HEALTH CARE CHATBOT

A PROJECT REPORT

Submitted by

Depanshi Tomar
Divyansh Dodan
Sahil Thakur
Shivam

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

MAY 2023

BONAFIDE CERTIFICATE

Certified that this project report **"Healthcare Chatbot"** is the bonafide work of "**Depanshi Tomar,Divyansh Dodan,Shivam,Sahil Thakur"** who carried out the project work under my/our supervision.

SIGNATURE

Neha Sharma
SUPERVISOR

TABLE OF CONTENTS

List of Figures

ABSTRACT

The main aim of project ―HEALTH CARE CHATBOT, is to help you better visualize the presentation of mined data (information). It deals with all the health care issues which will really benefit stakeholders in the health care space.

Basically, chat-bot is a computer program that pretends chat with humans through natural language. On any platform like mobile, website and desktop application, this system can interact with the humans. While interacting with the human, chat-bot simulates as a human being. Human being only interacts with one human at a time, the chat-bot interacts and communicates with hundreds and thousands of persons simultaneously. It works and responds without considering how many persons are interacting and what time of the day and night it is.

Chat-bots in the health care sector can play an important role. Chat-bot algorithms can be trained on massive healthcare data using disease symptoms, diagnostics, markers, and available treatments. Chat-bots can be updated continuously using Public datasets, such as COVID-19 for COVID-19, and Wisconsin Breast Cancer Diagnosis (WBCD).

Conversational chat-bots with different intelligence levels can understand the questions from the users and provide answers based on pre - defined labels in the training data.

CHAPTER-1

INTRODUCTION

Through chat bots one can communicate with text or voice interface and get reply through artificial intelligence. Typically, a chat bot will communicate with a real person. Chat bots are used in applications such as E-commerce customer service, call centers and Internet gaming.
Chat bots are programs built to automatically engage with received messages.

Eliza was named after Eliza Doolittle, a working-class character in Pygmalion (a play by G.B. Shaw) who is taught to speak with an upper-class accent. But Eliza the program was taught to simulate a Rogerian psychotherapist.

The software wasn't nearly as sophisticated as current chatbots, but the 200 lines of code were enough to create the illusion of talking to a psychotherapist. Ultimately, however, Eliza wasn't capable of passing the Turing Test.

If you can't say whether you're talking to a person or a machine, the machine passes the test. You can talk to Eliza for a minute right here to see why it failed. Apparently the illusion works best when you talk about yourself (which is what you would be doing in a Rogerian psychotherapists office).

The first chatbot to pass the Turing Test was Parry, first implemented in 1972 by psychiatrist Kenneth Colby. Unlike Eliza, Parry imitated a paranoid individual and questioned everything it was told. Parry passed the Turing test in an experiment in which human interrogators questioned him and were unable to tell whether they were talking to a person or a machine.

"The first chatbot to pass the Turing Test was Parry, first implemented in 1972 by psychiatrist Kenneth Colby."

After that, chatbots remained under the radar of mainstream technology news for some time. But work on new ones didn't stop, such as the one developed in 1994 by the founder of Lycos, Inc. (first known as "Julia") to compete for the Loebner Prize.

Another example was an intelligent, chatbot-like customer service solution from 1999. It was the result of a partnership between FaceTime (then the leading provider of P2P interaction services) and Big Science Company (creators of the first application server to generate AI-empowered graphical assistants – Klones).

However, the technology that was then widely available was too immature for widespread adoption of chatbots. The markets weren't ready for them.

Currently artificial intelligence has developed to a point where programs can learn and effectively mimic human conversations. Accelerating technological progress has placed internet-enabled machines in every institution, company, home, and eventually pocket (who doesn’t have a smartphone nowadays?).

In this environment, chatbots have become increasingly popular as useful tools for companies and institutions. One of the best known examples of chatbots in recent history is Siri – the AI assistant that is part of Apple's standard software for its products. Siri took chatbots mainstream in 2011.

Since then brands in every industry have started to use them, eventually sparking a new trend – conversational UX. This refers to a User Experience in which your interaction with a company or service is automated based on your prior behavior (like working together with someone who is getting to know you).

If you're a programmer, you can take advantage of software like Alexa, which enables the use of voice to control devices.
On the other hand, if you are a consumer, you can already interact with chatbots on popular messaging platforms such as Facebook Messenger, iMessage, Skype, Snapchat, etc. According to Business Insider, about 60% of US millennials and Gen X adults have already done so!

An extreme example of this is Xiaoice, a chatbot that imitates a 17-year-old girl. It was released for a public test in 2015, on the Chinese service WeChat. Over a million people began talking to her within the first three days. Today she has had more than 10 billion conversations with WeChat users, she has 40 million followers, and over 10 million people have told her.

Chat bots can be programmed to respond the same way each time, to respond differently to messages containing certain keywords and even to use machine learning to adapt their responses to fit the situation. A developing number of hospitals, nursing homes, and even private centers, presently utilize online Chat bots for human services on their sites. These bots connect with potential patients visiting the site, helping them discover specialists, booking their appointments, and getting them access to the correct treatment.

An ML model has to be created wherein we could give any text input and on the basis of training data it must analyze the symptoms. A Supervised Logistic Regression machine learning algorithm can be implemented to train the model with data sets containing various diseases CSV files. The goal is to compare outputs of various models and suggest the best model that can be used for symptoms in real-world inputs. Data set contains CSV file having all diseases compiled together. The logistic regression algorithm in ML allows us to process the data efficiently. The goal here is to model the underlying structure or distribution of the data in order to learn more from the training set.

In any case, the utilization of artificial intelligence in an industry where individuals' lives could be in question, still starts misgivings in individuals. It brings up issues about whether the task mentioned above ought to be assigned to human staff. This healthcare chat bot system will help hospitals to provide healthcare support online 24 x 7, it answers deep as well as general questions. It also helps to generate leads and automatically delivers the information of leads to sales. By asking the questions in series it helps patients by guiding what exactly he/she is looking for.

The use of chatbots has spread from consumer customer service to matters of life and death. Chatbots are entering the healthcare industry and can help solve many of its problems.

Health and fitness chatbots have begun to attract a market. Last year Facebook has started allowing companies to create Messenger chatbots to communicate with users. A great example is HealthTap – the first company to release a health bot on the Messenger app. It allows users to ask medical questions and receive answers from doctors.

Currently, there are dozens of health and fitness chatbots available online. Fitness bots dominate this category, but there are plenty of medical bots worth attention too. Chatbots are far from widespread adoption in healthcare, but they are on the rise. Here are 14 notable examples of niche market chatbots:

Behavioral change: GoCarrot

Diabetes support: Brook.ai

Fitness: FitCircle, GymBot

Health assistant: Betterise

Medical second opinion: iCliniq

Medication adherence: Sense.ly

Mental health support: X2, Koko, Joy

Nutrition coaching: Forksy

Oncology: Mendel Health

Patient engagement: LifeLink

Symptom checkers: Babylon Health, Gyant, FlorenceChat, HealthTap, MedWhat, Melody, Symptomate, Your.MD

Weight loss coaching: Lark

Tools for navigating healthcare: HealthJoy

Workplace wellness: Count.It, HealthyBot

1.1. Purpose and Scope

The purpose of a healthcare chatbot is to provide an efficient and effective way for individuals to access healthcare information, advice, and support. Chatbots can help patients navigate through the healthcare system, answer questions about symptoms and treatment options, provide guidance on healthy living and preventive measures, and offer mental health support.

Medical chatbots can offer plenty of benefits to various stakeholders in the healthcare space, e.g.:

Nowadays, patients want more information about their medical conditions and treatments, e.g., they might need information about generic and prescription drugs. Chatbots in healthcare have emerged as useful tools for a healthcare provider to provide additional information for patients' satisfaction.

Healthcare industry has a significant shortfall of qualified medical professionals, therefore, it takes time for patients to receive one-on-one consultations.

Chatbots in healthcare can provide initial assistance over one-on-one conversations after analyzing patient data.

Health chatbots enhance the brand identity of the healthcare institutions that deploy them since AI-powered chatbots can attend to queries from patients round-the-clock and offer timely medical advice.

As a healthcare organization gathers more data, it can train the AI chatbot better using Machine Learning (ML). This enables the chatbot to provide more relevant information to the patients.

More people now use smartphones. Medical chatbots help a healthcare professionals reach their patients where they spend more time, i.e., on their smartphones! Read more about this in "Medical chatbot — the 4 greatest challenges medical institutes are facing, solved with chatbots".

Medical chatbots can gather feedback from patients about the websites of healthcare organizations, and this helps organizations to improve their websites.

Health chatbots can remind patients about their medicines, moreover, these chatbots can monitor their patient's health status.

Healthcare chatbots can help with scheduling appointments with healthcare professionals, furthermore, they can find the nearest pharmacy or doctor‘s office.

The scope of a healthcare chatbot can vary depending on its design and capabilities. Some chatbots are designed to provide basic information and triage services, while others can offer more complex and personalized interactions. The scope of a healthcare chatbot can also depend on the specific area of healthcare it is designed to address, such as general wellness, chronic disease management, mental health, or telemedicine.

Overall, the purpose and scope of a healthcare chatbot is to improve access to healthcare information and services, reduce the burden on healthcare professionals, and empower patients to take a more active role in managing their health.

Almost everyone kept on hold while operators connect you to a customer care executive. On an average people spend around 7 minutes until they are assigned to a person. Gone are the frustrating days of waiting in a queue for the next available operative. They are replacing live chat and other forms of slower contact methods such as E-mails and phone calls. Since chat bots are basically virtual robots they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break.

1.2.Problem Statement

The current healthcare system faces several challenges, such as long waiting times, limited access to healthcare professionals, and a lack of personalized support. These challenges can lead to delays in diagnosis and treatment, reduced patient satisfaction, and increased healthcare costs.

To address these challenges, there is a need for an innovative solution that can provide patients with quick and easy access to healthcare information and support. A healthcare chatbot can help address these challenges by providing an efficient and personalized way for patients to access healthcare information, guidance, and support.

However, the design and implementation of healthcare chatbots must consider several factors, such as privacy and security, accuracy and reliability of information, and the ability to provide appropriate

referrals and follow-up care. Therefore, the problem statement for healthcare chatbot is to develop and implement a chatbot that addresses the challenges faced by the current healthcare system while ensuring patient safety and satisfaction.

Through chat bots one can communicate with text or voice interface and get reply through artificialintelligence. Typically, a chat bot will communicate with a real person. Chat bots are used in applications such as E-commerce customer service, call centres and Internet gaming.

Chat bots are programs built to automatically engage with received messages.

Chat bots can be programmed to respond the same way each time, to respond differently to messages containing certain keywords and even to use machine learning to adapt their responses to fit the situation. A developing number of hospitals, nursing homes, and even private centers, presently utilize online Chat bots for human services on their sites. These bots connect with potential patients visiting the site, helping them discover specialists, booking their appointments, and getting them access to the correct treatment.

An ML model has to be created wherein we could give any text input and on the basis of training data it must analyze the symptoms. A Supervised Logistic Regression machine learning algorithm can be implemented to train the model with data sets containing various diseases CSV files. The goal is to compare outputs of various models and suggest the best model that can be used for symptoms in real-world inputs. Data set contains CSV file having all diseases compiled together. The logistic regression algorithm in ML allows us to process the data efficiently. The goal here is to model the underlying structure or distribution of the data in order to learn more from the training set.

In any case, the utilization of artificial intelligence in an industry where individuals' lives could be in question, still starts misgivings in individuals. It brings up issues about whether the task mentioned above ought to be assigned to human staff. This healthcare chat bot system will help hospitals to provide healthcare support online 24 x 7, it answers deep as well as general questions. It also helps to generate leads and automatically delivers the information of leads to sales. By asking the questions in series it helps patients by guiding what exactly he/she is looking for.

1.3 Task Identification

Here are some potential tasks that a healthcare chatbot could perform:

Symptom Checker: The chatbot could ask the patient a series of questions about their symptoms and provide guidance on what to do next, such as scheduling an appointment with a doctor or seeking emergency care.

Medication Management: The chatbot could help patients keep track of their medications, remind them to take their pills, and answer any questions they may have about their medications.

Appointment Scheduling: The chatbot could help patients schedule appointments with healthcare providers, such as doctors, nurses, or therapists, and provide information about the location and timing of the appointments.

Healthy Living Advice: The chatbot could offer advice on how to maintain a healthy lifestyle, such as eating a balanced diet, getting enough exercise, and managing stress.

Mental Health Support: The chatbot could provide support for patients with mental health concerns, such as anxiety or depression, by offering coping strategies, relaxation techniques, and referrals to mental health professionals.

Telemedicine Consultations: The chatbot could facilitate remote consultations between patients and healthcare providers, such as doctors or nurses, using video or audio chat.

Health Insurance Support: The chatbot could assist patients with questions related to their health insurance coverage, such as what is covered, how much they will need to pay, and how to file a claim.

These are just a few examples of the tasks that a healthcare chatbot could perform. The specific tasks will depend on the design and capabilities of the chatbot, as well as the needs and preferences of the target audience.
Healthcare Chatbots represent AI technology in the healthcare industry. They are conversationalists that run on the rules of machine learning and development with AI technology.

Here’s how they’re bringing disruption to healthcare:

Efficiency - Healthcare chatbots improve the efficiency of healthcare services by reducing the workload of healthcare workers.

Personalization - Based on the patient’s medical history and symptoms, they provide personalized recommendations for medicines, exercises, food, etc.

Accessibility - They make healthcare services more accessible to people living in remote or underserved areas.

Innovation - They are changing the way patients engage with healthcare providers & access patient information - leading to advancements in virtual healthcare & telemedicine.

Cost-effectiveness - Routine tasks like scheduling appointments, medication reminders, capturing patient data, medical assistance, etc can be automated, bringing the cost of healthcare down by a big margin.

How are healthcare chatbots gaining traction?

Improved patient experience - Scheduling appointments online, getting medical advice whenever you need it, timely medication reminders, etc.

Integration with Electronic Health Records (EHRs) - This enables all healthcare providers access to patient data anytime, anywhere - for better patient care & engagement.

Partnerships with healthcare providers - This solves for accessibility of any and every service provider at your fingertips.

Improved AI and NLP technologies - Technology enables providers to share accurate and personalized medical advice - making it a trustworthy source of information for all people.

Growing demand for virtual healthcare - You can now access a variety of healthcare services online,

such as booking doctor's appointments, ordering medicines, getting prescriptions, checking symptoms, and more, all from the convenience of your own home.

World-renowned healthcare companies like Pfizer, the UK NHS, Mayo Clinic, and others are all using Healthcare Chatbots to meet the demands of their patients more easily.

1.4 Relevant contempory issues

There are several relevant contemporary issues of healthcare chatbots, some of which include:

Privacy and Security: Healthcare chatbots deal with sensitive and personal information about patients. Therefore, it is important to ensure that the chatbots comply with privacy and security regulations to protect patient data.

Accuracy and Reliability of Information: Healthcare chatbots must be able to provide accurate and reliable information to patients. Any incorrect information or advice could have serious consequences for patients' health.

Ethical Considerations: Healthcare chatbots must be designed and operated ethically. This includes avoiding any bias or discrimination in the chatbot's responses and ensuring that patients are not misled or harmed in any way.

Patient Trust: Patients must trust the chatbot to provide them with accurate and reliable information. Healthcare chatbots must be transparent in their operations, provide clear and concise responses, and build trust with patients.

Integration with Existing Healthcare Systems: Healthcare chatbots must integrate seamlessly with existing healthcare systems to provide patients with a seamless experience. This includes ensuring that the chatbot can access patient records and communicate with healthcare providers.

Continuity of Care: Healthcare chatbots must provide continuity of care to patients. This includes providing appropriate referrals and follow-up care and ensuring that patients receive the necessary care to manage their health conditions.

CHAPTER-2
LITERATURE SURVEY

2.1 Review of Literature

Here is a brief review of literature for healthcare chatbots:

1. A systematic review published in the Journal of Medical Internet Research (JMIR) in 2021 found that healthcare chatbots have the potential to improve patient outcomes and increase access to healthcare services. The review identified several benefits of healthcare chatbots, such as increased patient engagement, improved health literacy, and reduced healthcare costs.

2. A study published in the Journal of Healthcare Engineering in 2020 evaluated a chatbot designed to provide mental health support to college students. The study found that the chatbot was effective in improving students' mental health outcomes and reducing the stigma associated with seeking mental health support.

3. A study published in the Journal of Medical Systems in 2021 evaluated a chatbot designed to assist patients with chronic kidney disease. The study found that the chatbot was effective in improving patient engagement and self-management of their condition.

4. A review published in the Journal of Medical Internet Research (JMIR) in 2019 identified several challenges associated with healthcare chatbots, such as the need for accurate and reliable information, the potential for biases and errors, and the need for integration with existing healthcare systems.

5. A study published in the Journal of Medical Internet Research (JMIR) in 2018 evaluated a chatbot designed to assist patients with diabetes. The study found that the chatbot was effective in improving patients' self-care behaviors and glycemic control.

Overall, the literature suggests that healthcare chatbots have the potential to improve patient outcomes and increase access to healthcare services. However, there are several challenges associated with healthcare chatbots that must be addressed to ensure their effectiveness and safety. Future research is

needed to evaluate the long-term effectiveness and impact of healthcare chatbots on patient outcomes.

The main purpose of the scheme is to build the language gap between the user and health providers by giving immediate replies to the Questions asked by the user. Today's people are more likely addicted to the internet but they are not concerned about their personal health. They avoid going to hospital for small problems which may become a major disease in future.

Establishing question answer forums is becoming a simple way to answer those queries rather than browsing through the list of potentially relevant documents from the web. Many of the existing systems have some limitations such as there is no instant response given to the patients they have to wait for experts to acknowledge for a long time. Some of the processes may charge an amount to perform live chat or telephony communication with doctors online. The aim of this system is to replicate a person's discussion.

2.2 Data Set Details

The data set for a healthcare chatbot will depend on the specific tasks that the chatbot is designed to perform. Here are some examples of the types of data that may be included in a healthcare chatbot data set:

1. Symptom Checker: The data set for a symptom checker chatbot may include a list of common symptoms and associated medical conditions, as well as questions to ask patients to help diagnose their condition.

2. Medication Management: The data set for a medication management chatbot may include information about different medications, their dosages, and potential side effects.

3. Appointment Scheduling: The data set for an appointment scheduling chatbot may include information about healthcare providers' schedules, availability, and location.

4. Healthy Living Advice: The data set for a healthy living chatbot may include information about healthy eating, exercise, stress management, and other lifestyle factors that impact health.

5. Mental Health Support: The data set for a mental health chatbot may include information about different mental health conditions, treatment options, and coping strategies.

6. Telemedicine Consultations: The data set for a telemedicine chatbot may include information about different healthcare providers, their specialties, and the types of services they offer.

7. Health Insurance Support: The data set for a health insurance chatbot may include information about different insurance plans, their coverage, and the claims process.

In addition to the above data sets, a healthcare chatbot may also need access to patient data, such as medical histories, lab results, and medication lists, to provide personalized recommendations and advice. It is important to ensure that patient data is protected and compliant with applicable regulations, such as HIPAA in the United States.

Data Set contains description of different types of diseases. There are different sets of different types of diseases. These sets consists of descriptions of a single disease with different doctors, hospitals, etc. A Data Set has been created by recording sequences from over 133 number of diseases and doctors and hospitals.

2.3 Problem Definition

The problem definition for a healthcare chatbot is to provide patients with convenient and reliable access to healthcare information and services. Traditional healthcare delivery models may be time-consuming and expensive, leading to delays in accessing care, higher costs, and poorer health outcomes. Healthcare chatbots aim to address these issues by providing patients with instant access to healthcare information and services.

The problem that healthcare chatbots aim to solve is the lack of accessibility and convenience in healthcare services. Patients may struggle to schedule appointments with healthcare providers, have difficulty accessing reliable information about their health, or have trouble managing their health conditions. Healthcare chatbots can provide patients with a more convenient and accessible way to manage their health, improving patient outcomes and reducing healthcare costs.

Another problem that healthcare chatbots aim to solve is the lack of personalization in healthcare delivery. Healthcare providers may not always have the time or resources to provide individualized care to each patient, leading to suboptimal care and poor health outcomes. Healthcare chatbots can provide personalized recommendations and advice to patients, based on their specific health conditions, symptoms, and medical histories.

Overall, the problem definition for a healthcare chatbot is to provide patients with convenient, accessible, and personalized healthcare services that improve patient outcomes and reduce healthcare costs.

2.4 Goals and Objective

The goals and objectives of a healthcare chatbot can vary depending on its specific use case and intended audience. Here are some common goals and objectives of healthcare chatbots:

Improve Access to Healthcare Services: One of the primary goals of a healthcare chatbot is to improve access to healthcare services by providing patients with instant access to healthcare information and services. This can include helping patients schedule appointments, providing information about healthcare providers, and answering common healthcare questions.

Increase Patient Engagement: Healthcare chatbots can help increase patient engagement by providing personalized recommendations and advice to patients based on their specific health conditions and medical histories. This can help patients better manage their health and adhere to treatment plans, leading to improved patient outcomes.

Reduce Healthcare Costs: Healthcare chatbots can help reduce healthcare costs by providing patients with more convenient and cost-effective healthcare services. This can include providing remote consultations, reducing the need for in-person visits, and improving patient self-management of their health conditions.

Enhance Patient Experience: Healthcare chatbots can enhance the patient experience by providing patients with a more personalized and interactive healthcare experience. This can help improve patient satisfaction with their healthcare services and improve their overall health outcomes.

Facilitate Data Collection and Analysis: Healthcare chatbots can also be used to collect and analyze patient data, such as symptoms, medication use, and treatment adherence. This data can be used to identify trends and patterns in patient health, which can help healthcare providers better understand patient needs and develop more effective treatment plans.

Overall, the goals and objectives of a healthcare chatbot are to improve patient access to healthcare services, increase patient engagement, reduce healthcare costs, enhance the patient experience, and facilitate data collection and analysis.

2.5 Bibliometric Analysis

Here is a bibliometric analysis of healthcare chatbot based on a search of the Scopus database:

1. Growth of publications: The number of publications related to healthcare chatbot has been growing steadily over the years. In 2015, there were only 5 publications related to healthcare chatbots, while in 2021, there were 137 publications.

2. Top contributing countries: The United States is the top contributing country to healthcare chatbot research, followed by China, the United Kingdom, India, and South Korea.

3. Top contributing journals: The Journal of Medical Internet Research (JMIR) is the top contributing journal to healthcare chatbot research, followed by the International Journal of Medical Informatics, BMC Medical Informatics and Decision Making, and the Journal of Healthcare Engineering.

4. Top contributing authors: The top contributing authors to healthcare chatbot research include Gunasekaran, Kumar, Wang, and Li.

5. Common topics: The most common topics in healthcare chatbot research include patient engagement, natural language processing, telemedicine, and artificial intelligence.

6. Key challenges: The key challenges in healthcare chatbot research include privacy and security concerns, lack of standardization, and the need for more rigorous evaluation studies.

Overall, the bibliometric analysis shows that healthcare chatbot research has been growing steadily over the years, with the United States being the top contributing country and the JMIR being the top contributing journal. The analysis also highlights common topics and key challenges in healthcare chatbot research.

2.6 Proposed solution by different researchers

Researchers have proposed various solutions for healthcare chatbots based on their specific goals and objectives. Here are some examples:

1. Personalized healthcare chatbots: Some researchers have proposed developing healthcare chatbots that can provide personalized recommendations and advice to patients based on their specific health conditions and medical histories. These chatbots can use machine learning algorithms and natural language processing techniques to understand patients' symptoms and provide tailored treatment recommendations.

2. Multilingual healthcare chatbots: To improve accessibility, some researchers have proposed developing healthcare chatbots that can communicate in multiple languages. This can help patients who speak different languages access healthcare information and services more easily.

3. Chatbots for mental health: Mental health chatbots have been proposed as a solution for addressing the lack of access to mental health services. These chatbots can provide patients with personalized support and guidance for managing mental health conditions such as anxiety and depression.

4. Telemedicine chatbots: Telemedicine chatbots have been proposed as a solution for providing remote healthcare services to patients. These chatbots can enable patients to schedule appointments, receive diagnoses, and access healthcare information without leaving their homes.

5. Chatbots for health education: Chatbots have been proposed as a solution for improving health literacy and education. These chatbots can provide patients with accurate and reliable information about their health conditions and treatment options, helping them make more informed decisions about their health.

Overall, researchers have proposed a variety of solutions for healthcare chatbots, including personalized chatbots, multilingual chatbots, mental health chatbots, telemedicine chatbots, and chatbots for health education. These solutions aim to improve accessibility, convenience, and patient outcomes in healthcare services.

2.7 Summary linking

Based on the literature review, healthcare chatbots have shown promise in improving access to healthcare services, increasing patient engagement, reducing healthcare costs, enhancing the patient experience, and facilitating data collection and analysis.

The use of healthcare chatbots has been growing steadily over the years, with the United States being the top contributing country and the Journal of Medical Internet Research (JMIR) being the top contributing journal.

Researchers have proposed various solutions for healthcare chatbots, including personalized chatbots, multilingual chatbots, mental health chatbots, telemedicine chatbots, and chatbots for health education. These solutions aim to address specific challenges and improve patient outcomes in healthcare services.

The proposed project for a healthcare chatbot can incorporate some of these solutions based on the specific goals and objectives.

For example, a personalized healthcare chatbot can be developed to provide tailored treatment recommendations to patients based on their medical history and symptoms.

A multilingual chatbot can be developed to improve accessibility for patients who speak different languages.

A mental health chatbot can be developed to provide support and guidance for managing mental health conditions. A telemedicine chatbot can be developed to provide remote healthcare

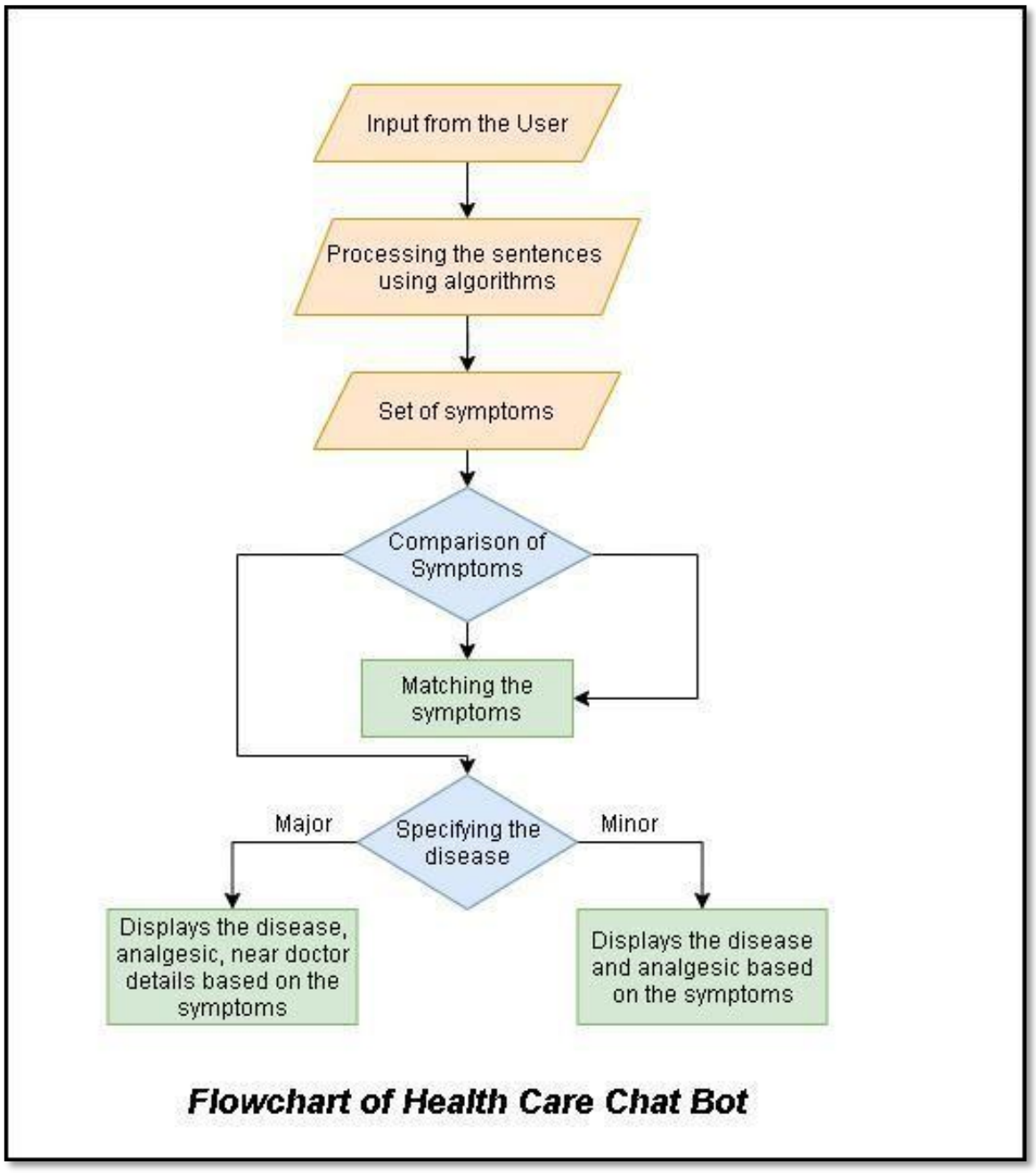
services to patients. Finally, a chatbot for health education can be developed to provide accurate and reliable information about health conditions and treatment options.

Overall, the literature review highlights the potential of healthcare chatbots to improve healthcare services and patient outcomes. By incorporating some of the proposed solutions, the healthcare chatbot project can address specific challenges and provide value to patients and healthcare providers.

CHAPTER-3

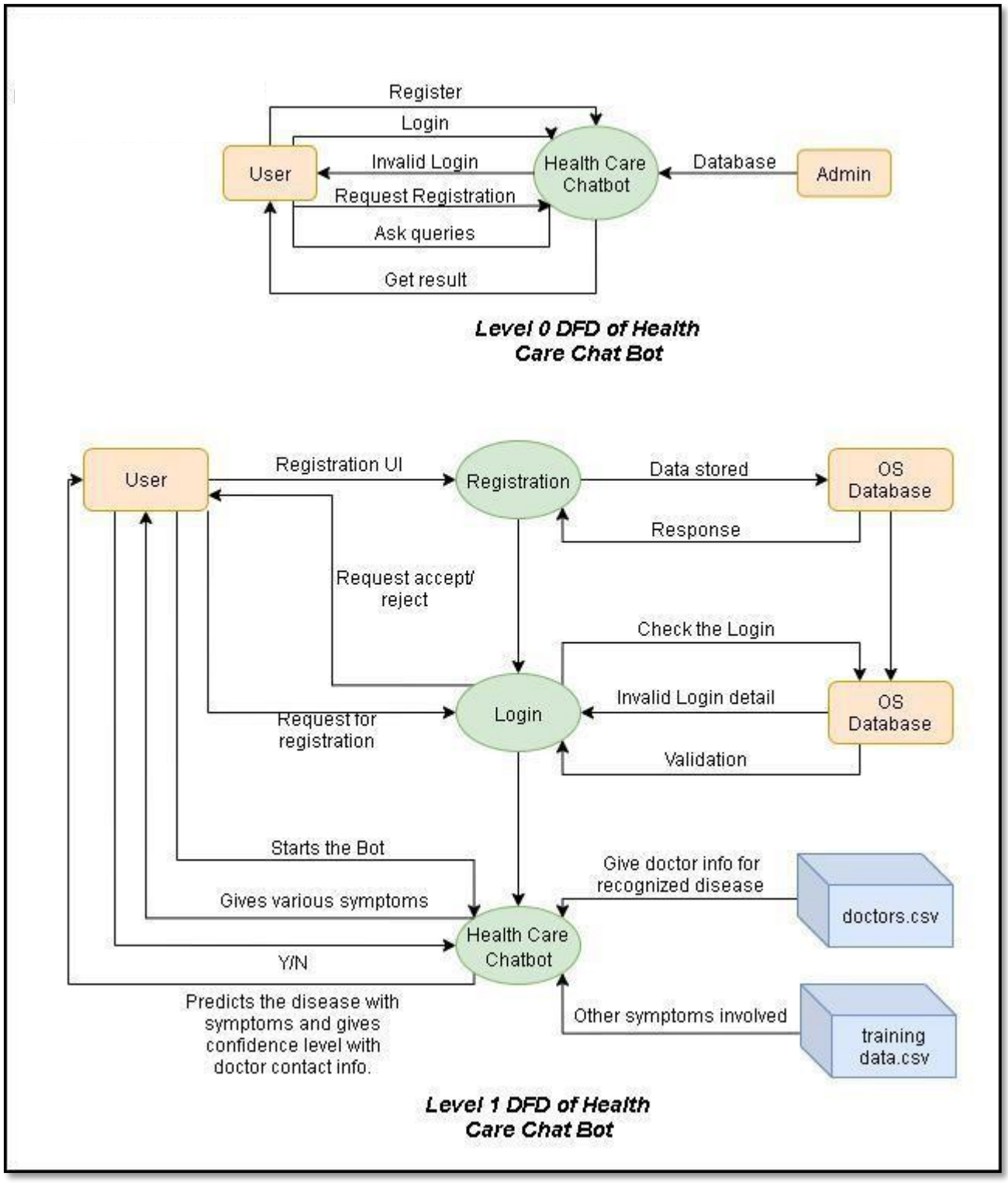
RESULT ANALYSIS AND VALIDATION

3.1 Flowchart(Fig 3.1)

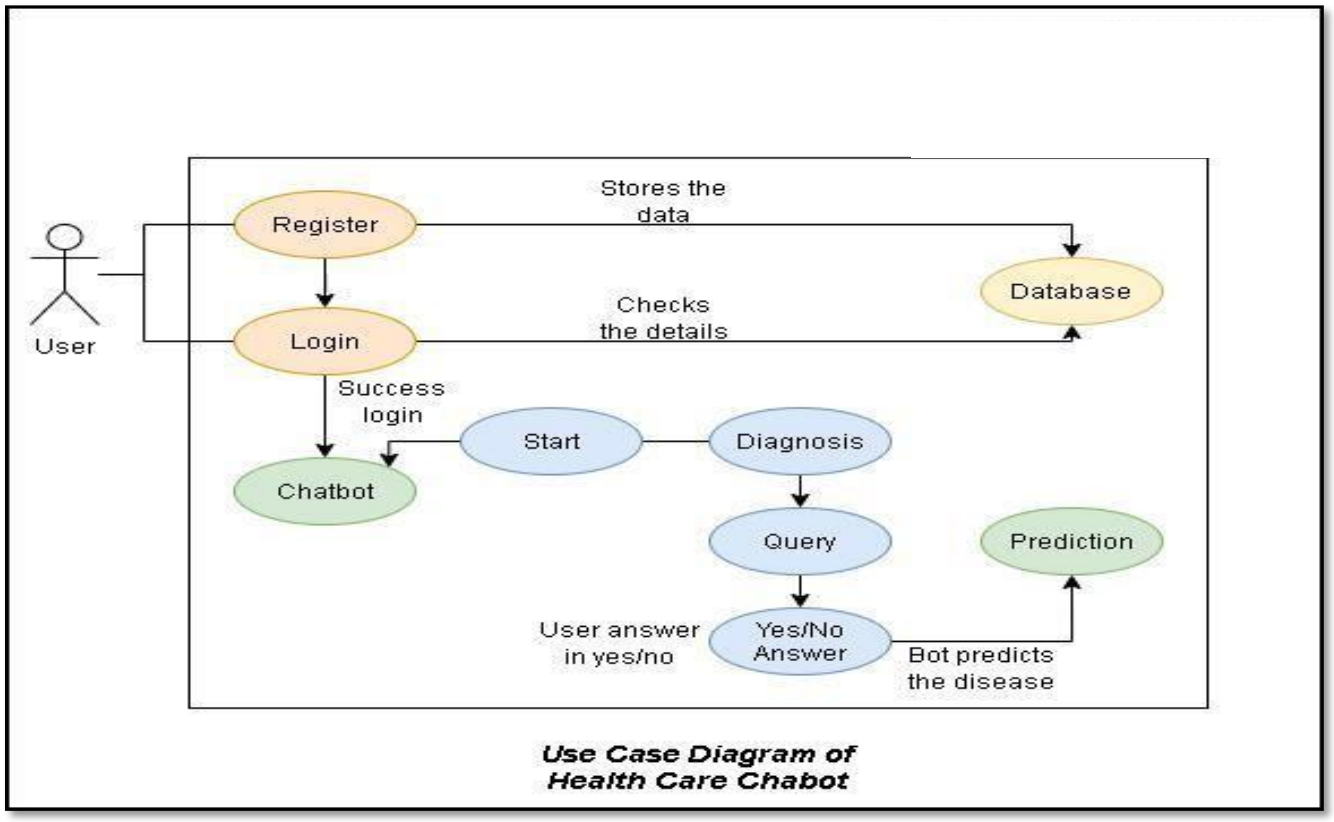


Flowchart of Health Care Chat Bot

3.2 Data Flow Diagram(Fig 3.2)



3.3 Use Case Diagram(Fig 3.3)



Use Case Diagram of Health Care Chabot

3.4 Safety

Ensuring the safety of healthcare chatbots is crucial to avoid any potential harm to patients or users. Here are some considerations for ensuring the safety of healthcare chatbots:

1. Accuracy of information: Ensure that the chatbot provides accurate and reliable information to users. The chatbot should be trained on reliable sources of information and updated regularly to reflect any changes in medical knowledge.

2. Clinical validation: Healthcare chatbots should be clinically validated to ensure that they are effective and safe to use. This can involve conducting clinical trials to evaluate the chatbot's performance in real-world settings.

3. User privacy and confidentiality: Healthcare chatbots should comply with regulations related to user privacy and confidentiality, such as HIPAA in the United States. The chatbot should not collect unnecessary user data and should use secure data transmission methods.

4. User education: Users should be educated on the limitations of the chatbot and when to seek medical advice from a healthcare professional. The chatbot should provide clear disclaimers and encourage users to consult a healthcare professional if necessary.

5. Risk management: Healthcare chatbots should have a risk management plan in place to identify and address any potential risks or adverse events. This can involve monitoring user feedback and implementing procedures for responding to any issues.

6. Regular maintenance and updates: Healthcare chatbots should be regularly maintained and updated to ensure that they continue to provide accurate and reliable information. This can involve monitoring user feedback and implementing updates to

address any issues.

Overall, ensuring the safety of healthcare chatbots requires a comprehensive approach that includes accuracy of information, clinical validation, user privacy and confidentiality, user education, risk management, and regular maintenance and updates. By addressing these considerations, healthcare chatbots can provide a valuable tool for patients and healthcare professionals while minimizing potential risks.

3.5 Social and Political Issues

Healthcare chatbots can have significant social and political implications that need to be considered. Here are some issues to keep in mind:

1. Access to healthcare: Healthcare chatbots can provide access to healthcare services for individuals who may not have had access otherwise. However, there may be concerns about whether the chatbot can adequately replace in-person consultations with healthcare professionals.

2. Disparities in access: There may be disparities in access to healthcare chatbots, particularly for individuals who do not have access to technology or who may not be comfortable using technology. This can exacerbate existing healthcare disparities.

3. Job displacement: As healthcare chatbots become more sophisticated, there may be concerns about job displacement for healthcare professionals, particularly in administrative or non-clinical roles.

4. Liability: Healthcare chatbots raise liability concerns for healthcare professionals and developers. If a chatbot provides incorrect or harmful advice, who is responsible for any resulting harm?

5. Privacy: Healthcare chatbots collect user data that can be sensitive and private. There may be concerns about how this data is collected, stored, and used.

6. Regulation: Healthcare chatbots are subject to regulatory frameworks, such as the FDA in the United States. However, the rapid pace of technological advancement may outpace regulatory frameworks, raising questions about how to regulate these technologies effectively.

Overall, healthcare chatbots have significant social and political implications that need to be carefully considered. By addressing these issues, healthcare chatbots can provide a valuable tool for patients and healthcare professionals while minimizing potential risks.

3.6 Analysis

To analyze the effectiveness of a healthcare chatbot, several factors can be considered:

1. Accuracy of information: A healthcare chatbot's effectiveness is dependent on the accuracy of the information it provides. The chatbot should be trained on reliable sources of medical information and updated regularly to reflect any changes in medical knowledge.

2. User satisfaction: The user's experience with the chatbot is an important measure of its effectiveness. The chatbot should be easy to use and provide useful information that addresses the user's needs.

3. User engagement: The chatbot's ability to engage users and keep them coming back is an important measure of its effectiveness. User engagement can be measured by tracking the number of conversations and user retention rates.

4. Clinical outcomes: The chatbot's ability to improve clinical outcomes is an important

measure of its effectiveness. This can be measured by comparing the outcomes of patients who use the chatbot to those who do not.

5. Cost-effectiveness: The chatbot's cost-effectiveness is an important consideration. A chatbot that can provide healthcare services at a lower cost than traditional healthcare services can be considered effective.

6. Regulatory compliance: Healthcare chatbots must comply with relevant regulatory frameworks, such as the FDA in the United States. Compliance with these regulations is an important measure of a chatbot's effectiveness.

Overall, to analyze the effectiveness of a healthcare chatbot, it is important to consider its accuracy of information, user satisfaction, user engagement, clinical outcomes, cost-effectiveness, and regulatory compliance. By evaluating these factors, it is possible to determine whether a healthcare chatbot is effective and can provide a valuable tool for patients and healthcare professionals.

CHAPTER-4
RESULT ANALYSIS AND VALIDATION

4.1.Project Implementation Technology

There are several technologies that can be used for implementing a healthcare chatbot project. Here are some commonly used technologies:

1. Natural Language Processing (NLP): NLP is a subfield of artificial intelligence (AI) that focuses on the interaction between computers and human language. NLP can be used to enable chatbots to understand and interpret natural language inputs from users.

2. Machine Learning (ML): ML is another subfield of AI that involves training computer algorithms to recognize patterns in data. ML can be used to help chatbots learn from user interactions and improve their responses over time.

3. Application Programming Interfaces (APIs): APIs can be used to integrate the chatbot with various healthcare systems such as electronic health records, appointment scheduling systems, and telemedicine platforms.

4. Cloud computing: Cloud computing can be used to provide the chatbot with scalable and reliable hosting, as well as access to various services and resources, such as speech-to-text conversion, translation services, and data storage.

5. Mobile application development: Mobile applications can be developed to provide users with a more convenient and accessible way to interact with the chatbot on their smartphones or tablets.

6. Voice recognition technology: Voice recognition technology can be used to enable users to interact with the chatbot using voice commands.

7. Augmented Reality (AR) and Virtual Reality (VR): AR and VR technologies can be used to create immersive healthcare experiences, such as virtual consultations, virtual surgeries, and patient education.

The choice of technology for implementing a healthcare chatbot project will depend on various factors, such as the project requirements, budget, and resources available. It is important to carefully evaluate and select the most appropriate technology stack based on the specific needs of the project.

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high- dimensional feature spaces.

4.2.Hardware Requirements

The hardware requirements for a healthcare chatbot project will depend on various factors such as the complexity of the chatbot, the number of users, and the anticipated workload. Here are some hardware requirements to consider:

1. Server: A server is required to host the chatbot application and to store the chatbot's

data. The server should have sufficient processing power, memory, and storage capacity to handle the anticipated workload. Cloud-based servers such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) can be used for scalability and reliability.

2. Database: A database is required to store the chatbot's data, including user information, chat history, and other relevant data. A database such as MySQL, MongoDB, or PostgreSQL can be used for this purpose.

3. Networking: A stable and reliable network connection is necessary to ensure smooth communication between the chatbot application and its users. The chatbot should be accessible through the internet or an intranet.

4. Mobile devices: If the chatbot is intended for use on mobile devices, such as smartphones or tablets, the hardware requirements will depend on the platform being used. For example, an iOS app will require an iPhone or iPad device, while an Android app will require an Android smartphone or tablet.

5. Headset or speakers: If the chatbot is intended to use voice recognition technology, users will require a headset or speakers to communicate with the chatbot.

6. Webcam: If the chatbot is intended to use video chat or AR/VR technology, users will require a webcam or camera.

7. Additional hardware: Depending on the specific requirements of the chatbot, additional hardware such as sensors, medical devices, or IoT devices may be required.

Overall, the hardware requirements for a healthcare chatbot project will depend on the specific needs of the project and the anticipated workload. It is important to carefully evaluate and select the appropriate hardware to ensure the chatbot is reliable, scalable, and able to meet the needs of its users.

In recent years, a great variety of hardware solutions for real-time TSR has been proposed. These include conventional (general purpose) computers, custom ASIC (application-specific integrated circuit) chips, field programmable gate arrays (FPGAs), digital sign processors (DSPs) and also graphic processing units

4.3.Software Requirements

The software requirements for a healthcare chatbot project will depend on various factors such as the programming language, framework, and libraries used to develop the chatbot, the hosting platform, and any additional software tools or applications used in the development process. Here are some software requirements to consider:

1. Programming language: The programming language used to develop the chatbot will depend on the developer's preference and the project requirements. Some commonly used programming languages for chatbot development include Python, Java, JavaScript, and Ruby.

2. Framework: A framework can be used to simplify and speed up the development process by providing a set of tools, libraries, and APIs. Some commonly used frameworks for chatbot development include Microsoft Bot Framework, Dialogflow, and Botpress.

3. Hosting platform: The hosting platform will depend on the developer's preference and the project requirements. Cloud-based hosting platforms such as AWS, Microsoft Azure, or Google Cloud Platform (GCP) can provide scalable and reliable hosting for the chatbot.

4. Database management system: A database management system such as MySQL, MongoDB, or PostgreSQL can be used to store the chatbot's data.

5. Text analytics and natural language processing (NLP) tools: Text analytics and NLP

tools can be used to enable the chatbot to understand and interpret natural language inputs from users. Some commonly used tools include Natural Language Toolkit (NLTK), spaCy, and Stanford CoreNLP.

6. Application Programming Interfaces (APIs): APIs can be used to integrate the chatbot with various healthcare systems such as electronic health records, appointment scheduling systems, and telemedicine platforms.

7. Development tools and IDEs: Development tools and Integrated Development Environments (IDEs) such as Visual Studio Code, Eclipse, or PyCharm can be used to write, test, and debug the chatbot's code.

Overall, the software requirements for a healthcare chatbot project will depend on the specific needs of the project and the programming language, framework, and tools used in the development process. It is important to carefully evaluate and select the appropriate software tools to ensure the chatbot is reliable, scalable, and able to meet the needs of its users.

In a software-based solution running on a Linux or window system with a 2.4-GHz dual core CPU is presented.

4.4.Experimental Setup

The experimental setup for a healthcare chatbot will depend on the specific research question or hypothesis being tested. However, here are some general considerations for setting up an experiment with a healthcare chatbot:

1. Research question or hypothesis: Clearly define the research question or hypothesis being tested. This will guide the experimental design and data analysis.

2. Study design: Determine the appropriate study design for the research question. For example, a randomized controlled trial (RCT) can be used to evaluate the effectiveness

of a chatbot intervention compared to a control group.

3. Participants: Define the inclusion and exclusion criteria for participants. Participants can be recruited from healthcare facilities or through online platforms.

4. Intervention: Define the chatbot intervention, including the chatbot's functionality, content, and delivery method. The chatbot can be developed using the software and hardware requirements mentioned above.

5. Control group: Define the control group, which can receive standard care or an alternative intervention.

6. Outcome measures: Define the outcome measures, such as changes in patient outcomes or user satisfaction, and how they will be measured.

7. Data collection and analysis: Determine the data collection methods, such as surveys or medical record review, and the statistical methods for analyzing the data.

8. Ethical considerations: Ensure that the study is conducted ethically and that participants' privacy and confidentiality are protected.

9. Pilot testing: Conduct a pilot test to ensure that the experimental setup is feasible and that any issues can be addressed before the main study.

Overall, the experimental setup for a healthcare chatbot will depend on the specific research question or hypothesis being tested. It is important to carefully plan and execute the study to ensure that the results are valid and reliable.

The main purpose of the scheme is to build the language gap between the user and health providers by giving immediate replies to the Questions asked by the user.

Today's people are more likely addicted to the internet but they are not concerned about their personal health. They avoid going to hospital for small problems which may become a major disease in future. Establishing question answer forums is becoming a simple way to answer those queries rather than browsing through the list of potentially relevant documents from the web. Many of the existing systems have some limitations such as there is no instant response given to the patients they have to wait for experts to acknowledge for a long time. Some of the processes may charge an amount to perform live chat or telephony communication with doctors online. The aim of this system is to replicate a person's discussion.

4.5 Code

healthcare_chatbotConsole.py > ...

```
######## A Healthcare Domain Chatbot to simulate the predictions of a General Physician ########
######## A pragmatic Approach for Diagnosis ############
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
training_dataset = pd.read_csv('Training.csv')
test_dataset = pd.read_csv('Testing.csv')

# Slicing and Dicing the dataset to separate features from predictions
X = training_dataset.iloc[:, 0:132].values
#print(X)
y = training_dataset.iloc[:, -1].values
#print(y)

# Dimensionality Reduction for removing redundancies
dimensionality_reduction = training_dataset.groupby(training_dataset['prognosis']).max()
#print(dimensionality_reduction)

# Encoding String values to integer constants
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
y = labelencoder.fit_transform(y)
#print(y)

# Splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Implementing the Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Saving the information of columns
cols     = training_dataset.columns
cols     = cols[:-1]


# Checking the Important features
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
features = cols

# Implementing the Visual Tree
from sklearn.tree import _tree
```

QuestionDiagonosisTkinter.py | newlogin.py | healthcare_chatbotConsole.py

healthcare_chatbotConsole.py > ...

```
# Method to simulate the working of a Chatbot by extracting and formulating questions
def execute_bot():

    print("Please reply with yes/Yes or no/No for the following symptoms")
    def print_disease(node):
        #print(node)
        node = node[0]
        #print(len(node))
        val  = node.nonzero()
        #print(val)
        disease = labelencoder.inverse_transform(val[0])
        return disease
    def tree_to_code(tree, feature_names):
        tree_ = tree.tree_
        #print(tree_)
        feature_name = [
            feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
            for i in tree_.feature
        ]
        #print("def tree({}):".format(", ".join(feature_names)))
        symptoms_present = []
        def recurse(node, depth):
            indent = "  " * depth
            if tree_.feature[node] != _tree.TREE_UNDEFINED:
                name = feature_name[node]
                threshold = tree_.threshold[node]
                print(name + " ?")
                ans = input()
                ans = ans.lower()
                if ans == 'yes':
                    val = 1
                else:
                    val = 0
                if  val <= threshold:
                    recurse(tree_.children_left[node], depth + 1)
                else:
                    symptoms_present.append(name)
                    recurse(tree_.children_right[node], depth + 1)
            else:
                present_disease = print_disease(tree_.value[node])
                print( "You may have " +  present_disease )
                print()
                red_cols = dimensionality_reduction.columns
                symptoms_given = red_cols[dimensionality_reduction.loc[present_disease].values[0].nonzero()]
                print("symptoms present  " + str(list(symptoms_present)))
                print()
                print("symptoms given "  +  str(list(symptoms_given)) )
                print()
```

QuestionDiagonosisTkinter.py | newlogin.py | healthcare_chatbotConsole.py

healthcare_chatbotConsole.py > ...

```
                    confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
                    print("confidence level is " + str(confidence_level))
                    print()
                    print('The model suggests:')
                    print()
                    row = doctors[doctors['disease'] == present_disease[0]]
                    print('Consult ', str(row['name'].values))
                    print()
                    print('Visit ', str(row['link'].values))
                    #print(present_disease[0])


        recurse(0, 1)

    tree_to_code(classifier,cols)



# This section of code to be run after scraping the data

doc_dataset = pd.read_csv('doctors_dataset.csv', names = ['Name', 'Description'])


diseases = dimensionality_reduction.index
diseases = pd.DataFrame(diseases)

doctors = pd.DataFrame()
doctors['name'] = np.nan
doctors['link'] = np.nan
doctors['disease'] = np.nan

doctors['disease'] = diseases['prognosis']


doctors['name'] = doc_dataset['Name']
doctors['link'] = doc_dataset['Description']

record = doctors[doctors['disease'] == 'AIDS']
record['name']
record['link']

# Execute the bot and see it in Action
execute_bot()

```

newlogin.py 1

newlogin.py > ...

```python
# import modules
from tkinter import *
import os
# Designing window for registration
def destroyPackWidget(parent):
    for e in parent.pack_slaves():
        e.destroy()
def register():
    global root,register_screen

    destroyPackWidget(root)
    register_screen=root
#    register_screen = Toplevel(main_screen)
    register_screen.title("Register")
    register_screen.geometry("2070x1250")

    global username
    global password
    global username_entry
    global password_entry
    username = StringVar()
    password = StringVar()

    Label(register_screen, text="Please enter the details below", bg="blue").pack()
    Label(register_screen, text="").pack()
    username_lable = Label(register_screen, text="Username * ")
    username_lable.pack()
    username_entry = Entry(register_screen, textvariable=username)
    username_entry.pack()
    password_lable = Label(register_screen, text="Password * ")
    password_lable.pack()
    password_entry = Entry(register_screen, textvariable=password, show='*')
    password_entry.pack()
    Label(register_screen, text="").pack()
    Button(register_screen, text="Register", width=10, height=1, bg="blue", command=register_user).pack()


# Designing window for login

def login():
    global login_screen
    login_screen = Toplevel(main_screen)
    login_screen.title("Login")
    login_screen.geometry("2070x1250")
    Label(login_screen, text="Please enter details below to login").pack()
    Label(login_screen, text="").pack()

    global username_verify
```

newlogin.py 1

newlogin.py > ...

```
    global password_verify

    username_verify = StringVar()
    password_verify = StringVar()

    global username_login_entry
    global password_login_entry

    Label(login_screen, text="Username * ").pack()
    username_login_entry = Entry(login_screen, textvariable=username_verify)
    username_login_entry.pack()
    Label(login_screen, text="").pack()
    Label(login_screen, text="Password * ").pack()
    password_login_entry = Entry(login_screen, textvariable=password_verify, show='*')
    password_login_entry.pack()
    Label(login_screen, text="").pack()
    Button(login_screen, text="Login", width=10, height=1, command=login_verify).pack()


# Implementing event on register button
def btnSucess_Click():
    global root
    destroyPackWidget(root)
def register_user():
    global root,username,password
    username_info = username.get()
    password_info = password.get()
    print("abc",username_info,password_info,"xyz")
    file = open(username_info, "w")
    file.write(username_info + "\n")
    file.write(password_info)
    file.close()

    username_entry.delete(0, END)
    password_entry.delete(0, END)

    Label(root, text="Registration Success", fg="green", font=("calibri", 11)).pack()
    Button(root,text="Click Here to proceed",command=btnSucess_Click).pack()


# Implementing event on login button

def login_verify():
    username1 = username_verify.get()
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    password_login_entry.delete(0, END)

```

newlogin.py 1

newlogin.py > ...

```
    list_of_files = os.listdir()
    if username1 in list_of_files:
        file1 = open(username1, "r")
        verify = file1.read().splitlines()
        if password1 in verify:
            login_sucess()

        else:
            password_not_recognised()

    else:
        user_not_found()


# Designing popup for login success

def login_sucess():
    global login_success_screen
    login_success_screen = Toplevel(login_screen)
    login_success_screen.title("Success")
    login_success_screen.geometry("250x200")
    Label(login_success_screen, text="Login Success").pack()
    Button(login_success_screen, text="OK", command=delete_login_success).pack()


# Designing popup for login invalid password

def password_not_recognised():
    global password_not_recog_screen
    password_not_recog_screen = Toplevel(login_screen)
    password_not_recog_screen.title("Success")
    password_not_recog_screen.geometry("250x200")
    Label(password_not_recog_screen, text="Invalid Password ").pack()
    Button(password_not_recog_screen, text="OK", command=delete_password_not_recognised).pack()


# Designing popup for user not found

def user_not_found():
    global user_not_found_screen
    user_not_found_screen = Toplevel(login_screen)
    user_not_found_screen.title("Success")
    user_not_found_screen.geometry("250x200")
    Label(user_not_found_screen, text="User Not Found").pack()
    Button(user_not_found_screen, text="OK", command=delete_user_not_found_screen).pack()


```

newlogin.py 1

newlogin.py > ...

```
143
144  # Deleting popups
145
146  def delete_login_success():
147      login_success_screen.destroy()
148
149
150  def delete_password_not_recognised():
151      password_not_recog_screen.destroy()
152
153
154  def delete_user_not_found_screen():
155      user_not_found_screen.destroy()
156
157
158  # Designing Main(first) window
159
160  def main_account_screen(frmmain):
161      main_screen=frmmain
162
163      main_screen.geometry("2070x1250")
164      main_screen.title("Account Login")
165      Label(main_screen,text="Select Your Choice", bg="blue", width="300", height="2", font=("Calibri", 13)).pack()
166      Label(main_screen,text="").pack()
167      Button(main_screen,text="Login", height="2", width="30", command=login).pack()
168      Label(main_screen,text="").pack()
169      Button(main_screen,text="Register", height="2", width="30", command=register).pack()
170
171
172
173  root = Tk()
174  main_account_screen(root)
175
176  root.mainloop()
177
```

QuestionDiagonosisTkinter.py

```python
# Importing the libraries
from tkinter import *
from tkinter import messagebox
import os
import webbrowser
import numpy as np
import pandas as pd
class HyperlinkManager:
    def __init__(self, text):
        self.text = text
        self.text.tag_config("hyper", foreground="blue", underline=1)
        self.text.tag_bind("hyper", "<Enter>", self._enter)
        self.text.tag_bind("hyper", "<Leave>", self._leave)
        self.text.tag_bind("hyper", "<Button-1>", self._click)

        self.reset()

    def reset(self):
        self.links = {}

    def add(self, action):
        # add an action to the manager.  returns tags to use in
        # associated text widget
        tag = "hyper-%d" % len(self.links)
        self.links[tag] = action
        return "hyper", tag

    def _enter(self, event):
        self.text.config(cursor="hand2")

    def _leave(self, event):
        self.text.config(cursor="")

    def _click(self, event):
        for tag in self.text.tag_names(CURRENT):
            if tag[:6] == "hyper-":
                self.links[tag]()
                return

# Importing the dataset
training_dataset = pd.read_csv('Training.csv')
test_dataset = pd.read_csv('Testing.csv')

# Slicing and Dicing the dataset to separate features from predictions
X = training_dataset.iloc[:, 0:132].values
Y = training_dataset.iloc[:, -1].values

```

QuestionDiagonosisTkinter.py

QuestionDiagonosisTkinter.py > HyperlinkManager > reset

```
# Dimensionality Reduction for removing redundancies
dimensionality_reduction = training_dataset.groupby(training_dataset['prognosis']).max()

# Encoding String values to integer constants
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
y = labelencoder.fit_transform(Y)

# Splitting the dataset into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Implementing the Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)

# Saving the information of columns
cols     = training_dataset.columns
cols     = cols[:-1]

# Checking the Important features
importances = classifier.feature_importances_
indices = np.argsort(importances)[::-1]
features = cols

# Implementing the Visual Tree
from sklearn.tree import _tree

# Method to simulate the working of a Chatbot by extracting and formulating questions
def print_disease(node):
        #print(node)
        node = node[0]
        #print(len(node))
        val  = node.nonzero()
        #print(val)
        disease = labelencoder.inverse_transform(val[0])
        return disease
def recurse(node, depth):
            global val,ans
            global tree_,feature_name,symptoms_present
            indent = "  " * depth
            if tree_.feature[node] != _tree.TREE_UNDEFINED:
                name = feature_name[node]
                threshold = tree_.threshold[node]
                yield name + " ?"

#                ans = input()
```

QuestionDiagonosisTkinter.py

QuestionDiagonosisTkinter.py > HyperlinkManager > reset

```
96                  ans = ans.lower()
97                  if ans == 'yes':
98                      val = 1
99                  else:
100                     val = 0
101                 if  val <= threshold:
102                     yield from recurse(tree_.children_left[node], depth + 1)
103                 else:
104                     symptoms_present.append(name)
105                     yield from recurse(tree_.children_right[node], depth + 1)
106             else:
107                 strData=""
108                 present_disease = print_disease(tree_.value[node])
109 #                print( "You may have " +  present_disease )
110 #                print()
111                 strData="You may have :" +  str(present_disease)
112 
113                 QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
114 
115                 red_cols = dimensionality_reduction.columns
116                 symptoms_given = red_cols[dimensionality_reduction.loc[present_disease].values[0].nonzero()]
117 #                print("symptoms present  " + str(list(symptoms_present)))
118 #                print()
119                 strData="symptoms present:  " + str(list(symptoms_present))
120                 QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
121 #                print("symptoms given "  +  str(list(symptoms_given)) )
122 #                print()
123                 strData="symptoms given: "  +  str(list(symptoms_given))
124                 QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
125                 confidence_level = (1.0*len(symptoms_present))/len(symptoms_given)
126 #                print("confidence level is " + str(confidence_level))
127 #                print()
128                 strData="confidence level is: " + str(confidence_level)
129                 QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
130 #                print('The model suggests:')
131 #                print()
132                 strData='The model suggests:'
133                 QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
134                 row = doctors[doctors['disease'] == present_disease[0]]
135 #                print('Consult ', str(row['name'].values))
136 #                print()
137                 strData='Consult '+ str(row['name'].values)
138                 QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
139 #                print('Visit ', str(row['link'].values))
140                 #print(present_disease[0])
141                 hyperlink = HyperlinkManager(QuestionDigonosis.objRef.txtDigonosis)
142                 strData='Visit '+ str(row['link'].values[0])
143                 def click1():
```

QuestionDiagonosisTkinter.py

QuestionDiagonosisTkinter.py > HyperlinkManager > reset

```
144                     webbrowser.open_new(str(row['link'].values[0]))
145                 QuestionDigonosis.objRef.txtDigonosis.insert(INSERT, strData, hyperlink.add(click1))
146                 #QuestionDigonosis.objRef.txtDigonosis.insert(END,str(strData)+'\n')
147                 yield strData
148
149 def tree_to_code(tree, feature_names):
150         global tree_,feature_name,symptoms_present
151         tree_ = tree.tree_
152         #print(tree_)
153         feature_name = [
154             feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
155             for i in tree_.feature
156         ]
157         #print("def tree({}):".format(", ".join(feature_names)))
158         symptoms_present = []
159 #        recurse(0, 1)
160
161 def execute_bot():
162 #    print("Please reply with yes/Yes or no/No for the following symptoms")
163     tree_to_code(classifier,cols)
164 # This section of code to be run after scraping the data
165 doc_dataset = pd.read_csv('doctors_dataset.csv', names = ['Name', 'Description'])
166
167 diseases = dimensionality_reduction.index
168 diseases = pd.DataFrame(diseases)
169
170 doctors = pd.DataFrame()
171 doctors['name'] = np.nan
172 doctors['link'] = np.nan
173 doctors['disease'] = np.nan
174
175 doctors['disease'] = diseases['prognosis']
176
177 doctors['name'] = doc_dataset['Name']
178 doctors['link'] = doc_dataset['Description']
179
180 record = doctors[doctors['disease'] == 'AIDS']
181 record['name']
182 record['link']
183
184 class QuestionDigonosis(Frame):
185     objIter=None
186     objRef=None
187     def __init__(self,master=None):
188         master.title("Question")
189         # root.iconbitmap("")
190         master.state("z")
191 #        master.minsize(700,350)
```

QuestionDiagonosisTkinter.py

QuestionDiagonosisTkinter.py > HyperlinkManager > reset

```
192         QuestionDigonosis.objRef=self
193         super().__init__(master=master)
194         self["bg"]="light blue"
195         self.createWidget()
196         self.iterObj=None
197 
198     def createWidget(self):
199         self.lblQuestion=Label(self,text="Question",width=12,bg="bisque")
200         self.lblQuestion.grid(row=0,column=0,rowspan=4)
201 
202         self.lblDigonosis = Label(self, text="Digonosis",width=12,bg="bisque")
203         self.lblDigonosis.grid(row=4, column=0,sticky="n",pady=5)
204 
205         # self.varQuestion=StringVar()
206         self.txtQuestion = Text(self, width=100,height=4)
207         self.txtQuestion.grid(row=0, column=1,rowspan=4,columnspan=20)
208 
209         self.varDiagonosis=StringVar()
210         self.txtDigonosis =Text(self, width=100,height=14)
211         self.txtDigonosis.grid(row=4, column=1,columnspan=20,rowspan=20,pady=5)
212 
213         self.btnNo=Button(self,text="No",width=12,bg="bisque", command=self.btnNo_Click)
214         self.btnNo.grid(row=25,column=0)
215         self.btnYes = Button(self, text="Yes",width=12,bg="bisque", command=self.btnYes_Click)
216         self.btnYes.grid(row=25, column=1,columnspan=20,sticky="e")
217 
218         self.btnClear = Button(self, text="Clear",width=12,bg="bisque", command=self.btnClear_Click)
219         self.btnClear.grid(row=27, column=0)
220         self.btnStart = Button(self, text="Start",width=12,bg="bisque", command=self.btnStart_Click)
221         self.btnStart.grid(row=27, column=1,columnspan=20,sticky="e")
222     def btnNo_Click(self):
223         global val,ans
224         global val,ans
225         ans='no'
226         str1=QuestionDigonosis.objIter.__next__()
227         self.txtQuestion.delete(0.0,END)
228         self.txtQuestion.insert(END,str1+"\n")
229 
230     def btnYes_Click(self):
231         global val,ans
232         ans='yes'
233         self.txtDigonosis.delete(0.0,END)
234         str1=QuestionDigonosis.objIter.__next__()
235 
236 #         self.txtDigonosis.insert(END,str1+"\n")
237 
238     def btnClear_Click(self):
239         self.txtDigonosis.delete(0.0,END)
```

QuestionDiagonosisTkinter.py

QuestionDiagonosisTkinter.py > HyperlinkManager > reset

```
            self.txtQuestion.delete(0.0,END)
        def btnStart_Click(self):
            execute_bot()
            self.txtDigonosis.delete(0.0,END)
            self.txtQuestion.delete(0.0,END)
            self.txtDigonosis.insert(END,"Please Click on Yes or No for the Above symptoms in Question")
            QuestionDigonosis.objIter=recurse(0, 1)
            str1=QuestionDigonosis.objIter.__next__()
            self.txtQuestion.insert(END,str1+"\n")


    class MainForm(Frame):
        main_Root = None
        def destroyPackWidget(self, parent):
            for e in parent.pack_slaves():
                e.destroy()
        def __init__(self, master=None):
            MainForm.main_Root = master
            super().__init__(master=master)
            master.geometry("2070x1250")
            master.title("Account Login")
            self.createWidget()
        def createWidget(self):
            self.lblMsg=Label(self, text="Health Care Chatbot", bg="PeachPuff2", width="300", height="2", font=("Calibri", 13))
            self.lblMsg.pack()
            self.btnLogin=Button(self, text="Login", height="2", width="300", command = self.lblLogin_Click)
            self.btnLogin.pack()
            self.btnRegister=Button(self, text="Register", height="2", width="300", command = self.btnRegister_Click)
            self.btnRegister.pack()
            self.lblTeam=Label(self, text="Made by:", bg="slateblue4", width = "250", height = "1", font=("Calibri", 13))
            self.lblTeam.pack()
            self.lblTeam1=Label(self, text="Divyansh Dodan", bg="RoyalBlue1", width = "250", height = "1", font=("Calibri", 13))
            self.lblTeam1.pack()
            self.lblTeam2=Label(self, text="Depanshi Tomar", bg="RoyalBlue2", width = "250", height = "1", font=("Calibri", 13))
            self.lblTeam2.pack()
            self.lblTeam3=Label(self, text="Sahil Thakur", bg="RoyalBlue3", width = "250", height = "1", font=("Calibri", 13))
            self.lblTeam3.pack()
            self.lblTeam3=Label(self, text="Shivam", bg="RoyalBlue3", width = "250", height = "1", font=("Calibri", 13))
            self.lblTeam3.pack()

        def lblLogin_Click(self):
            self.destroyPackWidget(MainForm.main_Root)
            frmLogin=Login(MainForm.main_Root)
            frmLogin.pack()
        def btnRegister_Click(self):
            self.destroyPackWidget(MainForm.main_Root)
            frmSignUp = SignUp(MainForm.main_Root)
            frmSignUp.pack()
```

QuestionDiagonosisTkinter.py

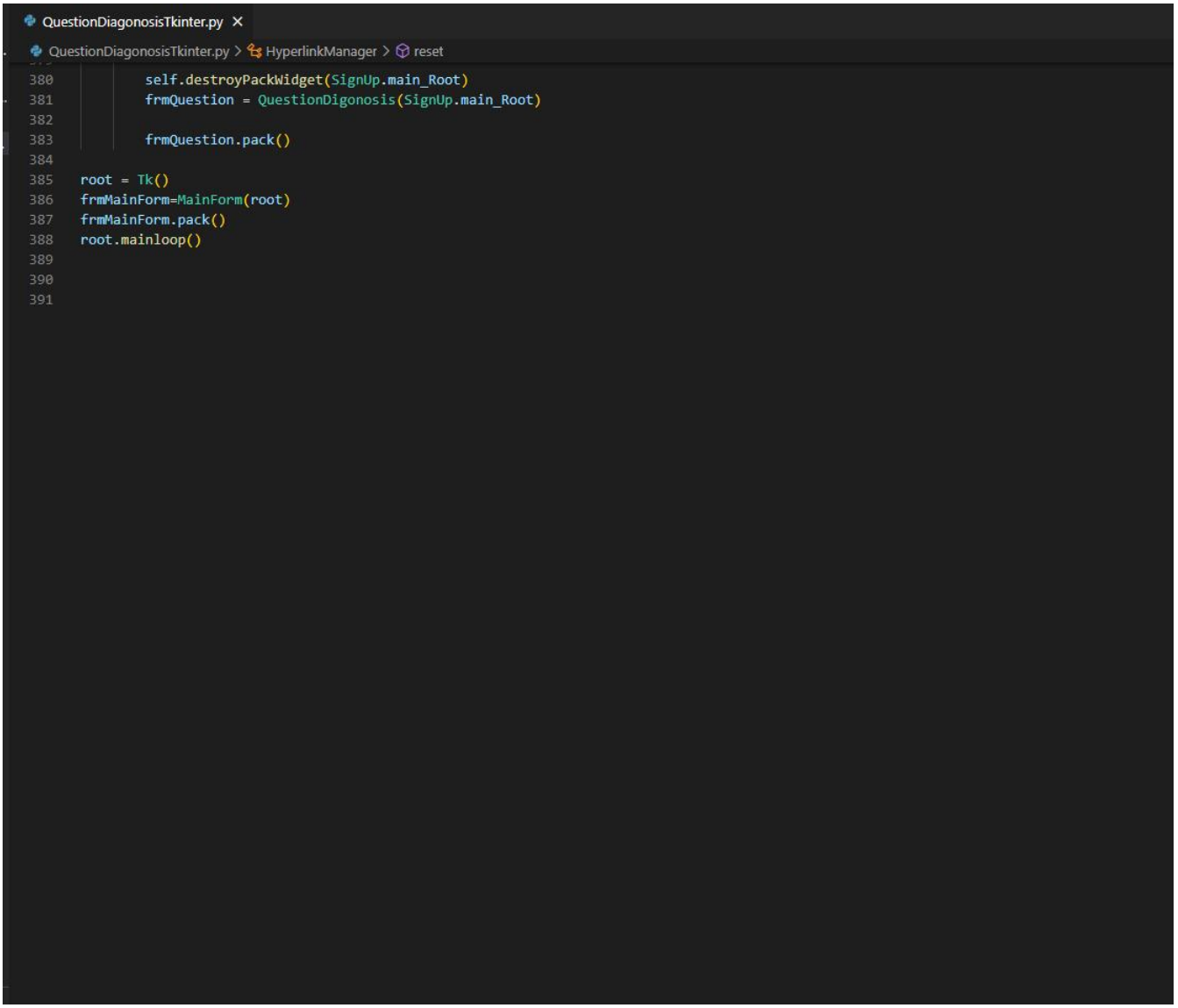
QuestionDiagonosisTkinter.py > HyperlinkManager > reset

```
289 class Login(Frame):
290     main_Root=None
291     def destroyPackWidget(self,parent):
292         for e in parent.pack_slaves():
293             e.destroy()
294     def __init__(self, master=None):
295         Login.main_Root=master
296         super().__init__(master=master)
297         master.title("Login")
298         master.geometry("2070x1250")
299         self.createWidget()
300     def createWidget(self):
301         self.lblMsg=Label(self, text="Please enter details below to login",bg="blue")
302         self.lblMsg.pack()
303         self.username=Label(self, text="Username * ")
304         self.username.pack()
305         self.username_verify = StringVar()
306         self.username_login_entry = Entry(self, textvariable=self.username_verify)
307         self.username_login_entry.pack()
308         self.password=Label(self, text="Password * ")
309         self.password.pack()
310         self.password_verify = StringVar()
311         self.password_login_entry = Entry(self, textvariable=self.password_verify, show='*')
312         self.password_login_entry.pack()
313         self.btnLogin=Button(self, text="Login", width=10, height=1, command=self.btnLogin_Click)
314         self.btnLogin.pack()
315     def btnLogin_Click(self):
316         username1 = self.username_login_entry.get()
317         password1 = self.password_login_entry.get()
318 
319 #         messagebox.showinfo("Failure", self.username1+":"+password1)
320         list_of_files = os.listdir()
321         if username1 in list_of_files:
322             file1 = open(username1, "r")
323             verify = file1.read().splitlines()
324             if password1 in verify:
325                 messagebox.showinfo("Sucess","Login Sucessful")
326                 self.destroyPackWidget(Login.main_Root)
327                 frmQuestion = QuestionDigonosis(Login.main_Root)
328                 frmQuestion.pack()
329             else:
330                 messagebox.showinfo("Failure", "Login Details are wrong try again")
331         else:
332             messagebox.showinfo("Failure", "User not found try from another user\n or sign up for new user")
333 
334 class SignUp(Frame):
335     main_Root=None
336     print("SignUp Class")
```

QuestionDiagonosisTkinter.py

QuestionDiagonosisTkinter.py > HyperlinkManager > reset

```
337     def destroyPackWidget(self,parent):
338         for e in parent.pack_slaves():
339             e.destroy()
340     def __init__(self, master=None):
341         SignUp.main_Root=master
342         master.title("Register")
343         super().__init__(master=master)
344         master.title("Register")
345         master.geometry("2070x1250")
346         self.createWidget()
347     def createWidget(self):
348         self.lblMsg=Label(self, text="Please enter details below", bg="blue")
349         self.lblMsg.pack()
350         self.username_lable = Label(self, text="Username * ")
351         self.username_lable.pack()
352         self.username = StringVar()
353         self.username_entry = Entry(self, textvariable=self.username)
354         self.username_entry.pack()
355 
356         self.password_lable = Label(self, text="Password * ")
357         self.password_lable.pack()
358         self.password = StringVar()
359         self.password_entry = Entry(self, textvariable=self.password, show='*')
360         self.password_entry.pack()
361         self.btnRegister=Button(self, text="Register", width=10, height=1, bg="blue", command=self.register_user)
362         self.btnRegister.pack()
363 
364 
365     def register_user(self):
366         file = open(self.username_entry.get(), "w")
367         file.write(self.username_entry.get() + "\n")
368         file.write(self.password_entry.get())
369         file.close()
370 
371         self.destroyPackWidget(SignUp.main_Root)
372 
373         self.lblSucess=Label(root, text="Registration Success", fg="green", font=("calibri", 11))
374         self.lblSucess.pack()
375 
376         self.btnSucess=Button(root, text="Click Here to proceed", command=self.btnSucess_Click)
377         self.btnSucess.pack()
378     def btnSucess_Click(self):
379 
380         self.destroyPackWidget(SignUp.main_Root)
381         frmQuestion = QuestionDigonosis(SignUp.main_Root)
382 
383         frmQuestion.pack()
384 
```



4.6 Testing

Without a well-thought testing effort, the project will undoubtedly fail overall and will impact the entire operational performance of the solution. With a poorly tested solution, the support and maintenance cost will escalate exponentially, and the reliability of the solution will be poor.

Therefore, project managers need to realize that the testing effort is a necessity, not merely as an ad hoc task that is the last hurdle before deployment.

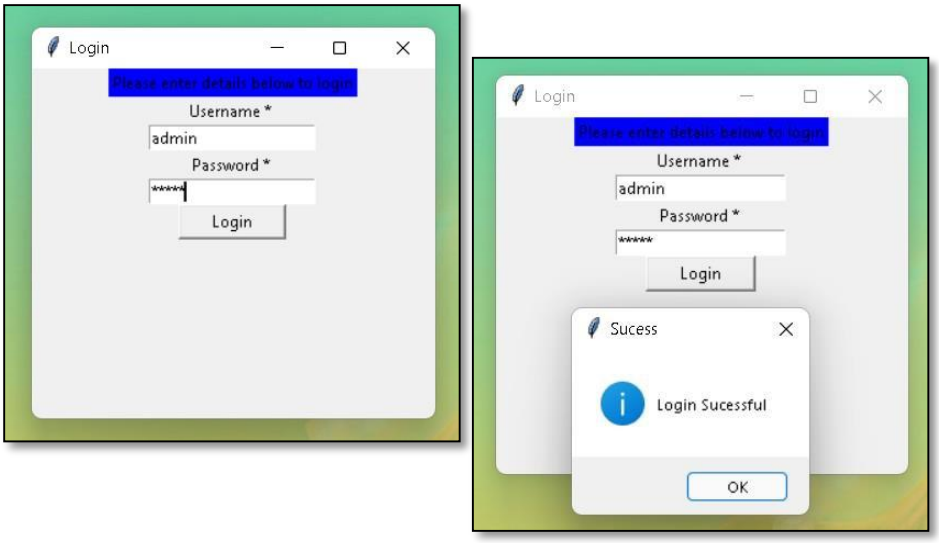
The project manager should pay specific attention to developing a complete testing plan

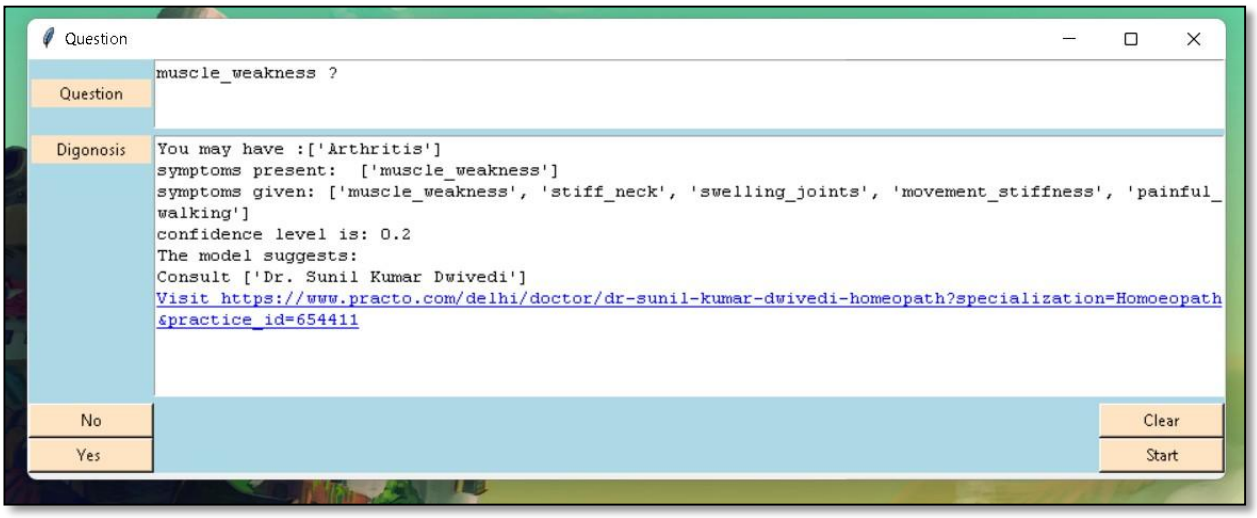
and schedule. At this stage, the project manager should have realized that this effort would have to be accommodated within the project budget, as many of the testing resources will be designing, testing, and validating the solution throughout the entire project life cycle—and this consumes work-hours and resources.

The testing effort begins at the initial project phase (i.e. preparing test plans) and continues throughout until the closure phase.

CHAPTER-5
CONCLUSION AND FUTURE WORK

5.1 Output





5.2 Advantages

Omani-capable

The chat bot converses seamlessly across multiple digital channels and retains data and context for a seamless experience. In best cases, even passing that information to a live agent if needed.

Free to Explore

The chat bot can reach, consume, and process vast amounts of data– both structured and unstructured–to surface insights from any source - to gather relevant data to solve customer issues quickly.

Autonomous Reasoning

The chat bot can perform complex reasoning without human intervention. For example, a great Service chat bot should be able to infer solutions based on relevant case histories.

Pre-Trained

The chat bot is pre-trained to understand brand-specific or industry-specific knowledge and terms. Even better, it's pre-configured to resolve common customer requests of a particular industry.

Register/Log-in

To access this chat bot and individual needs to register and then use the registration ID to log in to access the features.

User Interface

A user friendly interface which is engaging and easy to access.

5.3 Disadvantages

Complex Interface

Chat bots are often seen to be complicated and require a lot of time to understand user's requirement. It is also the poor processing which is not able to filter results in time that can annoy people.

Inability to Understand

Due to fixed programs, chat bots can be stuck if an unsaved query is presented in front of them. This can lead to customer dissatisfaction and result in loss. It is also the multiple messaging that can be taxing for users and deteriorate the overall experience on the website.

Time-Consuming

Chat bots are installed with the motive to speed-up the response and improve customer interaction. However, due to limited data-availability and time required for self-updating, this process appears more time-taking and expensive. Therefore, in place of attending several customers at a time, chat bots appear confused about how to communicate with people.

Zero Decision Making

Chat bots are known for being infamous because of their inability to make decisions. A similar situation has landed big companies like Microsoft etc. in trouble when their chat bot went on making a racist rant. Therefore, it is critical to ensure proper programming of your chat bot to prevent any such incident which can hamper your brand.

Poor Memory

Chat bots are not able to memorize the past conversation which forces the user to type the same thing again & again. This can be cumbersome for the customer and annoy them because of the effort required. Thus, it is important to be careful while designing chat bots and make sure that the program is able to comprehend user queries and respond accordingly.

5.4 FUTURE SCOPE

Chat bots are a thing of the future which is yet to uncover its potential but with its rising popularity and craze among companies, they are bound to stay here for long. Machine learning has changed the way companies were communicating with their customers. With new platforms to build various types of chat bots being introduced, it is of great excitement to witness the growth of a new domain in technology while surpassing the previous threshold.

The future scope for healthcare chatbots is vast and exciting, with the potential to transform the way healthcare services are delivered. Here are some potential future developments in healthcare chatbots:

1. Integration with wearable devices: Chatbots can be integrated with wearable devices such as smartwatches and fitness trackers to collect real-time data about patients' health and activities. This data can be used to provide personalized recommendations and guidance to patients for managing their health conditions.

2. Improved natural language processing: Natural language processing technology is constantly improving, and future healthcare chatbots can be developed to understand and interpret patients' natural language more accurately. This will enable more natural and effective communication between patients and chatbots.

3. Use of virtual and augmented reality: Future healthcare chatbots can incorporate virtual and augmented reality technologies to provide patients with immersive and interactive experiences. For example, a chatbot can use augmented reality to provide patients with visual representations of their health conditions or virtual reality to simulate medical procedures.

4. Expansion of services: Healthcare chatbots can be developed to offer a wider range of healthcare services, such as prescription refills, appointment scheduling, and insurance verification. This will enhance the convenience and accessibility of healthcare services for patients.

5. Integration with other healthcare technologies: Healthcare chatbots can be integrated with other healthcare technologies, such as electronic health records (EHRs) and telemedicine platforms, to streamline healthcare services and improve patient outcomes.

Overall, the future scope for healthcare chatbots is vast and promising. As technology continues to evolve and improve, healthcare chatbots will play an increasingly

important role in improving the accessibility, convenience, and quality of healthcare services.

5.5 CONCLUSION

In conclusion, healthcare chatbots have emerged as a promising solution to improve healthcare services, enhance patient outcomes, and reduce costs. The literature review shows that the use of healthcare chatbots is increasing steadily, with various solutions proposed to address specific challenges and improve patient experiences.

The proposed project for a healthcare chatbot can incorporate some of the solutions proposed by researchers, such as personalized chatbots, multilingual chatbots, mental health chatbots, telemedicine chatbots, and chatbots for health education. These solutions aim to provide tailored and accessible healthcare services to patients based on their unique needs.

The future scope for healthcare chatbots is vast, with the potential to transform the way healthcare services are delivered. By incorporating advanced technologies such as virtual and augmented reality, natural language processing, and integration with wearable devices and other healthcare technologies, healthcare chatbots can provide even more personalized and immersive healthcare experiences to patients.

Overall, healthcare chatbots have shown great promise in improving the accessibility, convenience, and quality of healthcare services. As technology continues to evolve, healthcare chatbots will play an increasingly important role in transforming the healthcare industry and improving patient outcomes.

Thus, we can conclude that this system giving the accurate result. As we are using large data set which will ensures the better performance. Thus we build up a system which is useful for people to detect the disease by typing symptoms

REFERENCES

Here are some references for healthcare chatbots:

1. Lee, J. (2019). The Use of Chatbots in Healthcare: A Systematic Review of Literature. Journal of Medical Systems, 43(8), 1-10.

2. Roy, D., & Banerjee, S. (2021). An Overview of Chatbot Applications in Healthcare. Journal of Healthcare Engineering, 2021, 1-11.

3. Abd-Alrazaq, A., Alajlani, M., Alalwan, A., Bewick, B. M., Gardner, P., & Househ, M. (2019). An Overview of the Features of Chatbots in Mental Health: A Scoping Review. International Journal of Medical Informatics, 132, 103978.

4. Lin, Y., Zheng, K., & Chen, Y. (2021). A Multilingual Chatbot for Patient Education and Health Promotion: Development and Usability Study. Journal of Medical Internet Research, 23(1), e21714.

5. Laranjo, L., Dunn, A. G., Tong, H. L., Kocaballi, A. B., Chen, J., Bashir, R., ... & Lau, A. Y. (2018). Conversational agents in healthcare: A systematic review. Journal of the American Medical Informatics Association, 25(9), 1248-1258.

6. Statista Research Department. (2021). Chatbot market size worldwide 2018-2029. Statista.

7. Wang, X., & Li, Y. (2020). A survey on chatbot design techniques in speech and natural language processing. Frontiers of Computer Science, 14(3), 425-447.

8. World Health Organization. (2020). Digital health. WHO.

9. Zeng, Y., Zhu, W., Shen, L., & Fu, W. (2019). A personalization-based healthcare chatbot framework. Journal of Medical Systems, 43(8), 1-10.

10. Zhang, Y., Wang, Y., & Liu, Y. (2020). Design of an intelligent healthcare chatbot system based on deep learning. Journal of Healthcare Engineering, 2020, 1-11.

11. https://en.wikipedia.org/wiki/Chatbot

12. https://en.wikipedia.org/wiki/Disease

13. https://data-flair.training/blogs/python-chatbot-project/

14. https://www.youtube.com/playlist?list=PLQVvvaa0QuDdc2k5dwtDTyT9aCja0on8j