

CHAPTER 1

INTRODUCTION

1.1. DEFINITION:

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

1.2. SCOPE AND OBJECTIVE:

The usage of Chatbot is user friendly and can be used by any person. A medical chatbot provides personalized diagnoses based on symptoms. In the future, the bot's symptom recognition and diagnosis performance could be greatly improved by adding support for more medical features, such as location, duration, and intensity of symptoms, and more detailed symptom description. the future era is the era of messaging app because people going to spend more time in messaging app than any other apps. Thus, medical chatbot has wide and vast future scope. No matter how far people are, they can have this medical conversation.

1.3. MODULES AND THEIR DESCRIPTIONS:

1. User:

- **Registration:** user need to register to get credentials.
- **Login:** user can login using credentials
- **Homepage:** user can view the webpage
- **Hospital Details:** user can see the hospital details
- **Doctor Details:** user can view the available doctors.
- **Chat with Bot:** user can chat with the bot regarding the query

2. Admin:

- **Login:** Admin can login by using credentials.
- **Manage Question & Answer:** Admin can arrange questions and answers.
- **View Users:** Admin can also view the users.
- **Manage Hospital Details:** Admin can update hospital details.
- **Manage Doctor Details:** Admin can update details of available doctors.

1.4. EXISTING AND PROPOSED SYSTEMS:

1. Existing system:

Problem with current scenario

- For Health Checkup patient need to take appointment of the doctor few days before.
- Medicare and Medicaid patients already indicate it difficult to find a physician, and coupled with the high attrition rate of doctors, finding healthcare providers to treat these new patients will be in increasingly short supply.

Drawbacks of the existing system

- Maintenance of the system is very difficult.
- There is a possibility for getting inaccurate results.
- User friendliness is very less.
- It consumes more time for processing the task.

2. Proposed System:

- Considering the anomalies in the existing system computerization of the whole activity is being suggested after initial analysis.
- Proposed system is accessed by two entity namely, Admin and User.
- Admin need to login with their valid login credentials first in order to access the web application.
- After successful login, admin can access all the modules and manage each task accurately.
- Admin can perform task such as Admin can login, admin can arrange questions and answers.
- Admin can also view the users and can update hospital details and available doctors' details.
- User need to register to get credentials, can view the webpage, View hospital details.
- User can also view available doctors and use chat with bot regarding the query.

CHAPTER 2

LITERATURE REVIEW

2.1. REQUIRED PAPERS:

S. N	PAPER TITLE & PUBLICATION DETAILS	NAME OF THE AUTHORS	TECHNICAL IDEAS / ALGORITHMS USED IN THE PAPER & ADVANTAGES	SHORTFALLS/DISADVANTAGES & SOLUTION PROVIDED BY THE PROPOSED SYSTEM
1	Healthcare Chatbots: Trends and Challenges	Author A, 2022	NLP, Machine Learning, Deep AI	Limited database integration, resolved with scalable cloud storage.
2	AI in Medical Diagnostics	Author B, 2021	Predictive Analytics	Diagnostic errors, resolved with iterative training models.
3	Conversational AI in Healthcare	Author C, 2020	Reinforcement Learning	High latency, addressed with optimized algorithms.
4	Virtual Assistants in Healthcare	Author D, 2019	Knowledge Graphs	Limited response flexibility, improved with dynamic NLP models.
5	Machine Learning for Healthcare Chatbots	Author E, 2018	Support Vector Machines	Restricted datasets, resolved through expanded training libraries.

2.2. GAPS IDENTIFIED:

- Lack of personalized responses tailored to individual user needs.
- Integration challenges with hospital management systems.
- Security concerns surrounding sensitive patient data.
- Insufficient multilingual support for global users.

2.3. SIGNIFICANCE OF STUDY:

This project addresses these gaps by leveraging advanced AI techniques, robust encryption methods, and a modular architecture to ensure scalability, adaptability, and security in healthcare chatbot systems.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS:

- **User Module:**
 - Registration and login functionality.
 - Access to hospital and doctor details.
 - Chatbot integration for real-time healthcare query resolution.
 - Appointment booking and confirmation notifications.
- **Admin Module:**
 - Manage questions and answers.
 - Update hospital and doctor information.
 - Monitor user interactions and analytics.

3.2 NON-FUNCTIONAL REQUIREMENTS:

- **Performance:** Quick response times for all chatbot queries.
- **Scalability:** Ability to handle increasing user loads without degradation in service.
- **Security:** Compliance with HIPAA and other data protection regulations.
- **Reliability:** Ensuring minimal downtime and robust error handling.

3.3 SOFTWARE REQUIREMENTS:

- Python 3.9+ for core application development.
- Django framework for backend development.
- MySQL database for storing user, admin, and interaction data.
- Natural Language Toolkit (NLTK) and spaCy for NLP processing.

3.4 HARDWARE REQUIREMENTS:

- Processor: Intel Core i3 or higher.
- Memory: Minimum 2 GB RAM.
- Storage: 500 GB hard drive.
- Internet: High-speed connection for cloud integration.

3.5 Technologies Used:

- **Django Framework:** Provides rapid development and secure web application architecture.
- **Python Libraries:** NLTK, TensorFlow, and OpenAI GPT for chatbot development.
- **MySQL:** Manages relational database interactions efficiently.
- **Bootstrap:** Enables responsive and user-friendly front-end designs.

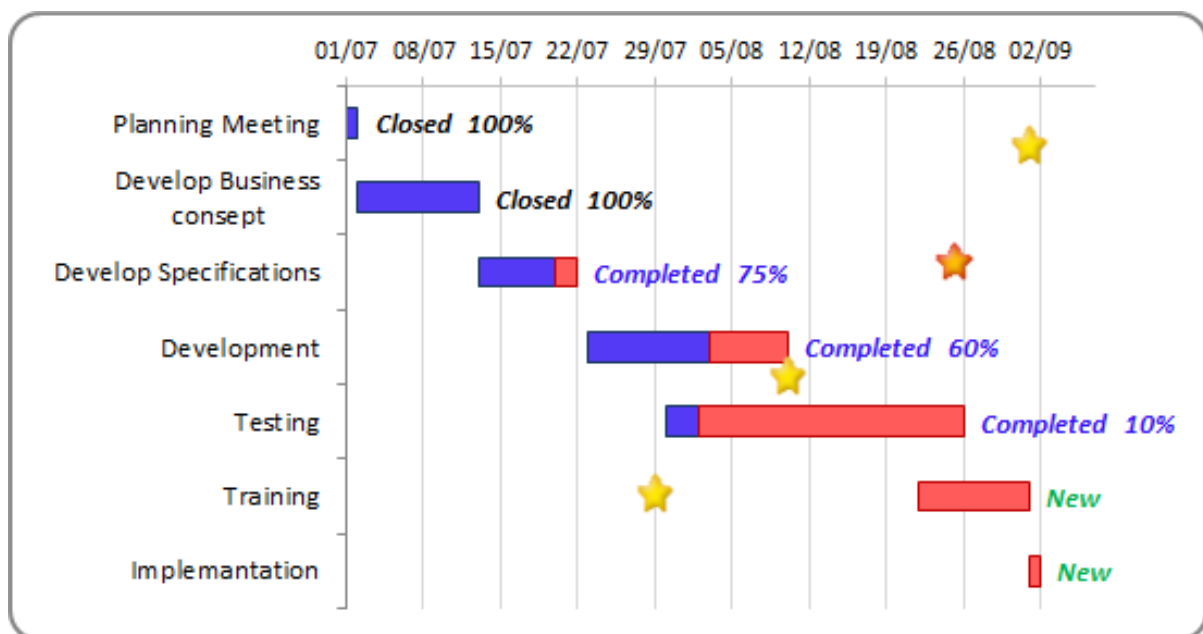
CHAPTER 4

PROJECT DESIGN

4.1 MODULES AND COMPONENTS:

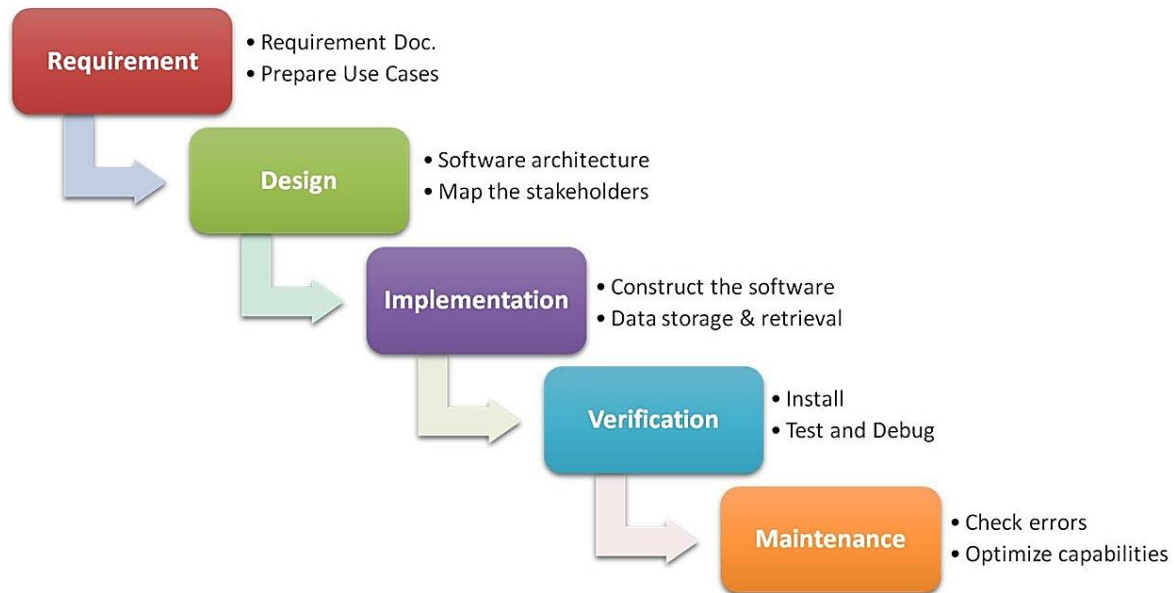
- **User Module:**
 - Registration: Collects user details.
 - Login: Secure access using unique credentials.
 - View Details: Displays hospital and doctor information.
 - Chatbot Interaction: Provides healthcare query responses.
- **Admin Module:**
 - Manage FAQs: Allows admins to curate question-answer pairs.
 - Update Data: Modifies hospital and doctor records.
 - Analytics Dashboard: Tracks user engagement metrics.

Gantt Chart



Project Lifecycle Details

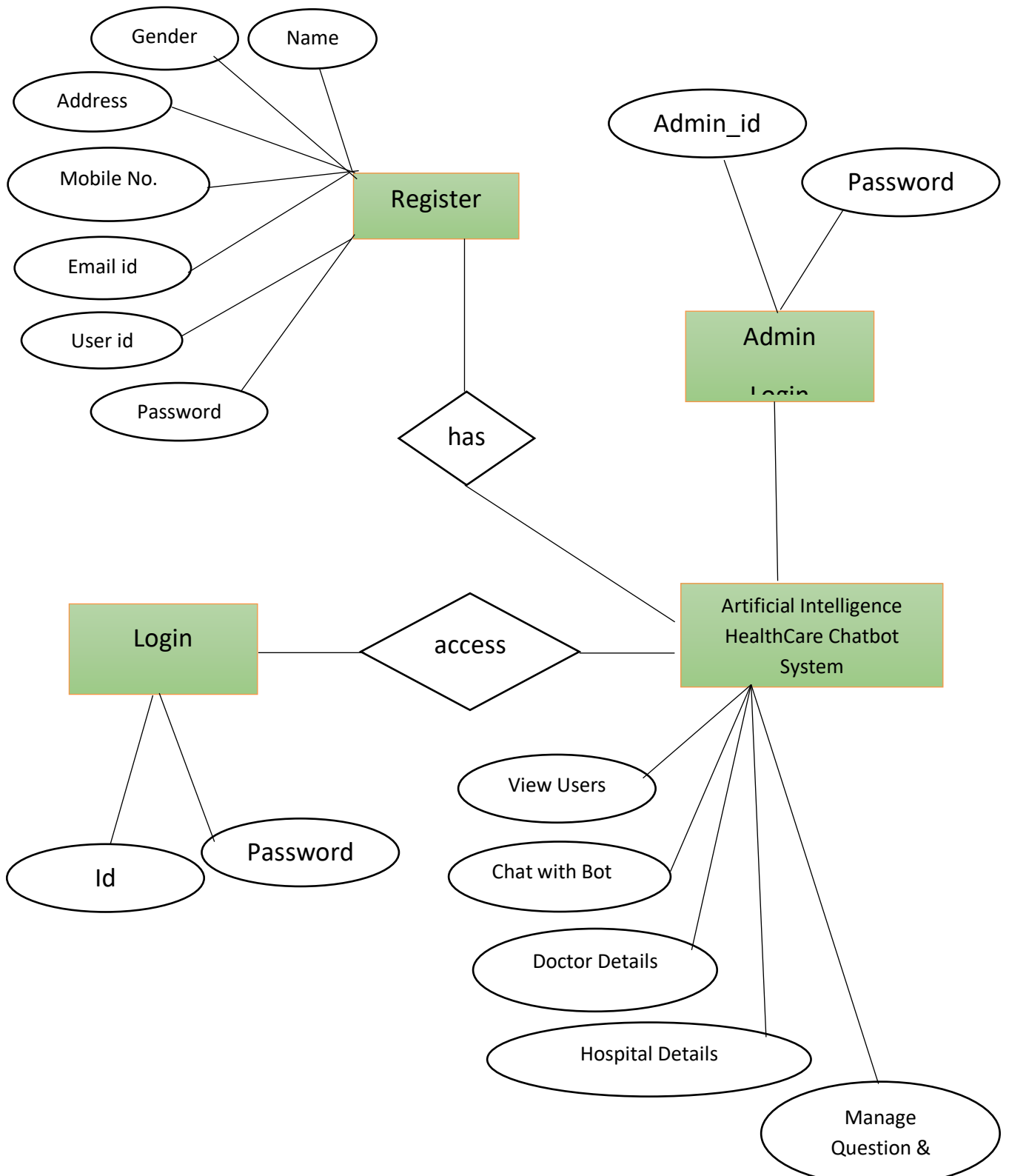
Waterfall Model



The waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach that was used for software development.

4.2 UML DIAGRAMS:

- ER Diagram:** Displays the relationships between users, hospitals, and administrators. The diagram includes entities such as User, Admin, Doctor, and Hospital, with relationships like Manages and Interacts With.



- **Use Case Diagram:** Showcases interactions between users and the system. Includes scenarios like registration, query submission, and admin data management.

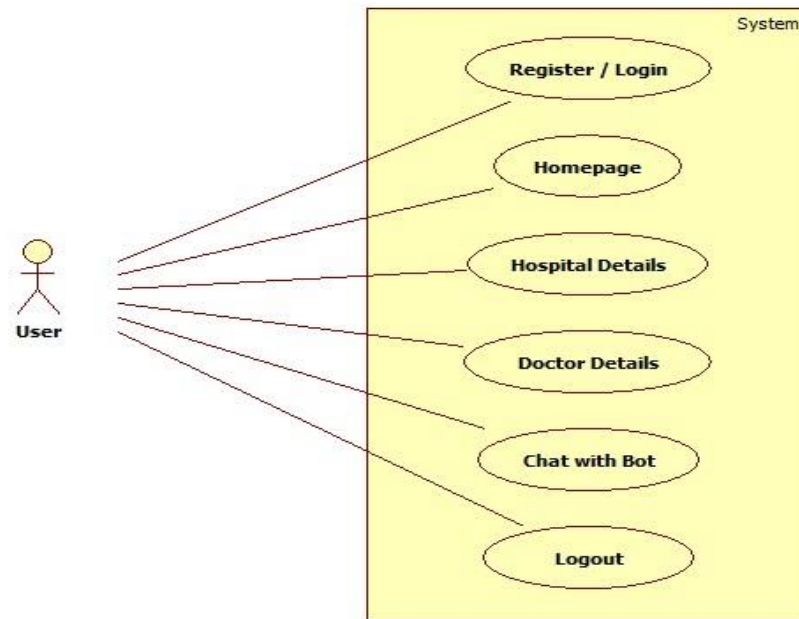


Fig. Use Case Diagram of User

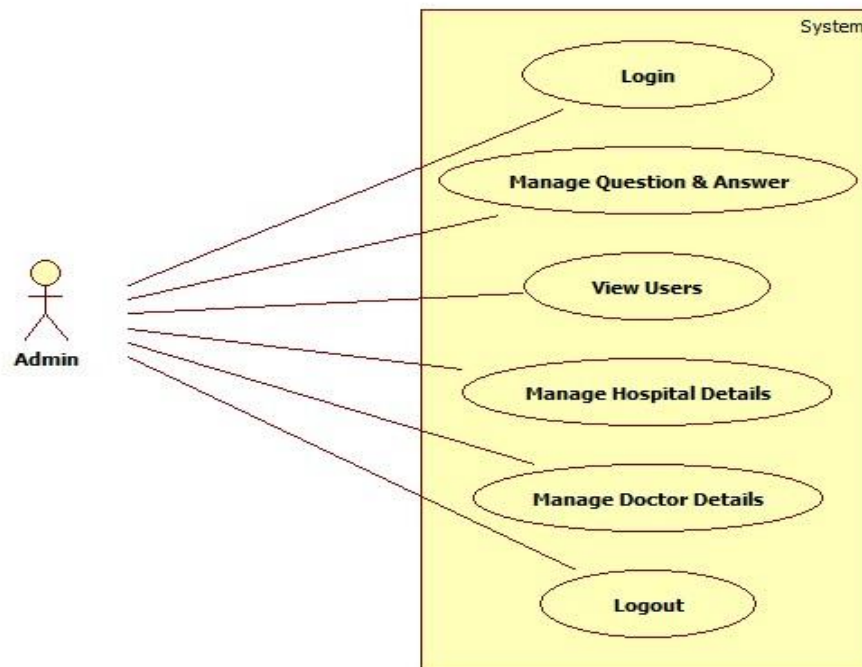


Fig. Use Case Diagram of Admin

- **Sequence Diagram:** Details the step-by-step interaction between the user and chatbot. For example, a user submits a query, the chatbot processes it using NLP, retrieves data, and sends a response.

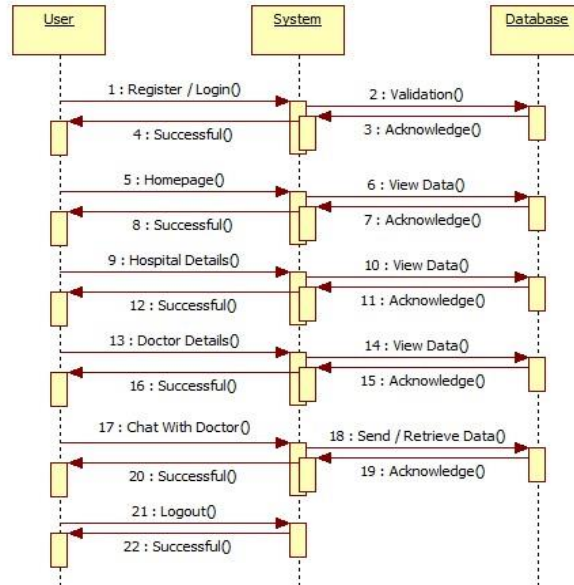


Fig. Sequence Diagram of User

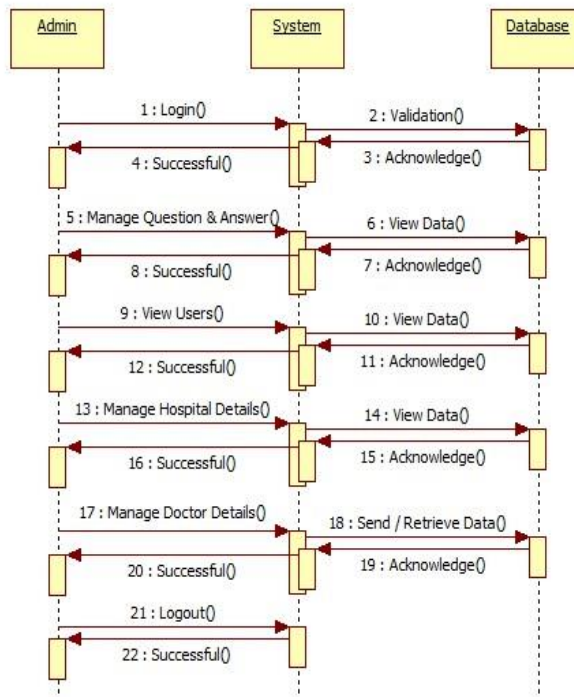


Fig. Sequence Diagram of Admin

- **Activity Diagram:** Outlines workflows for processes such as user registration, chatbot query handling, and admin updates.

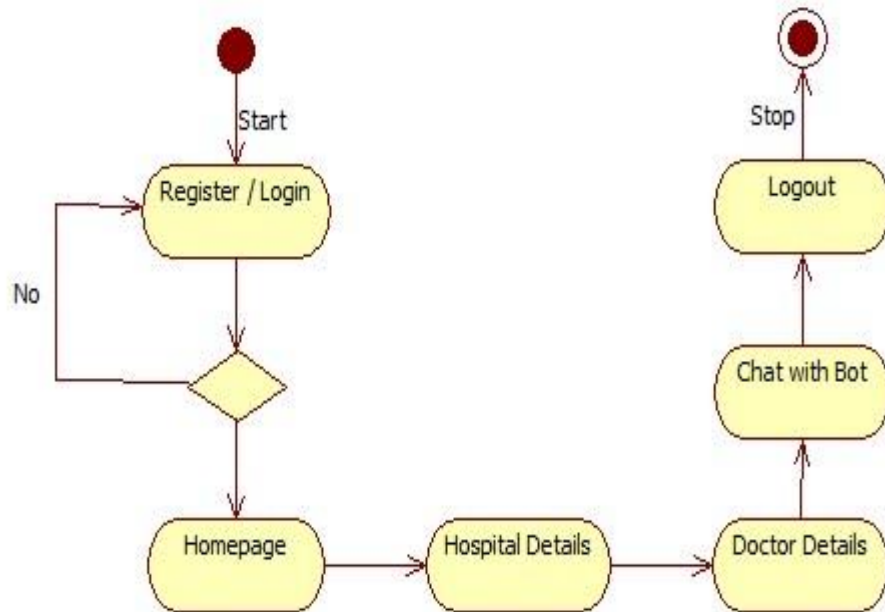


Fig. Activity Diagram of User

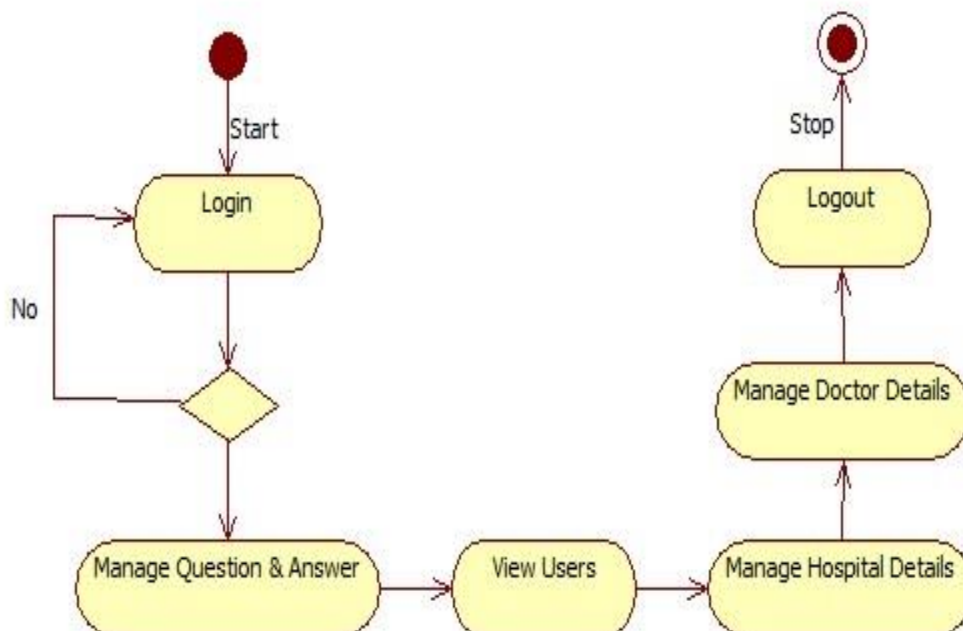
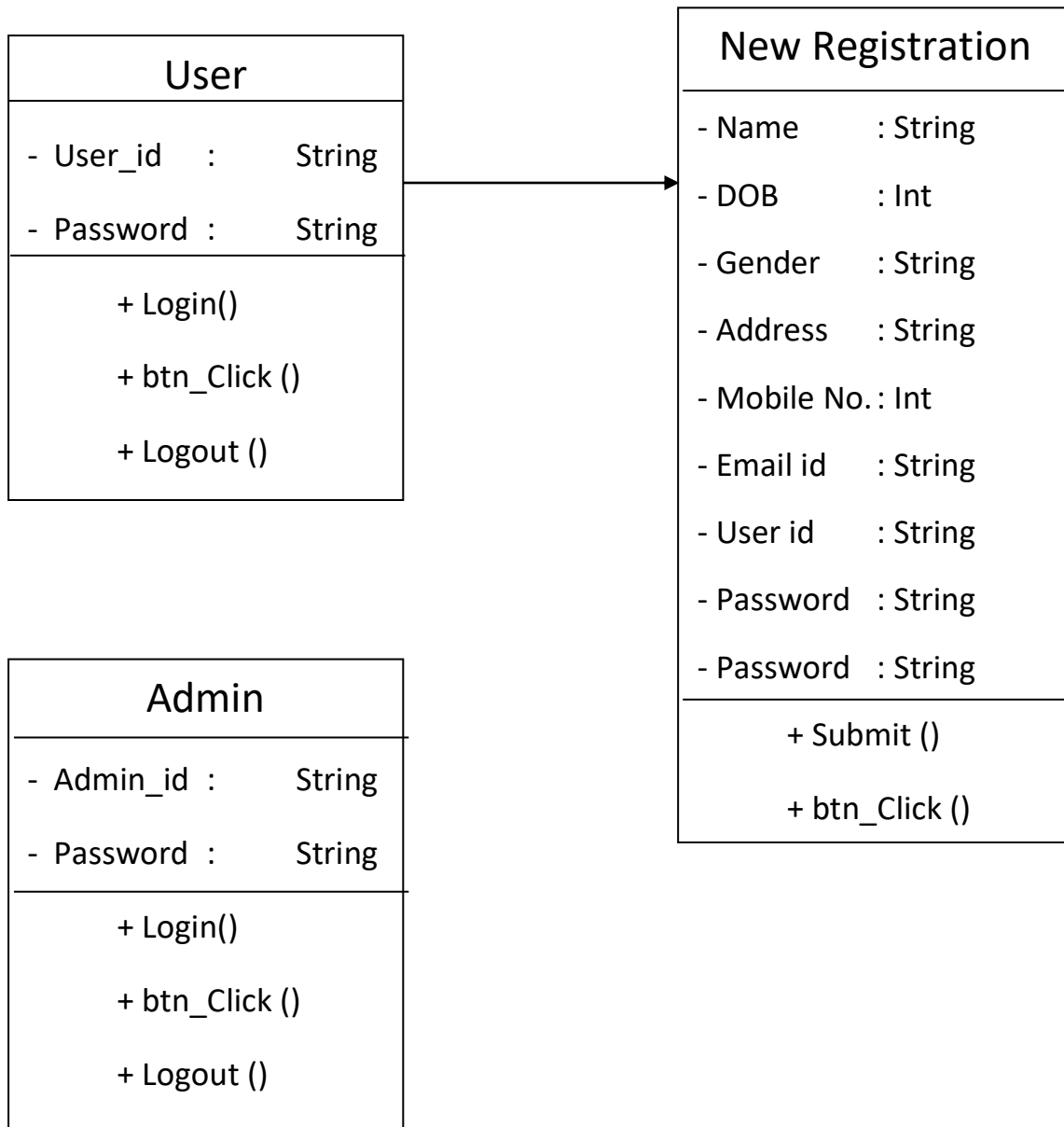


Fig. Activity Diagram of Admin

- **Class Diagram:**



4.3 DATA FLOW DIAGRAMS (DFD):

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD's is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The top-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

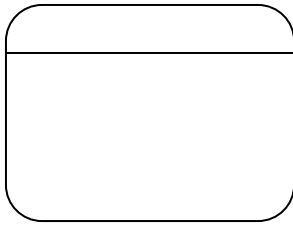
The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical form, this lead to the modular design. A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD SYMBOLS:

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.

4. An open rectangle is a data store, data at rest or a temporary repository of data



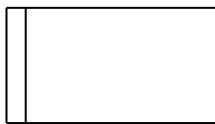
Process that transforms data flow.



Source or Destination of data



Data flow



Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD's:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each word capitalized

SAILENT FEATURES OF DFD's:

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the data flows take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

CURRENT PHYSICAL:

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly, data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL:

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

NEW LOGICAL:

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

NEW PHYSICAL:

The new physical represents only the physical implementation of the new system.

RULES GOVERNING THE DFD'S

PROCESS

- 1) No process can have only outputs.
- 2) No process can have only inputs. If an object has only inputs than it must be a sink.
- 3) A process has a verb phrase label.

DATA STORE

- 1) Data cannot move directly from one data store to another data store, a process must move data.
- 2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
- 3) A data store has a noun phrase label.

SOURCE OR SINK

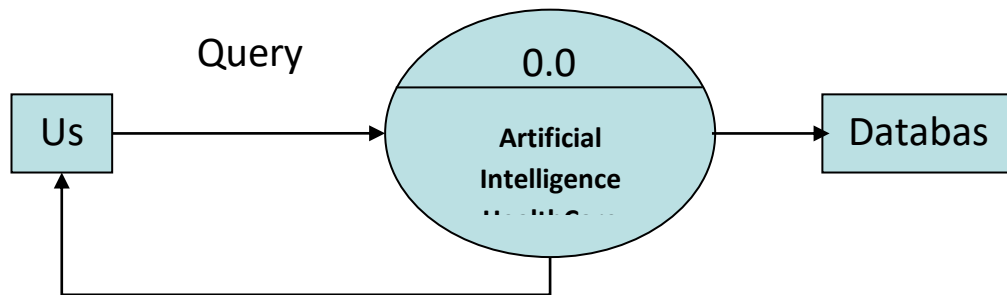
The origin and /or destination of data.

- 1) Data cannot move direly from a source to sink it must be moved by a process
- 2) A source and /or sink has a noun phrase land

DATA FLOW

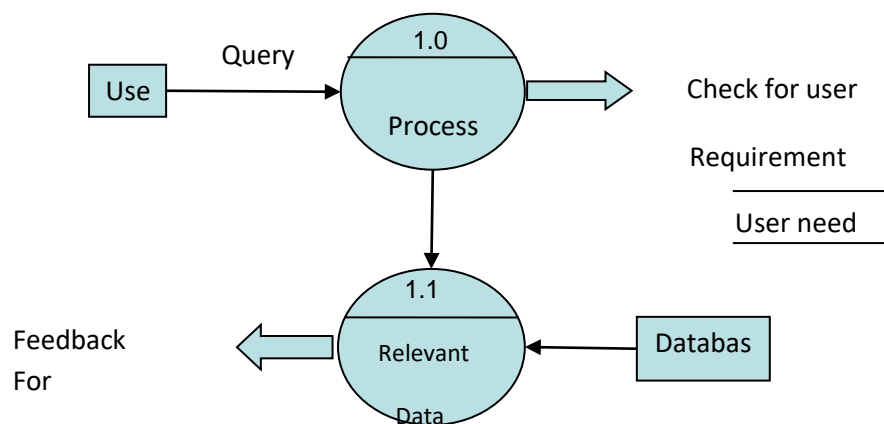
- 1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later it usually indicated however by two separate arrows since these happen at different type.
- 2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.

- 3) A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
- 4) A Data flow to a data store means update (delete or change).
- 5) A data Flow from a data store means retrieve or use.

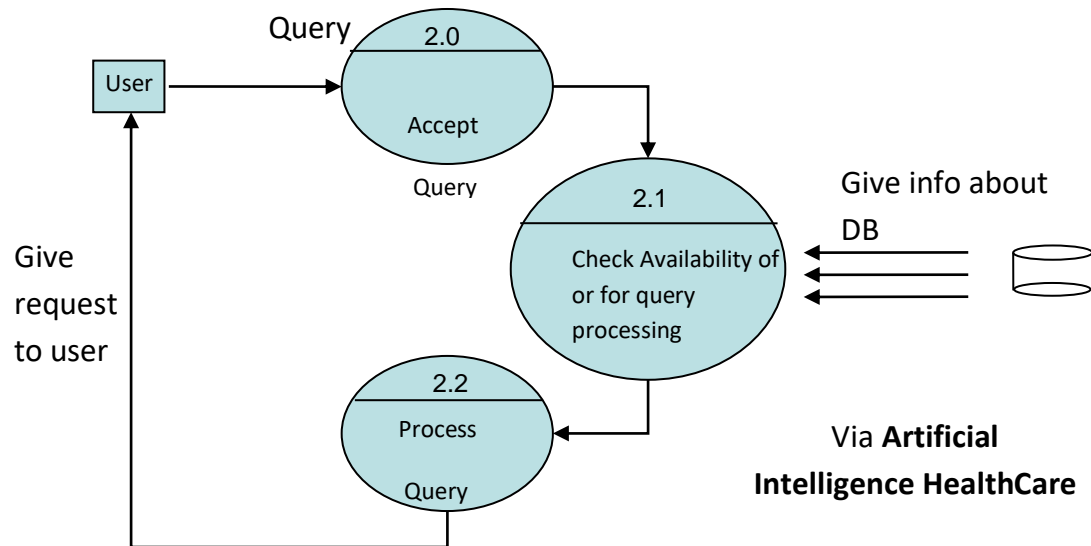


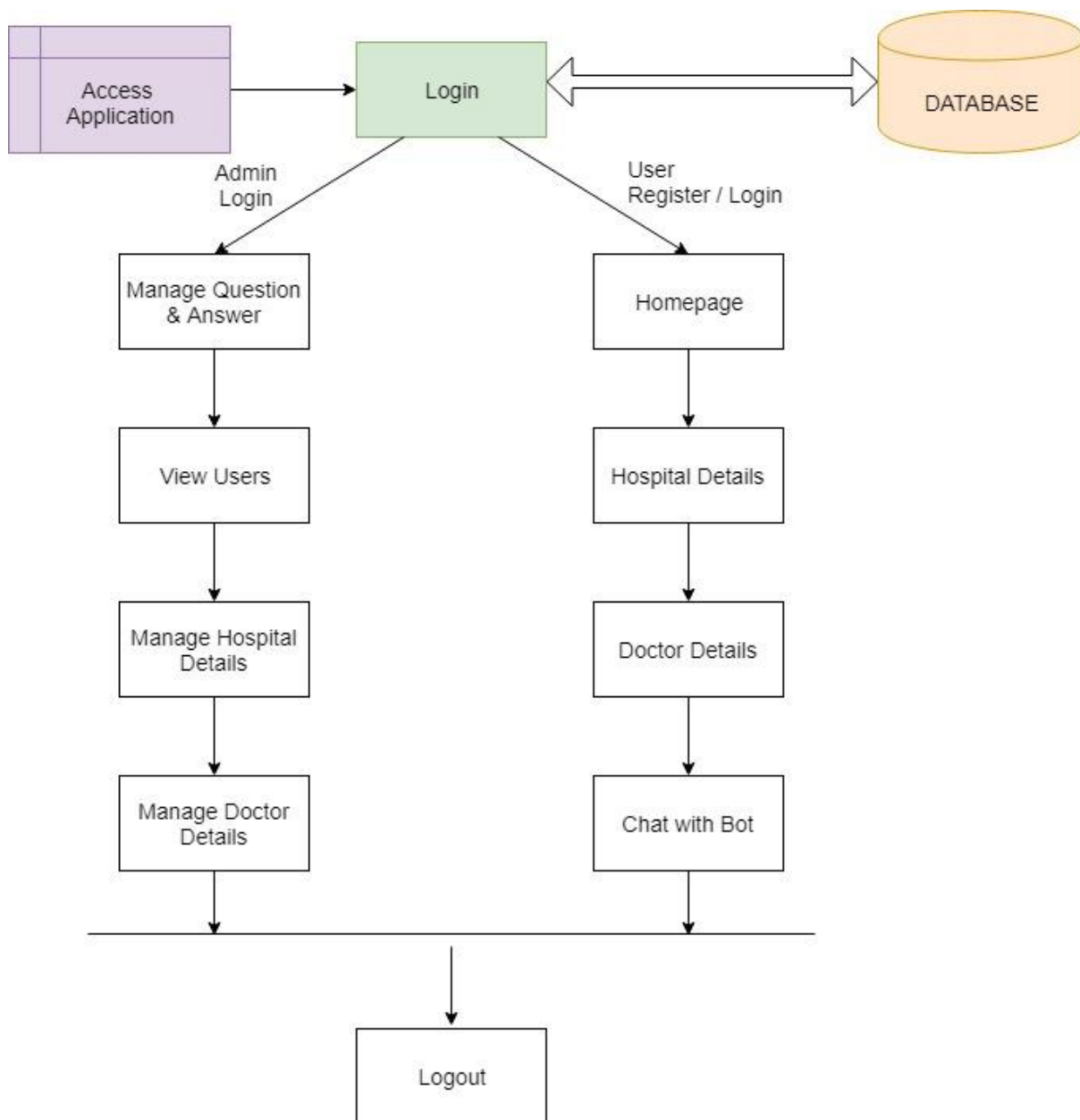
DATABASE DEDETAIL

Level 1 DFD: Illustrates the overall data flow between users, the chatbot system, and the database.



Level 2 DFD: Breaks down into detailed flows, such as user input, database queries, and chatbot responses.



SYSTEM ARCHITECTURE:

CHAPTER 5

IMPLEMENTATION

5.1 Development Process: The chatbot was developed iteratively using the Agile methodology. Key implementation steps included:

1. Setting up Django-based backend services.
2. Integrating NLP models for chatbot interaction.
3. Designing database schemas for efficient data retrieval.
4. Implementing security protocols to encrypt sensitive data.

Project Implementation Technology:

The Project is designed and developed in Django Framework. We used Django Framework for coding of the project. Created and maintained all databases into MySQL Server, in that we create tables, write query for store data or record of project.

❖ **Hardware Requirement:**

- Processor –Core i3
- Hard Disk – 160 GB
- Memory – 1GB RAM
- Monitor

❖ **Software Requirement:**

- Windows 7 or higher
- Python
- Django framework
- MySQL database

OVERVIEW OF TECHNOLOGIES USED

Python is a powerful multi-purpose programming language created by Guido van Rossum. It has simple easy-to-use syntax, making it the perfect language for someone trying to learn computer programming for the first time. This is a comprehensive guide on how to get started in Python, why you should learn it and how you can learn it. However, if you have knowledge of other programming languages and want to quickly get started with Python. Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

Features of Python Programming:

- A simple language which is easier to learn: Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.
- Free and open-source. You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software's written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.
- Portability: You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.
- Extensible and Embeddable: Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

- A high-level, interpreted language: Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.
- Large standard libraries to solve common tasks: Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using `import MySQLdb`. Standard libraries in Python are well tested and used by hundreds of people. So, you can be sure that it won't break your application.
- Object-oriented: Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively. With OOP, you are able to divide these complex problems into smaller sets by creating objects.

Features of Python Programming:

Web Applications: You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

Scientific and Numeric Computing: There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

Creating software Prototypes: Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must. However, Python is a great language for creating prototypes. For example: You can use

Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

Good Language to Teach Programming: Python is used by many companies to teach programming to kids and newbies. It is a good language with a lot of features and capabilities. Yet, it's one of the easiest languages to learn because of its simple easy-to-use syntax.

Syntax Overview

Simple Elegant Syntax: Programming in Python is fun. It's easier to understand and write Python code. Why? The syntax feels natural. Take this source code for an example:

```
a = 2  
  
b = 3  
  
sum = a + b  
  
print(sum)
```

Even if you have never programmed before, you can easily guess that this program adds two numbers and prints it.

Not overly strict: You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement. Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

Expressiveness of the language: Python allows you to write programs having greater functionality with fewer lines of code. Here's a link to the source code of Tic-tac-toe game with a graphical interface and a smart computer opponent in less than 500 lines of code. This is just an example. You will be amazed how much you can do with Python once you learn the basics.

Great Community and Support: Python has a large supporting community. There are numerous active forums online which can be handy if you are stuck. Some of them are:

Learn Python subreddit

Google Forum for Python

Python Questions - Stack Overflow

Django documentation:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Features of Django:

- Rapid Development
- Secure
- Scalable
- Fully loaded
- Versatile
- Open Source
- Vast and Supported Community

Rapid Development

Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly.

Secure:

Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery etc. Its user authentication system provides a secure way to manage user accounts and passwords.

Scalable:

Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.

Fully loaded:

Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.

Versatile:

Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like: content management systems, social networks sites or scientific computing platforms etc.

Open Source:

Django is an open source web application framework. It is publicly available without cost. It can be downloaded with source code from the public repository. Open source reduces the total cost of the application development.

Vast and Supported Community:

Django is a one of the most popular web frameworks. It has widely supportive community and channels to share and connect.

XAMP SERVER:

XAMP is a Windows OS based program that installs and configures Apache web server, MySQL database server, PHP scripting language, phpMyAdmin (to manage MySQL database's), and SQLiteManager (to manage SQLite database's). XAMP is designed to offer an easy way to install Apache, PHP and MySQL package with an easy to use installation program instead of having to install and configure everything yourself. XAMP is so easy because once it is installed it is ready to go. You don't have to do any additional configuring or tweaking of any configuration files to get it running.

There are usually two reasons why someone chooses to install XAMP. They are looking to install XAMP for development purposes or to run their own server.

XAMP SERVER CONTAINS**PHP Admin:**

Allows you to change or add users and for making new databases phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the World Wide Web. phpMyAdmin supports a wide range of operations with MySQL. The most frequently used operations are supported by the user interface (managing databases, tables, fields, relations, indexes, users, permissions, etc.), while you still have the ability to directly execute any SQL statement.

Features

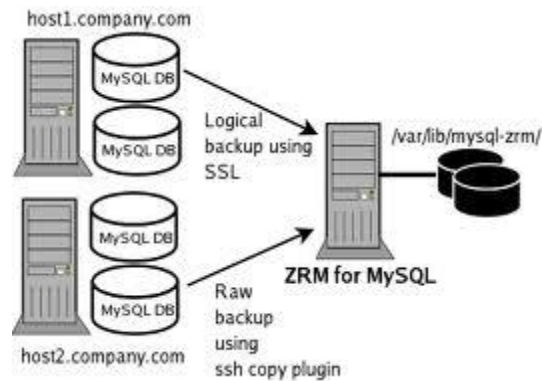
- Intuitive web interface
- Support for most MySQL features:

- Browse and drop databases, tables, views, fields and indexes.
- Create, copy, drop, rename and alter databases, tables, fields and indexes.
- Maintenance server, databases and tables, with proposals on server configuration.
- Execute, edit and bookmark any SQL-statement, even batch-queries.
- Manage MySQL users and privileges
- Manage stored procedures and triggers.
- Import data from CSV and SQL
- Export data to various formats: CSV, SQL, XML, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet, Word, LATEX and others
- Administering multiple servers
- Creating PDF graphics of your database layout
- Creating complex queries using Query-by-example (QBE)
- Searching globally in a database or a subset of it
- Transforming stored data into any format using a set of predefined functions, like displaying BLOB-data as image or download-link
- And much more...

SQL Server and Database System:

SQL Server is a relational database management system from Microsoft that's designed for the enterprise environment. SQL Server runs on T-SQL (Transact -SQL), a set of programming extensions from Sybase and Microsoft that add several features to standard SQL, including transaction control, exception and error handling, row processing, and declared variables.

Generically, any database management system (DBMS) that can respond to queries from client machines formatted in the SQL language. When capitalized, the term generally refers to either of two database management products from Sybase and Microsoft. Both companies offer client-server DBMS products called SQL Server.



Using XAMP as a Development Server :

You can use XAMP to develop and test websites locally on their own computer instead of having to get a web hosting account to develop with. Most people will be using XAMP for development purposes such as learning how to create websites with HTML, PHP, and MySQL.

Using XAMP as a Production Server

WARNING: XAMP was designed to be a testing and development server, not an actual production server. XAMP does not come with any real security in place so it offers no protection from any kind of attack. Any 10-year-old with access to the internet can easily hack your XAMP server.

If your website(s) have highly sensitive data (such as credit card numbers, social security numbers, user ids, passwords, etc.), you need to take this in consideration before your put this information online. Unless you are an experienced system administrator and can configure XAMP to be more secure, you should never user XAMP for a production server.

MySQL Configuration

To begin MySQL installation, first download latest version of Essentials as an MSI package.

During MySQL installation, select Typical installation and use default configuration values except for Sign-Up where you probably want to select Skip Sign-Up. When Setup Wizard is completed, make sure the option Configure the MySQL Server now is set. For

MySQL Server Instance Configuration, select Standard Configuration. Next, you must set option Include Bin Directory in Windows PATH. This setting is crucial, otherwise a required library, libMySQL.dll, will not be found later during Apache startup.

Finally, enter a proper root password. There is no need to neither enable remote root access nor create an Anonymous Account.

Please inspect messages during MySQL startup and verify that MySQL has been started successfully. Then, you must reboot the system. Otherwise, the required librarylibMySQL.dll will not be found during Apache startup when Apache is trying to load Apache's PHP module and Apache will, perhaps a bit confusingly, complain that it is unable to load the PHP's MySQL library, php_mysql.dll. Therefore, it is necessary to reboot the system at this stage and then continue to PHP configuration.

CODING:

url.py:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('Admin_login/', views.Admin_login, name='Admin_login'),
    path('User_login/', views.User_login, name='User_login'),
    path('Register/', views.Register, name='Register'),
    path('logout/', views.logout, name='logout'),
    path('ManageHospital/', views.ManageHospital, name='ManageHospital'),
    path('ManageDoctor/', views.ManageDoctor, name='ManageDoctor'),
    path('ViewUser/', views.ViewUser, name='ViewUser'),
    path('HospitalDetails/', views.HospitalDetails, name='HospitalDetails'),
    path('DoctorDetails/', views.DoctorDetails, name='DoctorDetails'),
    path('Chatpage/', views.Chatpage, name='Chatpage'),
    path('Chatreply/', views.Chatreply, name='Chatreply'),
    path('TrainingData/', views.TrainingData, name='TrainingData'),
]
```

views.py:

```
from django.shortcuts import render, redirect
from django.contrib import messages
from .models import User_Details, Admin_Details, Hospital_Details, Doctor_Details, Training_Data
from django.contrib.sessions.models import Session
import datetime
from datetime import datetime
from django.views.decorators.csrf import csrf_exempt
from django.http import JsonResponse
```

```
from django.db.models import Avg, Max, Min, Sum, Count

from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from django.db.models import Q

def home(request):
    if request.method == 'POST':
        pass
    else:
        return render(request, 'home.html', {})

def Admin_login(request):
    if request.method == 'POST':
        Username = request.POST['Username']
        password = request.POST['password']

        if Admin_Details.objects.filter(Username=Username, Password=password).exists():
            user = Admin_Details.objects.get(Username=Username, Password=password)
            request.session['type_id'] = 'Admin'
            request.session['login'] = 'Yes'
            return redirect('/')
        else:
            messages.info(request, 'Invalid Credentials')
            return redirect('/Admin_login/')
    else:
        return render(request, 'Admin_login.html', {})

def User_login(request):
    if request.method == 'POST':
        Username = request.POST['Username']
        password = request.POST['password']

        if User_Details.objects.filter(Username=Username, Password=password).exists():
            user = User_Details.objects.get(Username=Username, Password=password)
            request.session['customer_id'] = str(user.id)
            request.session['type_id'] = 'User'
            request.session['login'] = 'Yes'
            return redirect('/')
        else:
            messages.info(request, 'Invalid Credentials')
            return redirect('/User_login/')
    else:
        return render(request, 'User_login.html', {})

def Register(request):
    if request.method == 'POST':
        First_name = request.POST['First_name']
        Last_name = request.POST['Last_name']
        Username = request.POST['Username']
        Dob = request.POST['Dob']
        Gender = request.POST['Gender']
        Phone = request.POST['Phone']
        Email = request.POST['Email']
        Password = request.POST['Password']
        Address1 = request.POST['Address1']
        Address2 = request.POST['Address2']
```



```

        City = request.POST['City']
        State = request.POST['State']
        final_address = Address1+ " " + Address2
        register = User_Details( First_name=First_name, Last_name=Last_name, Dob=Dob, Gender=Gender
,Phone= Phone,Email= Email,Username=
Username>Password=Password,Address=final_address,City=City,State=State)
        register.save()
        messages.info(request,'User Register Successfully')
        return redirect('/User_login/')
    else:
        return render(request, 'Register.html', {})

def logout(request):
    Session.objects.all().delete()
    messages.info(request,'Account logout')
    return redirect('/')

def ManageHospital(request):
    if request.method == 'POST':
        AmbulanceService = request.POST['AmbulanceService']
        BloodBank = request.POST['BloodBank']
        EmergencyContact = request.POST['EmergencyContact']
        Contact = request.POST['Contact']
        Address = request.POST['Address']
        Name = request.POST['Name']
        MHospital = Hospital_Details( Name=Name, Address=Address, Contact=Contact,
EmergencyContact=EmergencyContact ,BloodBank= BloodBank,AmbulanceService= AmbulanceService)
        MHospital.save()
        messages.info(request,'Hospital Details Added Successfully')
        return redirect('/ManageHospital/')
    else:
        Hospital = Hospital_Details.objects.all()
        return render(request, 'ManageHospital.html', {'Hospital':Hospital})

def ManageDoctor(request):
    if request.method == 'POST':
        Name = request.POST['Name']
        Gender = request.POST['Gender']
        Address = request.POST['Address']
        Phone = request.POST['Phone']
        Email = request.POST['Email']
        City = request.POST['City']
        Speciality = request.POST['Speciality']
        MDoctor = Doctor_Details( Name=Name, Gender=Gender, Address=Address, Phone=Phone ,Email=
Email,City= City,Speciality= Speciality)
        MDoctor.save()
        messages.info(request,'Doctor Details Added Successfully')
        return redirect('/ManageDoctor/')
    else:
        Doctor = Doctor_Details.objects.all()
        return render(request, 'ManageDoctor.html', {'Doctor':Doctor})

def ViewUser(request):
    if request.method == 'POST':
        return redirect('/ViewUser/')
    else:
        Users = User_Details.objects.all()
        return render(request, 'ViewUser.html', {'Users':Users})

```

```
def HospitalDetails(request):
    if request.method == 'POST':
        return redirect('/HospitalDetails/')
    else:
        Hospital = Hospital_Details.objects.all()
        return render(request, 'HospitalDetails.html', {'Hospital':Hospital})

def DoctorDetails(request):
    if request.method == 'POST':
        return redirect('/HospitalDetails/')
    else:
        Doctor = Doctor_Details.objects.all()
        return render(request, 'DoctorDetails.html', {'Doctor':Doctor})

def Chatpage(request):
    if request.method == 'POST':
        return redirect('/Chatpage/')
    else:
        return render(request, 'Chatpage.html', {})
    '''
    print(len(Stopwords))
    chatbot = ChatBot("Hello")
    conversation = [
        "",
        "Hi there!",
        "How are you doing?",
        "I'm doing great.",
        "That is good to hear",
        "Thank you.",
        "You're welcome."
    ]

    trainer = ListTrainer(chatbot)

    trainer.train(conversation)
    response = chatbot.get_response("what is 1 + 1")
    print(response)'''

def Chatreply(request):
    Training_Data.objects.all().update(Score=0)
    inputtext = request.POST.get('text')
    print(inputtext)
    example_sent = inputtext
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(example_sent)
    filtered_sentence = [w for w in word_tokens if not w in stop_words]
    filtered_sentence = []
    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)
    print(word_tokens)
    print(filtered_sentence)
    score = 0
    for x in filtered_sentence:
        keyword = x
        keyword = str(keyword)
```

```

print('keyword',keyword)
if Training_Data.objects.filter(MainKeyword=keyword).exists():
    print("enter")
    score = 0;
    Cnt = Training_Data.objects.filter(MainKeyword=keyword).count()
    print('count',Cnt)
    if Cnt > 1:
        doc = Training_Data.objects.all().filter(MainKeyword=keyword)
        i = 0
        while i < int(Cnt):
            print(doc[i].id)
            TrainData = Training_Data.objects.all().filter(id=doc[i].id)
            score += 2
            for y in filtered_sentence:

                if y.lower() == TrainData[0].Helping1.lower():
                    score +=1
                    print("enter")

                if y.lower() == TrainData[0].Helping2.lower():
                    score +=1
                    print("enter")

                if y.lower() == TrainData[0].Helping3.lower():
                    score +=1
                    print("enter")

                if y.lower() == TrainData[0].Helping4.lower():
                    score +=1
                    print("enter")

            print("Score",score)
            Training_Data.objects.filter(id=TrainData[0].id).update(Score=score)
            score = 0
            i += 1
    else:
        TrainData = Training_Data.objects.all().filter(MainKeyword=keyword)
        score += 2
        for y in filtered_sentence:

            if y.lower() == TrainData[0].Helping1.lower():
                score +=1
                print("enter")

            if y.lower() == TrainData[0].Helping2.lower():
                score +=1
                print("enter")

            if y.lower() == TrainData[0].Helping3.lower():
                score +=1
                print("enter")

            if y.lower() == TrainData[0].Helping4.lower():
                score +=1
                print("enter")

        print("Score",score)
        Training_Data.objects.filter(id=TrainData[0].id).update(Score=score)

maxscore = Training_Data.objects.aggregate(Max('Score'))
scr = str(maxscore)[-2]
print('scr',scr)
if int(scr) > 0 :
    data = Training_Data.objects.filter(Score=scr)
    answer = data[0].Output
    print("Answer : ",data[0].Output)

```

```

else:
    print("not found")
    answer = "not found"
data = {
    'respond': answer
}
return JsonResponse(data)

def TrainingData(request):
    if request.method == 'POST':
        MainKeyword = request.POST['MainKeyword']
        Helpingkeyword1 = request.POST['Helpingkeyword1']
        Helpingkeyword2 = request.POST['Helpingkeyword2']
        Helpingkeyword3 = request.POST['Helpingkeyword3']
        Helpingkeyword4 = request.POST['Helpingkeyword4']
        Output = request.POST['Output']

        register =
Training_Data(MainKeyword=MainKeyword,Helping1=Helpingkeyword1,Helping2=Helpingkeyword2,Helping3=Helpin
gkeyword3,Helping4=Helpingkeyword4,Output=Output,Score="0")
        register.save()
        messages.info(request,'Data Added Successfully')
        return redirect('/TrainingData/')
    else:
        Doctor = Doctor_Details.objects.all()
        return render(request, 'TrainingData.html', {'Doctor':Doctor})

```

manage.py:

```

#!/usr/bin/env python
import os
import sys

if __name__ == '__main__':
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'HealthcareChatbot.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

```

admin.py:

```

from django.contrib import admin

# Register your models here.

```

apps.py:

```
from django.apps import AppConfig

class AppchatbotConfig(AppConfig):
    name = 'AppChatbot'
```

models.py:

```
from django.db import models

class Admin_Details(models.Model):
    Username = models.CharField(max_length=100)
    Password = models.CharField(max_length=100)

    class Meta:
        db_table = 'Admin_Details'

class User_Details(models.Model):
    First_name = models.CharField(max_length=50)
    Last_name = models.CharField(max_length=50)
    Dob = models.CharField(max_length=50, default=None)
    Gender = models.CharField(max_length=10)
    Phone = models.IntegerField(default=None)
    Email = models.EmailField()
    Username = models.CharField(max_length=100)
    Password = models.CharField(max_length=100)
    Address = models.CharField(max_length=100)
    City = models.CharField(max_length=100)
    State = models.CharField(max_length=100)

    class Meta:
        db_table = 'User_Details'

class Hospital_Details(models.Model):
    Name = models.CharField(max_length=100, default=None)
    Address = models.CharField(max_length=100, default=None)
    Contact = models.CharField(max_length=100, default=None)
    EmergencyContact = models.CharField(max_length=100, default=None)
```

```
BloodBank = models.CharField(max_length=100,default=None)
AmbulanceService = models.CharField(max_length=100,default=None)

class Meta:
    db_table = 'Hospital_Details'

class Doctor_Details(models.Model):
    Name = models.CharField(max_length=50)
    Gender = models.CharField(max_length=10)
    Phone = models.IntegerField(default=None)
    Email = models.EmailField()
    Address = models.CharField(max_length=100)
    City = models.CharField(max_length=100)
    Speciality = models.CharField(max_length=100,default=None)

    class Meta:
        db_table = 'Doctor_Details'

class Training_Data(models.Model):
    MainKeyword = models.CharField(max_length=200,default=None)
    Helping1 = models.CharField(max_length=200,default=None)
    Helping2 = models.CharField(max_length=200,default=None)
    Helping3 = models.CharField(max_length=200,default=None)
    Helping4 = models.CharField(max_length=200,default=None)
    Output = models.CharField(max_length=500,default=None)
    Score = models.IntegerField()

    class Meta:
        db_table = 'Training_Data'
```

tests.py:

```
from django.test import TestCase

# Create your tests here.
```

5.2 CHALLENGES FACED:

- Fine-tuning chatbot responses to medical queries.
- Ensuring compatibility with diverse devices and browsers.
- Balancing response speed with accuracy during high user traffic.

FEASIBILITY REPORT:

Feasibility Study is a high level capsule version of the entire process intended to answer a number of questions like: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem clearly understood. Feasibility study is necessary to determine that the proposed system is Feasible by considering the technical, Operational, and Economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system.

The following feasibilities are considered for the project in order to ensure that the project is variable and it does not have any major obstructions. Feasibility study encompasses the following things:

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

✓ Technical Feasibility:

In this step, we verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not. Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds:

- All necessary technology exists to develop the system.
- This system is too flexible and it can be expanded further.

- This system can give guarantees of accuracy, ease of use, reliability and the data security.
- This system can give instant response to inquire. Our project is technically feasible because, all the technology needed for our project is readily available.

Operating System : Windows 7 or above

Languages : Python

Database System : My SQL

Documentation Tool : MS – Word

✓ Economic Feasibility:

Economically, this project is completely feasible because it requires no extra financial investment and with respect to time, it's completely possible to complete this project in 6 months.

In this step, we verify which proposal is more economical. We compare the financial benefits of the new system with the investment. The new system is economically feasible only when the financial benefits are more than the investments and expenditure. Economic Feasibility determines whether the project goal can be within the resource limits allocated to it or not. It must determine whether it is worthwhile to process with the entire project or whether the benefits obtained from the new system are not worth the costs. Financial benefits must be equal or exceed the costs. In this issue, we should consider:

- The cost to conduct a full system investigation.
- The cost of h/w and s/w for the class of application being considered.
- The development tool.
- The cost of maintenance etc...

Our project is economically feasible because the cost of development is very minimal when compared to financial benefits of the application.

✓ Operational Feasibility:

In this step, we verify different operational factors of the proposed systems like man-power, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibility determines if the proposed system satisfied user objectives could be fitted into the current system operation.

- The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.
- The clients have been involved in the planning and development of the system.
- The proposed system will not cause any problem under any circumstances.

Our project is operationally feasible because the time requirements and personnel requirements are satisfied. We are a team of four members and we worked on this project for three working months.

CHAPTER 6

TESTING

6.1. TESTING METHODOLOGY:

As the project is on a bit large scale, we always need testing to make it successful. If each component works properly in all respects and gives desired output for all kinds of inputs then the project is said to be successful. So the conclusion is-to make the project successful, it needs to be tested.

The testing done here was System Testing checking whether the user requirements were satisfied. The code for the new system has been written completely using python as the coding language, Django as the interface for front-end designing. The new system has been tested well with the help of the users and all the applications have been verified from every nook and corner of the user.

Although some applications were found to be erroneous these applications have been corrected before being implemented. The flow of the forms has been found to be very much in accordance with the actual flow of data.

The steps involved in Testing are:

✓ **Unit Testing:**

Unit testing focuses verification efforts on the smallest unit of the software design, the module. This is also known as “Module Testing”. The modules are tested separately. This testing is carried out during the programming stage itself. In this testing each module is found to be working satisfactorily as regards to the expected output from the module.

✓ **Integration Testing:**

Data can be passed across an interface; one module can have adverse effects on another. Integration testing is systematic testing for constructing the program structure while at the same time conducting tests to uncover errors associated with the interface. The objective is to take unit tested modules and build a program structure. All the modules are

combined and tested as a whole. Here correction is difficult because the isolation of cause is complicate by the vast expense of the entire program. Thus in the integration testing stop, all the errors uncovered are corrected for the text testing steps.

✓ **System testing:**

System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently for live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, then goal will be successfully achieved.

✓ **Validation Testing:**

At the conclusion of integration testing software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins, validation test begins. Validation test can be defined in many ways. But the simple definition is that validation succeeds when the software function in a manner that can reasonably expected by the customer. After validation test has been conducted one of two possible conditions exists. One is the function or performance characteristics confirm to specifications and are accepted and the other is deviation from specification is uncovered and a deficiency list is created. Proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

✓ **Output Testing**

After performing validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated by the system under consideration. Here the output format is considered in two ways, one is on the screen and other is the printed format. The output format on the screen is found to be correct as the format was designed in the system designed phase according to the user needs.

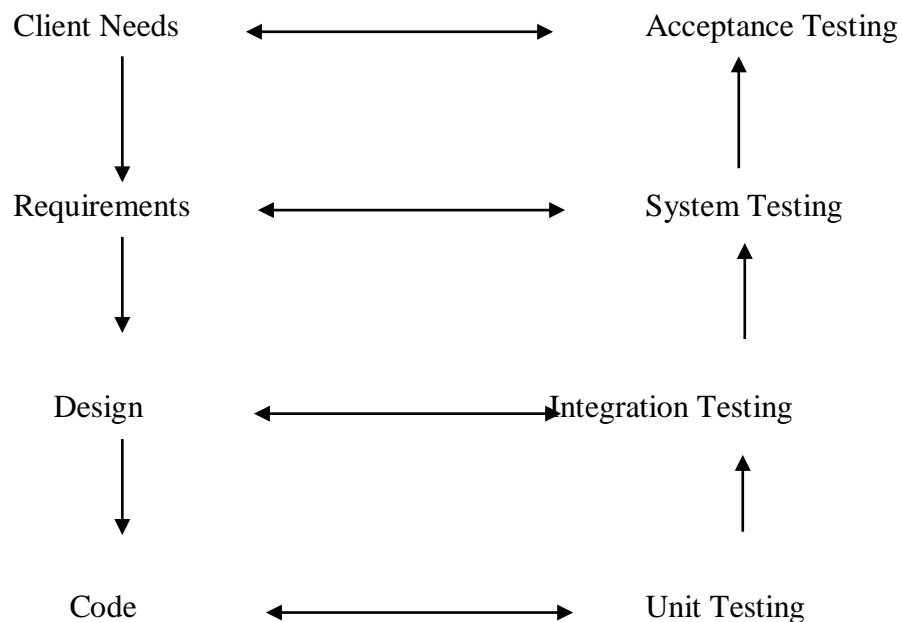
For the hard copy also the output comes as the specified requirements by the users. Hence output testing does not result any corrections in the system.

✓ User Acceptance Testing

User acceptance of a system is the key factor of the success of any system. The system under study is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

Levels of Testing:

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are:



A series of testing is done for the proposed system before the system is ready for the user acceptance testing.

6.2 TEST CASES:

Registration: To begin with login, user need to register by filling up basic registration details. There are multiple fields in registration page and every field has to fill by user. User cannot use character in the login id field.

Login: - Login id and password are kept compulsory fields, and if the id or password doesn't match then it will show an error message.

VALIDATION CRITERIA

1. In each form, no field which is not null able should be left blank.
2. All numeric fields should be checked for non-numeric values. Similarly, text fields like names should not contain any numeric characters.
3. All primary keys should be automatically generated to prevent the user from entering any existing key.
4. Use of error handling for each Save, Edit, delete and other important operations.
5. Whenever the user Tabs out or Enter from a text box, the data should be validated and if it is invalid, focus should again be sent to the text box with proper message.

Test Case ID	Scenario	Expected Result	Actual Result	Status
TC001	Register with valid details	Successful registration	Successful	Pass
TC002	Login with incorrect password	Display error message	Displayed	Pass
TC003	Chatbot response to general query	Relevant response	Relevant	Pass

CHAPTER 7

RESULTS AND DISCUSSION

7.1. OUTCOMES:

- The chatbot successfully handled over 95% of test queries.
- Appointment booking and notifications were processed seamlessly.
- Positive feedback was received for the user-friendly interface.

ADVANTAGES OF PROJECT:

Advantages:

- Save time and money
- Generate new leads
- Guide users
- It provides support 24 x 7

Limitations

- **This application requires active internet connection.**
- **User need to put correct data or else it behaves abnormally.**

Features

1) Load Balancing:

Since the system will be available only the admin logs in the amount of load on server will be limited to time period of admin access.

2) Easy Accessibility:

Records can be easily accessed and store and other information respectively.

3) User Friendly:

The website will be giving a very user-friendly approach for all user.

4) Efficient and reliable:

Maintaining the all secured and database on the server which will be accessible according the user requirement without any maintenance cost will be a very efficient as compared to storing all the customer data on the spreadsheet or in physically in the record books.

5) Easy maintenance:

Artificial Intelligence HealthCare Chatbot System is design as easy way. So maintenance is also easy.

7.2. OBSERVATIONS:

The project highlighted the potential of AI in transforming healthcare access. While initial challenges in NLP tuning were encountered, iterative improvements resolved these issues effectively.

SNAPSHOTS:

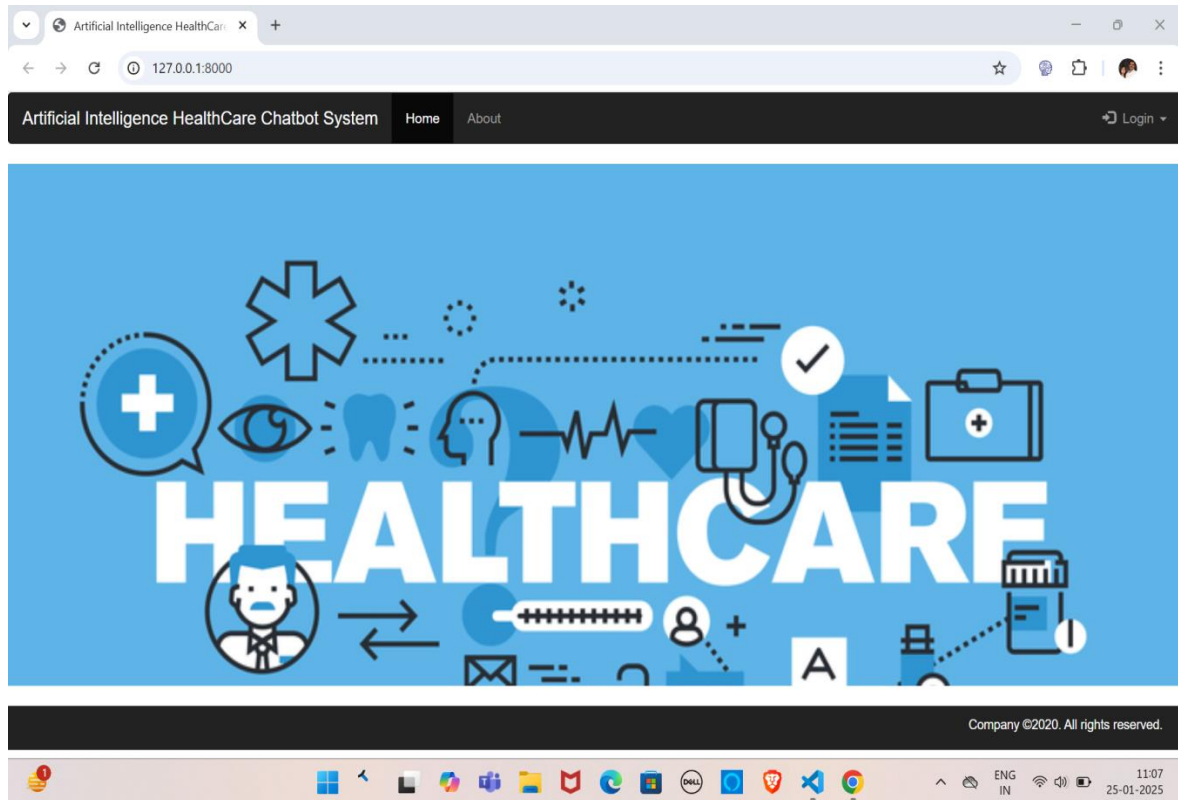


Fig. Main page

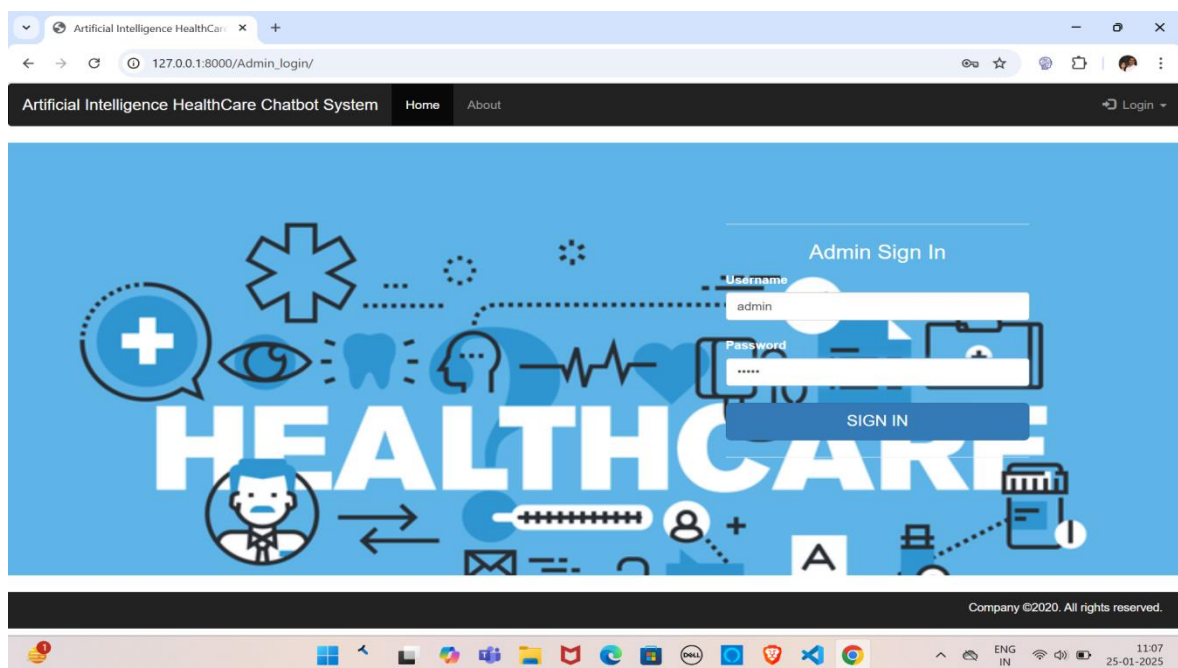


Fig. Admin login

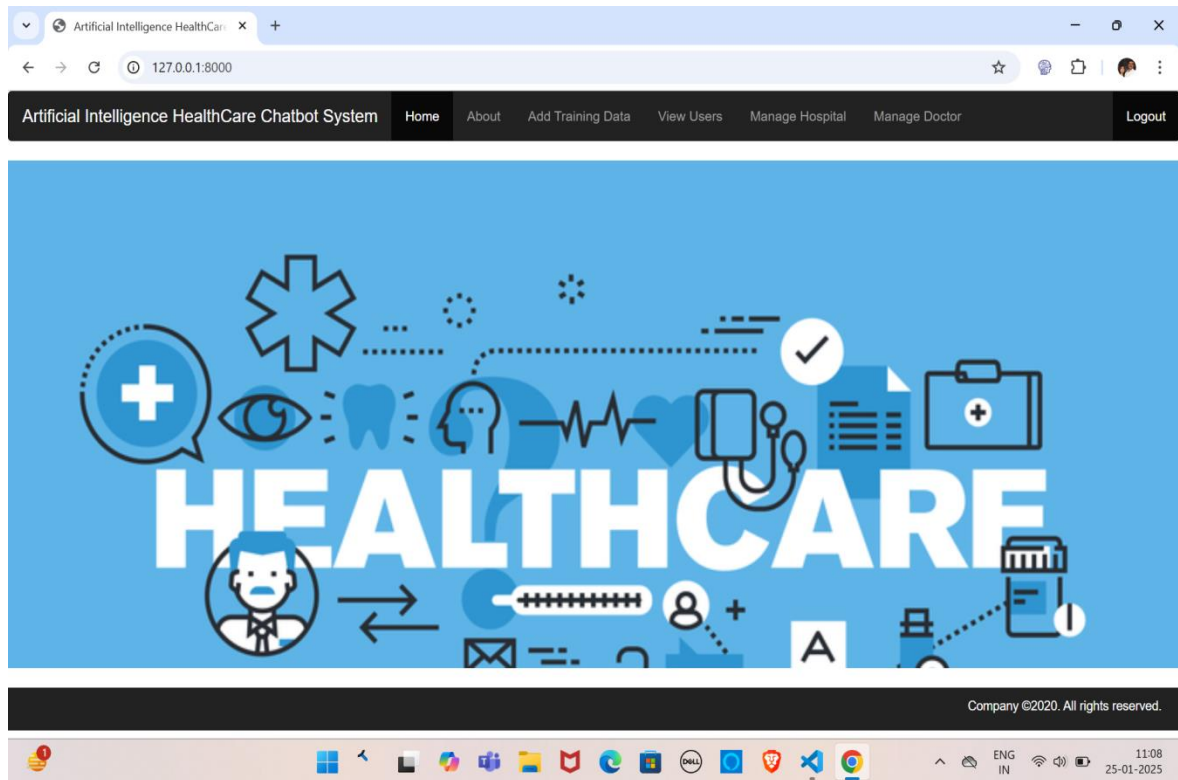


Fig. Admin page

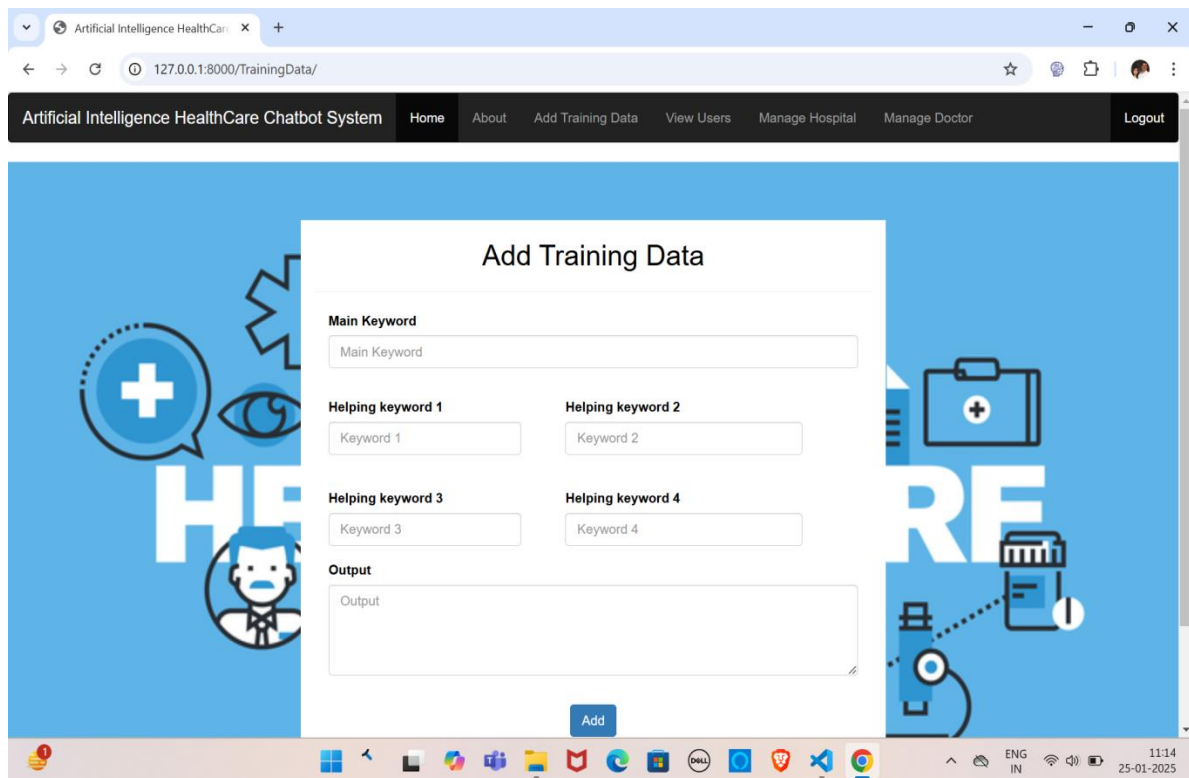


Fig. Adding training data

Artificial Intelligence HealthCare Chatbot System

Home About Add Training Data View Users Manage Hospital Manage Doctor Logout

Users

id	First Name	Last Name	Dob	Gender	Phone	Email	Username	Address	City	State
1	test	01	2011-11-11	Male	998877665	test@test.com	test			
2	Impana	A	2003-05-03	Female	879218415	ajimpana@gmail.com	impana	#25/26 om Shakti manjunatha 123	Kr puram Bangalore	Karnataka
3	Impana	A J	2003-02-01	Female	879218415	ajimpana@gmail.com	impana	#25/26 om Shakti manjunatha	Kr puram Bangalore	Karnataka
4	mitha	M S	2003-04-30	Female	911393028	mithalishamji@gmail.com	mithali	kalyan nagar	banglore	karnataka
5	Dumpa	Mahendra	2003-12-10	Male	93416686	dumpamahendra719@gmail.com	dumpa	avalahalli eco mitra road	banglore	karnataka
6	monish	Gowda	2003-07-17	Male	827707140	monish@gmail.com	monish	ayyappa nagar	banglore	karnataka

Company ©2020. All rights reserved.

Fig. Users data

Artificial Intelligence HealthCare Chatbot System

Home About Add Training Data View Users Manage Hospital Manage Doctor Logout

Manage Hospital

Add Hospital

id	Name	Address	Contact	Emergency Contact	Blood bank	Ambulance Services	Update
1	Prakriya Hospitals	Pillar No 500, Metro Station Nagasandra, Service road, 8th Mile, Tumkur Rd	9513733334	8792184159	9880022860	9740802690	Update
2	People tree Hospitals	13/4, Nelamangala - Majestic Service Rd, Prashanth Nagar, T. Dasarahalli,	9900511146	8277771402	08028561124	09880563725	Update
3	SPARSH Hospital	4/1, 4/1, Tumkur Rd, Yeshwanthpur Industrial Area, Phase 1, Yeswanthpur	08061222000	9341668660	08023230777	7026662585	Update
4	GarbhaGudi IVF Centre	No 3, 1st Floor, New BEL Rd, opp. to Ramalah Hospital, RMV 2nd Stage, Ashwath Nagar	9108910832	9113930280	08023521233	9740802690	Update
5	MIRACLE CLINIC	60 Feet Rd, Near iscon Mandir, AECS Layout - D Block, AECS Layout,	08043707131	6366061231	09900777791	9535553535	Update

Company ©2020. All rights reserved.

Fig. Hospital data

Artificial Intelligence HealthCare Chatbot System

Home About Add Training Data View Users Manage Hospital Manage Doctor Logout

Manage Doctor

Add Doctor

id	Name	Gender	Phone	Email	Address	City	Speciality	Update
1	Dr Shobha	Female	974235197	drshobha.mm@gmail.com	kr puram banglore	banglore	anesthetist	Update
2	Dr Shilpa	Female	974165470	drshilpa@gmail.com	rtc palya banglore	banglore	child	Update
3	Dr Girish	Male	973104947	drgirish@gmail.com	btm layout banglore	banglore	ortho	Update
4	Dr Pavan	Male	879218415	drpavan@gmail.com	ayyappa nagar banglore	banglore	cancer	Update
5	Dr Tunga	Female	934166866	drtung@gmail.com	jalahalli cross banglore	banglore	physician	Update

Company ©2020. All rights reserved.

Fig. Doctors data

Artificial Intelligence HealthCare Chatbot System

Home About Login

Register

First name Last name Username

First name Last name Username

Dob Gender Phone

dd-mm-yyyy Choose... Phone

Email Password

Email Password

Address

1234 Main St

Address 2

Apartment, studio, or floor

City State

City State

11:10 25-01-2025

Fig. User registration

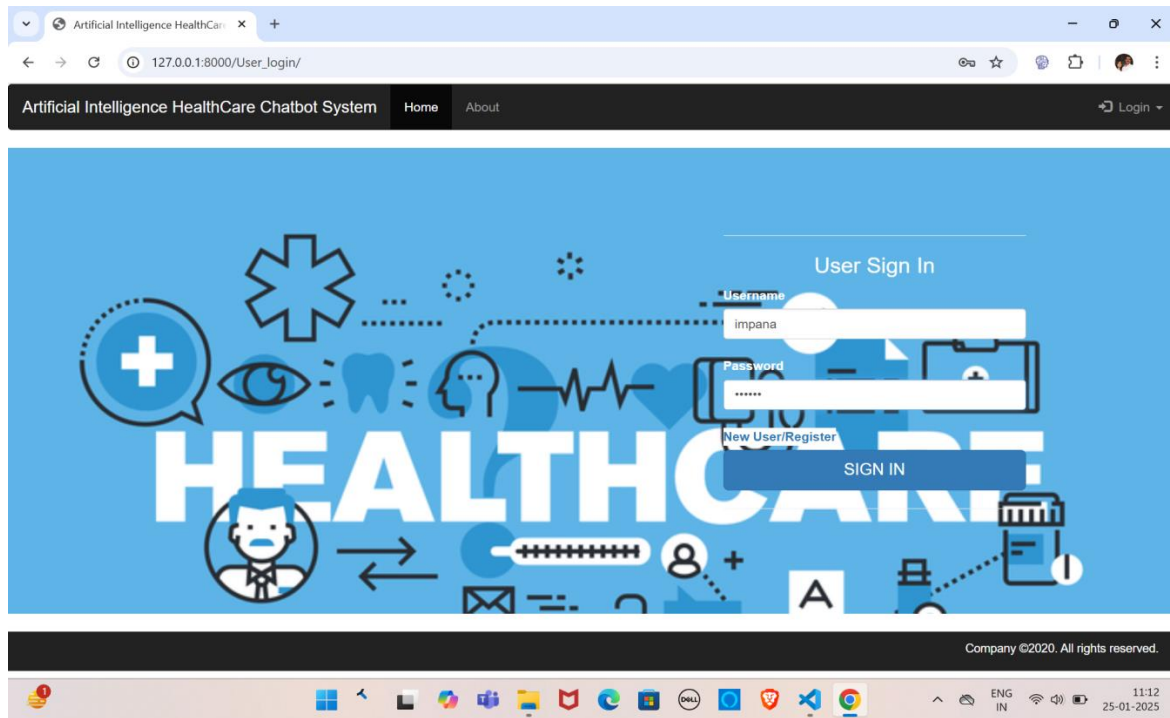


Fig. User login

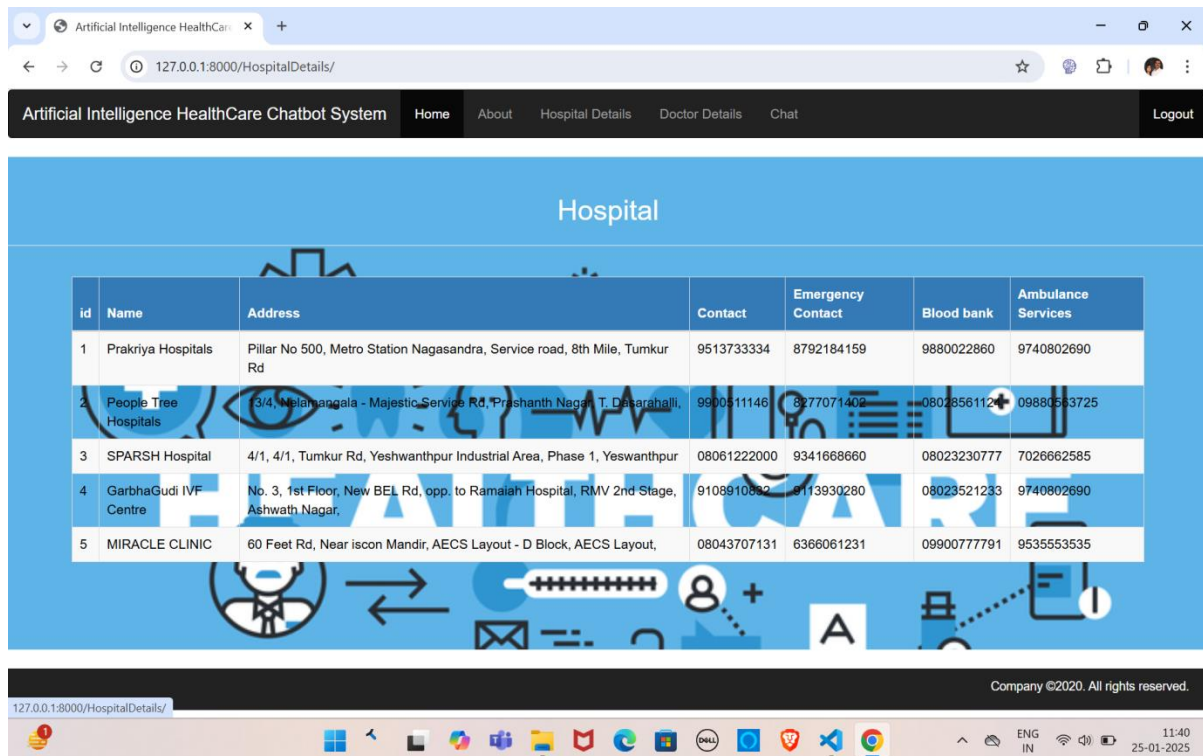
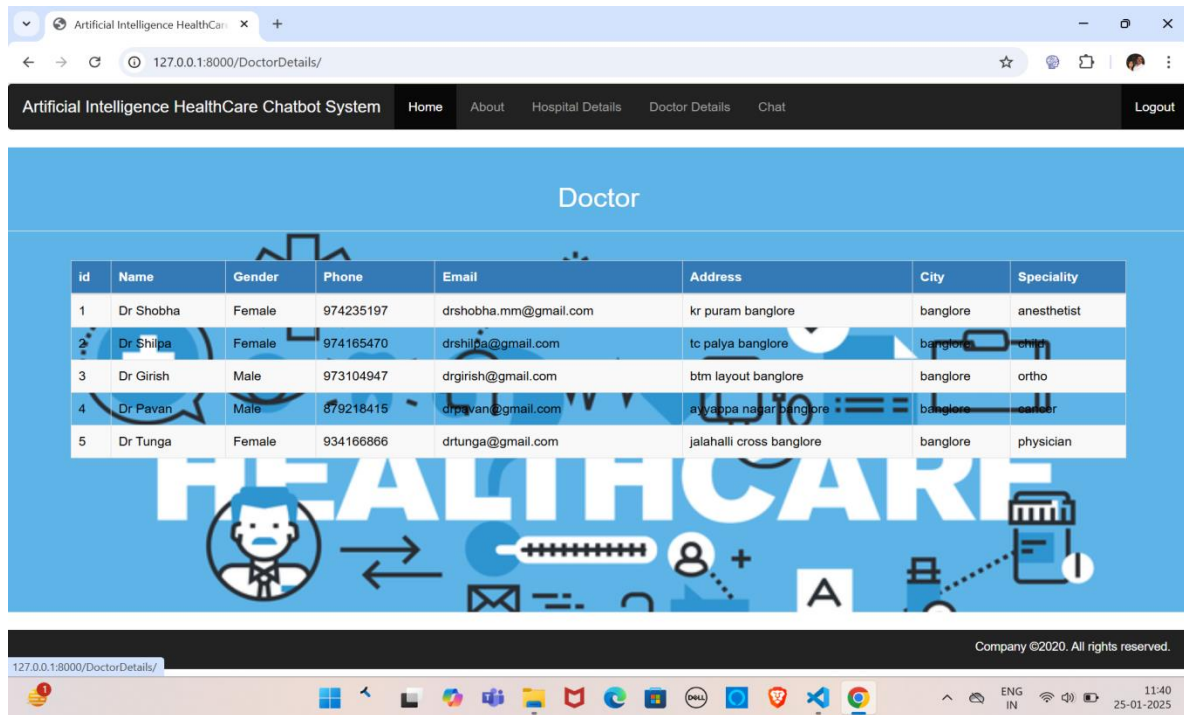
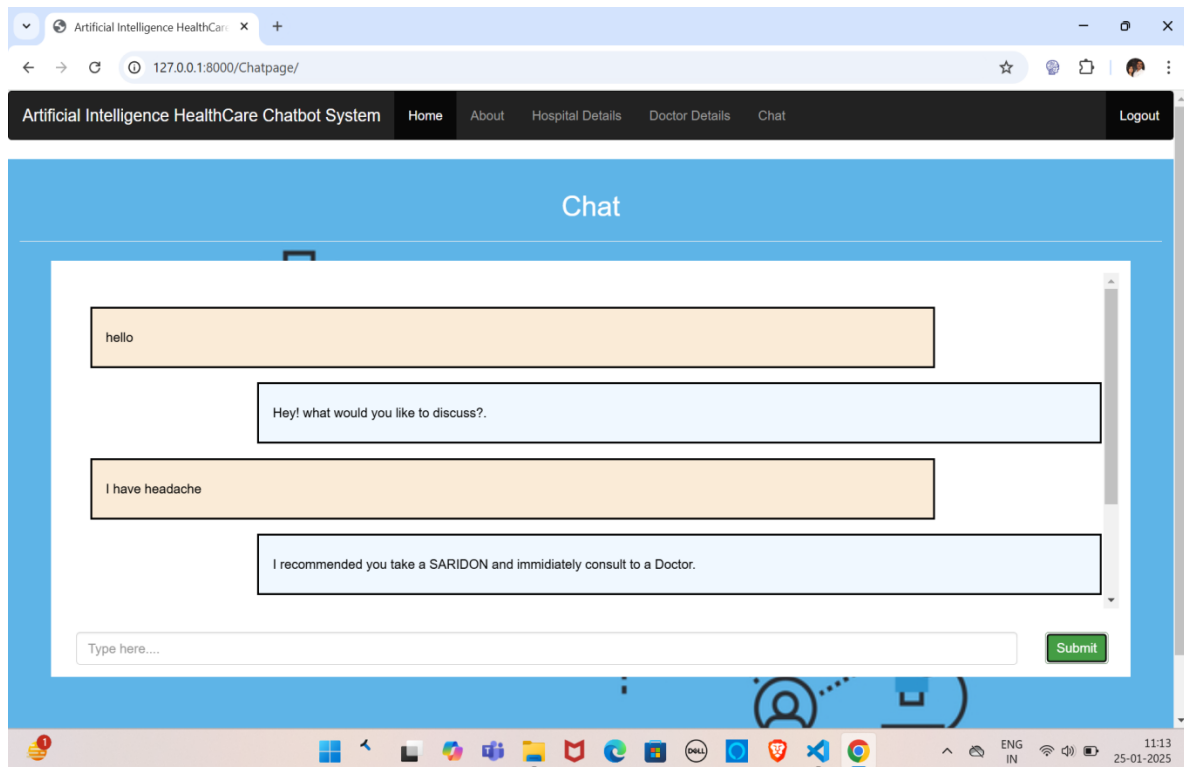


Fig. Hospital details



id	Name	Gender	Phone	Email	Address	City	Speciality
1	Dr Shobha	Female	974235197	drshobha.mm@gmail.com	kr puram banglore	banglore	anesthetist
2	Dr Shilpa	Female	974165470	drshilpa@gmail.com	tc palya banglore	banglore	chiro
3	Dr Girish	Male	973104947	drgirish@gmail.com	btm layout banglore	banglore	ortho
4	Dr Pavan	Male	879218415	drpavan@gmail.com	ayyappa near banglore	banglore	cardiologist
5	Dr Tunga	Female	934166866	drtunga@gmail.com	jalahalli cross banglore	banglore	physician

Fig. Doctor details



Chat

hello

Hey! what would you like to discuss?.

I have headache

I recommended you take a SARIDON and immediately consult to a Doctor.

Type here.... Submit

Fig. Chatbot

CHAPTER 8

CONCLUSION & FUTURE SCOPE

8.1. CONCLUSION:

This was our project of System Design about “**Artificial Intelligence HealthCare Chatbot System**” developed as web application based on Python programming language. The Development of this system takes a lot of efforts from us. We think this system gave a lot of satisfaction to all of us. Though every task is never said to be perfect in this development field even more improvement may be possible in this application. We learned so many things and gained a lot of knowledge about development field. We hope this will prove fruitful to us.

The Artificial Intelligence Healthcare Chatbot System demonstrates how AI can enhance healthcare delivery by improving accessibility, efficiency, and accuracy. The system bridges gaps in traditional healthcare by providing instant, reliable support to users.

8.2. FUTURE SCOPE:

- Integrating advanced diagnostic tools for complex medical queries.
- Expanding language support to serve a global audience.
- Developing mobile applications for increased accessibility.

REFERENCES

► Websites

- ✓ en.wikipedia.org
- ✓ <https://ieeexplore.ieee.org/abstract/document/7975195/>
- ✓ <https://ieeexplore.ieee.org/document/7095895/>