CHAPTER 37 Experimental Investigation and Implementation of Healthcare System Using Artificial Intelligence

Mr. M. K. Sampath
Knowledge Institute of Technology, Salem, India

Mrs. R. Pushpalatha
Knowledge Institute of Technology, Salem, India

Mrs. K. Gowdhami
Knowledge Institute of Technology, Salem, India

Mrs. M. Saranya
Knowledge Institute of Technology, Salem, India

ABSTRACT

With increasing population, increasing birth rate and decreasing death rate due to advancement in the medical field it's found that numbers of doctors are less to serve the need of the increasing population. This scenario can be better understood while walking through the cities government hospitals where the less availability of the doctors is the major cause behind the improper treatment of the patients and in certain scenario the resultant death. Sometime even doctors can make mistake in providing the correct treatment result in death of patient. To encounter such cases there is a need of the smart and Intelligent chatbot who can provide advice to the doctors and sometime even patients about what to do in such cases which ultimately results in the saving the life of hundreds of people. The AI based medical chatbot on which this research topic is based deals with providing medical advice in such scenario because sometime doctors can even make mistake while observing the symptoms but the machine which is specifically developed for it can't make such mistake. This AI based medical chatbot can take decision as per the request of the patient.

Keywords: *AI based medical chatbot , government hospitals, ultimately results etc*

INTRODUCTION

There are many applications that are incorporating a human appearance and intending to simulate human dialog, but in most of thecases the knowledge of the conversational bot is stored in a database created by a human experts. However, very few researches haveinvestigated the idea of creating a chat-bot with an artificial character and personality starting from web pages or plain text about acertain person. This Projectdescribes an approach to the idea of identifying the most important facts in texts describing the life(including the personality) of an historical figure for building a conversational agent that could be used in middle-school CSCLscenarios.Although numerous agent frameworks have been proposed inthe vast body of literature, none of these available frameworks proved to be simple enough to be used by first-year students ofcomputer science. Hence, the authors set out to create a novel framework that would be suitable for the aims of the course, for thelevel of computing skills of the intended group of students, and for the size of this group of students. The content of theintroductory AI course in question is a set of assignments that requires the students to useintelligent agents and other AI techniques to monitor, filter, and retrieve relevant information from the World Wide Web. It represents,therefore, a synthesis of the traditional objectivist approach and a real-world-oriented, constructivist approach to teachingprogramming to novices. The main aim of implementing such a pedagogy was to engage the students in learning to which theypersonally relate while attaining intellectual rigor.

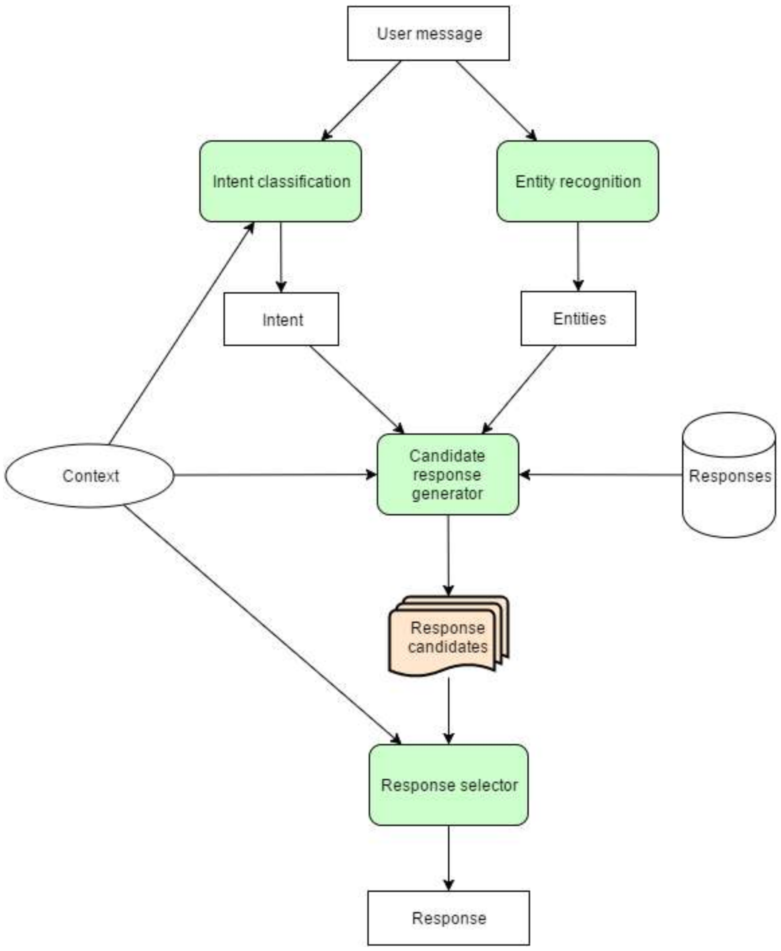

Fig 3.1 Existing system design

Classroom experience indicates that students learn more effectively when thetraditional objectivist approach is combined with a constructivist approach than when this orthodox approach to teachingprogramming to novices is used alone.

STEPS INVOLVED IN THE SYSTEM

A. User Login to System

User registers on Chatbot application. Then ask queries regarding to the healthcare and medical details.
B. Ask some Questions
You can ask some questions regarding some healthcare. And its related to voice- text and text-voice conversation. Using Google API for inter conversion of text-voice and vice versa.
C. Age based Medicine dosage details
You can ask medical dosage related queries to this app in voice and system gets output for medicine API and speak out and display all data. Get your age fromregistration data and provide data related to your data like age, area, gender and so on. Give me age
Then predict disease using SVM Algorithm.
D. Get Medicine Details on medicine name
You can ask about medicine related details on the basis of medicine names.
E. Disease Prediction
Depending on the disease symptoms SVM algorithm can predict the disease.
F. Online API
Use Google API for voice-text and textvoice conversion. The Chatbot API sends query to chatbot and get related answer and refer this answer analysis onthat and display answer on android app. Get medicine related data like medicine name, medicine expiry details and so on from medicine API.

PROPOSED SYSTEM

INTRODUCTION

The Health-Care Chat Bot System should be written in Python, GUI links and a simple, accessible network API. The system must provide a capacity for parallel operation and system design should not introduce scalability issues with regard to the number of surface computers, tablets or displays connected at any one time. The end system should also allow for seamless recovery, without data loss, from individual device failure. There must be a strong audit chain with all system actions logged. While interfaces are worth noting that this system is likely to conform to what is available. With that in mind, the most adaptable and portable technologies should be used for the implementation. The system has criticality in so far as it is a live system

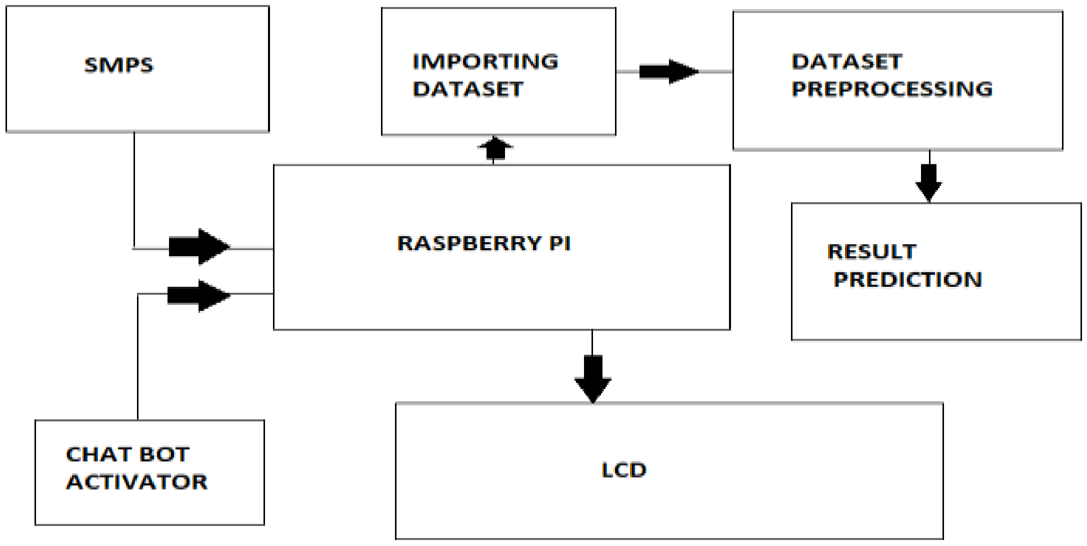

Fig.4.1 System design

If the system is down, then customers must not notice, or notice that the system recovers quickly (seconds). The system must be reliable enough to run, crash and glitch free more or less indefinitely, or facilitate error recovery strong enough such that glitches are never revealed to its end-users.

PROJECT IMPLEMENTATION TECHNOLOGY:

In machine learning, support-vector machines DCNN are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an DCNN training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use DCNN in a probabilistic classification setting). An DCNN model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high dimensional feature spaces.

HARDWARE DESCRIPTION

RASPBERRY PI:

The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market. The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Fig.4.2 The Raspberry Pi 3 Model B+

The Raspberry Pi Compute Module 3+ (CM3+) is a range of DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (on non-Lite variants) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these modules have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards, opening up more options for the designer. The CM3+ contains a BCM2837B0 processor (as used on the Raspberry Pi

3B+), 1Gbyte LPDDR2 RAM and eMMC Flash. The CM3+ is currently available in 4 variants, CM3+/8GB, CM3+/16GB, CM3+/32GB and CM3+ Lite, which have 8, 16 and 32 Gigabytes of eMMC Flash, or no eMMC Flash, respectively. The CM3+ Lite product is the same as CM3+ except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device. Note that the CM3+ is electrically identical and, with the exception of higher CPU z-height, physically identical to the legacy CM3 products. CM3+ modules require a software/firmware image dated November 2018 or newer to function correctly

• Low cost

• Low power

• High availability

• High reliability – Tested over millions of Raspberry Pis Produced to date – Module IO pins have 15 micro-inch hard gold plating over 2.5 micron Nickel

• Mature and stable Linux software stack – Latest Linux Kernel support – Many drivers upstreamed – Stable and well supported userland – Full availability of GPU functions using standard APIs

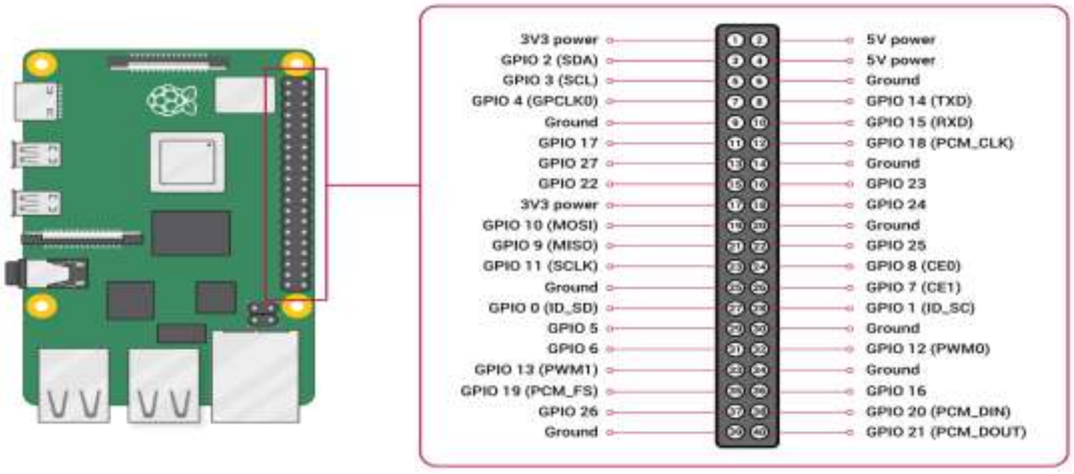

Fig.4.3 Raspberry pi pin diagram

Mechanical Specification The CM3+ modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules and therefore should work with the many DDR2 SODIMM sockets available on the market. (Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.) The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture. The maximum component height on the underside of the Compute Module is 1.2mm. The maximum component height on the top side of the Compute Module is 2.5mm. The Compute Module PCB thickness is 1.0mm +/- 0.1mm. Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified. Figure 2 gives the CM3+ mechanical dimensions
The Compute Module 3+ has six separate supplies that must be present and powered at all times; you

cannot leave any of them unpowered, even if a specific interface or GPIO bank is unused. The six supplies are as follows: 1. VBAT is used to power the BCM2837 processor core. It feeds the SMPS that generates the chip core voltage. 2. 3V3 powers various BCM2837 PHYs, IO and the eMMC Flash. 3. 1V8 powers various BCM2837 PHYs, IO and SDRAM. 4. VDAC powers the composite (TV-out) DAC. 5. GPIO0-27 VREF powers the GPIO 0-27 IO bank. 6. GPIO28-45 VREF powers the GPIO 28-45 IO bank.

LCD DISPLAY:

Product features: The I2C 1602 LCD module is a 2 line by 16 character display interfaced to an I2C daughter board. The I2C interface only requires 2 data connections, +5 VDC and GND For in depth information on I2C interface and history, visit: http://www.wikipedia/wiki/i2c Specifications: 2 lines by 16 character I2C Address Range 0x20 to 0x27 (Default=0x27, addressable) Operating Voltage 5 Vdc Backlight White Adjustable by potentiometer on I2c interface 80mm x 36mm x 20 mm 66mm x 16mm Power: The device is powered by a single 5Vdc connection

This is a 16x2 LCD display screen with I2C interface. It is able to display 16x2 characters on 2 lines, white characters on blue background.

Usually, Arduino LCD display projects will run out of pin resources easily, especially with Arduino Uno. And it is also very complicated with the wire soldering and connection. This I2C 16x2 Arduino LCD Screen is using an I2C communication interface. It means it only needs 4 pins for the LCD display: VCC, GND, SDA, SCL. It will save at least 4 digital/analog pins on Arduino. All connectors are standard XH2.54 (Breadboard type). You can connect with the jumper wire directly

To avoid the confliction of I2C address with other I2C devices, such ultrasonic sensor, IMU, accelerometers, and gyroscope, the I2C address of the module is configurable from 0x20-0x27. And its contrast can be adjusted manually.

Another alternative option is I2C 20x4 Arduino LCD Display Module if more characters are required.

The I2 C LCD component drives an I2 C interfaced 2 line by 16 character LCD. The I2 C LCD component is a wrapper around an I2 C Master component and makes use of an existing I2 C Master component. If a project does not already have an I2 C Master component, one is required in order to operate. When one of the API functions is called, that function calls one or more of the I 2 C Master functions in order to communicate with the LCD.

The I2 C LCD component is used in applications that require a visual or textual display. This component is also used where a character display is needed but seven consecutive GPIOs on a single GPIO port are not possible. In cases where the project already includes an I2 C master, no additional GPIO pins are required.

POWER SUPPLY:

TRANSFORMER:

This document presents the solution for a 12V 1A flyback converter based on the Infineon OPTIREG™ TLE8386-2EL controller and IPD50N08S4-13 OptiMOS™-T2. The user is guided through the component selections, the circuit design and, finally, an overview of the experimental results are presented. The TLE8386-2EL is part of the Automotive OPTIREG™ family and it implements a low-side-sense current mode controller with built in protection features. The device is AECQ-100 qualified.

The IPD50N08S4-13 is an AEC-Q101 qualified 80V N-channel enhanced mode MOSFET, it is part of the OptiMOS™-T2 family.

Intended audience This document is intended for power supply design engineers, application engineers, students, etc., who need to design a Flyback converter for automotive power applications where a galvanic isolation between two voltage domains is required. In particular the focus is on a battery connected flyback that delivers up to 12W at 12V output voltage; the intention is to provide the user with all of the needed information to fully design and characterize the SMPS bringing it from an engineering concept to its production. Specific features and applications are: - 48V to 12V Automotive applications - Isolated current mode SMPS - Flyback regulators with auxiliary sensing

CENTRE TAPPED TRANSFORMER SPECIFICATIONS

- Step-down Centre tapped Transformer
- Input Voltage: 220V AC at 50Hz
- Output Voltage: 24V, 12V or 0V
- Output Current: 1A
- Vertical mount type
- Low cost and small package

A centre-tapped transformer also known as two phase three wire transformer is normally used for rectifier circuits. When a digital project has to work with AC mains a Transformer is used to step-down the voltage (in our case, to 24V or 12V) and then convert it to DC by using a rectifier circuit. In a center-tapped transformer the peak inverse voltage is twice as in bridge rectifier hence this transformer is commonly used in full wave rectifier circuits.

The operation and theory behind a Center tapped transformer is very similar to a normal secondary transformer. A primary voltage will be induced in the primary coil (I1 and I3) and due to magnetic induction the voltage will be transferred to the secondary coil. Here in the secondary coil of a centre tapped transformer, there will be an additional wire (T2) which will be placed exactly at the center of the secondary coil, hence the voltage here will always be zero.

If we combine this zero potential wire (T2) with either T1 or T2, we will get a voltage of 12V AC. If this wire is ignored and voltage across T1 and T2 is considered then we will get a voltage of 24V AC. This feature is very useful for the function of a full wave rectifier.

Let us consider the voltage given by the first half of the secondary coil as Va and the voltage across the second half of the secondary coil as Vb as shown

RECTIFER CIRCUIT:

We have learnt in rectifier circuits about converting a sinusoidal ac voltage into its corresponding pulsating dc. Apart from the dc component, this pulsating dc voltage will have unwanted ac components like the components of its supply frequency along with its harmonics (together called ripples). These ripples will be the highest for a single-phase half wave rectifier and will reduce further for a single-phase full wave rectifier. The ripples will be minimum for 3-phase rectifier circuits. Such supply is not useful for driving complex electronic circuits. For most supply purposes constant dc voltage is required than the pulsating output of the rectifier. For most applications the supply from a rectifier will make the operation of the circuit poor. If the rectifier output is smoothened and steady and then passed on as the supply voltage, then the overall operation of the circuit becomes better. Thus, the output of the rectifier has to be passed though a filter circuit to filter the ac components. The filter is a device that allows passing the dc component of the load and blocks the ac component of the rectifier output. Thus the output of the filter circuit will be a steady dc voltage. The filter circuit can be constructed by the combination of components like capacitors, resistors, and inductors. Inductor is used for its property that it allows only dc components to pass and blocks ac signals. Capacitor is used so as to block the dc and allows ac to pass. All the combinations and their working are explained in detail below. Series Inductor Filter The circuit diagram of a full wave rectifier with a series inductor filter is given below. As the name of the filter circuit suggests, the Inductor L is connected in series between the rectifier circuit and the load. The inductor carries the property of opposing the change in current that flows through it. In other words, the inductor offers high impedance to the ripples and no impedance to the desired dc components. Thus the ripple components will be eliminated. When the rectifier output current increases above a certain value, energy is stored in it in the form of a magnetic field and this energy is given up when the output current falls below the average value.

Thus all the sudden changes in current that occurs in the circuit will be smoothened by placing the inductor in series between the rectifier and the load. The waveform below shows the use of inductor in the circuit. From the circuit, for zero frequency dc voltage, the choke resistance Ri in series with the load resistance RL forms a voltage divider circuit, and thus the dc voltage across the load is Vdc = RL/(Ri + RL) Vdc is the output from a full wave rectifier. In this case, the value of Ri is negligibly small when compared to RL. The effect of higher harmonic voltages can be easily neglected as better filtering for the higher harmonic components take place. This is because of the fact that with the increase in frequency, the reactance of the inductor also increases. It should be noted that a decrease in the value of load resistance or an increase in the value of load current will decrease the amount of ripples in the circuit. So, the series inductor filter is mostly used in cases of high load current or small load resistance. A simple series inductor filter may not be properly used. It is always better to use a shunt capacitor (C) with series inductor (L) to form an LC Filter. Shunt Capacitor Filter As the name suggests, a capacitor is used as the filter and this high value capacitor is shunted or placed across the load impedance. This capacitor, when placed across a rectifier gets charged and stores the charged energy during the conduction period. When the rectifier is not conducting, this energy charged by the capacitor is delivered back to the load. Through this energy storage and delivery process, the time duration during which the current flows through the load resistor gets increased and the ripples are decreased by a great amount. Thus for the ripple component with a frequency of ‘f’ megahertz, the capacitor ‘C’ will offer a very low impedance. The value of this impedance can be written as: Shunt Capacitor Impedance = 1/2 fC Thus the dc components of the input signal along with the few residual ripple components, is only allowed to go through the load resistance RLoad. The high amount of ripple components of current gets bypassed through the capacitor C.

Now let us look at the working of Half-wave rectifier and Full-wave rectifier with Capacitor filters, their output filtered waveform, ripple factor, merits and demerits in detail.

SOFTWARE DESCRIPTION

PYTHON 3.7

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object- oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in manya reason most platforms and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation. The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications. This tutorial introduces the reader informally to the basic concepts and features of the Python language and system. It helps to have a Python interpreter handy for hands-on experience, but all examples are self-contained, so the tutorial can be read off- line as well. For a description of standard objects and modules, see library-index. Reference-index gives a more formal definition of the language. To write extensions in C or C++, read extending-index and c-api-index. There are also several books covering Python in depth. This tutorial does not attempt to be comprehensive and cover every single feature, or even every commonly used feature. Instead, it introduces many of Python's most notes worthy features, and will give you a good idea of the language's flavor and style. After reading it, you will be able to read and write Python modules and programs, and you will be ready to learn more about the various Python library modules described in library-index. If you do much work on computers, eventually you find that there's some task you'd like to automate. For example, you may wish to perform a search-and-replace over a large number of text files, or rename and rearrange a bunch of photo files in a complicated way. Perhaps you'd like to write a small custom database, or a specialized

GUI application or a simple game. If you're a professional software developer, you may have to work with several C/C++/Java libraries but find the usual write/compile/test/re-compile cycle is too slow. Perhaps you're writing a test suite for such a library and find writing the testing code a tedious task. Or maybe you've written a program that could use an extension language, and you don't want to design and implement a whole new language for your application.

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero exit status. If that doesn't work, you can exit the interpreter by typing the following command: quit(). The interpreter's line-editing features include interactive editing, history substitution and code completion on systems that support read line. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get. If it beeps, you have command line editing; see Appendix Interactive Input Editing and History Substitution for an introduction to the keys. Ifnothing appears to happen, or if ^P is echoed, command line editing isn't available; you'll only be able to use backspace to remove characters from the current line. The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a script from that file. A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in command, analogous to the shell's -c option. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote commands in its entiretywithsinglequotes.SomePythonmodulesarealsousefulasscripts. These can be invoked using python-m module [arg]...,which executes the source file for the module as if you had spelled out its full name on the command line. When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -i before the script.

There are tools which use doc strings to automatically produce online or printed documentation or to let the user interactively browse through code; it's good practice to include doc strings in code that you write, so make a habit of it. The execution of a function introduces a new symbol table usedfor the local variables of the function. More precisely, all variable assignments in a functions to read the value in the local symbol table; whereas variable references first look in the local symbol table, then in the local symbol tables of enclosing functions, then in the global symbol table, and finally in the table of built-in names. Thus, global variables cannot be directly assigned a value within a function (unless named in a global statement), although they may be referenced. The actual parameters (arguments) to a function call are introduced in the local symbol table of the called function when it is called; thus, arguments are passed using call by value (where the value is always an object reference, not the value of the object).1 When a function calls another function, a new local symbol table is created for that call. A function definition introduces the function name in the current symbol table. The value of the function name has a type that is recognized by the interpreter as a user-defined function. This value can be assigned to another name which can then also be used as a function.

Annotations are stored in the annotations attribute of the function as a dictionary and haven o effect on any other part of the function. Parameter annotations are defined by a colon after the parameter name, followed by an expression evaluating to the value of the annotation. Return annotationsare defined by a literal ->, followed by an expression, between the parameter list and the colon denoting the end of the def statement.

The comparison operators in and not in check whether a value occurs (does not occur) in a sequence. The operator is and does not compare whether two objects are really the same object; this only matters for mutable objects like lists. All comparison operators have the same priority, which is lower than that of all numerical operators. Comparisons can be chained. For example,a<b==ctestswhetheraislessthanbandmoreoverbequalsc. Comparisons may be combined using the Boolean operators and the outcome of a comparison (or of any other Boolean expression) may be negated with not. These have lower priorities than comparison operators; between them, not has the highest priority and or the lowest, so that A and not B or C is equivalent to (A and (not B)) or C. As always, parentheses can be used to express the desired composition. The Boolean operators and are so-called short-circuit operators: their arguments are evaluated from left to right, and evaluation stops as soon as the outcome is determined. For example, if A and C are true but Bis false, A and B and C does not evaluate the expression C. When used as a general value and not as a Boolean, the return value of a short-circuit operator is the last evaluated argument.

Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state. Compared with other programming languages, Python's class mechanism adds classes with a minimum of new syntax and semantics. It is a mixture of the class mechanisms found in C++ and Modula-3. Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or classes, and a method can call the method of a base class with the same name. Objects can contain arbitrary amounts and kinds of data. As is true for modules, classes partake of the dynamic nature of Python: they are created at runtime, and can be modified further after creation. In C++ terminology, normally class members (including the data members) are public (except see below Private Variables), and all member functions are virtual. A sin Modula-3, there are no short hands for referencing the object's members from its methods: the method function is declared with an explicit first argument representing the object, which is provided implicitly by the call. A sin Small talk, classes themselves are objects. This providesSemantics for importing and renaming. Unlike C++ and Modula-3, built-in types can be used as base classes for extension by the user. Also, like in C++, most built-in operators with special syntax (arithmetic operators, sub scripting etc.) can be redefined for class instances.(Lacking universally accepted terminology to talk about classes, I will make occasional use of Smalltalk and C++ terms. I would use Modula-3 terms, since its object- oriented semantics are closer to those of Python than C++, but I expect that few readers have heard of it.)

Objects have individuality, and multiple names (in multiple scopes) can be bound to the same object. This is known as aliasing in other languages. This is usually not appreciated on a first glance at Python, and can be safely ignored when dealing with immutable basic types (numbers, strings, tuples).However, aliasing has a possibly surprising effect on these mantic of Python code involving mutable objects such as lists, dictionaries, and most other types. This is usually used to the benefit of the program, since aliases behave like pointers in some respects. For example, passing an object is cheap since only a pointer is passed by the implementation; and if a function modifies an object passed as an argument, the caller will see the change — this eliminates the need for two different argument passing mechanisms as in Pascal.

A namespace is a mapping from names to objects. Most name spaces are currently implemented as Python dictionaries, but that’s normally not noticeable in any way (except for performance), and it may change in the future. Examples of name spaces are: these to f built-in names (containing functions such as abs(), and built-in exception names); the global names in a module; and the local names in a function invocation. In a sense the set of attributes of an object also form a namespace. The important thing to know about namespaces is that there is absolutely no relation between names in different namespaces; for instance, two different modules may both define a function maximize without confusion — users of the modules must prefix it with the module name. By the way, I use the word attribute for any name following a dot — for example, in the expression z. real, real is an attribute of the object z. Strictly speaking, references to names in modules are attribute references: in the expression modname.funcname, modname is a module object and funcname is an attribute of it. In this case there happens to be a straight forward mapping between the module’s attributes and the global names defined in the module: they share the same namespace!1 Attributes may be read-only or writable. In the latter case, assignment to attributes is possible.Module attributes are writable: you canwrite modname.the_answer = 42. Writable attributes may also be deleted with the del statement. For example, del mod name .the_ answer will remove the attribute the_answer from the object named by mod name. Namespaces are created at different moments and have different lifetimes. The namespace containing the built-in names is created when the Python interpreter starts up, and is never deleted. The global namespace for a module is created when the module definition is read in; normally, module namespaces also last until the interpreter quits.The statements executed by the top-level invocation of the interpreter, either read from a script file or interactively, are considered part of a module called main, so they have their own global namespace.(The built-in names actually also live in a module; this is called built ins.) The local namespace for a function is created when the function is called, and deleted when the function returns or raises an exception that is not handled within the function. (Actually, forgetting would be a better way to describe what actually happens.) Of course, recursive invocations each have their own local namespace. Python checks the modification date of the source against the compiled version to see if it’s out of date and needs to be recompiled. This is a completely automatic process. Also, the compiled modules are platform-independent, so the same library can be shared among systems with different architectures. Python does not check the cache in two circumstances. First, it always recompiles and does not store the result for the module that’s loaded directly from the command line. Second, it does not check the cache if there is no source module. To support anon-source (compiled only) distribution, the compiled module must be in the source directory, and there must not be a source module. Some tips for experts:

• You can use the -O or -OO switches on the Python command to reduce the size of a compiled module. The -O switch removes assert statements, the -OO switch removes both assert statements and docstrings. Since some programs may rely on having these available, you should only use this option if you know what you’re doing. “Optimized”modules have an opt- tag and are usually smaller. Future releases may change the effects of optimization.

• A program doesn’t run any faster when it is read from a .pyc file than when it is read from a .py file; the only thing that’s faster about .pyc files is the speed with which they are loaded.

• The module compile all can create .pyc files for all modules in a directory.

• There is more detail on this process, including a flow chart of the decisions

THONNY IDE

Thonny is as mall and light weight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME, so Thonny only requires the GTK2 toolkit and therefore you only need the GTK2 runtime libraries installd to run it.

For compiling Thonny yourself, you will need the GTK (>= 2.6.0) libraries and header files. You will also need the Pango, Gliband ATK libraries and header files. All these files are available at http://www.gtk.org. Furthermore you need, of course, a C compiler and the Make tool; a C++ compiler is also required for the included Scintilla library. The GNU versions of these tools are recommended.

There are also some compile time options which can be found in src/Thonny .h. Please see Appendix C for more information. In the case that your system lacks dynamic linking loader support, you probably want to pass the option --disable-vte to the configure script. This prevents compiling Thonny with dynamic linking loader support to automatically load libvte.so.4 if available. Thonny has been successfully compiled and tested under Debian 3.1 Sarge, Debian 4.0 Etch, Fedora Core 3/4/5, Linux From Scratch and FreeBSD 6.0. It also compiles under Microsoft Windows

At startup, Thonny loads all files from the last time Thonny was launched. You can disable this feature in the preferences dialog (see Figure 3-4). If you specify some files on the command line, only these files will be opened, but you can find the files from the last session in the file menu under the "Recent files" item. By default this contains the last 10 recently opened files. You can change the amount of recently opened files in the preferences dialog. You can start several instances of Thonny , but only the first will load files from the last session. To run a second instance of Thonny , do not specify any file names on the command-line, or disable opening files in a running instance using the appropriate command line option.

Thonny detects an already running instance of itself and opens files from the command-line in the already running instance. So, Thonny can be used to view and edit files by opening them from other programs such as a filemanager. If you do not like this for some reason, you can disable using the first instance by using the appropriate command line option

If you have installed libvte.so in your system, it is loaded automaticallyThonny , and you will have a terminal widget in the notebook at the bottom. If Thonny cannot find libvte.so at startup, the terminal widget will not be loaded. So there is no need to install the package containing this file in order to run Thonny . Additionally, you can disable the use of the terminal widget by command line option, for more information see Section3.2.You can use this terminal (from now on called VTE) nearly as an usual terminal program like xterm. There is basic clipboard support. You can paste the contents of the clipboard by pressing the right mouse button to open the popup menu and choosing Paste. To copy text from the VTE, just select the desired text and then press the right mouse button and choose Copy from the pop up menu. On systems running the X Window System you can paste the last selected text by pressing the middle mouse button in the VTE (on 2-button mice, the middle button can often be simulated by pressing both mouse buttons together).

As long as a project is open, the Make and Run commands will use the project's settings, instead of the defaults. These will be used whichever document is currently displayed. The current project's settings are saved when it is closed, or when Thonny is shut down. When restarting Thonny , the previously opened project file that was in use at the end of the last session will be reopened.

Execute will run the corresponding executable file, shell script or interpreted script in a terminal window. Note that the Terminal tool path must be correctly set in the Tools tab of the Preferences dialog - you can use any terminal program that runs a Bourne compatible shell and accept the "-e" command line argument to start a command. After your program or script has finished executing, you will be prompted to press the return key. This allows you to review any text output from the program before the terminal window is closed.

By default the Compile and Build commands invoke the compiler and linker with only the basic arguments needed by all programs. Using Set Includes and Arguments you can add any include paths and compile flags for the compiler, any library names and paths for the linker, and any arguments you want to use when running Execute.

Thonny has basic printing support. This means you can print a file by passing the filename of the current file to a command which actually prints the file.

However, the printed document contains no syntax highlighting.

RESULT

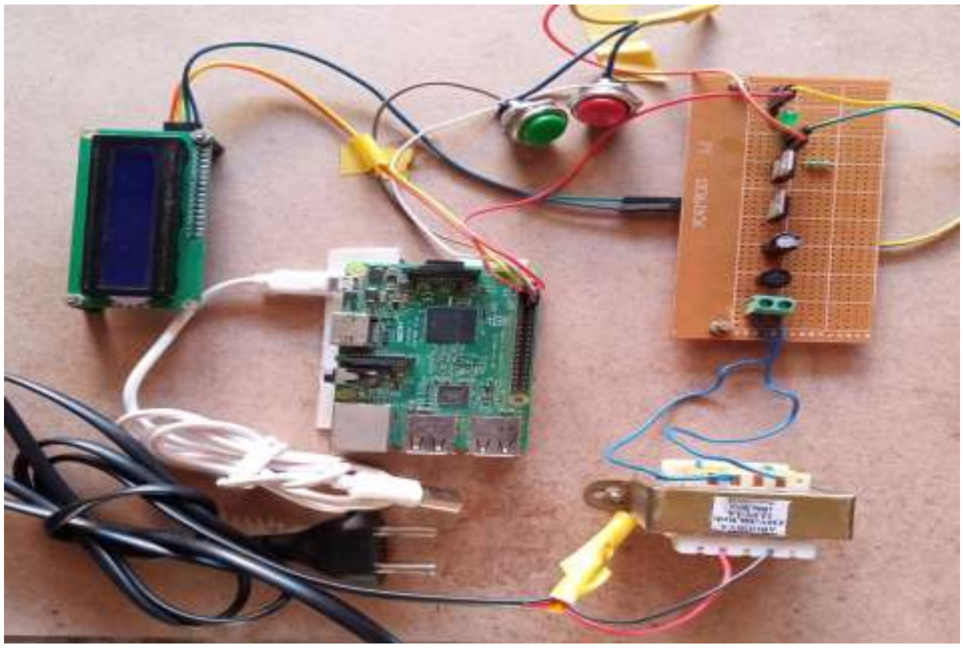

Fig 6.2 Hardware result

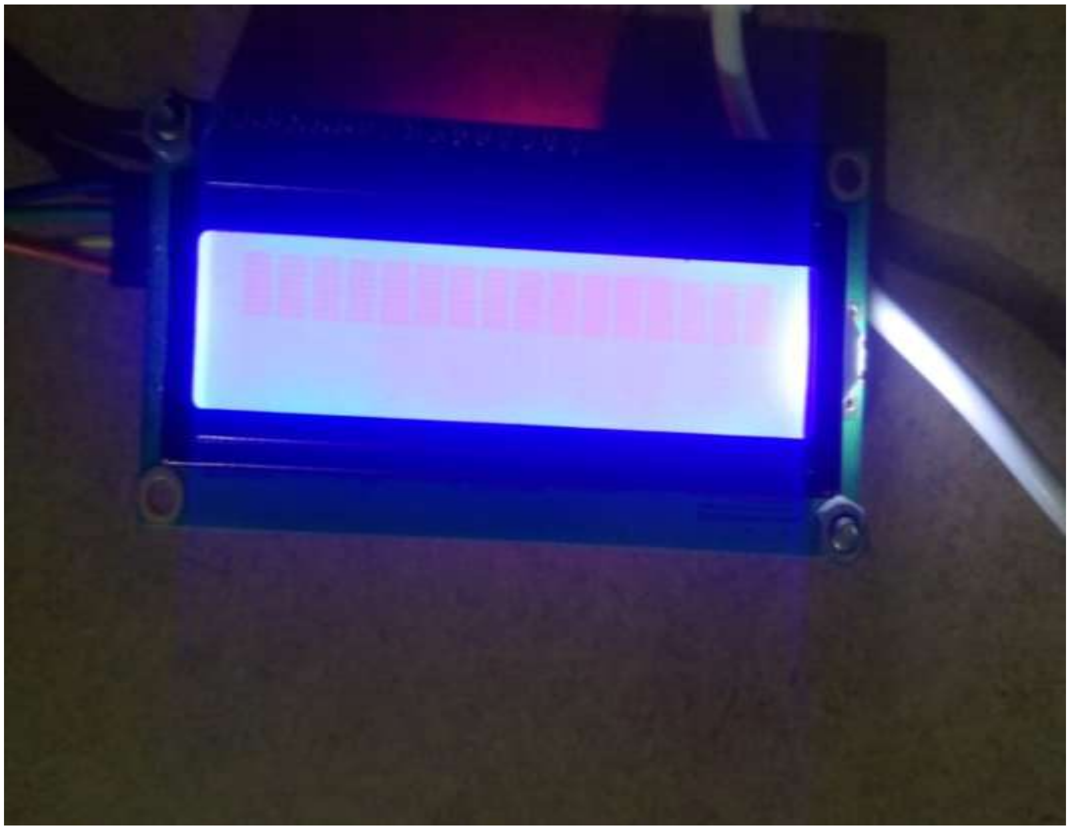

Fig 6.3 LCD showing decease and certainty

CONCLUSION AND FUTURE SCOPE

Thus, we can conclude that this system giving the accurate result. As we are using large dataset which will ensures the better performance. Thus, we build up a system which is useful for people to detect the disease by typing symptoms
Chat bots are a thing of the future which is yet to uncover its potential but with its rising popularity and craze among companies, they are bound to stay here for long. Machine learning has changed the way companies were communicating with their customers. With new platforms to build various types of chat bots being introduced, it is of great excitement to witness the growth of a new domain in technology while surpassing the previous threshold

REFERENCES

[1] Flora Amato, Stefano Marrone, “Chatbots meet eHealth: automat zing healthcare”, proceeding of diet, May-2018.

[2] BenildaEleonor V. Comendador, “Pharmabot: A pediatric generic Medicine consultant Chatbot”, proceeding of the JACE, April 2015.

[3] Divya, Indumathi, Ishwarya, Priyasankari, “A SelfDiagnosis Medical Chatbot Using Artificial Intelligence”, proceeding MAT Journal, October-2017.

[4] Tobias Kowatsch,” Text-based Healthcare Chatbots Supporting Patient and Health”, 01 October 2017.

[5] Chin-Yuan Huang, Ming-Chin Yang, Chin-Yu Huang, “A Chatbot-supported Smart Wireless Interactive Healthcare System for Weight Control and Health Promotion”, proceeding of the IEEE, April-2018.

[6] Boukricha, H., Wachsmuth, I.: Modeling Empathy for a Virtual Human: How, When and to What Extent. The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3. International Foundation for Autonomous Agents and Multiagent Systems, 2011., pp. 1135–1136

[7] Agarwal, R., Gao, G., DesRoches, C., et al.: The Digital Transformation of Healthcare: Current Status and the Road Ahead. Information Systems Research 21, 796-809 (2010).

[8] Aron, A., Aron, E.N., Smollan, D.: Inclusion of Other in the Self Scale and the structure of interpersonal closeness. Journal of Personality and Social Psychology 63, 596-612 (1992).

[9] Bickmore, T., Cassell, J.: Social Dialogue with Embodied Conversational Agents. In: Kuppevelt, J.C.J., Bernsen, N.O., Dybkjær, L. (eds.) Advances in Natural Multimodal Dialogue Systems, vol. 30, pp. 23–54. Springer, Dordrecht (2005).

[10] Bickmore, T., Gruber, A., Picard, R.: Establishing the computer–patient working alliance in automated health behavior change interventions. Patient Education and Counseling 59, 21-30 (2005).

[11] K. Oh, D. Lee, B. KO and H. Choi, "A Chatbot for Psychiatric Counseling in Mental Healthcare Service Based on Emotional Dialogue Analysis and Sentence Generation," 2017 18th IEEE International Conference on Mobile Data Management (MDM), Daejeon, 2017, pp. 371-375. doi: 10.1109/MDM.2017.64

[12] Du Preez, S.J. & Lall, Manoj & Sinha, S. (2009). An intelligent webbased voice chat bot. 386 - 391.10.1109/EURCON.2009.5167660

[13] Bayu Setiaji, Ferry Wahyu Wibowo, "Chatbot Using a Knowledge in Database: Human-to-Machine Conversation Modeling", Intelligent Systems Modelling and Simulation (ISMS) 2016 7th International Conference on, pp. 72-77, 2016.

[14] Dahiya, Menal. (2017). A Tool of Conversation: Chatbot. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING. 5. 158-161.2017[15] C.P. Shabariram, V.

Srinath, C.S. Indhuja, Vidhya (2017). Ratatta: Chatbot Application Using Expert System, International Journal of Advanced Research in Computer Science and Software Engineering,2017