

Mini Project On Medical Chatbot

By

**Shashank Agrawal (2021510002)
Prithviraj Sathyajit (2021510059)**

Under the guidance of
Internal Supervisor

Prof. Pallavi Thakur



Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University
2022-23

CERTIFICATE OF APPROVAL

This is to certify that the following students

Shashank Agrawal (2021510002)
Prithviraj Sathyajit (2021510059)

Have satisfactorily carried out work on the project
entitled

“Medical Chatbot”

Towards the fulfilment of project, as laid down
by
Sardar Patel Institute of Technology
during year
2022-23.

Project Guide:
Prof. Pallavi Thakur

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Shashank Agrawal (2021510002)
Prithviraj Sathyajit (2021510059)

Have successfully completed the Project report on

“Medical Chatbot”,

which is found to be satisfactory and is approved

at

**SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI**

INTERNAL EXAMINER

Prof. Pallavi Thakur

EXTERNAL EXAMINER

HEAD OF DEPARTMENT

Dr. Pooja Raundale

PRINCIPAL

Dr. B.N Chaudhari

Contents

Abstract	i
Objectives	i
List Of Figures	ii
List Of Tables	ii
1 Introduction	1
1.1 Problem Definition	1
1.2 Objectives and Scope	1
1.2.1 Objectives	1
1.2.2 Scope	1
1.3 Existing System	2
1.4 Proposed System	2
1.5 System Requirements	3
2 Software Requirement Specification (SRS) and Design	4
2.1 Purpose	4
2.2 Definition	4
2.3 Overall Description	4
2.3.1 Product Functions	4
2.3.2 User Characteristics	5
3 Project Analysis and Design	6
3.1 Methodologies Adapted	6
3.2 Modules	7
3.2.1 Activity diagram	7
3.2.2 Sequence Diagram	8
3.2.3 Work Breakdown Structure	8
3.2.4 Gantt Chart	9
3.2.5 Use-Case	10
4 Project Implementation and Testing	13
4.1 Landing Page	13
4.2 Bot Conversation - Introduction	14
4.3 Bot conversation	15
4.4 Code 1	16
4.5 Code 2	16
4.6 Code 3	17
5 Test Cases	18
6 Limitations	19

7	Future Enhancements	19
8	User Manual	20
9	Bibliography	21
9.1	Web References	21

Abstract

Complex, unreliable, and unsustainable is the new healthcare delivery system. The use of machine learning (ML) to improve system performance has revolutionized the way businesses and individuals collect and analyze data. It is possible to use machine learning algorithms for the analysis of data.

Medical bot is an online web application in which, A virtual assistant can communicate with patients to understand their symptoms, provide information related to the user input. To further analyze customer needs, natural language processing algorithms are employed with the help of which patients are informed about the disease and allowed to book visits with doctors at location. This web application allows reduction of human contact and allow patients to receive care irrespective of time and their location

The app will provide a general diagnosis based on the patient's symptom and give them an option to make an appointment if required.

Objectives

The Android based Application "Medical Chatbot" is used

- To save patient's information
- To allow the patient to find information on the disease they suspect
- To provide information of the disease mentioned.
- To allow the patient to book an appointment to listed hospitals/doctors.

List of Figures

3.1.1Diagrammatic Representation of Waterfall Model	6
3.2.1Activity Diagram	7
3.2.2Sequence Diagram	8
3.2.3Work Breakdown Structure	8
3.2.4Gantt Chart	9
3.2.5Use-Case Diagram	10
4.1.1 Landing Page	13
4.2.1 Bot conversation - Introduction	14
4.3.1Bot conversation - Symptom entry	15

List of Tables

1.5.1 Hardware Requirements on Server Side	3
1.5.2 Hardware Requirements on Client Side	3
1.5.3 Software Requirements on Server Side	3
1.5.3 Software Requirements on Client Side	3
4.2.1 Use Case Table - Chat	11
4.2.2 Use Case Table - Add Info	11
4.2.3 Use Case Table - View Details	11
4.2.4 Use Case Table - Update details	12
4.2.5 Use Case Table - Delete details	12
6.1 Test Case - Login and Register	18
6.2 Test Case - Others	18

1 Introduction

1.1 Problem Definition

To provide efficient and accurate information to the user. Reducing human contact and maintaining the social forms of social distancing. As well as allow the user to book appointments in process squashing time waste and avoid misinformation in human Interactions.

1.2 Objectives and Scope

1.2.1 Objectives

The web based "Medical Chatbot" is

- To provide accurate information about their disease to the user
- To keep required details regarding Hospitals and doctors handy
- To provide patients an easy and quicker way to book visits to the doctors with the specialities required.
- To provide a better service to patients.

1.2.2 Scope

The patients can view and access their past interactions with the bot. Many people despite their platform of use or different language of use be able to access the bot and be able to work with it.

This application is based on the fact that human interaction needs to be limited so Medical experts are able to give on the spot advice through the application.

In future allowing the patients to have video calls in form of appointments through the application is also a scope of the system

1.3 Existing System

Currently no such kind of application exists for the users to allow them to interact with a system to provide information about their symptoms.

Some of the disadvantages of existing system are as follows :

- Time consuming, Manual work
Every time a patient needs to physically go to the hospital/clinic and go through the process just to even receive information about their disease.
- Redundancy
Many times the patient information or the explanation about their symptoms get lost in transition.
- Loss of information
Many times staff might miss some information regarding a particular patient due to scattering of the information in various places.

1.4 Proposed System

The User is the patient seeking information related to their symptoms. This System will be connected to the data where the information related to any symptoms mentioned by the patient will be available and further care in future.

The user will also be able to book visits to the doctors/Hospitals stored in the database for the symptoms corresponding to their specialities.

Some of the advantages of our system are as follows :

- User Friendly
It provides attractive interface to the user for navigation through a dynamic flow of placement process in the app.
- Information of Diseases in a place
Users can view their disease information as well as the list of doctors available

1.5 System Requirements

- Hardware Requirements on Server Side

Table 1.5.1: Hardware Requirements on Server Side

Processor	Dual Core Processor or Above
RAM	Minimum 4 GB RAM
Storage	Minimum 10 GB Hard Disk Space for smooth run

- Hardware Requirements on Client Side

Table 1.5.2: Hardware Requirements on Client Side

Device	Android Device with Touch Screen minimum 5" inch Display
Processor	Dual Core Processor or Above
RAM	Minimum 2 GB RAM
Storage	Minimum 250 MB Storage Space

- Software Requirements on Server Side

Table 1.5.3: Software Requirements on Server Side

Operating System	OS Independent
Database	Firestore

- Software Requirements on Client Side

Table 1.5.3: Software Requirements on Client Side

Operating System	Android/IOS Smartphone
Server	Not Required

2 Software Requirement Specification (SRS) and Design

2.1 Purpose

The purpose of our project is to develop an Web application that can help user (patient) to easily access and to keep all the required information handy. This can save lots of time of the user and the hospital staff and it will be easier to store each and every single and small detail regarding each visit booking. This application will keep track of all the information regarding the disease including doctor and the hospitals nearby.

2.2 Definition

To build a Medical Bot so the patient can easily gain access to the information required as well book appointments.

2.3 Overall Description

2.3.1 Product Functions

The product function includes:

1. Information gathering: Users are required to give their personal history before starting any process.
2. Inquiry: The user is asked about the symptoms or disease in question
3. Disease: The user is given information about the disease mentioned by the user.
4. View Doctors: The user is able to see the doctors available near the mentioned location
5. Book appointment: The user is able to book a visit to the doctor mentioned if they wish to do so.

2.3.2 User Characteristics

There is one user:

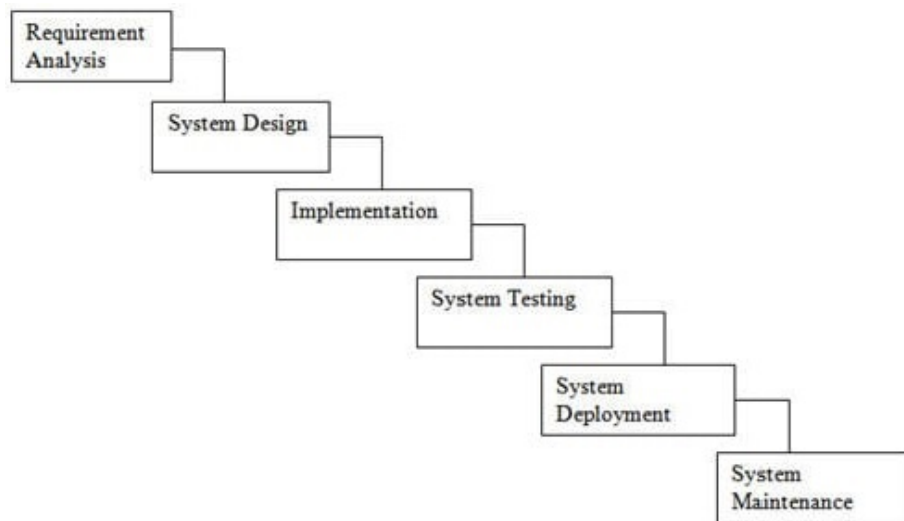
- Patient: Once the user gives their personal history the user able to get information about their disease and book visits based on their interactions with the application.

3 Project Analysis and Design

3.1 Methodologies Adapted

In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

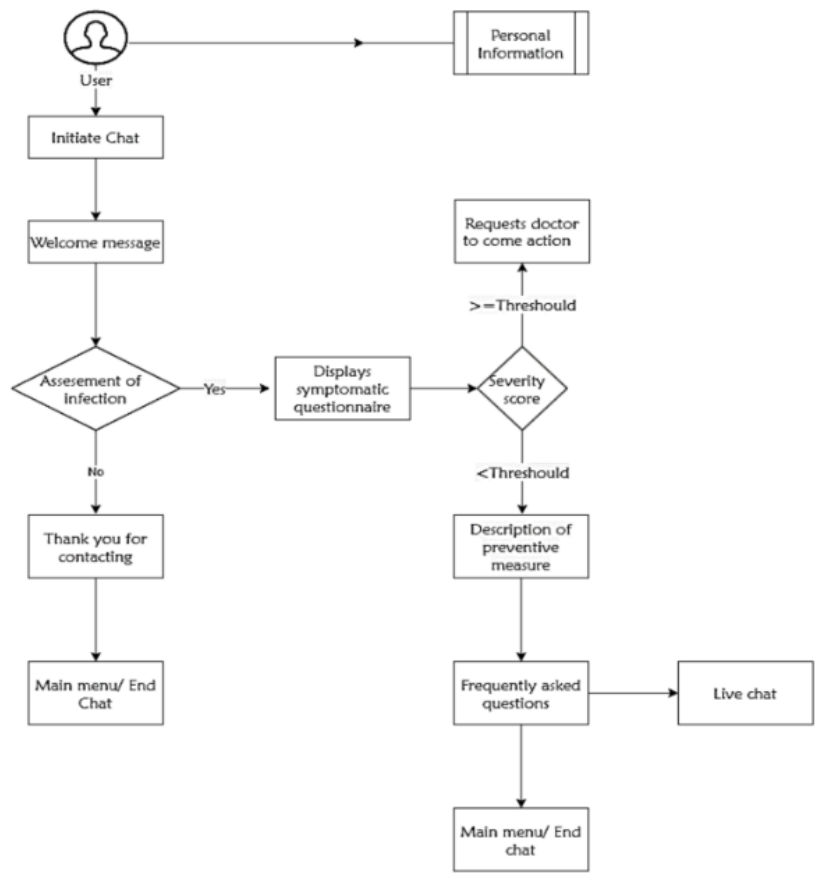
Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



3.1.1: Diagrammatic Representation of Waterfall Model

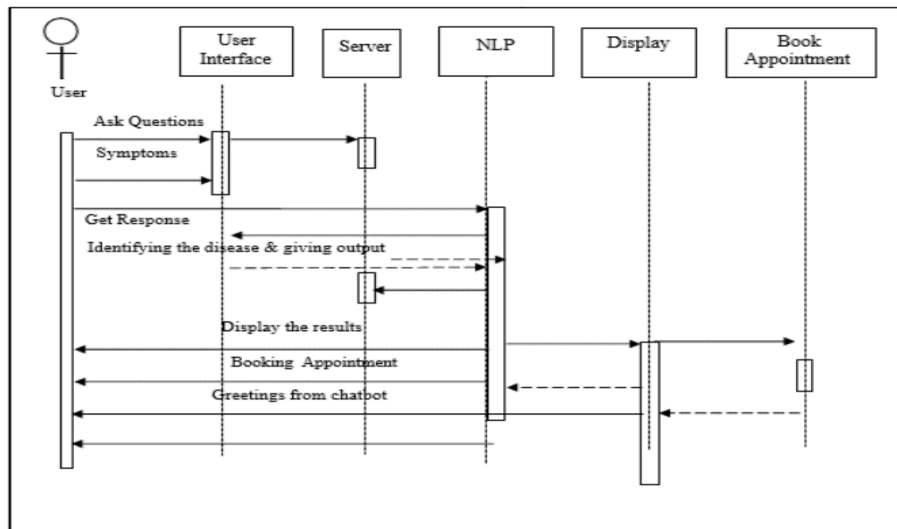
3.2 Modules

3.2.1 Activity diagram



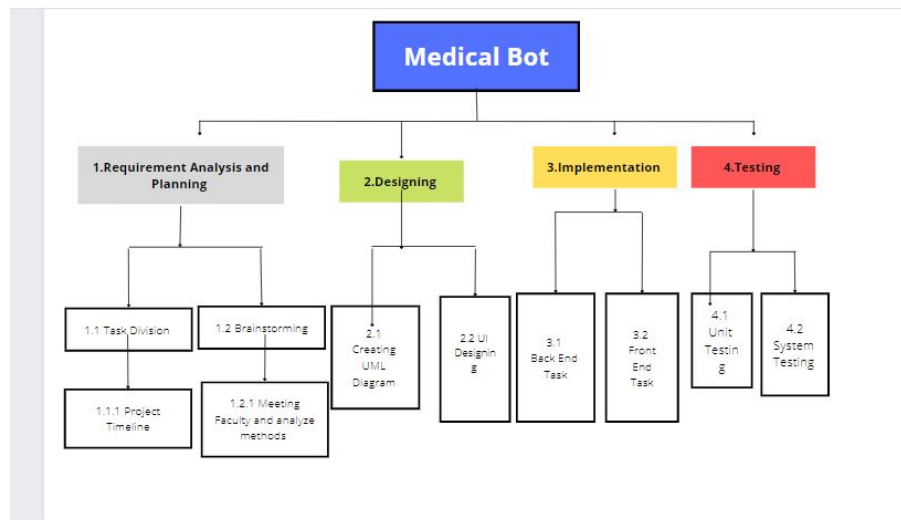
3.2.1: Activity Diagram

3.2.2 Sequence Diagram



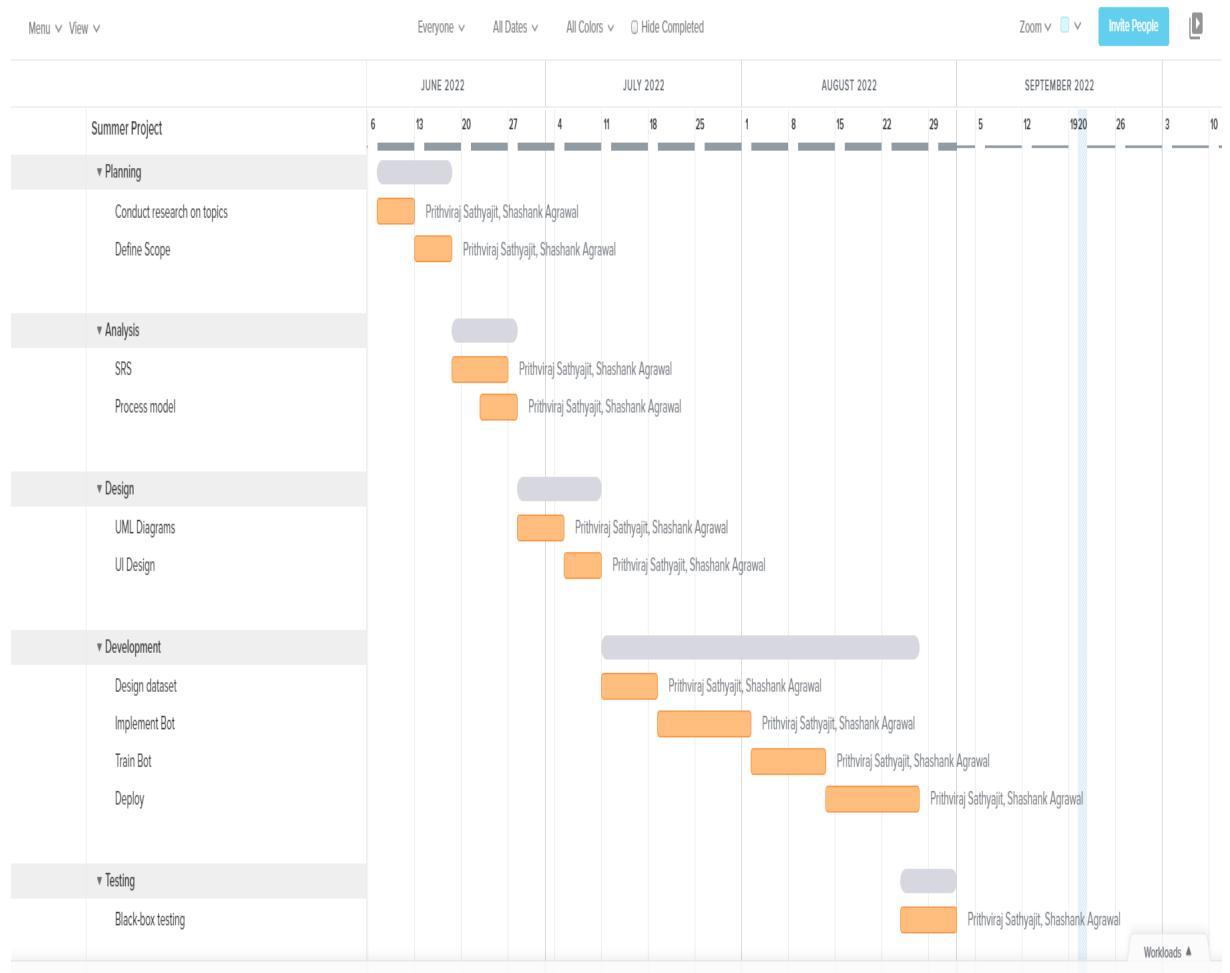
3.2.2: Sequence Diagram

3.2.3 Work Breakdown Structure



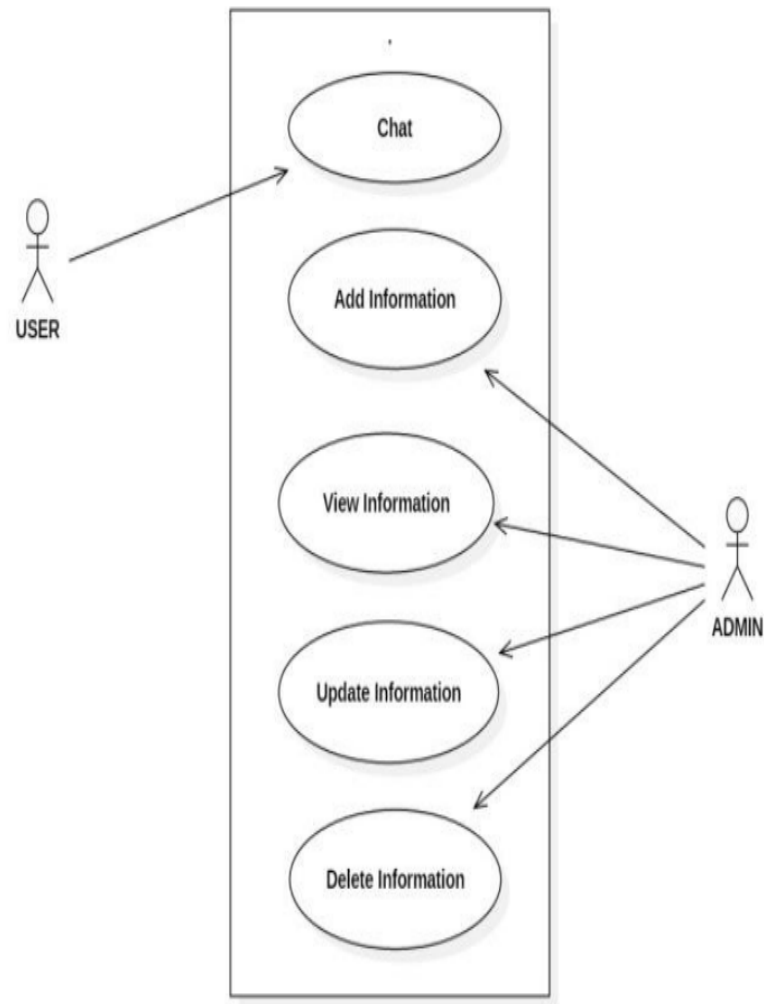
3.2.3: Work Breakdown Structure

3.2.4 Gantt Chart



3.2.4: Gantt Chart

3.2.5 Use-Case



3.2.5: Use-Case Diagram

Use Cases:

1. Chat
2. Add Information
3. View Information
4. Update Information
5. Delete Information

Table 4.2.1: Use Case Table - Chat

Use Case ID	1
Use Case	Chat
Actor	Patient
Pre-Condition	They must enter their details
Post-Condition	User can chat
Flow of events	Enter details and enter symptoms

Table 4.2.2: Use Case Table - Add Info

Use Case ID	2
Use Case Name	Add information
Actor	Admin
Pre-Condition	-
Post-Condition	Admin can add patient details
Flow of events	Add details

Table 4.2.3: Use Case Table - View Details

Use Case ID	3
Use Case Name	View details
Actor	Admin
Pre-Condition	User details must exist
Post-Condition	Admin can view details

Table 4.2.4: Use Case Table - Update details

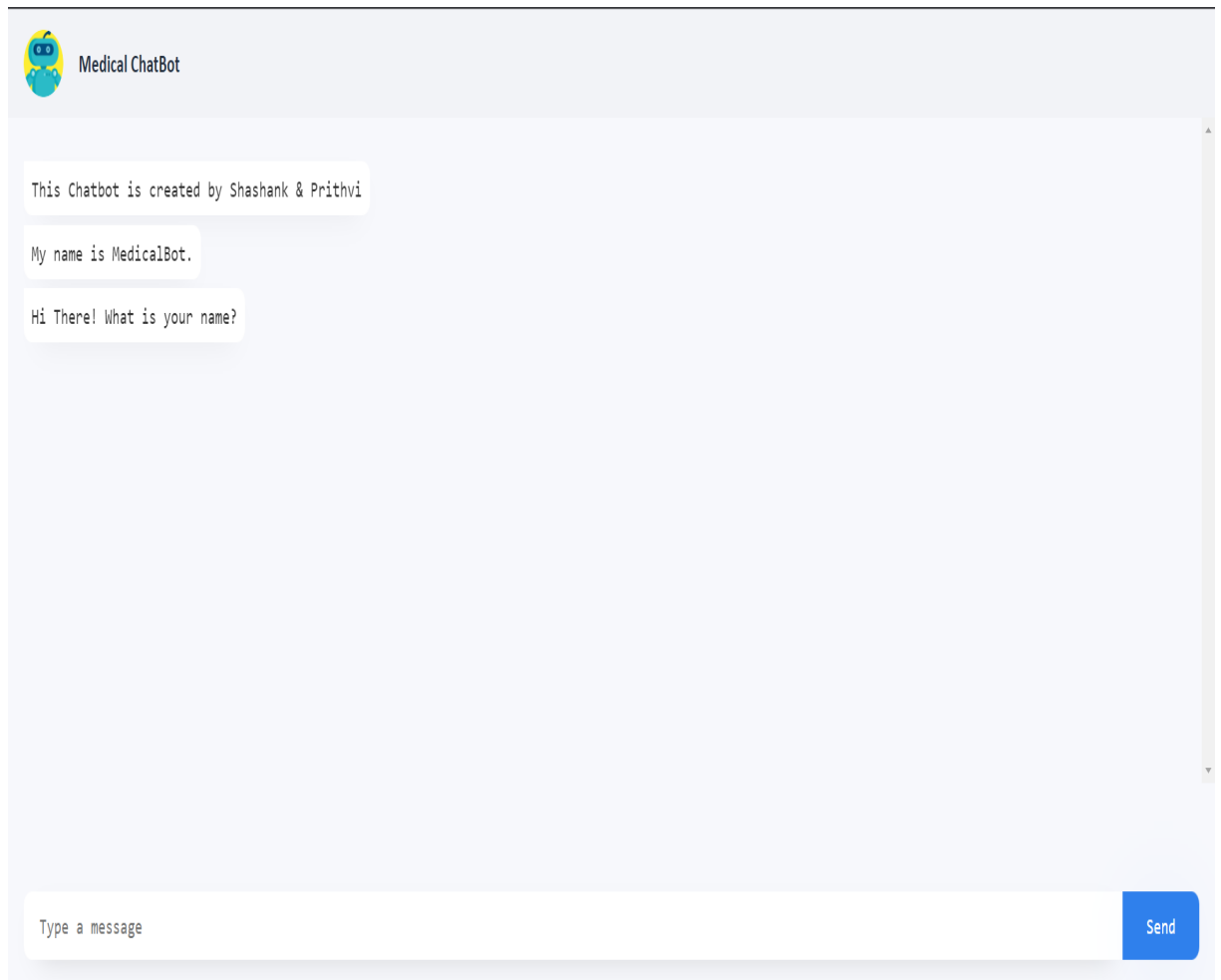
Use Case ID	4
Use Case Name	Update details
Actor	Admin
Pre-Condition	User details must exist
Post-Condition	Admin can update details

Table 4.2.5: Use Case Table - Delete details

Use Case ID	5
Use Case Name	Delete patient details
Actor	Admin
Pre-Condition	User details must exist
Post-Condition	Admin can delete details

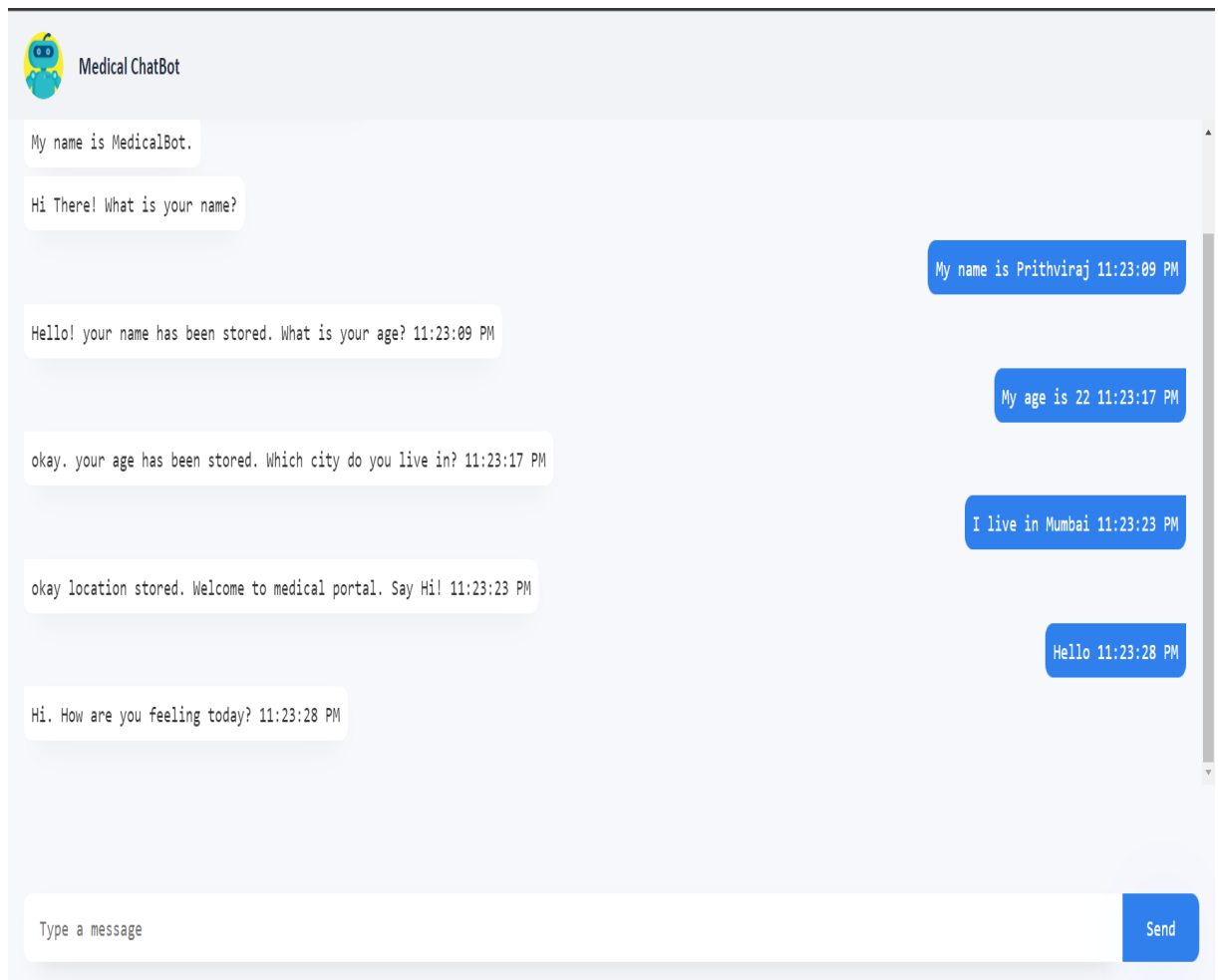
4 Project Implementation and Testing

4.1 Landing Page



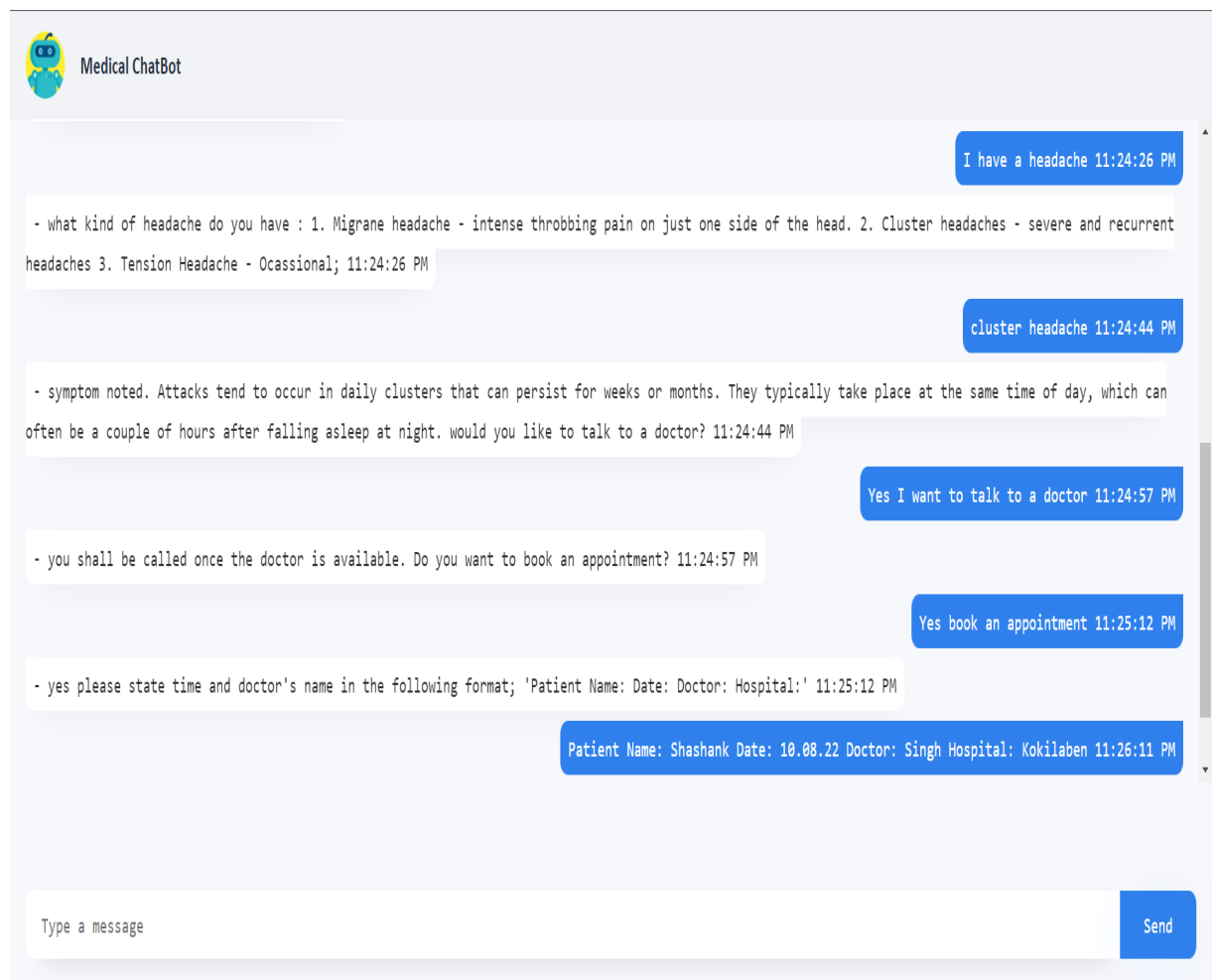
4.1.1: Landing Page

4.2 Bot Conversation - Introduction



4.2.1: Bot conversation - Introduction

4.3 Bot conversation



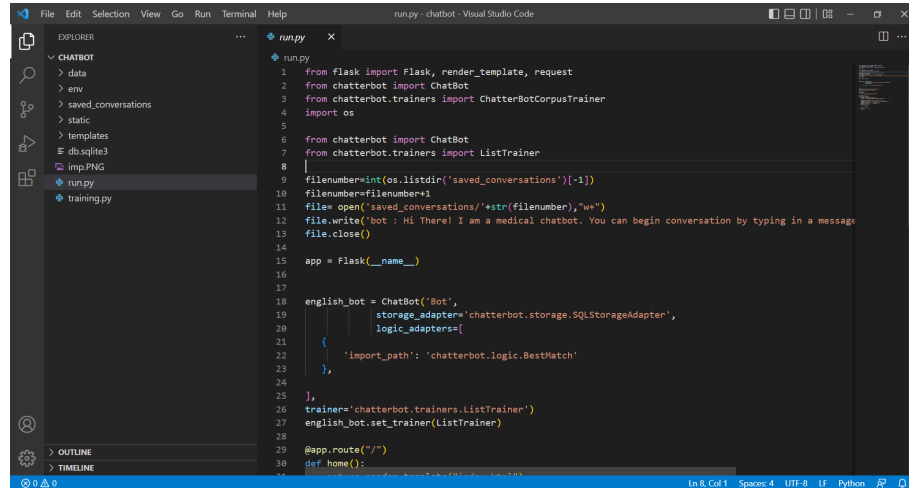
The screenshot displays a chat interface for a 'Medical ChatBot'. The header shows a blue robot icon and the text 'Medical ChatBot'. The conversation history is as follows:

- User:** I have a headache 11:24:26 PM
- Bot:** - what kind of headache do you have : 1. Migrane headache - intense throbbing pain on just one side of the head. 2. Cluster headaches - severe and recurrent headaches 3. Tension Headache - Ocassional; 11:24:26 PM
- User:** cluster headache 11:24:44 PM
- Bot:** - symptom noted. Attacks tend to occur in daily clusters that can persist for weeks or months. They typically take place at the same time of day, which can often be a couple of hours after falling asleep at night. would you like to talk to a doctor? 11:24:44 PM
- User:** Yes I want to talk to a doctor 11:24:57 PM
- Bot:** - you shall be called once the doctor is available. Do you want to book an appointment? 11:24:57 PM
- User:** Yes book an appointment 11:25:12 PM
- Bot:** - yes please state time and doctor's name in the following format; 'Patient Name: Date: Doctor: Hospital:' 11:25:12 PM
- User:** Patient Name: Shashank Date: 10.08.22 Doctor: Singh Hospital: Kokilaben 11:26:11 PM

At the bottom, there is a text input field labeled 'Type a message' and a blue 'Send' button.

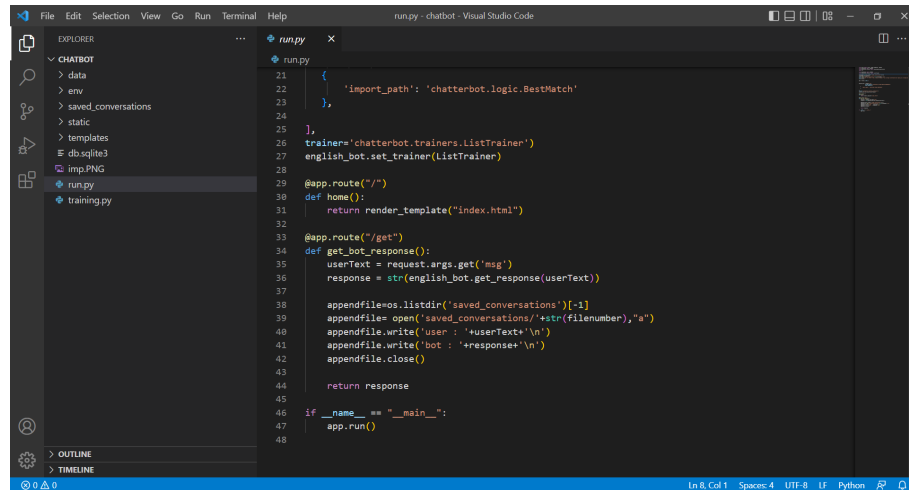
4.3.1: Bot conversation - Symptom entry

4.4 Code 1



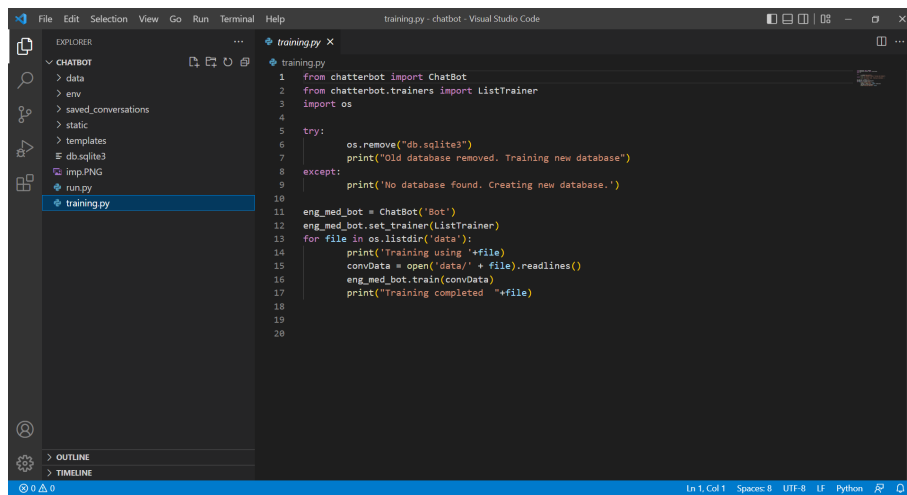
```
1 from flask import Flask, render_template, request
2 from chatterbot import ChatBot
3 from chatterbot.trainers import ChatterBotCorpusTrainer
4 import os
5
6 from chatterbot import ChatBot
7 from chatterbot.trainers import ListTrainer
8
9 filename=os.listdir('saved_conversations')[-1]
10 filename=filename+1
11 file= open('saved_conversations/'+str(filename),"w")
12 file.write('bot : Hi There! I am a medical chatbot. You can begin conversation by typing in a message')
13 file.close()
14
15 app = Flask(__name__)
16
17
18 english_bot = ChatBot('Bot',
19                      storage_adapter='chatterbot.storage.SQLStorageAdapter',
20                      logic_adapters=[
21                          {
22                              'import_path': 'chatterbot.logic.BestMatch'
23                          },
24                      ],
25                      trainers='chatterbot.trainers.ListTrainer')
26 english_bot.set_trainer(ListTrainer)
27
28
29 @app.route("/")
30 def home():
31     return render_template("index.html")
32
33
34 def get_bot_response():
35     userText = request.args.get('msg')
36     response = str(english_bot.get_response(userText))
37
38     appendfile=os.listdir('saved_conversations')[-1]
39     appendfile= open('saved_conversations/'+str(filename),"a")
40     appendfile.write('user : '+userText+'\n')
41     appendfile.write('bot : '+response+'\n')
42     appendfile.close()
43
44     return response
45
46 if __name__ == "__main__":
47     app.run()
```

4.5 Code 2



```
21
22     'import_path': 'chatterbot.logic.BestMatch'
23 },
24 ],
25 ],
26 trainers='chatterbot.trainers.ListTrainer')
27 english_bot.set_trainer(ListTrainer)
28
29 @app.route("/")
30 def home():
31     return render_template("index.html")
32
33
34 @app.route("/get")
35 def get_bot_response():
36     userText = request.args.get('msg')
37     response = str(english_bot.get_response(userText))
38
39     appendfile=os.listdir('saved_conversations')[-1]
40     appendfile= open('saved_conversations/'+str(filename),"a")
41     appendfile.write('user : '+userText+'\n')
42     appendfile.write('bot : '+response+'\n')
43     appendfile.close()
44
45     return response
46
47 if __name__ == "__main__":
48     app.run()
```

4.6 Code 3



```
1 from chatterbot import ChatBot
2 from chatterbot.trainers import ListTrainer
3 import os
4
5 try:
6     os.remove("db.sqlite3")
7     print("Old database removed. Training new database")
8 except:
9     print("No database found. Creating new database.")
10
11 eng_med_bot = ChatBot('Bot')
12 eng_med_bot.set_trainer(ListTrainer)
13 for file in os.listdir('data'):
14     print('Training using '+file)
15     convData = open('data/' + file).readlines()
16     eng_med_bot.train(convData)
17     print("Training completed "+file)
```


5 Test Cases

Table 6.1: Test Case - Login and Register

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User enters name	User enters their name	Bot greets user with name	Home Page	Pass
2	User enters name	Enters name in different format	Prompt re-enter name	Prompt re-enter name	Pass
3	User enter symptoms	User enters existing symptoms	General diagnosis displayed	General diagnosis displayed	Pass
4	User enter symptoms	Symptoms non-existent in database	Prompt re-enter	Prompt re-enter	Pass

Table 6.2: Test Case - Others

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User enters patient details in invalid format	Invalid patient details	Prompt Message showing error	Successful Updated	Fail
2	User enters valid information	Valid patient details	Details stored successfully	Details stored successfully	Pass

6 Limitations

- It needs internet to be accessed.
- It does not have a feature to connect with hospital directly for queries and support.
- PDFs regarding reports can not be uploaded
- It does not have a feature to apply for contacting doctor directly.
- Diagnosis may be too general at times.

7 Future Enhancements

- Automatic generation of previous reports of the patients
- Ability to connect to doctor if required.
- Make payment for appointment.
- More specific diagnoses.

8 User Manual

Part 1 – Enter Details

Upon opening the website, the user will be prompted to enter their name, age and location

Part 2 – Enter Symptoms

User must enter their symptoms when prompted by the chatbot.

Part 3 – Book appointment

User can choose to book an appointment

Part 4 – Enter appointment details

User has to enter details such as patient name, doctor's name, hospital name

9 Bibliography

9.1 Web References

- [1] <https://www.geeksforgeeks.org/>
- [2] <https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/>
- [3] <https://app.teamgantt.com/>
- [4] <https://stackoverflow.com/>
- [5] <https://www.draw.io/>