed by your doctor. If your symptoms indicate internal piles, you will need to get a digital rectal exam. Here are some surgery options your doctor might suggest: Laser piles surgery,Haemorrhoidectomy,Haemorrhoidal artery ligation,Haemorrhoidal artery ligation"]
},
{
"tag":"migraine prevention",
"patterns":["What medicines can I buy to help me with my migraine?","tell me some prevention method from migraine","What should I eat or drink if i am suffering from migraine?","How can I keep from getting migraine?","I have migraine"],
"responses":["medicines you can consume :Triptans (5-hydroxytryptamine),Ditans (lasmiditan),Gepants (rimegepant and ubrogepant),Dihydroergotamine (prochlorperazine),Antiemetic medications (metoclopramide). You can’t prevent all migraines. But you can take preventive migraine medications as directed by your healthcare provider to reduce how often and how severe migraine symptoms affect you."]

```
        },
        {
            "tag":"pink eyes prevention",
            "patterns":["What medicines can I buy to help me with my pink eyes?","tell me some prevention method from pink eyes","What should I eat or drink if i am suffering from pink eyes?","How can I keep from getting pink eyes?"],
            "responses":["medicines you can consume :polymyxin b/trimethoprim (Polytrim) ciprofloxacin (Ciloxan) are some Antibiotic eye drops may help reduce the infection's duration. To reduce the symptoms of bacterial or viral pink eye you can: Take ibuprofen or another over-the-counter pain killer. Use over-the-counter lubricating eye drops (artificial tears). Put a warm, damp washcloth over your eyes for a few minutes. "]
        },
        {
            "tag":"indigestion prevention",
            "patterns":["What medicines can I buy to help me with my indigestion?","tell me some prevention method from indigestion","What should I eat or drink if i am suffering from indigestion?","How can I keep from getting indigestion?"],
            "responses":["medicines you can consume :antacid medications, like Tums ®, Rolaids ® and Pepto-Bismol ®. Antacids neutralize the acid in your stomach so it doesn’t irritate your tissues. Small sips of water when you’re experiencing indigestion may help a little."]
        },
        {
            "tag":"vertigo prevention",
            "patterns":["What medicines can I buy to help me with my vertigo?","tell me some prevention method from vertigo","What should I eat or drink if i am suffering from vertigo?","How can I keep from getting vertigo?"],
            "responses":["There is no medicines for vertigo. Benign paroxysmal positional vertigo (BPPV) occurs when tiny calcium carbonate crystals (canaliths) move out of the utricle in your inner ear (where they belong) into your semicircular canals. \n This can cause vertigo symptoms, especially when you change your head position.Medication may help in some cases of acute (sudden onset, short duration) vertigo."]
        },
        {
         "tag":"Consultation",
         "patterns": ["who should i contact for consultation?","is there any doctor available?","can you give me some suggestions for doctor consultations?","can you set up a meeting with a doctor for consultation?","is there any doctor available for consultation"],
          "responses":["You can contact various doctors here for any kind of consultation: 1. https://www.1mg.com/online-doctor-consultation, 2. https://www.tatahealth.com/online-doctor-consultation/general-physician, 3. https://www.doconline.com/, or you can pay a visit to your local area doctor or family doctor."]
        }
]}
```

7.3 REFERENCE:

[1] R Jegadeesan, Dava Srinivas, N Umapathi, G Karthick, N Venkateswaran, "PERSONAL HEALTHCARE CHATBOT FOR MEDICAL SUGGESTIONS USING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING", Jyothishmathi Institute of Technology and Science, Karimnagar, Telangana. Eur. Chem. Bull. 12 (S3), 6004 – 6012. 13 March 2023

[2] T. Padmavathy, K. Rajarajeshwari, J. Kavvya, M. Reni, "A Survey on Healthcare Chatbot Using Machine Learning", International Journal of Innovative Research in Engineering & Management (IJIREM) ISSN: 2350-0557, Volume-7, Issue-2, March 2020

[3] LekhaAthota, Vinod Kumar Shukla, Nitin Pandey, Ajay Rana, "Chatbot for Healthcare System Using Artificial Intelligence", International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) Amity University, Noida, India. June 4-5, 2020.

[4] Duckki Lee," AI-based Healthcare Chatbot", International Research Journal of Engineering and Technology (IRJET), Yonam Institute of Technology in South Korea, Feb 2023.

[5] Mark Lawrence, MD Istiyak, Mohd Aman, "HEALTHCARE CHATBOT SYSTEM", International Journal of Novel Research and Development, © 2024 IJNRD | Volume 9, Issue 3 March 2024| ISSN: 2456-4184