**Disease Diagnosis using Chatbot**

**INTRODUCTION**

A prosperous society is when its entire people are healthy. It is important to maintain the health if one wishes to be happy. Only a healthy body can have a healthy mind and it has a positive impact on the performance of people. Nowadays, people are less aware of their health. In their busy life, they forget to take suitable measures to maintain their health and are less aware of their health status. In the latest news by TOI [1], we can see that people give no importance to their health and find it time consuming to undergo check-ups at hospitals. The busy-scheduled life has got no place for health. Most people comprising the working section of the society claim that their hectic schedule gives them no time for periodic medical check-ups and that they disregard any uneasiness shown by their body until it is too severe. In this proposed system, a medical chatbot is built to be a conversational agent that motivates users to discuss about their health issues and based on the symptoms provided by them; chatbot returns the diagnosis [2]. This chatbot system will be able to identify symptoms from user interaction. Using these extracted symptoms, chatbot predicts the disease and recommends treatment. The machine learning algorithm [3] employed here is K-nearest neighbor algorithm (KNN). This clearly shows that a medical chatbot can somewhat accurately diagnose patients with simple symptom analysis and a conversational approach done with the help of natural language processing. Medical chatbot has a high impact on the health culture of the state. It has improved reliability and is less prone to human errors. Today‟s people are more likely addicted to internet but they are not concerned about their personal health. They avoid hospital treatment for small issues which may become a major disease in future. This proposed idea solves this problem. This idea focuses on creating a chatbot which is free of cost and available throughout the day. The facts that the chatbot is free and can be accessed wherever the user is, be it their working environment, prompt the user to have it and use it. It saves the overhead involved in consulting specialized doctors. Executing the proposed system can convey more awareness among people regarding their health status and a need to take measures to remain healthy. With the new proposed system, there will be reduction in the number of people disregarding their health because of the tedious process of hospital appointments. People can interact with the chatbot just like they do with another human and can continue with their other works. It ensures that there will be no disruption with their working and is user friendly. This provides a way to help people aware of their health by using the chatbot and thereby helps people with their health, thus playing a bigger role in healthcare.

**1.1 Objective of the project:**

Hospitals are the most widely used means by which a sick person gets medical check-ups, disease diagnosis and treatment recommendation. This has been a practice by almost all the people over the world. People consider it as the most reliable means to check their health status. The proposed system is to create an alternative to this conventional method of visiting a hospital and making an appointment with a doctor to get diagnosis. This research intends to apply the concepts of natural language processing and machine learning to create a chatbot application. People can interact with the chatbot just like they do with another human and through a series of queries, chatbot will identify the symptoms of the user and thereby, predicts the disease and recommends treatment. This system can be of great use to people in conducting daily check-ups, makes people aware of their health status and encourages people to make proper measures to remain healthy. According to this research, such a system is not widely used and people are less aware of it. Executing this proposed framework can help people avoid the time-consuming method of visiting hospitals by using this free of cost application, wherever they are.

**2. LITERATURE SURVEY:**

**Survey on Chatbot Design Techniques in Speech Conversation Systems**

Human-Computer Speech is gaining momentum as a technique of computer interaction. There has been a recent upsurge in speech based search engines and assistants such as Siri, Google Chrome and Cortana. Natural Language Processing (NLP) techniques such as NLTK for Python can be applied to analyse speech, and intelligent responses can be found by designing an engine to provide appropriate human like responses. This type of programme is called a Chatbot, which is the focus of this study. This paper presents a survey on the techniques used to design Chatbots and a comparison is made between different design techniques from nine carefully selected papers according to the main methods adopted. These papers are representative of the significant improvements in Chatbots in the last decade. The paper discusses the similarities and differences in the techniques and examines in particular the Loebner prizewinning Chatbots.

**Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents.**

The aim of this paper is to explore commercial applications of chatbots, as well as to propose several measurement metrics to evaluate performance, usability and overall quality of an embodied conversational agent. On the basis of these metrics we examine existing Polish-speaking commercial chatbots that a) work in the B2C sector, b) reach the widest possible range of users, and c) are presumably the most advanced commercial deployments of their creators. We analyse various aspects of functioning of each embodied conversational agent: visual look, form of implementation on the website, speech synthesis unit, built-in knowledge base (with general and specialized information), presentation of knowledge and additional functionalities, conversational abilities and context sensitiveness, personality traits, personalization options, emergency responses in unexpected situations, possibility of rating chatbot and the website by the user. Our study reveals the current condition of Polish market of commercial virtual assistants and emphasizes the importance of a multidimensional evaluation of any commercial chatbot deployment.

**A Novel Approach for Medical Assistance Using Trained Chatbot**

There are lot of treatments that are available for various diseases. No human can possibly know about all the medicines and the diseases. So, the problem is that there isn't any place where anyone can have the details of the diseases or the medicines. What if there is a place where you can find your health problem just by entering symptoms or just scanning an ECG or you can check whether the prescribed medicine is supposed to be used the way you are told to. Then it will help us to deduce the problem and to verify the solution. The proposed idea is to create a system with artificial intelligence that can meet the requirements. The AI can predict the diseases based on the symptoms and give the list of available treatments. The System can also give the composition of the medicines and their prescribed uses. It helps them to take the correct treatment. Hence the people can have an idea about their health and can have the right protection.

**"A Self-Diagnosis Medical Chatbot Using Artificial Intelligence",**

Medical care is very important for a healthy life. However, it is very difficult to seek medical attention if you have a health problem. The recommended notion is to develop a medical chatbot that can adopt AI to analyze the ailment and produce necessary information concerning the conditions were discussing with a doctor. Medical chatbots were built to reduce medical costs and improve access to medical knowledge. Some chatbots serve as medical manuals to help patients become aware of their illness and improve their health. Users can assuredly benefit from chatbots if they can diagnose several kinds of illness and render the required data. Text diagnosis bot enables sufferers to join in analyses of their medicinal matters and present a personalized analysis report with reference to the symptoms.

**“Mobile based healthcare management using artificial intelligence”**

In this growing age of technology it is necessary to have a proper health care management system which should be cent percent accurate but also should be portable so that every person carry with it as personalized health care system. The health care management system which will consist of mobile based Heart Rate Measurement so that the data can be transferred and diagnosis based on heart rate can be provided quickly with a click of button. The system will consist of video conferencing to connect remotely with the Doctor. The Doc-Bot which was developed earlier is now being transferred to mobile platform and will be further advanced for diagnosis of common diseases. The system will also consist of Online Blood Bank which will provide up-to-date details about availability of blood in different hospitals.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

In existing system, A rule-based medical chatbot uses predefined rules and a set of programmed responses to interact with users. These chatbots do not leverage machine learning but instead rely on a decision tree or a flowchart approach to provide information based on user inputs

**Disadvantages:**

1. Less Accuracy
2. More time taking process

**3.2 PROPOSED SYSTEM**

In this project we are developing Chatbot which can analyse input symptoms and then predict disease and then display diet and doctor appointment booking. It’s not real time application to make booking with the doctor but we will display predicted disease, diet information along with doctor and hospital details.

To identify disease we need to train Chatbot with machine learning so it can take symptoms as input and then predict disease and to train Chatbot we have use CNN algorithm and this algorithm get trained on dataset.

**Advantages**

1. High Accuracy
2. Less Time Taken

To implement this project we have designed following modules

1. Register: using this module users can sign up with the application along with email ID and contact no so email can be sent along with predicted disease and diet details
2. User: using this module user can login to application
3. Chatbot: using this module user can enter symptoms and then Chatbot will predict disease, diet, doctor details and then display output as well as send email to registered email ID
4. Lifestyle & Disease Information: using this module use can select disease name and then system will suggest foods to take, avoid along with doctor details.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* Business requirements describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**SYSTEM REQUIREMENT:**

**HARDWARE REQUIREMENTS:**

# Processor - Intel i3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500 GB

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows10(min)
* Programming Language - Python

**4. SYSTEM DESIGN**

**UML Diagram:**

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioural Model View**

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

**Class Diagram:**

The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake.



**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

****

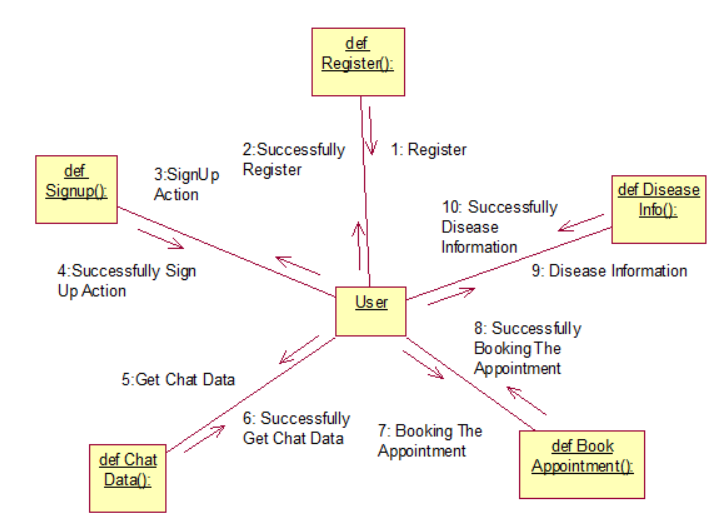
**Sequence diagram:**

A sequence diagram 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. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Collaboration diagram:**

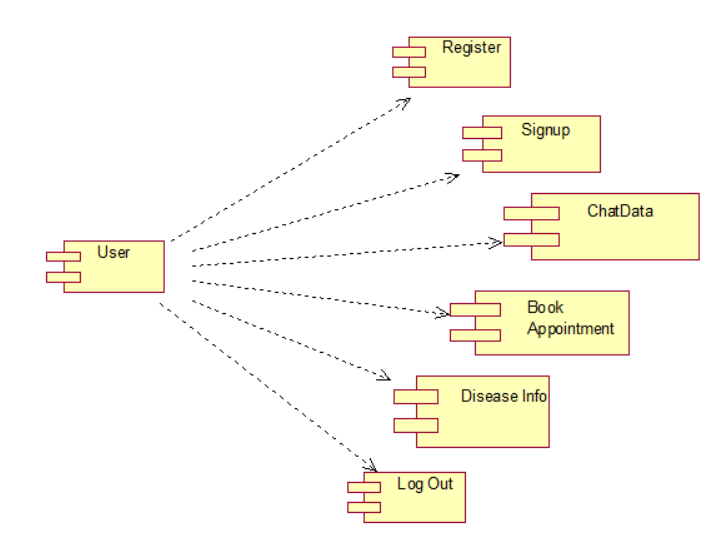
A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**Component Diagram:**

In the Unified Modelling 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.

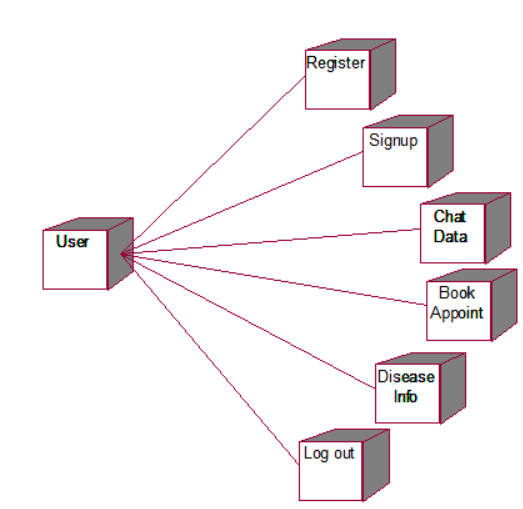
Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

****

**Deployment Diagram:**

A **deployment diagram** in the Unified Modelling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

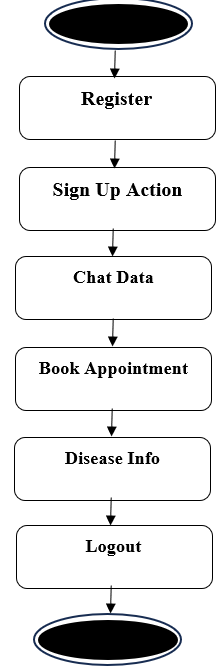
The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another

activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

****

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

1. Register 2. Successfully Register

3. Sign Up 4. Successfully Sign Up

5. Get Chat Data 6. Successfully Get Chat Data

7. Book Appointment 8. Successfully Book Appointment

9. Disease Info 10. Successfully Disease Info

**5. IMPLEMETATION**

**5.1 PYTHON**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**5.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**5.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

**5.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI applicatioMakeske the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**5.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**5.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics

**5.2 Sample Code:**

**from django.shortcuts import render**

**from django.template import RequestContext**

**from django.contrib import messages**

**import pymysql**

**from django.http import HttpResponse**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**import re**

**import numpy as np**

**from django.core.files.storage import FileSystemStorage**

**from django.views.decorators.csrf import csrf\_exempt**

**import os**

**import pandas as pd**

**import smtplib**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**import string**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.metrics import confusion\_matrix**

**import seaborn as sns**

**from sklearn.metrics import precision\_score**

**from sklearn.metrics import recall\_score**

**from sklearn.metrics import f1\_score**

**from sklearn.metrics import accuracy\_score**

**from string import punctuation**

**from nltk.corpus import stopwords**

**import nltk**

**from nltk.stem import WordNetLemmatizer**

**from keras.utils.np\_utils import to\_categorical**

**from keras.layers import MaxPooling2D**

**from keras.layers import Dense, Dropout, Activation, Flatten**

**from keras.layers import Convolution2D**

**from keras.models import Sequential**

**from keras.models import model\_from\_json**

**import pickle**

**from datetime import date**

**import smtplib**

**global filename**

**global word\_vector**

**global uname, email**

**stop\_words = set(stopwords.words('english'))**

**lemmatizer = WordNetLemmatizer()**

**def cleanData(doc):**

**tokens = doc.split()**

**table = str.maketrans('', '', punctuation)**

**tokens = [w.translate(table) for w in tokens]**

**tokens = [word for word in tokens if word.isalpha()]**

**tokens = [w for w in tokens if not w in stop\_words]**

**tokens = [word for word in tokens if len(word) > 1]**

**tokens = [lemmatizer.lemmatize(token) for token in tokens]**

**tokens = ' '.join(tokens)**

**return tokens**

**dataset = pd.read\_csv('Dataset/dataset.csv', encoding ="ISO-8859-1")**

**labels = dataset['Source'].unique().tolist()**

**symptoms = dataset.Target**

**diseases = dataset.Source**

**Y = []**

**for i in range(len(diseases)):**

**index = labels.index(diseases[i])**

**Y.append(index)**

**X = []**

**for i in range(len(symptoms)):**

**arr = symptoms[i]**

**arr = arr.strip().lower()**

**arr = arr.replace("\_", " ")**

**X.append(cleanData(arr))**

**vectorizer = TfidfVectorizer(use\_idf=True, smooth\_idf=False, norm=None, decode\_error='replace')**

**tfidf = vectorizer.fit\_transform(X).toarray()**

**X = pd.DataFrame(tfidf, columns=vectorizer.get\_feature\_names())**

**print(X.head())**

**print(X.shape)**

**Y = np.asarray(Y)**

**print(Y)**

**X = X.values**

**indices = np.arange(X.shape[0])**

**np.random.shuffle(indices)**

**X = X[indices]**

**Y = Y[indices]**

**Y = to\_categorical(Y)**

**X = X.reshape(X.shape[0],X.shape[1],1,1)**

**print(X.shape)**

**if os.path.exists('model/model.json'):**

**with open('model/model.json', "r") as json\_file:**

**loaded\_model\_json = json\_file.read()**

**classifier = model\_from\_json(loaded\_model\_json)**

**json\_file.close()**

**classifier.load\_weights("model/model\_weights.h5")**

**classifier.\_make\_predict\_function()**

**else:**

**classifier = Sequential()**

**classifier.add(Convolution2D(32, 1, 1, input\_shape = (X.shape[1], X.shape[2], X.shape[3]), activation = 'relu'))**

**classifier.add(MaxPooling2D(pool\_size = (1, 1)))**

**classifier.add(Convolution2D(32, 1, 1, activation = 'relu'))**

**classifier.add(MaxPooling2D(pool\_size = (1, 1)))**

**classifier.add(Flatten())**

**classifier.add(Dense(output\_dim = 256, activation = 'relu'))**

**classifier.add(Dense(output\_dim = Y.shape[1], activation = 'softmax'))**

**print(classifier.summary())**

**classifier.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])**

**hist = classifier.fit(X, Y, batch\_size=8, epochs=10, shuffle=True, verbose=2)**

**classifier.save\_weights('model/model\_weights.h5')**

**model\_json = classifier.to\_json()**

**with open("model/model.json", "w") as json\_file:**

**json\_file.write(model\_json)**

**f = open('model/history.pckl', 'wb')**

**pickle.dump(hist.history, f)**

**f.close()**

**def index(request):**

**if request.method == 'GET':**

**with open('model/model.json', "r") as json\_file:**

**loaded\_model\_json = json\_file.read()**

**classifier = model\_from\_json(loaded\_model\_json)**

**json\_file.close()**

**classifier.load\_weights("model/model\_weights.h5")**

**classifier.\_make\_predict\_function()**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)**

**predict = classifier.predict(X\_test)**

**predict = np.argmax(predict, axis=1)**

**testY = np.argmax(y\_test, axis=1)**

**for i in range(0,50):**

**predict[i] = 0**

**p = precision\_score(testY, predict,average='macro') \* 100**

**r = recall\_score(testY, predict,average='macro') \* 100**

**f = f1\_score(testY, predict,average='macro') \* 100**

**a = accuracy\_score(testY,predict)\*100**

**output='<tr><td><font size="" color="white">CNN Deep Learning<td><font size="" color="white">'+str(a)+'</td>'**

**output+='<td><font size="" color="white">'+str(p)+'</td>'**

**output+='<td><font size="" color="white">'+str(r)+'</td>'**

**output+='<td><font size="" color="white">'+str(f)+'</td></tr>'**

**context= {'data':output}**

**return render(request, 'index.html', context)**

**def User(request):**

**if request.method == 'GET':**

**return render(request, 'User.html', {})**

**def BookAppointment(request):**

**if request.method == 'GET':**

**return render(request, 'BookAppointment.html', {})**

**def ChatBotPage(request):**

**if request.method == 'GET':**

**return render(request, 'UserScreen.html', {})**

**def Logout(request):**

**if request.method == 'GET':**

**return render(request, 'index.html', {})**

**def DiseaseInfoAction(request):**

**if request.method == 'POST':**

**name = request.POST.get('t1', False)**

**diet = ""**

**with open('diets/'+name+".txt", "r") as file:**

**lines = file.readlines()**

**for i in range(len(lines)):**

**diet += lines[i]+"\n"**

**file.close()**

**context= {'data': diet}**

**return render(request, 'Info.html', context)**

**def DiseaseInfo(request):**

**if request.method == 'GET':**

**output= ""**

**for root, dirs, directory in os.walk('diets'):**

**for j in range(len(directory)):**

**name = directory[j].split(".")**

**name = name[0]**

**output += '<option value="'+name+'">'+name+'</option>'**

**context= {'data1': output}**

**return render(request, 'DiseaseInfo.html', context)**

**def Register(request):**

**if request.method == 'GET':**

**output = ""**

**for i in range(0,200):**

**output += "<option value="+str(i)+">"+str(i)+"</option>"**

**context= {'data1': output}**

**return render(request, 'Register.html', context)**

**def getDiet(filepath):**

**diet = ""**

**if os.path.exists("diets/"+filepath+".txt"):**

**with open("diets/"+filepath+".txt", "r") as file:**

**lines = file.readlines()**

**for i in range(len(lines)):**

**diet += lines[i]+"\n"**

**file.close()**

**else:**

**with open("diets/others.txt", "r") as file:**

**lines = file.readlines()**

**for i in range(len(lines)):**

**diet += lines[i]+"\n"**

**file.close()**

**return diet**

**def sendMails(email, message, disease):**

**with smtplib.SMTP\_SSL('smtp.gmail.com', 465) as connection:**

**email\_address = 'kaleem202120@gmail.com'**

**email\_password = 'xyljzncebdxcubjq'**

**connection.login(email\_address, email\_password)**

**connection.sendmail(from\_addr="kaleem202120@gmail.com", to\_addrs=email, msg="Subject : Disease Predictd As "+disease+"\n\n"+message+" Booking Confirmed with above doctor")**

**def appointmentMail(email, subject, message):**

**with smtplib.SMTP\_SSL('smtp.gmail.com', 465) as connection:**

**email\_address = 'kaleem202120@gmail.com'**

**email\_password = 'xyljzncebdxcubjq'**

**connection.login(email\_address, email\_password)**

**connection.sendmail(from\_addr="kaleem202120@gmail.com", to\_addrs=email, msg="Subject : "+subject+"\n\n"+message)**

**def BookAppointmentAction(request):**

**if request.method == 'POST':**

**global uname**

**doctor = request.POST.get('t1', False)**

**appointment\_date = request.POST.get('t2', False)**

**arr = doctor.split("-")**

**name = arr[0]**

**speciality = arr[1]**

**today = date.today()**

**db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'DiseasePrediction',charset='utf8')**

**db\_cursor = db\_connection.cursor()**

**student\_sql\_query = "INSERT INTO appointment(patient,doctor\_name,doctor\_speciality,booking\_date,appointment\_date) VALUES('"+uname+"','"+name+"','"+speciality+"','"+str(today)+"','"+appointment\_date+"')"**

**db\_cursor.execute(student\_sql\_query)**

**db\_connection.commit()**

**if db\_cursor.rowcount == 1:**

**appointmentMail(uname, "Appointment Confirmed with Doctor "+name, "Your appointment is confirmed on "+appointment\_date)**

**status = 'Your appointment is confirmed on '+appointment\_date+"<br/>with Doctor : "+name+"<br/>Doctor Speciality : "+speciality**

**context= {'data': status}**

**return render(request, 'BookAppointment.html', context)**

**def ChatData(request):**

**if request.method == 'GET':**

**global email**

**question = request.GET.get('mytext', False)**

**question = question.strip("\n").strip()**

**with open('model/model.json', "r") as json\_file:**

**loaded\_model\_json = json\_file.read()**

**classifier = model\_from\_json(loaded\_model\_json)**

**json\_file.close()**

**classifier.load\_weights("model/model\_weights.h5")**

**classifier.\_make\_predict\_function()**

**temp = []**

**query = question**

**print(query)**

**arr = query**

**arr = arr.strip().lower()**

**arr = arr.replace("\_", " ")**

**testData = vectorizer.transform([cleanData(arr)]).toarray()**

**print(testData.shape)**

**temp = testData.reshape(testData.shape[0],testData.shape[1],1,1)**

**predict = classifier.predict(temp)**

**predict = np.argmax(predict)**

**output = labels[predict]**

**diet = getDiet(output)**

**print(question+" "+output)**

**sendMails(email, diet, output)**

**savePrediction(output,question)**

**return HttpResponse("Chatbot: Disease Predicted as "+output+"\n\n"+diet, content\_type="text/plain")**

**def UserLogin(request):**

**if request.method == 'POST':**

**global uname, email**

**username = request.POST.get('username', False)**

**password = request.POST.get('password', False)**

**index = 0**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'DiseasePrediction',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select \* FROM register")**

**rows = cur.fetchall()**

**for row in rows:**

**if row[6] == username and password == row[7]:**

**uname = username**

**email = row[6]**

**index = 1**

**break**

**if index == 1:**

**context= {'data':'welcome '+username}**

**return render(request, 'UserScreen.html', context)**

**else:**

**context= {'data':'login failed'}**

**return render(request, 'User.html', context)**

**def savePrediction(output, symptoms):**

**global uname, email**

**today = date.today()**

**db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'DiseasePrediction',charset='utf8')**

**db\_cursor = db\_connection.cursor()**

**student\_sql\_query = "INSERT INTO predictionresult(username,symptoms,disease\_prediction,prediction\_date) VALUES('"+uname+"','"+symptoms+"','"+output+"','"+str(today)+"')"**

**db\_cursor.execute(student\_sql\_query)**

**db\_connection.commit()**

**#sendEmail(output, symptoms)**

**def Signup(request):**

**if request.method == 'POST':**

**name = request.POST.get('tf1', False)**

**age = request.POST.get('tf2', False)**

**gender = request.POST.get('tf3', False)**

**height = request.POST.get('tf4', False)**

**weight = request.POST.get('tf5', False)**

**disease = request.POST.get('tf6', False)**

**email = request.POST.get('tf7', False)**

**password = request.POST.get('tf8', False)**

**contact = request.POST.get('tf9', False)**

**print(str(name)+" "+str(age)+" "+str(gender)+" "+str(height)+" "+str(weight)+" "+str(disease)+" "+str(email)+" "+str(password)+" "+str(contact))**

**status = "none"**

**con = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'DiseasePrediction',charset='utf8')**

**with con:**

**cur = con.cursor()**

**cur.execute("select email FROM register where email='"+email+"'")**

**rows = cur.fetchall()**

**for row in rows:**

**status = "Email ID Already Exists"**

**break**

**if status == "none":**

**db\_connection = pymysql.connect(host='127.0.0.1',port = 3306,user = 'root', password = 'root', database = 'DiseasePrediction',charset='utf8')**

**db\_cursor = db\_connection.cursor()**

**student\_sql\_query = "INSERT INTO register(name,age,gender,height,weight,disease,email,password,contact) VALUES('"+name+"','"+age+"','"+gender+"','"+height+"','"+weight+"','"+disease+"','"+email+"','"+password+"','"+contact+"')"**

**db\_cursor.execute(student\_sql\_query)**

**db\_connection.commit()**

**print(db\_cursor.rowcount, "Record Inserted")**

**if db\_cursor.rowcount == 1:**

**status = 'Signup Process Completed'**

**context= {'data': status}**

**return render(request, 'Register.html', context)**

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## **Implementation**

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## **Testing**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### **System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

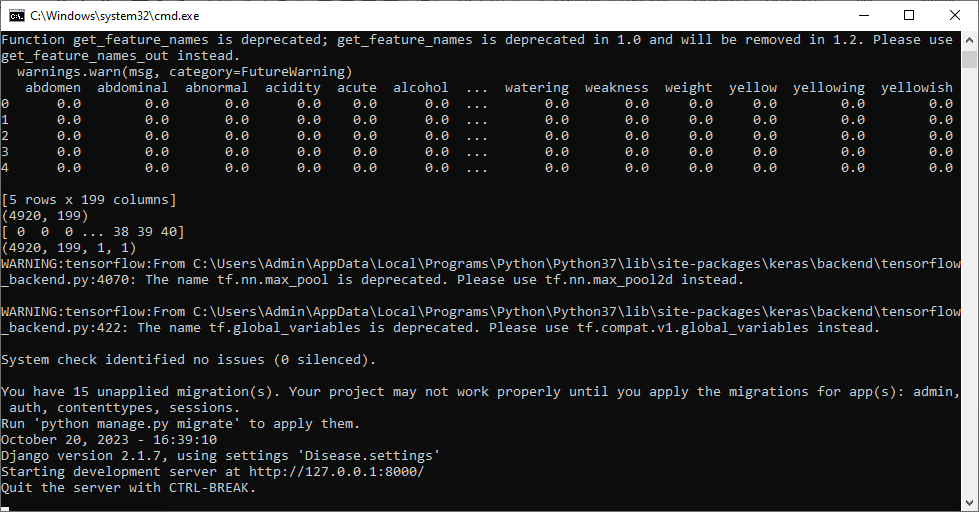
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation

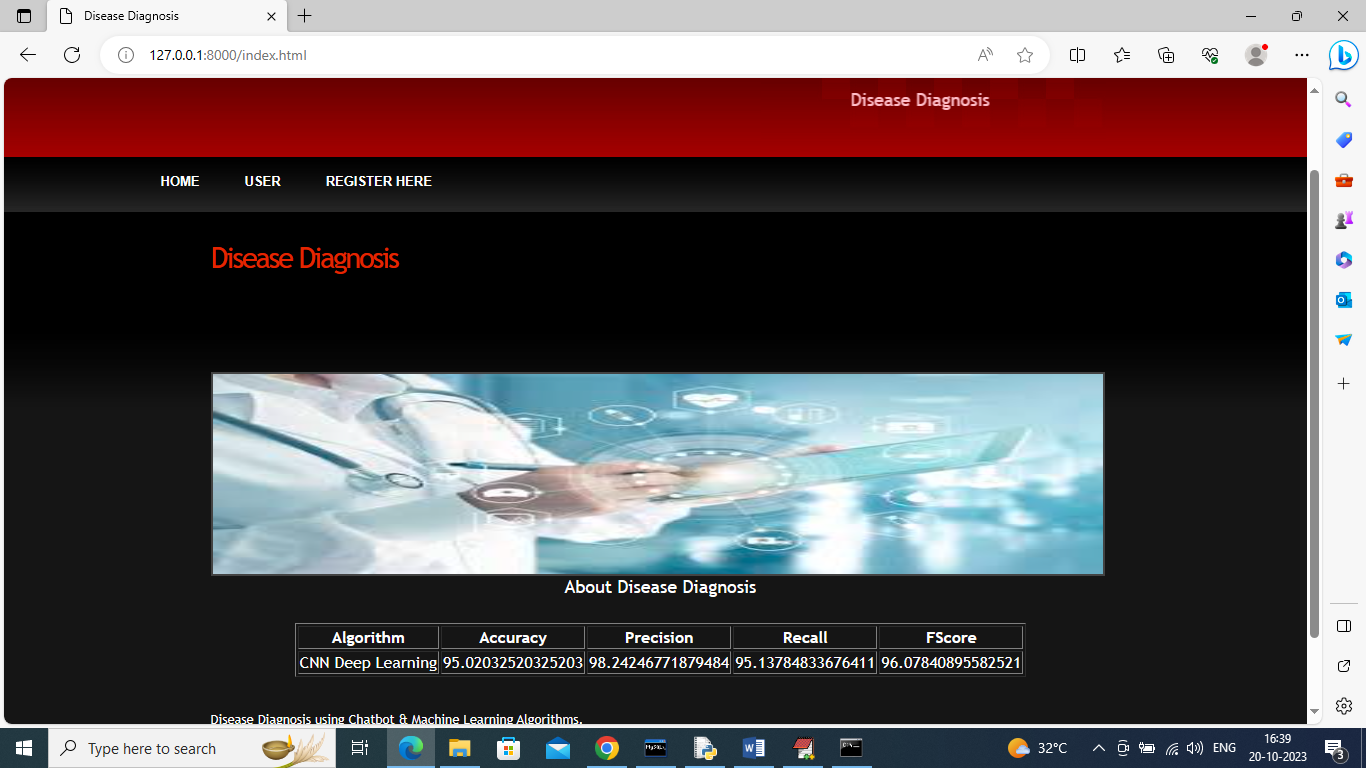
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Register | Test whether Register or not Register into the system | If the Register may not Registered | We cannot do further operations | If Register in we will do further operations | High | High |
| 02 | Sign Up | Test whether Sign Up Successfully or not | If the Sign Up Action may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 03 | Chat Data | Test whether Chat Data Successfully or not | If the  Chat Data may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 04 | Book Appointment | Test whether  Book Appointment Successfully or not | If the  Book Appointment Successfully | We cannot do further operations | we will do further  operations | High | High |
| 05 | Disease Info | Test whether  Disease Info Successfully or not | If the  Disease Info may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |

**7.SCREENSHOTS**:

To run project copy content from DB.txt file and then paste in MYSQL console to create database and then double click on ‘run.bat’ file to start python server and get below page

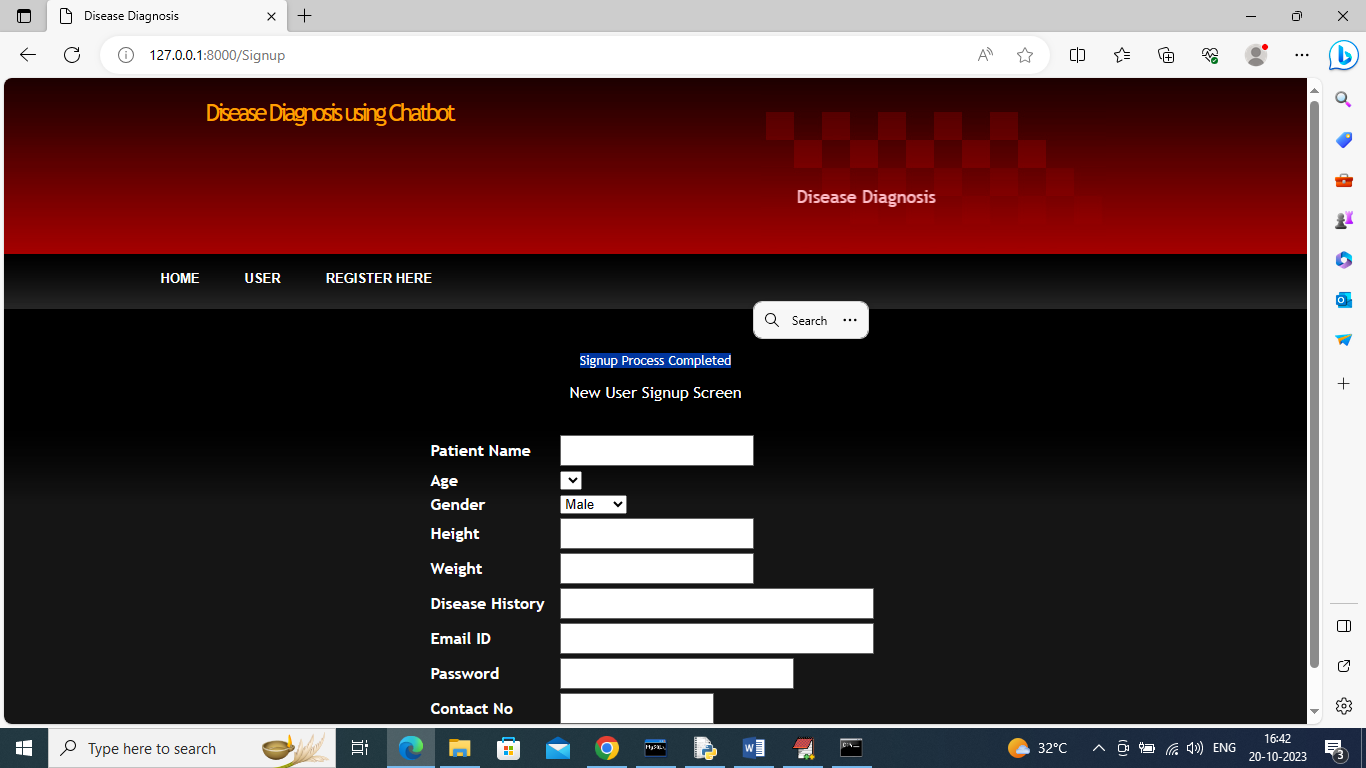


In above screen python server started and now open browser and entre URL as <http://127.0.0.1:8000/index.html> and then press enter key to get below page



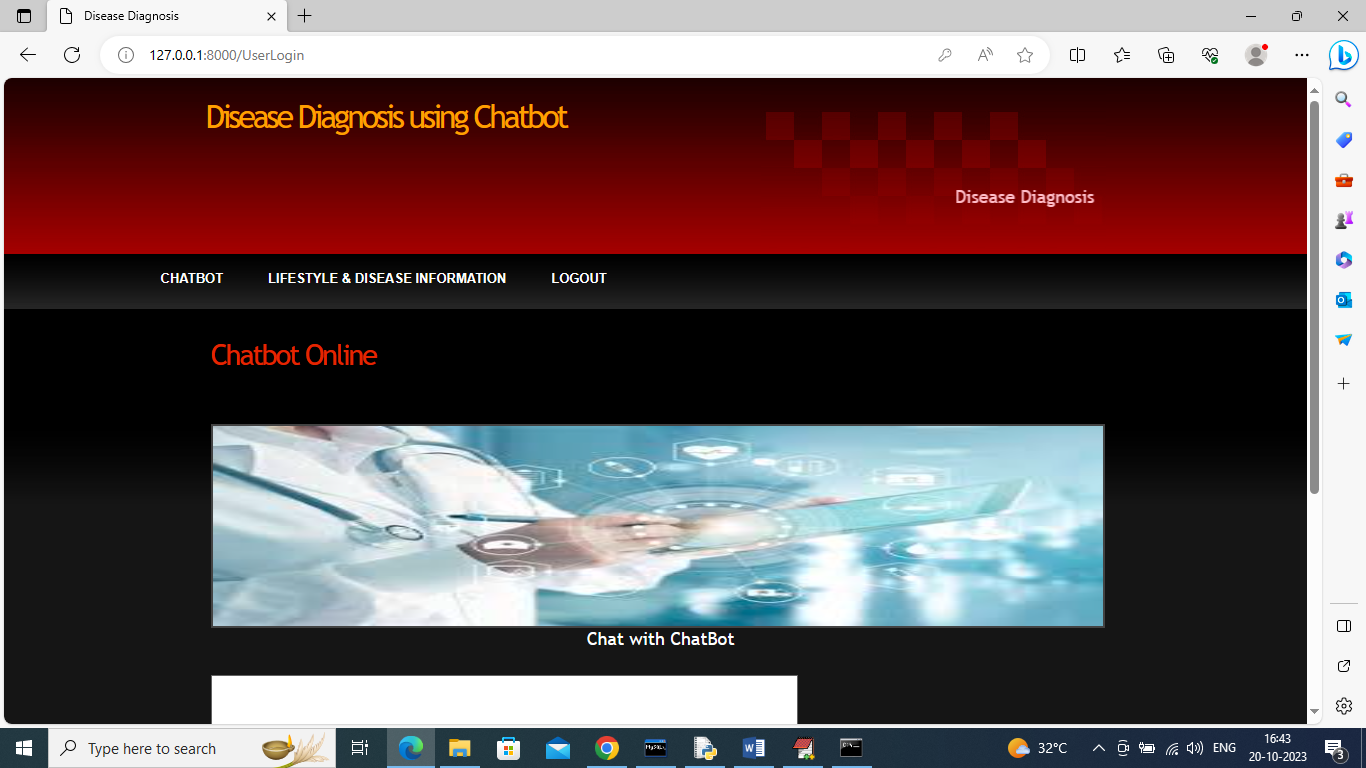
In above screen we can see application home page and then in table we can see CNN algorithm disease prediction accuracy and now click on ‘Register Here’ link to sign up with the application

In above step user is entering signup detail and give valid MAIL ID so you can receive mails and submit button to get below screen

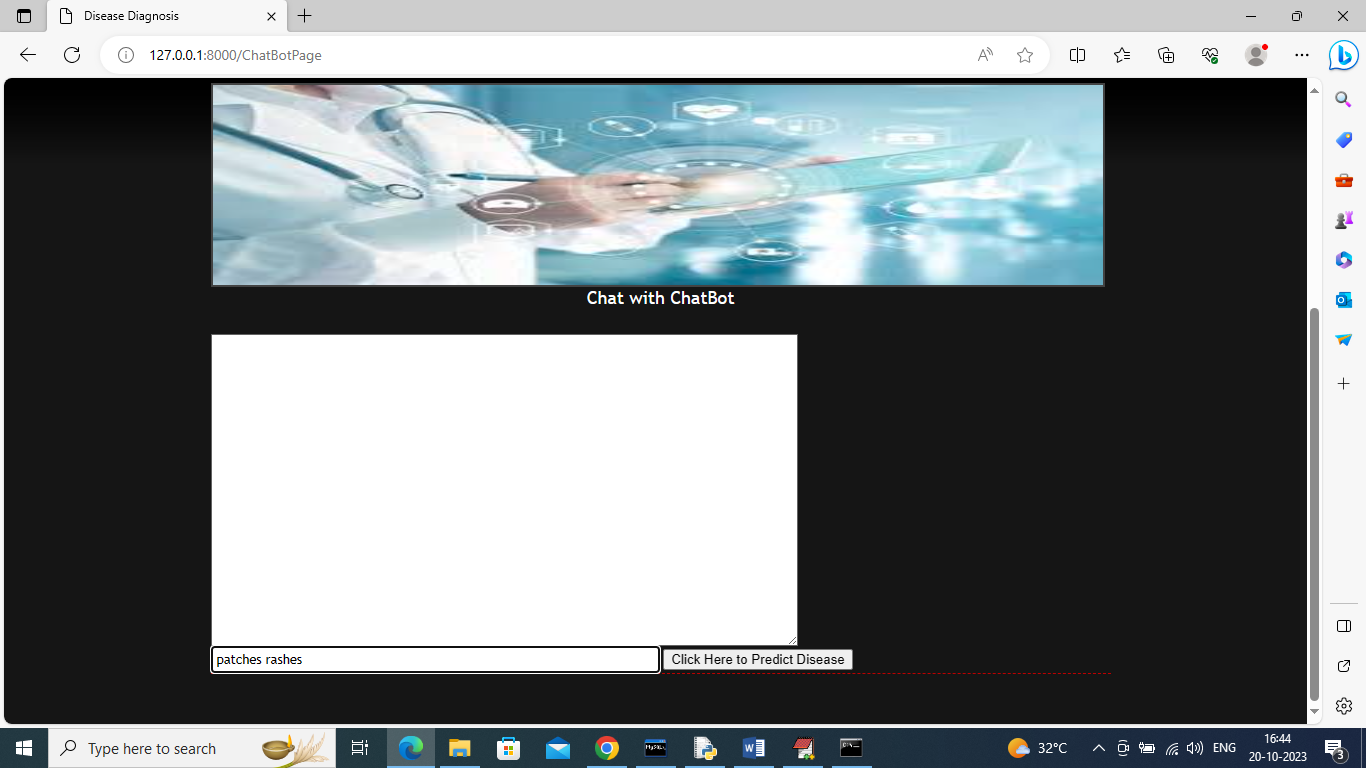


In above screen in blue colour text we can see sign up completed and now click on ‘User’ link to login as user with registered details

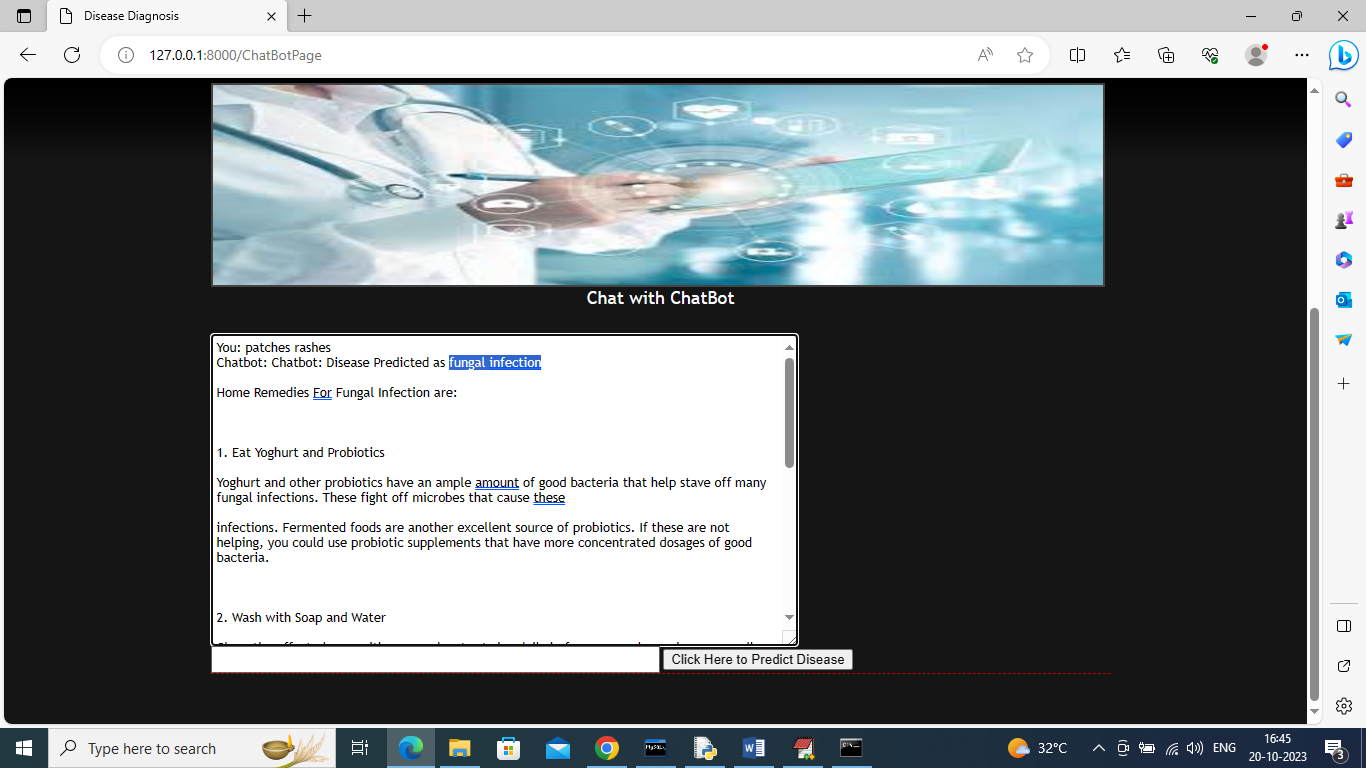
after login will get below page



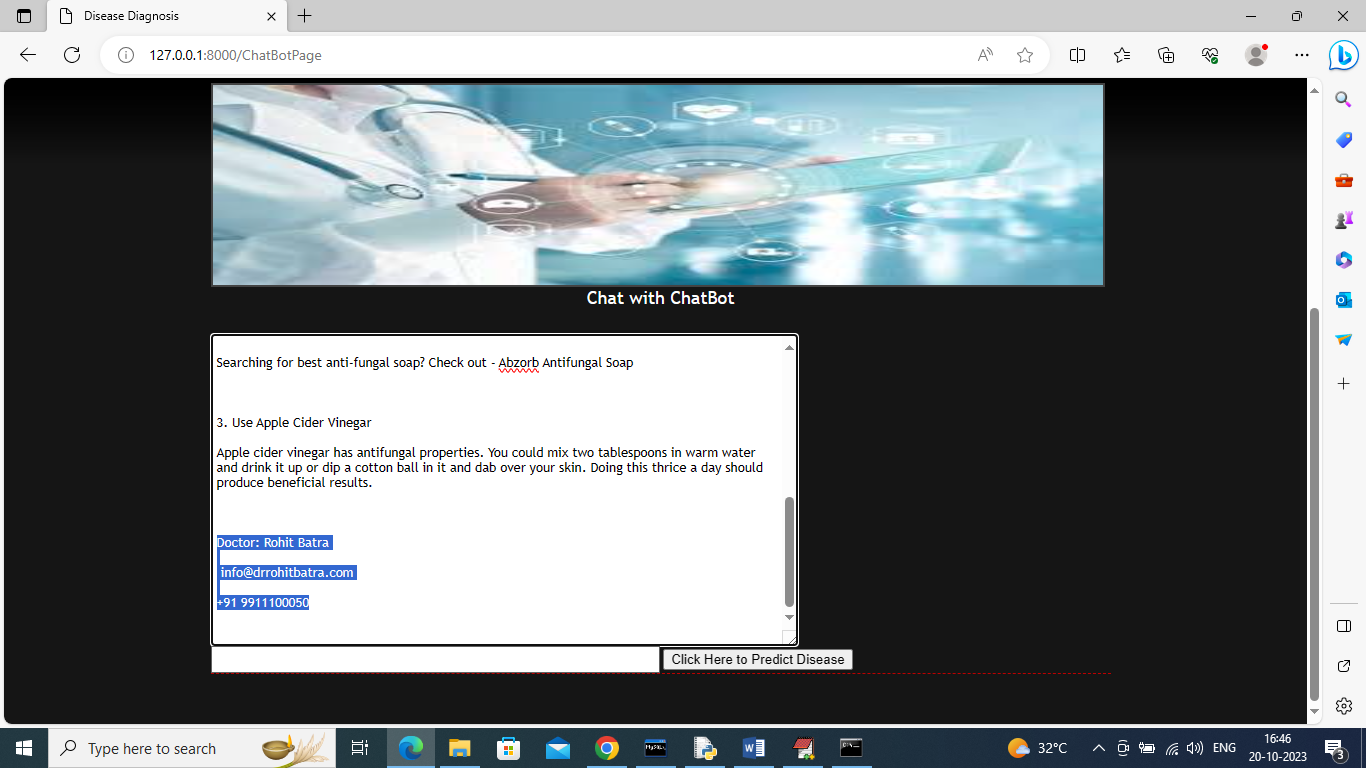
In above screen click on ‘Chatbot’ link to get below page



In above Chatbot page just type some symptoms and in above page I gave symptoms as ‘patches rashes’ and then press button to get reply from Chatbot like below screen

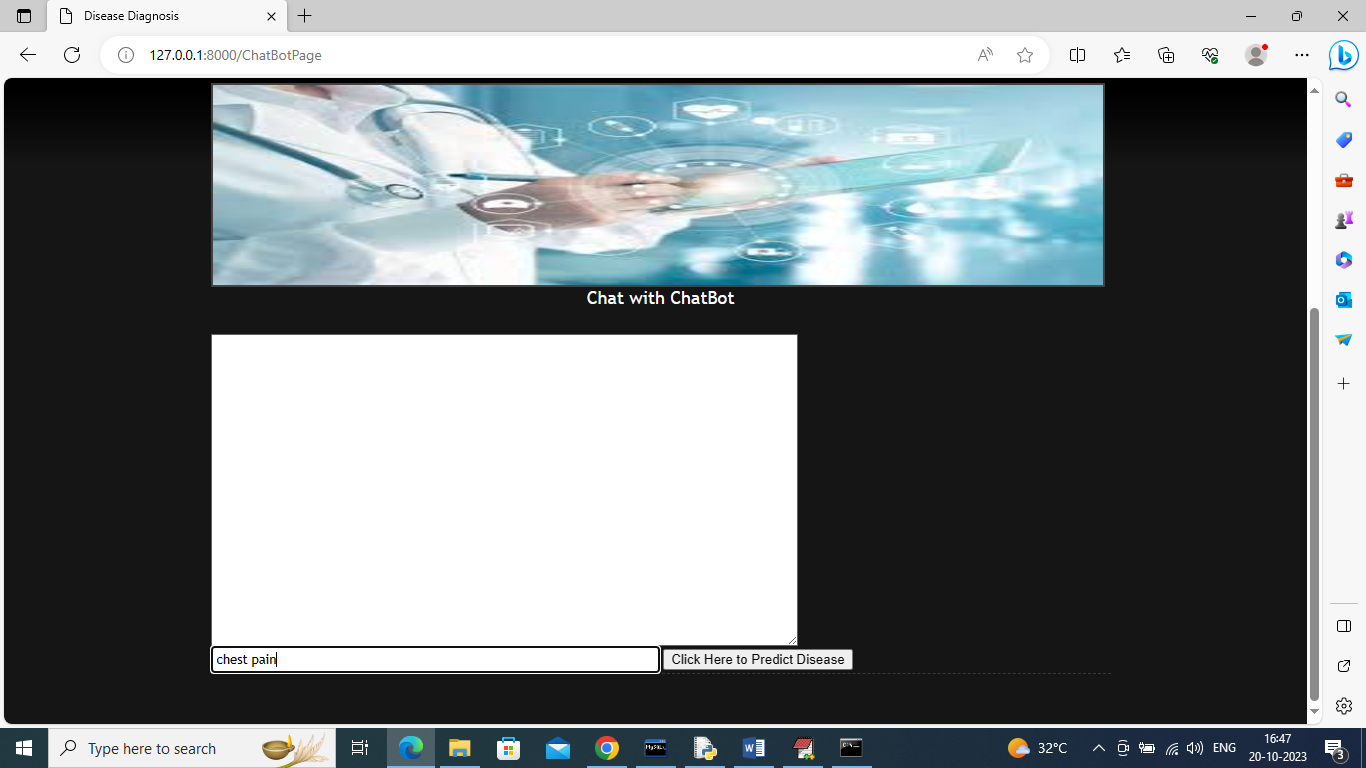


In above screen in blue colour text disease predicted as ‘Fungal Infection’ and then in below lines we can see home remedies along with diet details and scroll down above page to view complete details

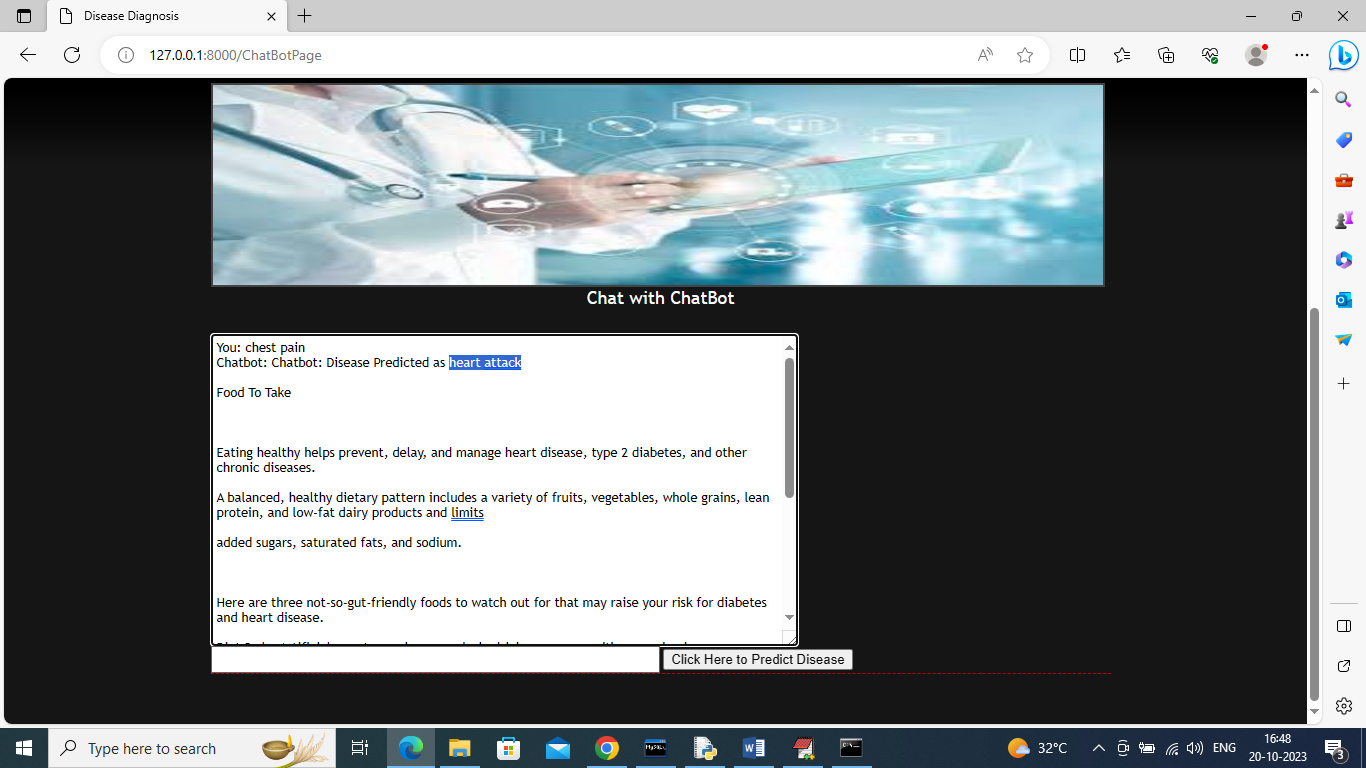


In above screen we can see doctor details and then same information will be sent to mail also sent

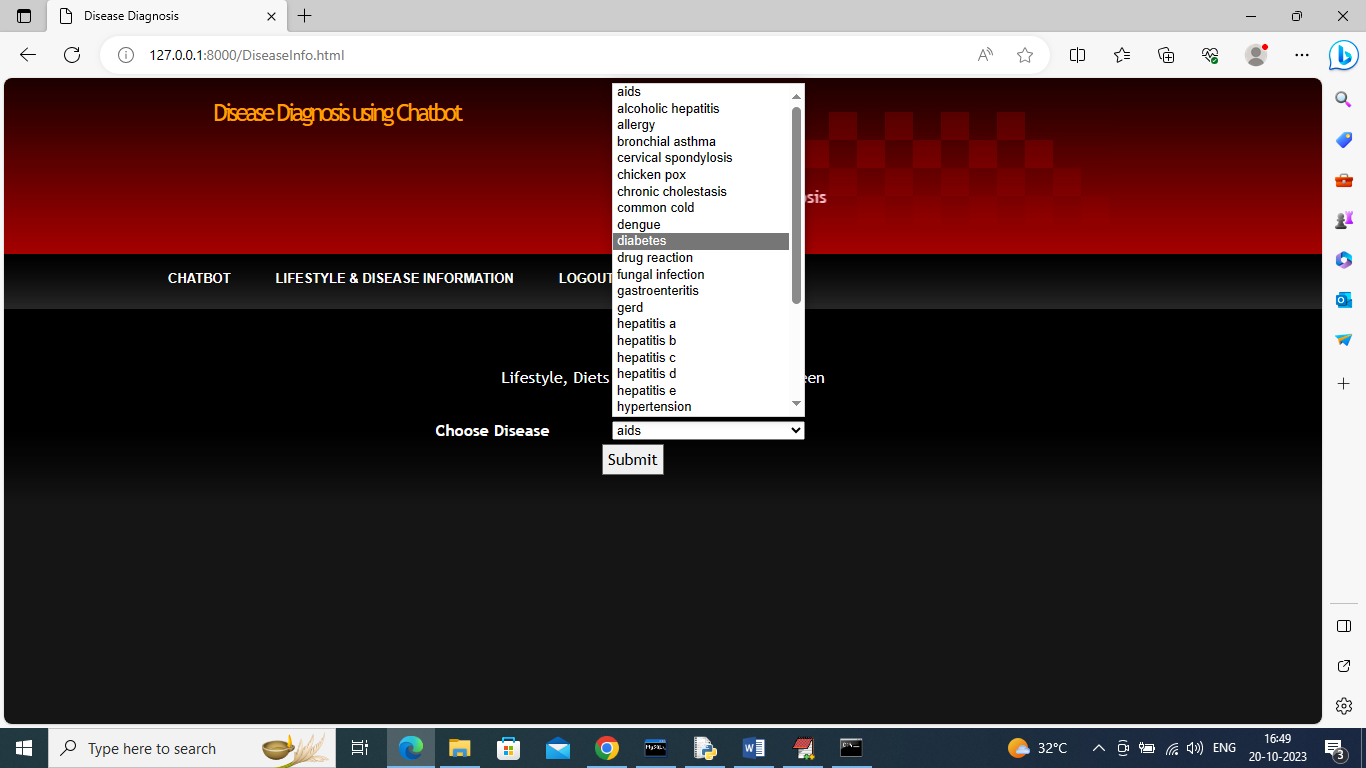
In above email we can see disease details with diet and remedies and similarly you can search for any symptoms and below is another example



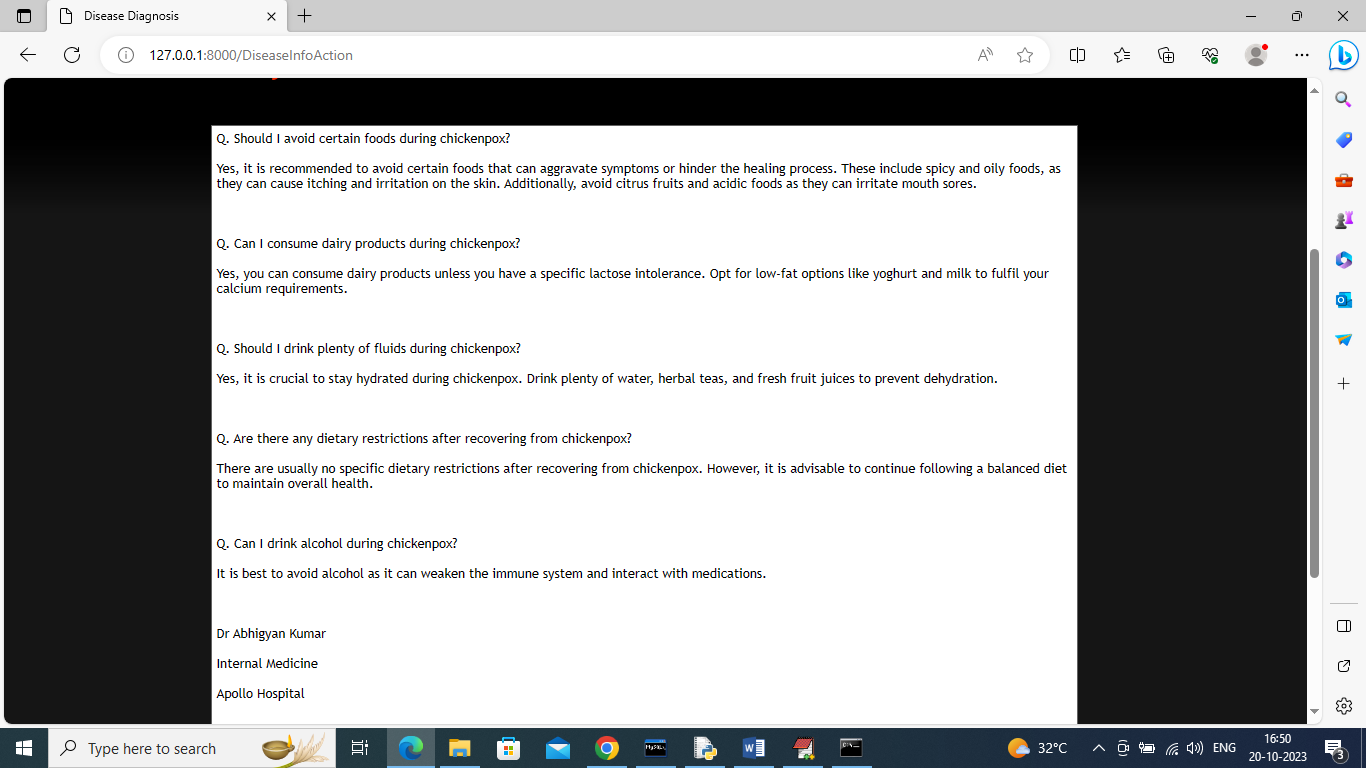
In above screen I gave symptoms as ‘Chest Pain’ and below is the output



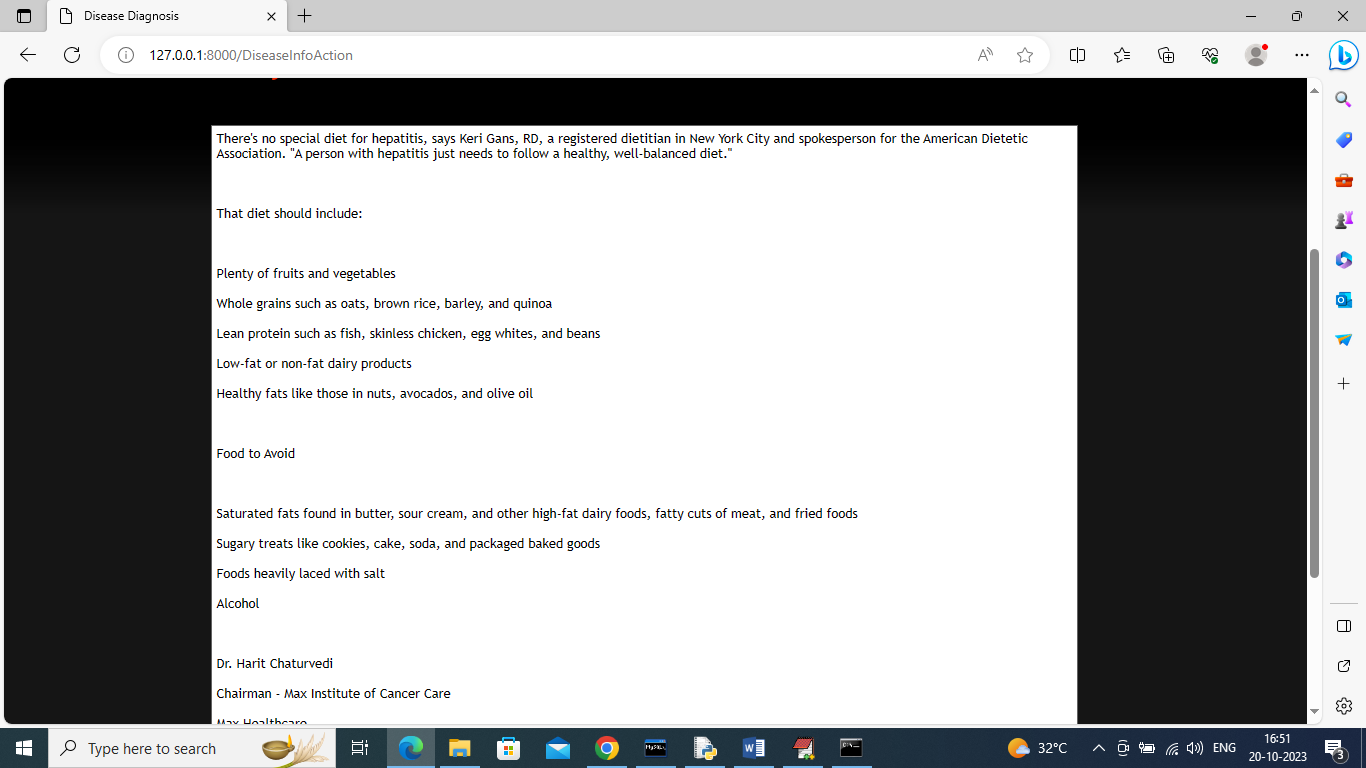
In above screen in blue colour text Chatbot predicted disease as ‘Heart Attack’ for symptom ‘Chest Pain’. Similarly you can search for any symptoms and now click on ‘Lifestyle & Disease Information’ link to view static information about disease



In above screen user can select specific disease and then press button to get disease, diet information like below screen



In above screen user can see some answers about selected disease along with doctor details.



In above screen for some disease we can see information about Food to take and avoid

**8. CONCLUSION:**

This paper explains a medical chatbot which can be used to replace the conventional method of disease diagnosis and treatment recommendation. Chatbot can act as a doctor. The chatbot acts as a user application. The user of this application can specify their symptoms to the chatbot and in turn, chatbot will specify the health measures to be taken. General information about symptom and diseases are available in the dataset and thus the chatbot instance can provide information about disease and treatment to the user. After analyzing the symptoms of the different users, it finally predicts the disease to the user and provides with a link where details about the treatment is visible.A smart medical chatbot can be useful to patients by identifying the symptoms as described by them, giving proper diagnosis and providing with suitable treatment for the disease. In the busy life, it is rare for people to frequently visit hospitals for check-ups. Chatbot is of great importance in such situations as they provide diagnostic assistance with a single click of button. Chatbot doesn‟t require the help of any physician to give proper health measures to the users and this is one of the major advantages of chatbot. Moreover, the cost- effectiveness in using chatbot is a major attractiveness to users. The chat with users is completely personal and this helps users to be more open with their health matters and paves way for chatbot to efficiently identify the disease.

**9. REFERENCES:**

1. The Times of India, https://timesofindia.indiatimes.com/life- style/health-fitness.
2. Sameera A. Abdul-Kader, Dr. John Woods. Survey on Chatbot Design Techniques in Speech Conversation Systems, School of Computer Science and Electronic Engineering/University of Essex Colchester/ UK.
3. Kuligowska, Karolina. (2015). Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents. Professionals Center for Business Research. 2. 1-16. 10.18483/PCBR.22.
4. Divya Madhu,Neeraj Jain C. J, Elmy Sebastain, Shinoy Shaji, Anandhu Ajayakumar. A Novel Approach for Medical Assistance Using Trained Chatbot, International Conference on Inventive Communication and Computational Technologies (ICICCT 2017).
5. Mrs. Rashmi Dharwadkar, Dr.Mrs. Neeta A. Deshpande "A Medical ChatBot"” in International Journal of Computer Trends and Technology (IJCTT) V60(1):41-45 , June 2018.
6. S. Divya, V. Indumathi, S. Ishwarya, M. Priyasankari, S. Kalpana Devi, "A Self-Diagnosis Medical Chatbot Using Artificial Intelligence", J. Web Dev. Web Des.*,* vol. 3, no. 1, pp. 1-7, 2018.
7. Amiya Kumar Tripathy , Rebeck Carvalho , Keshav Pawaskar , Suraj Yadav, “Mobile based healthcare management using artificial intelligence” in International Conference on Technologies for Sustainable Development (ICTSD), 2015, 4-6 Feb, 2015.
8. S. du Preez, M. Lall, S. Sinha, "An intelligent web-based voice chat bot", EUROCON 2009 EUROCON'09. IEEE, pp. 386-391, 2009.
9. R. Ranoliya, N. Raghuwanshi, S. Singh, "Chatbot for university related FAQs", 2017 Int. Conf. Adv. Comput. Commun. Informatics ICACCI 2017, pp. 1525-1530, September 2017.
10. M. Chandhana Surabhi, “Natural language processing future”, 2013 International Conference on Optical Imaging Sensor and Security (ICOSS). doi:10.1109/icoiss.2013.6678407
11. Fco. Mario Barcala, Jes´us Vilares, Miguel A. Alonso, Jorge Gra˜na, Manuel Vilares, “Tokenization and proper noun recognition for information retrieval”, 13th International Workshop on Database and Expert Systems Applications. doi:10.1109/dexa.2002.1045906
12. Pu Han , Si Shen , Dongbo Wang , Yanyun Liu, “The influence of word normalization in English document clustering”, 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE). doi:10.1109/csae.2012.6272740
13. Shweta Taneja , Charu Gupta , Kratika Goyal , Dharna Gureja, “An Enhanced K-Nearest Neighbor Algorithm Using Information Gain and Clustering”, 2014 Fourth International Conference on Advanced Computing & Communication Technologies. doi:10.1109/acct.2014.22