**MEDBOT: CONVERSATIONAL ARTIFICIAL INTELLIGENCE POWERED CHATBOT FOR DELIVERING TELE-HEALTH AFTER COVID-19**

**ABSTRACT:**

Telemedicine can be used by medical practitioners to connect with their patients during the recent Coronavirus outbreak, whilst attempting to reduce COVID-19 transmission among patients and clinicians. Amidst the pandemic, Telemedicine has the potential to help by permitting patients to receive supportive care without having to physically visit a hospital by using a conversational artificial intelligence-based application for their treatment. Thus, telehealth will rapidly and radically transform in-person care to remote consultation of patients. Because of this, it developed a Multilingual Conversational Bot based on Natural Language Processing (NLP) to provide free primary healthcare education, information, advice to chronic patients. The study introduces a novel computer application acting as a personal virtual doctor that has been opportunely designed and extensively trained to interact with patients like human beings. This application is based upon a server less architecture and it aggregates the services of a doctor by providing preventive measures, home remedies, interactive counseling sessions, healthcare tips, and symptoms covering the most prevalent diseases in rural India. The paper proposes a conversational bot “Aapka Chikitsak” on Google Cloud Platform (GCP) for delivering telehealth in India to increase the patient's access to healthcare knowledge and leverage the potentials of artificial intelligence to bridge the gap of demand and supply of human healthcare providers. This conversational application has resulted in reducing the barriers for access to healthcare facilities and procures intelligent consultations remotely to allow timely care and quality treatment, thereby effectively assisting the society.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS**  **LIST OF ABBREVIATIONS**  **LIST OF TABLES** | i  v  vii  xi  xii |
| 1. | **CHAPTER 1 : INTRODUCTION**   * + GENERAL   + OBJECTIVE   + EXISTING SYSTEM   1.3.1EXISTINGSYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 :PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.2.3 MODULE DIAGRAM  2.2.4GIVEN INPUTAND EXPECTED OUTPUT  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4 :SYSTEM DESIGN**  **4.1 general**  4.1.1 activity diagram  4.1.2 USE CASE DIAGRAM  4.1.3 DATA FLOW DIAGRAM  4.1.4SEQUENCE DIAGRAM  4.1.5 COLLABORATION DIAGRAM  4.1.6CLASS DIAGRAM  4.1.7 SYSTEM ARCHITECTURE  4.1.8 OBJECT DIAGRAM  4.1.9 STATE DIAGRAM  4.1.10 COMPONENT DIAGRAM  4.1.11 E-R DIAGRAM  4.2 DATABASE DESIGN (ALL LEVEL |  |
| 5. | **CHAPTER 5 :SOFTWARE SPECIFICATION**  5.1 general |  |
| 6. | **CHAPTER 6 :IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION  6.3 DATA BASE TABLE STRUCTURE |  |

|  |  |  |
| --- | --- | --- |
| 7. | **CHAPTER 7 :SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 :SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 :**  **APPLICATIONS AND FUTURE ENHANCEMENT**  9.1 GENERAL  9.2 APPLICATIONS  9.3 FUTURE ENHANCEMENTS |  |
| **10** | **CHAPTER 10 :**  10.1CONCLUSION  10.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2 | Activity Diagram |  |
| 4.3 | Use case Diagram |  |
| 4.4 | Data flow diagram |  |
| 4.5 | Sequence diagram |  |
| 4.6 | Collaboration diagram |  |
| 4.7 | Class diagram |  |
| 4.8 | Architecture Diagram |  |
| 4.9 | State Diagram |  |
| 4.1 | Component Diagram |  |
| 4.12 | E-R Diagram |  |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *+ public*  *-private*  *# protected*  *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class A  Class B  Class B  Class A | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class B  Class B  Class A  Class A | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processes. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |

|  |  |  |  |
| --- | --- | --- | --- |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

One of the major challenges that India as a country faces is to cater to good quality and affordable healthcare to its growing population. The World Health Report issued by WHO has ranked India’s healthcare system at 112 out of 190 countries [1]. This inaccessibility of healthcare facilities especially in rural India and the intricacy in accessing means of transport further causes patients to postpone their treatment, or opt for medical facilities that may be closer but at the applications to access and record the patient's data. At the patient's end, it is a cheaper alternative; AI-enabled virtual assistants that can render 24x7 care to a wide variety of patients. People suffering from chronic diseases, disabled patients, and patients living in rural and farther areas would benefit most from such powerful virtual assistants’ tools. These systems have many advantages: reduced time on the part of physicians, improved security of patient data, on demand healthcare information, thus, making healthcare accessible and affordable for all with an intuitive interface [4]. This paper describes integrating chatbots into telemedicine. Our solution “Aapka Chikitsak” includes a Multilingual Voice Application based on Natural Language Processing to provide primary healthcare education and advice to chronic patients and women needing antenatal care. Using AI, it converts the user's speech to text which is processed and understood using natural language processing, and an output is generated which is then converted back to speech and returned to the user. Our software covers the most prevalent diseases in rural India with a special emphasis on women's healthcare. Our application imparts the services of a doctor by providing preventive measures, home remedies, healthcare tips, symptoms, and location-based diet recommendations. Prevention is always better than cure and by having a personal healthcare assistant; our software will be extremely beneficial and provide an efficient and instant solution to those in need.

* 1. **OBJECTIVE**

The aim of this chapter is to provide a comparative analysis of different artificial intelligence approaches for Powered Chatbot for Delivering Tele-Health after COVID-19. Here we have significantly enhanced the training process to solvePre-processing, Token Identification.

**1.3 PROBLEM STATEMENT**

Amidst the pandemic, Telemedicine has the potential to help by permitting patients to receive supportive care without having to physically visit a hospital by using a conversational artificial intelligence-based application for their treatment. Thus, telehealth will rapidly and radically transform in-person care to remote consultation of patients. Because of this, it developed a Multilingual Conversational Bot based on Natural Language Processing (NLP) to provide free primary healthcare education, information, advice to chronic patients. The study introduces a novel computer application acting as a personal virtual doctor that has been opportunely designed and extensively trained to interact with patients like human beings. This application is based upon a server less architecture and it aggregates the services of a doctor by providing preventive measures, home remedies, interactive counseling sessions, healthcare tips, and symptoms covering the most prevalent diseases in rural India.

* 1. **Existing System**

One of the major challenges that India as a country faces is to cater to good quality and affordable healthcare to its growing population. The World Health Report issued by WHO has ranked India’s healthcare system at 112 out of 190 countries. This inaccessibility of healthcare facilities especially in rural India and the intricacy in accessing means of transport further causes patients to postpone their treatment, or opt for medical facilities that may be closer but at the same time are not cost-efficient and well-matched to their medical needs.

* + 1. **Disadvantage Of Existing System**
* More cost-efficient
* Well-matched to their medical needs

**1.3.2 LITERATURE SURVEY**

**TITLE:** Balinese Historian Chatbot using Full-Text Search and Artificial Intelligence Markup Language Method

**YEAR:** 2019

**AUTHOR:** I Made Sukarsa, I Putu Agung Bayupati

**DESCRIPTION**

In the era of technology, various information could be obtained quickly and easily. The history of Bali is one of the information that could be obtained. Balinese have known their history through Babad and stories which are told through generations. Babad is traditional historical writing which tells important event that has happened. As technology evolves, Balinese‘s interest in studying their own history has been decreased. It is caused by people interest in studying history books and chronicles tend to decrease over time. Therefore, an innovation of technology, which able to convert historical data from printed media to digital media, is needed. The technology that could be used is Chatbot technology; a computer program that could carry out conversations. Chatbot technology is used to make people learning history easily by using Instant Messenger LINE as a platform to communicate. This Chatbot uses two methods, namely the Artificial Intelligence Markup Language method and the Full-Text Search method. The Artificial Intelligence Markup Language method is used as the process of making characteristic of questions and answers. The Full-Text Search method is the process of matching answers based on user input. This chatbot only uses Indonesian to communicate. The results of this study are a Chatbot that could be accessed by using Instant Messenger LINE and could communicate like historian expert. In the era of technology, various information could be obtained quickly and easily.

**TITLE**: Rural India’s access to healthcare patchy study

**AUTHOR**: Pratyush Kumar1, Raman Kumar2

**YEAR**:2019

**DESCRIPTION**:

Half the world's people currently live in rural and remote areas. About 70% of the world's 1.4 billion people who are extremely poor live in rural areas. The problem is that the majority of healthcare providers prefer to serve in urban areas. Only a comprehensive and systematic approach can address these inequities. India, the largest democratic republic in the world, possesses 2.4% of the world's land area and supports 16% of the world's population. According to census 2011, 68.84% of population resides in rural areas. Nearly 86% of all the medical visits in India are made by rural inhabitants with a majority still traveling more than 100 km to avail healthcare facility, of which 70%–80% is born out of pocket landing them in poverty. A country's approach must systematically and simultaneously address legal coverage and rights, health worker shortages, extension of healthcare protection, and quality of care. Only then can equitable access for all be fully achieved. Those living in rural areas have access to health protection and services that meet the criteria of availability, affordability, accessibility, acceptability, and quality. Family medicine as a broad specialty has its role from womb till tomb. Family medicine is defined as a specialty of medicine which is concerned with providing comprehensive care to individuals and families by integrating biomedical, behavioral, and social sciences. As an academic discipline, it includes comprehensive healthcare services, education, and research. A family doctor provides primary and continuing care to the entire family within the communities; addresses physical, psychological, and social problems; and coordinates comprehensive healthcare services with other specialists, as needed. The practitioners in family medicine can play an important role in providing healthcare services to the suffering humanity.

**TITLE**: Beyond patient monitoring: Conversational agents role in telemedicine & healthcare support for home-living elderly individuals

**AUTHOR**: Ahmed Fadhil

**YEAR**:2018

**DESCRIPTION:**

There is a need for systems to dynamically interact with ageing populations to gather information, monitor health condition and provide support, especially after hospital discharge or at-home settings. Several smart devices have been delivered by digital health, bundled with telemedicine systems, smartphone and other digital services. While such solutions offer personalised data and suggestions, the real disruptive step comes from the interaction of new digital ecosystem, represented by chatbots. Chatbots will play a leading role by embodying the function of a virtual assistant and bridging the gap between patients and clinicians. Powered by AI and machine learning algorithms, chatbots are forecasted to save healthcare costs when used in place of a human or assist them as a preliminary step of helping to assess a condition and providing self-care recommendations. This paper describes integrating chatbots into telemedicine systems intended for elderly patient after their hospital discharge. The paper discusses possible ways to utilise chatbots to assist healthcare providers and support patients with their condition.

**TITLE**: Chatbots Meet eHealth: Automatizing Healthcare

**AUTHOR**: Ahmed Fadhil

**YEAR**:2017

**DESCRIPTION**:

The aim of this work is to investigate the effectiveness of novel human-machine interaction paradigms for eHealth applications. In particular, we propose to replace usual human-machine interaction mechanisms with an approach that leverages a chat-bot program, opportunely designed and trained in order to act and interact with patients as a human being. Moreover, we have validated the proposed interaction paradigm in a real clinical context, where the chat-bot has been employed within a medical decision support system having the goal of providing useful recommendations concerning several disease prevention pathways. More in details, the chat-bot has been realized to help patients in choosing the most proper disease prevention pathway by asking for different information (starting from a general level up to specific pathways questions) and to support the related prevention check-up and the final diagnosis. Preliminary experiments about the effectiveness of the proposed approach are reported

**TITLE**: Android Based e-Voting Mobile App Using Google Firebase as BaaS.

**AUTHOR**: Urmil Bharti,Deepali Bajaj

**YEAR**:2019

**DESCRIPTION**:

Making choices and choosing from options are always a part of life and everyone wants options to choose from. Similarly when it comes to voting and elections, it gives power of making choices to the people. Voting is a democratic way of making decisions. Counting Ballots takes a long time that causes delayed results. Furthermore calculating results could be biased and time consuming which causes voters to wait for the results. In today’s scenario as everything is pacing up and new ideas and inventions are always appreciated, our mobile app “Adhikaar” is also one of them. This Android voting app is more efficient and convenient to conduct and manage elections as compared to traditional manual methods. The app has a simple and interactive GUI for voting system and maintains its database using Google Firebase platform. Firebase is a Backend-as-a-Service—BaaS that lets users build more powerful, secure and scalable apps. Adhikaar enables user to cast their vote from anywhere, anytime without waiting in long queues. This app is purely eco-friendly since no paper is required. Election results are calculated automatically and declared instantly thus reducing human effort and chances of human errors.

* 1. **Proposed System**

Artificial Intelligence (AI) powered chatbots are playing a leading role by exemplifying the function of a virtual assistant that could manage a conversation via speech or textual methods. It makes use of voice queries to get answers, perform actions and recommendations according to user needs. They are adaptable to the user's individual language usages, searches, and preferences with continuing use. A conversational bot with a voice and/or chat interface can play a principal role by overcoming the current barriers towards making primary healthcare affordable, accessible, and potentially sustainable in the new digital economy. With the advent of AI, virtual assistants can be seen penetrating to the nook and corner of the world.

* + 1. **Advantages of Proposed System**
* It is robust
* High efficiency.
* More Quality it providing

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

In this chapter, various supervised artificial intelligence approaches are used. This section provides a general description of these approaches.

* 1. **METHODOLOGIES**

**MODULE NAMES:**

* Voice Assistant
* Speech Recognition
* Text-to-speech (TTS)
* User Module
* Information Extraction

**Voice Assistant:**

The key here is voice. A voice associate is an advanced accomplice that makes use of voice confirmation, talk mix, and everyday language getting geared up (nlp) to offer an enterprise by means of a selected application. Improvement is unremittingly progressing and changing over, and the voice associate market will develop close-with the aid of it. In april 2015, the exam company gartner anticipated that earlier than the of completing of 2020, 50 percentage of affiliation with improvement might be through "talks" with sharp machines, huge amounts of them by means of strategies for voice.

**Speech Recognition**

Speech recognition is an interdisciplinary subfield of computer science and computational linguistics that develops methodologies and technologies that enable the recognition and translation of spoken language into text by computers. It is also known as automatic speech recognition (ASR), computer speech recognition or speech to text (STT). It incorporates knowledge and research in the computer science, linguistics and computer engineering fields.

Some speech recognition systems require "training" (also called "enrollment") where an individual speaker reads text or isolated vocabulary into the system. The system analyzes the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased accuracy. Systems that do not use training are called "speaker independent"[1] systems. Systems that use training are called "speaker dependent".

Speech recognition applications include voice user interfaces such as voice dialing (e.g. "call home"), call routing (e.g. "I would like to make a collect call"), domotic appliance control, search key words (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), determining speaker characteristics,[2] speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed direct voice input).

**Text-to-speech (TTS)**

Text-to-speech (TTS) technology reads aloud digital text. It can take words on computers, smartphones, tablets and convert them into audio. Also, all kinds of text files can be read aloud, including Word, pages document, online web pages can be read aloud. TTS can help kids who struggle with reading. Many tools and apps are available to convert text into speech.

Python comes with a lot of handy and easily accessible libraries and we’re going to look at how we can deliver text-to-speech with Python in this article.

**User** **Module**

* The purpose of this module is to provide the user interface and view functions for the system.
* User registers into the system by giving basic information like name, age etc.
* It also provides communication services between clients of the system and the server by asking questions regarding healthcare.

**Information Extraction**

Noun Phrase Extraction:

* Noun Phrase Extraction takes into account parts of speech patterns that include a noun . In this stage all the nouns are extracted from given input.
* It is used to remove stop words and it does not take into account the words which are repeated again in a sentence.
* Medical Term Identifier:
* This phase includes extraction of all medical terms.
* For example, spondylolysis is a combination of "spondylo" which means vertebra, and "lysis," which means dissolve, and so means dissolution of a vertebra.
* Depending on the disease symptoms or the medical term, the SVM algorithm can predict the disease.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

These are the requirements for doing the project. Without using these tools and software’s we can’t do the project. So we have two requirements to do the project. They are

1. Hardware Requirements.

2. Software Requirements.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* FRONT END : PYTHON
* OPERATING SYSTEM : WINDOWS 7
* IDE : Spyder3
* **FUNCTIONAL REQUIREMENTS**

A useful prerequisite characterizes an element of a product framework or its segment. A capacity is portrayed as a lot of data sources, the conduct, and yields. The application is taken care of with different subtleties and the coronary illness related with those subtleties. The application permits client to share their heart related issues. It at that point forms client explicit subtleties to check for different ailment that could be related with it. Here we utilize some smart information mining procedures to figure the most precise sickness that could be related with patient's subtleties. In view of result, the can contact specialist appropriately for additional treatment. The framework permits client to see specialist's subtleties as well. The framework can be utilized with the expectation of complimentary coronary illness counseling on the web.

**NON-FUNCTIONAL REQUIREMENTS**

The major non-functional Requirements of the system are as follows

**Usability**

The framework is structured with totally mechanized procedure subsequently there is no or less client mediation.

**Reliability**

The framework is increasingly solid in light of the characteristics that are acquired from the picked stage java. The code worked by utilizing java is increasingly solid.

**Performance**

This framework is creating in the significant level dialects and utilizing the propelled front-end and back-end advances it will offer reaction to the end client on customer framework with in exceptionally less time.

**Supportability**

The framework is intended to be the cross stage acceptable. The framework is bolstered on a wide scope of equipment and any product stage, which is having JVM, incorporated with the framework.

**Implementation**

The framework is executed in web condition utilizing swaggers structure. The apache tomcat is utilized as the web server and windows xp proficient is utilized as the stage. Interface the UI depends on Struts gives HTML Tag.

**CHAPTER 4**

**DESIGN ENGINEERING**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.1 UML Diagrams**

**4.1.1 USE CASE dIAGRAM**

****

**Fig 4.1** Use Case Diagram

**Explanation:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. roles of the actors in the system can be depicted. the above diagram consists of user as actor. each will play a certain role to achieve the concept.

**4.1.2 Class diagram**



**Fig 4.2:** Class diagram

**EXPLANATION:**

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, and the relationships between the classes. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class.

**4.1.3 OBJECT DIAGRAM**

An **object diagram** in the Unified Modeling Language (UML) is a diagramthat shows a complete or partial view of the structure of a modeled system at a specific time.

An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time.

Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.



**Fig 4.3**: Object diagram

**4.1.4 STATE DIAGRAM**

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states.



**Fig 4.4:** Statechart diagram

**4.1.5 SEQUENCE DIAGRAM**

A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order.It is a construct of a message sequence chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.



**Fig 4.5**: Sequence diagram

**4.1.6 ACtivity diagram**



**Fig 4.6:** Activity diagram

**EXPLANATION:**

Activity diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. UML, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. UML activity diagrams could potentially model the internal logic of a complex operation. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structural development.

**4.1.7 COMPONENT DIAGRAM**

****

**EXPLANATION:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

**4.1.8 DATA FLOW DIAGRAM**

Input data

Preprocessing

Training dataset

Feature Extraction

Prediction/Classification

Testing Data

Medical Data

Moderate

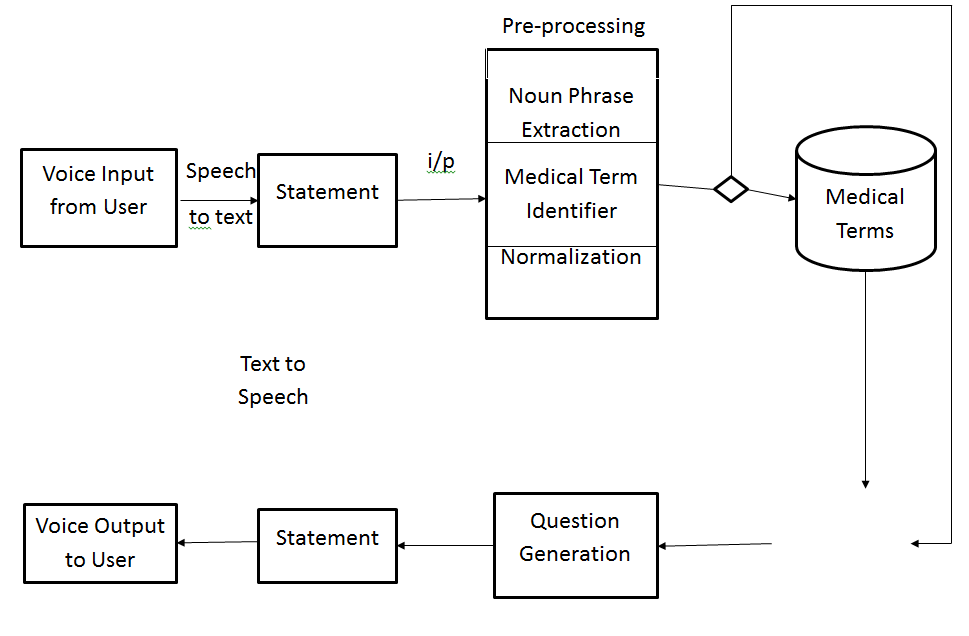
**4.1.9 DEPLOYMENT DIAGRAM**

****

**EXPLANATION:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger deployment and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination. Results are to be showed to user by data aggregators. All boxes are arrow indicates dependencies.

**SYSTEM ARCHITECTURE:**



**CHAPTER 5**

**DEVELOPMENT TOOLS**

**Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Importance of Python**

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**Features of Python**

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Libraries used in python:**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as dataframes.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure : NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

from flask import Flask, render\_template, request

import os

import pyttsx3

from MEDICAL\_ASSISTANT import mainCode

import LISTEN

from flask import jsonify

filenumber=int(os.listdir('saved\_conversations')[-1])

filenumber=filenumber+1

file= open('saved\_conversations/'+str(filenumber),"w+")

file.write('bot : Hi There! I am a medical chatbot. You can begin conversation by typing in a message and pressing enter.\n')

file.close()

app = Flask(\_\_name\_\_)

name = ""

age = ""

gender = ""

deases = ""

symptom = ""

@app.route("/")

def home():

return render\_template("index.html")

@app.route("/welcome")

def greet():

engine = pyttsx3.init()

engine.say('Hi There! I am a medical assistant. You can begin conversation by your voice in a message and pressing enter.')

engine.say('I can help you find out what\'s going on with a simple medical assisment.')

engine.runAndWait()

return ""

@app.route("/get")

def get\_bot\_response():

userText = LISTEN.listening()

response = str('')

engine = pyttsx3.init()

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('user : '+userText+'\n')

appendfile.write('bot : '+response+'\n')

appendfile.close()

engine.say(response)

engine.runAndWait()

return jsonify({'res':response,'inp':userText})

@app.route("/askname")

def askingname():

response = mainCode.askName()

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('bot : '+response+'\n')

appendfile.close()

engine = pyttsx3.init()

engine.say(response)

engine.runAndWait()

return response

@app.route("/askage")

def askingage():

response = mainCode.askAge()

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('bot : '+response+'\n')

engine = pyttsx3.init()

engine.say(response)

engine.runAndWait()

return response

@app.route("/askgender")

def askinggender():

response = mainCode.askGender()

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('bot : '+response+'\n')

engine = pyttsx3.init()

engine.say(response)

engine.runAndWait()

return response

@app.route("/asksymptom")

def askingsymptom():

response ='Can you please discribe your Symptoms'

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('bot : '+response+'\n')

engine = pyttsx3.init()

engine.say(response)

engine.runAndWait()

return response

@app.route("/askdeases")

def askingdeases():

response = 'Before we ask you your symptoms, we would like to know your health status.'

response += 'If yout have any existing Medical Conditions or Problems, please provide them here.'

response += 'If you dont, you can reply with a \'no\''

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('bot : '+response+'\n')

engine = pyttsx3.init()

engine.say(response)

engine.runAndWait()

return response

@app.route("/getname")

def gettingname():

userText = LISTEN.listening()

name = userText

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('user : '+userText+'\n')

#engine = pyttsx3.init()

#engine.say(response)

#engine.runAndWait()

return userText

@app.route("/getage")

def gettingage():

userText = LISTEN.listening()

age = userText

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('user : '+userText+'\n')

#engine = pyttsx3.init()

#engine.say(response)

#engine.runAndWait()

return userText

@app.route("/getgender")

def gettinggender():

userText = LISTEN.listening()

gender = userText

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('user : '+userText+'\n')

#engine = pyttsx3.init()

#engine.say(response)

#engine.runAndWait()

return userText

@app.route("/getdeases")

def gettingdeases():

userText = LISTEN.listening()

deases = userText

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('user : '+userText+'\n')

#engine = pyttsx3.init()

#engine.say(response)

#engine.runAndWait()

return userText

@app.route("/getsymptom")

def gettingsymptom():

userText = LISTEN.listening()

symptom = userText

existingDiseases = mainCode.extDisease(symptom)

response = "The most suitable medecine is for your Symptom is '"+existingDiseases+"'"

appendfile=os.listdir('saved\_conversations')[-1]

appendfile= open('saved\_conversations/'+str(filenumber),"a")

appendfile.write('user : '+userText+'\n')

appendfile.write('bot : '+response+'\n')

engine = pyttsx3.init()

engine.say(response)

engine.runAndWait()

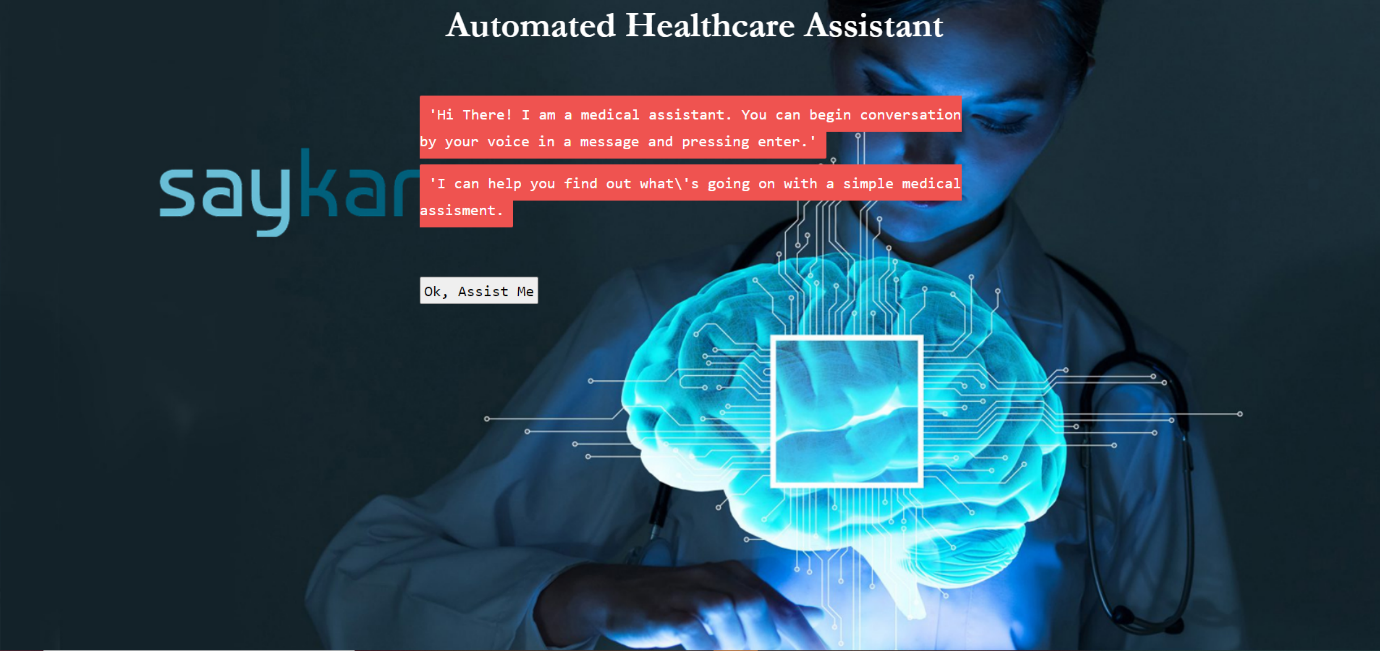
return jsonify({'res':response,'inp':userText})

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='localhost')

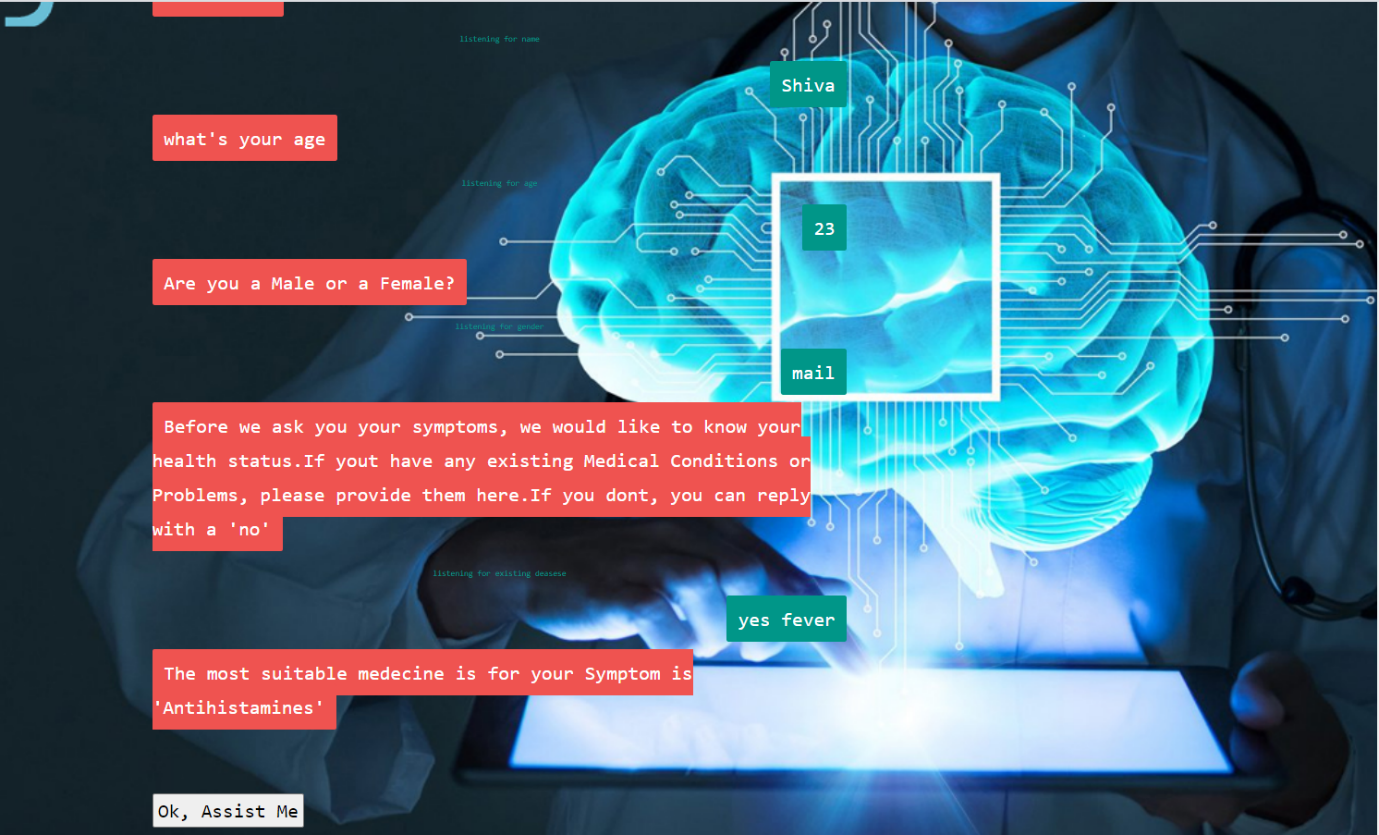
**CHAPTER 7**

**SNAPSHOTS**

****

****

****

****

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3Types of Tests**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**FUTURE ENHANCEMENT**

**9.1 FUTURE ENHANCEMENT**

In our future work, It has higher precision, recall, and accuracy. The result of the attrition prediction will be helpful for an organization to reduce the attrition rate of their company.

**CHAPTER 10**

**CONCLUSION & REFERENCE**

**10.1 CONCLUSION**

Keeping in mind the after-effects of a pandemic and the imbalance between the demand and healthcare services currently provided, especially in rural India have tried to bridge the gap by creating a Multilingual Conversational Application with Natural Language regular on-site consultations. Text is highly used, it can thus serve as a great opportunity to bridge the gap between the availability of healthcare advice to people. Processing (NLP). This is a one of a kind personalized healthcare bot which is sensitive to the needs and understanding of the Indian rural population provides generic healthcare information along with preventive measures for prevalent diseases and ailments indigenous to our country in a user simplified language; with special emphasis on interactive antenatal and postpartum healthcare. It has additional features including home remedies, location-based diet recommendations, age, and gender-specific health check-up advice, emergency helpline numbers, and can be linked with a real-time messaging application like WhatsApp. The aim of this application is not just to prevent malicious infectious diseases in the grappling population but to help achieve overall wellness. Our application is quite reliable in detecting various common diseases, suggesting home remedies and local food diets as long as problems and symptoms faced are well communicated by the user to the chatbot, and leading questions from the chatbot are appropriately answered.

**10.2 REFERENCES:**

[1] Rural India’s access to healthcare patchy: Studyhttps://economictimes.indiatimes.com/news/economy/indicators/rural-indias-access-to-healthcare-patchystudy/articleshow/21227645.cms, 2013, accessed : 2019- 10-24

[2] National Health Mission Health Management Information System. Available : https://nrhm-mis.nic.in/ accessed : 2020-05-20

[3] Emily Walsh, “How AI and Voice Assistants will Change Healthcare” https://voicebot.ai/2019/03/23/how-ai-andvoice-assistants-will-change-healthcare/, 2019, accessed : 2019-12-13

[4] KadekT eguhWirawan, I Made Sukarsa, I Putu Agung Bayupati, "Balinese Historian Chatbot using Full-Text Search and Artificial Intelligence Markup Language Method", International Journal of Intelligent Systems and Applications(IJISA), Vol.11, No.8, pp.21-34, 2019. DOI: 10.5815/ijisa.2019.08.03

[5] https://www.eclecticenergies.com/ego/eliza, accessed : 2019-1-17

[6] <https://phrasee.co/parry-the-a-i-chatterbot-from-1972/>, accessed : 2019-1-17

[7] Raij, A.B., Johnsen, K., Dickerson, R.F., Lok, B.C., Cohen, M.S., Duerson, M., Pauly, R.R., Stevens, A.O., Wagner, P. and Lind, D.S., 2007. Comparing interpersonal interactions with a virtual human to those with a real human. IEEE transactions on visualization and computer graphics, 13(3), pp.443-457.

[8] Fadhil, A., 2018.“Beyond patient monitoring: Conversational agents role in telemedicine & healthcare support for home-living elderly individuals”. arXiv preprint arXiv:1803.06000.

[9] Amato, F., Marrone, S., Moscato, V., Piantadosi, G., Picariello, A. and Sansone, C., 2017. Chatbots Meet eHealth: Automatizing Healthcare. In WAIAH@ AI\* IA (pp. 40-49).

[10] Comendador, BenildaEleonor V., et al. "Pharmabot: a pediatric generic medicine consultant chatbot." Journal of Automation and Control Engineering Vol 3.2 (2015)