INTRODUCTION OF THE PROJECT

A “chatbot” is a piece of software that conducts a conversation via auditory or textual methods. Such programs are often designed to convincingly simulate how a human would behave as a conversational partner

Through chatbots one can communicate with text or voice interface and get reply through artificial intelligence. Typically, a chat bot will communicate with a real person. Chat bots are used in applications such as ecommerce customer service, call centres and Internet gaming. Chatbots are programs built to automatically engage with received messages. Chatbots can be programmed to respond the same way each time, to respond differently to messages containing certain keywords and even to use machine learning to adapt their responses to fit the situation. A developing number of hospitals, nursing homes, and even private centers, presentlyutilize online

Chatbots for human services on their sites. These bots connect with potential patients visiting the site, helping them discover specialists, booking their appointments, and getting them access to the correct treatment. In any case, the utilization of artificial intelligence in an industry where individuals’ lives could be in question, still starts misgivings in individuals. It brings up issues about whether the task mentioned above ought to be assigned to human staff. This healthcare chatbot system will help hospitals

to provide healthcare support online 24 x 7, it answers deep as well as general questions. It also helps to generate leads and automatically delivers the information of leads to sales. By asking the questions in series it helps patients by guiding what exactly he/she is looking for.

BENEFITS:

- **On a Personal Level:** It allows you to communicate with bots (machines) and simulate that you are talking with a human.
- **On the Professional Level:** Various organization adapt chatbots, as itreduce cost of Human Resources.
- **At the Company Level:** It allows have a conversation with your audience gain customer feedback, and elevate your brand.

FEATURES:

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

Chatbots In Industry

Edtech Industries

EdTech was a promising industry before the pandemic forced all education to move online. HolonIQ expects a surge of $404 billion in valuation in this industry worldwide.

In India, the situation isn't very different. Last year alone, we've seen over $2.2 billion in investments in Edtech companies. In fact, the number of startups in this sector has gone up beyond 4500 and will see a steep increase in the future.

Artificial Intelligence in education is helping with personalisation and customisation, depending on the need. This will especially be useful as we are seeing a shift towards digital education, including in reputed institutions.

This brings upon an opportunity as well as a huge challenge for the market players to serve everyone right. As everyone is new to online education, there will be many support questions, from teachers, parents and students alike.

How To Schedule WhatsApp Messages On Android Phone
The questions can be on many different themes, like using the software optimally or education-related inquiries.

Another use case would be the replacement of notice boards. Sharing updates on upcoming events and results will be online too.

Chatbots are a natural saviour when one needs to introduce some automation to their business operations and services.

Foodtech Industries

Food Tech is one of the most challenging industries in the world. As one of the fastest-moving SKUs with incredibly short shelf lives and high stakes support, the industry needs technology that can help it scale.

The Food Tech ecosystem has seen a sharp rise in online users. Combined with the lockdown, we have seen a massive increase in the demand for food delivery services last year.

Because of this, investors have poured billions into the Indian Food Tech ecosystem. $10.8 billion, to be specific. And this trust has only grown with time.

Delivery-focused startups such as BigBasket, Faasos, Zomato, Milkbasket, Swiggy and Freshmenu, etc., are the primary recipients of investor funding – receiving over 97% of investments over the past 4 years.

So what are the challenges that this industry faces? Well, there are two primary blockers.

1. Surge and dips in volume
Food Tech companies process thousands of orders in a day. With scale, this number doubles and triples quickly over several months. This puts a lot of load on the human agents.

2. Unequal distribution of tickets
Support tickets are not created equally in the Food Tech industry. There is a peak during meal hours, that's lunch and dinner. The peak is seen on a day-to-day basis as well as on a week-on-week and month-on-month basis. For example, companies like Swiggy or Zomato will get a majority of their orders around lunch or dinner. This means that while support teams are slammed in the afternoon or evening, they're left largely unworked otherwise.

What Food Tech companies need is a means of support that's –

1. more effective,
2. less income intensive,
3. more interactive,
4. easy to use,
5. easy to scale,
6. and automated.

Using a Food Tech chatbot allows you to tackle your customer's most repeated queries, immediately. Because it's automated, questions like "where is my order", "my order is delayed", "I received an incorrect product", "my payment didn't go through", etc. are answered with a first response time of only seconds.

With fewer queries taking up human time, your agents can focus on the complex queries that need their attention. This personalised touch improves customer satisfaction while bringing down turnaround time.

Healthcare Chatbot

Physician billing hours are expensive but medical facilities have to balance cost with ensuring patients have valuable one-on-one time with their doctor. Using chatbots in the medical field can help. From scheduling appointments to answering simple, repeatable questions, chatbots can provide important services and information to patients without eating up precious time, or overbilling patients.

Chatbots in the healthcare industry can:

- Conduct post-surgery interviews.
- Offer tips and information about medicine.
- Track and moderate medication side effects.
- Offer mental health screenings.
- Track weight.
- Send prescription refill reminders.
-

Mental health chatbots such as Woebot and Replika can also help users learn therapy exercises, stress management, and coping skills—as well as provide tools in calming anxiety and thinking positively. These are especially useful for patients who may still have anxiety about speaking with a human therapist.

Hospitality Chatbot

In hotels, chatbots can do more than simply make or cancel reservations — they can inform consumers about room availability, answer frequently asked questions about rules and amenities, and even offer guest promotions quickly and easily without distracting front-of-house staff. They can also be used to build customer profiles, so that hospitality businesses can be informed and aware of customer preferences before they arrive and offer personalized recommendations for area attractions and restaurants.

The real estate industry uses chatbots more frequently than any other industry—the ability for these small businesses to answer customer questions around the clock in a timely fashion is critical when it comes to making a sale, or renting a unit.

TECHNOLOGY USED

ABOUT PYTHON:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system

What can PYTHON do?

Python is a general purpose programming language. Hence, you can use the programming language for developing both desktop and web applications. Also, you can use Python for developing complex scientific and numeric applications. Python is designed with features to facilitate data analysis and visualization.

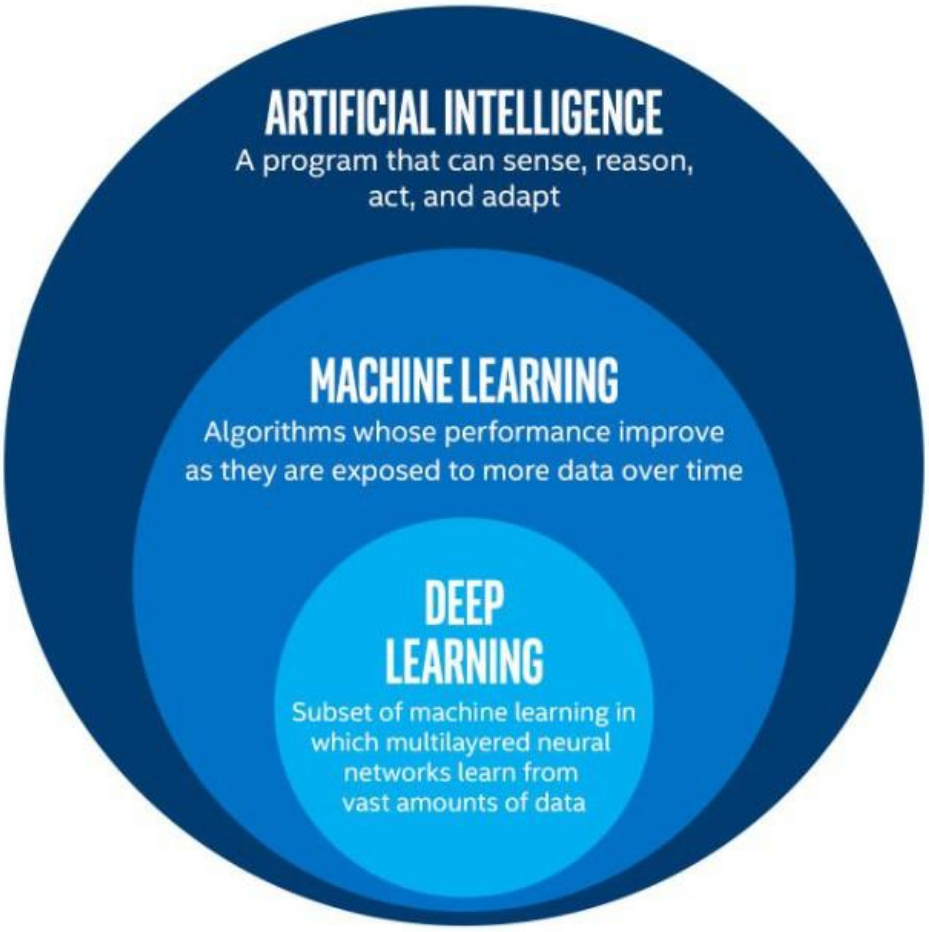
In, February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources. Already present at this stage in development were classes with inheritance, exception handling, functions, and the core data-types of list, dict, str and so on. Also in this initial release was a module system borrowed from Modula-3; Van Rossum describes the module as "one of Python's major programming units". Python's exception model also resembles Modula-3's, with the addition of an else clause. In 1994 comp.lang.python, the primary discussion forum for Python, was formed, marking a milestone in the growth of Python's userbase.

Python reached version 1.0 in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce. Van Rossum stated that "Python acquired lambda, reduce(), filter() and map().

INTRODUCTION TO AI:

Artificial intelligence (AI) is defined as intelligence exhibited by an artificial entity. Such a system is generally assumed to be a computer.

Although AI has a strong science fiction connotation, it forms a vital branch of computer science, dealing with intelligent behaviour, learning and adaptation in machines. Research in AI is concerned with producing machines to automate tasks requiring intelligent behavior. Examples include control, planning and scheduling, the ability to answer diagnostic and consumer questions, handwriting, speech, and facial recognition.



HISTORY OF AI:

The intellectual roots of AI, and the concept of intelligent machines, may be found in Greek mythology. Intelligent artifacts appear in literature since then, with real mechanical devices actually demonstrating behaviour with some degree of intelligence. After modern computers became available following World War-II, it has become possible to create programs that perform difficult intellectual tasks.

1950 - 1960:

The first working AI programs were written in 1951 to run on the Ferranti Mark I machine of the University of Manchester (UK): a draughts-playing program written by Christopher Strachey and a chess-playing program written by Dietrich Prinz.

1960 – 1970 :

During the 1960s and 1970s Marvin Minsky and Seymour Papert publish Perceptrons, demonstrating limits of simple neural nets and Alain Colmerauer developed the Prolog computer language. Ted Shortliffe demonstrated the power of rule-based systems for knowledge representation and inference in medical diagnosis and therapy in what is sometimes called the first expert system. Hans Moravec developed the first computer-controlled vehicle to autonomously negotiate cluttered obstacle courses.

1980's ONWARDS :

In the 1980s, neural networks became widely used with the back propagation algorithm, first described by Paul John Werbos in 1974. The 1990s marked major achievements in many areas of AI and demonstrations of various applications. Most notably Deep Blue, a chess-playing computer, beat Garry Kasparov in a famous six-game match in 1997.

APPLICATIONS OF PYTHON:

Game Playing :

You can buy machines that can play master level chess for a few hundred dollars. There is some AI in them, but they play well against people mainly through brute force computation--looking at hundreds of thousands of positions.

Speech Recognition :

In the 1990s, computer speech recognition reached a practical level for limited purposes. Thus United Airlines has replaced its keyboard tree for flight information by a system using speech recognition of flight numbers and city names. It is quite convenient. On the other hand, while it is possible to instruct some computers using speech, most users have gone back to the keyboard and the mouse as still more convenient.

Expert Systems :

A ``knowledge engineer'' interviews experts in a certain domain and tries to embody their knowledge in a computer program for carrying out some task. How well this works depends on whether the intellectual mechanisms required for the task are within the present state of AI. One of the first expert systems was MYCIN in 1974, which diagnosed bacterial infections of the blood and suggested treatments. It did better than medical students or practicing doctors, provided its limitations were observed.

Heuristic Classification :

One of the most feasible kinds of expert system given the present knowledge of AI is to put some information in one of a fixed set of categories using several sources of information. An example is advising whether to accept a proposed credit card purchase. Information is available about the owner of the credit card, his record of payment and also about the item he is buying and about the establishment from which he is buying it (e.g., about whether there have been previous credit card frauds at this establishment).

Tkinter Module

The tkinter package (“Tk interface”) is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, including macOS, as well as on Windows systems.

Running python -m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Tkinter supports a range of Tcl/Tk versions, built either with or without thread support. The official Python binary release bundles Tcl/Tk 8.6 threaded. See the source code for the _tkinter module for more information about supported versions.

Tkinter is not a thin wrapper, but adds a fair amount of its own logic to make the experience more pythonic. This documentation will concentrate on these additions and changes, and refer to the official Tcl/Tk documentation for details that are unchanged.

Tcl/Tk is not a single library but rather consists of a few distinct modules, each with separate functionality and its own official documentation. Python’s binary releases also ship an add-on module together with it.

Tcl

Tcl is a dynamic interpreted programming language, just like Python. Though it can be used on its own as a general-purpose programming language, it is most commonly embedded into C applications as a scripting engine or an interface to the Tk toolkit. The Tcl library has a C interface to create and manage one or more instances of a Tcl interpreter, run Tcl commands and scripts in those instances, and add custom commands implemented in either Tcl or C. Each interpreter has an event queue, and there are facilities to send events to it and process them. Unlike Python, Tcl’s execution model is designed around

cooperative multitasking, and Tkinter bridges this difference (see Threading model for details).

Tk

Tk is a Tcl package implemented in C that adds custom commands to create and manipulate GUI widgets. Each Tk object embeds its own Tcl interpreter instance with Tk loaded into it. Tk's widgets are very customizable, though at the cost of a dated appearance. Tk uses Tcl's event queue to generate and process GUI events.

Ttk

Themed Tk (Ttk) is a newer family of Tk widgets that provide a much better appearance on different platforms than many of the classic Tk widgets. Ttk is distributed as part of Tk, starting with Tk version 8.5. Python bindings are provided in a separate module, tkinter.ttk.

Internally, Tk and Ttk use facilities of the underlying operating system, i.e., Xlib on Unix/X11, Cocoa on macOS, GDI on Windows.

When your Python application uses a class in Tkinter, e.g., to create a widget, the tkinter module first assembles a Tcl/Tk command string. It passes that Tcl command string to an internal _tkinter binary module, which then calls the Tcl interpreter to evaluate it. The Tcl interpreter will then call into the Tk and/or Ttk packages, which will in turn make calls to Xlib, Cocoa, or GDI.

Important Tk Concepts

Even this simple program illustrates the following key Tk concepts:

widgets

A Tkinter user interface is made up of individual widgets. Each widget is represented as a Python object, instantiated from classes like ttk.Frame, ttk.Label, and ttk.Button.

widget hierarchy

Widgets are arranged in a hierarchy. The label and button were contained within a frame, which in turn was contained within the root window. When creating each child widget, its parent widget is passed as the first argument to the widget constructor.

configuration options

Widgets have configuration options, which modify their appearance and behavior, such as the text to display in a label or button. Different classes of widgets will have different sets of options.

geometry management

Widgets aren’t automatically added to the user interface when they are created. A geometry manager like grid controls where in the user interface they are placed.

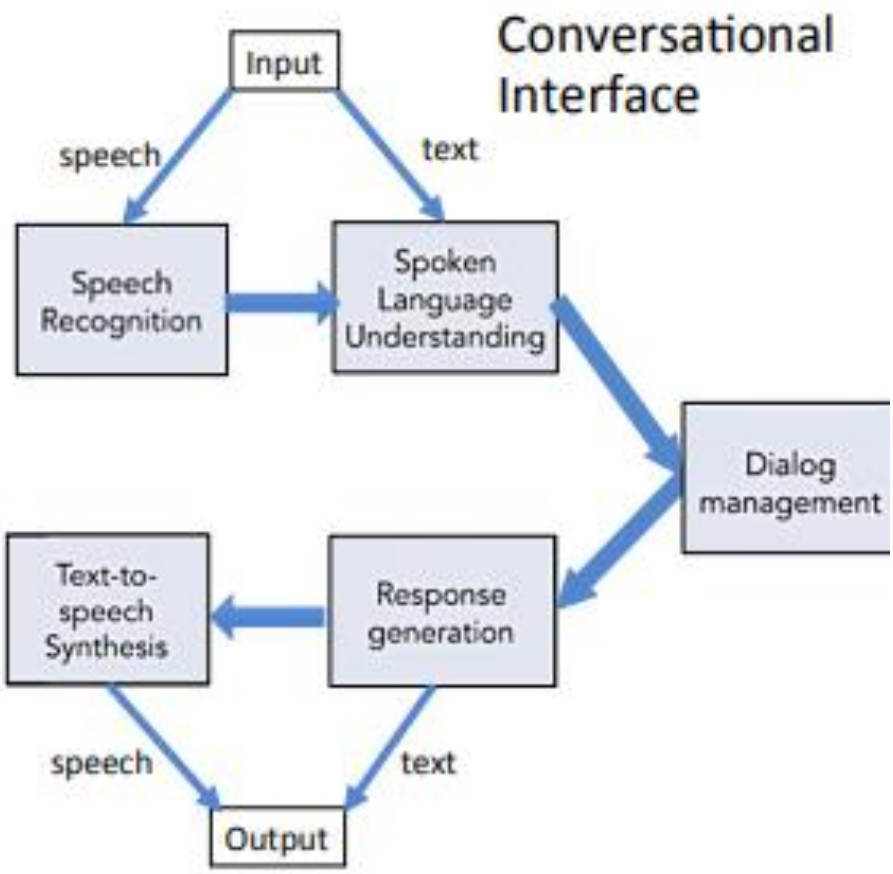
event loop

Tkinter reacts to user input, changes from your program, and even refreshes the display only when actively running an event loop. If your program isn’t running the event loop, your user interface won’t update.

Chatbot Architecture

Previously chatbots solely supported a single adjacency pair, also known as a one-shot conversation. However, modern chatbots can sustain multiple adjacency pairs, remembering states and contexts between conversations and have the capability to associate data in different adjacency pairs which is related. This is the bots ability to preserve the conversation.

A chatbot consists of four main parts: frontend, knowledge-base, back-end and corpus which is the training data. The front end is accountable for enabling communication between the bot and the user. The NLU utilises Artificial intelligence methods to identify the intent and context of the user input. An appropriate response is generated from the users' intent. The knowledge base defines the chatbots knowledge, which is created within the NLU and supported by the back-end, the back-end applies the domains corpus to produce the knowledge base



Data Flow Diagrams

DFD's have the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. A DFD is also known as "Bubble chart" has the purpose of clarifying system requirements and identifying major. Transformations that will become programs in system design.
So it's a starting point of the design phase that functionality decomposes the requirements specifications down lines. The bubbles represent data transformation and lines represent data flows in the system.

Context Diagrams:

A context diagram is a top level (also known as Level 0) data flow diagram. It only contains one process node (process 0) that generalizes the function of the entire system in relationship to external entities. The Context Diagram shows the system under consideration as a single high-level process and then shows the relationship that the system has with other external entities (systems, organizational groups, external data stores etc.). Another name for a Context Diagram is a Context-Level Data-Flow Diagram or a Level-0 Data Flow Diagram. Since a Context Diagram is a specialized version of Data-Flow Diagram, understanding a bit about Data-Flow Diagrams can be helpful.
A Data-Flow Diagram (DFD) is a graphical visualization of the movement of data through an information system. DFDs are one of the three essential components of the structured-systems analysis and design method (SSADM).

- Processes (circle)
- External Entities (rectangle)
- Data Stores (two horizontal, parallel lines or sometimes and ellipse)
- Data Flows (curved or straight line with arrowhead indicating flow direction)

DFD SYMBOLS:

In DFD there are four symbols:

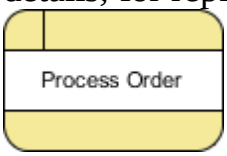
- A **SQUARE** defines the originator or the destination of the system data.
- An **ARROW** identifies the data flows in motion. It's a pipeline thru which information flows.
- A **CIRCLE** or a **BUBBLE** represents the process that transforms incoming data flow into outgoing data flow.
- An **OPEN RECTANGLE** is a data store-data at rest.

External Entity:

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process.

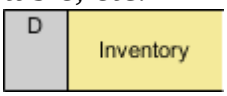
Process

A process is a business activity or function where the manipulation and transformation of data takes place. A process can be decomposed to finer level of details, for representing how data is being processed within the process.



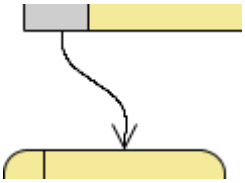
Data Store

A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database table, etc.



Data Flow

A data flow represents the flow of information, with its direction represented by an arrow head that shows at the end(s) of flow connector.



TYPES OF DFD:

a) Physical DFD
b) Logical DFD

PHYSICAL DFD VS. LOGICAL DFD :

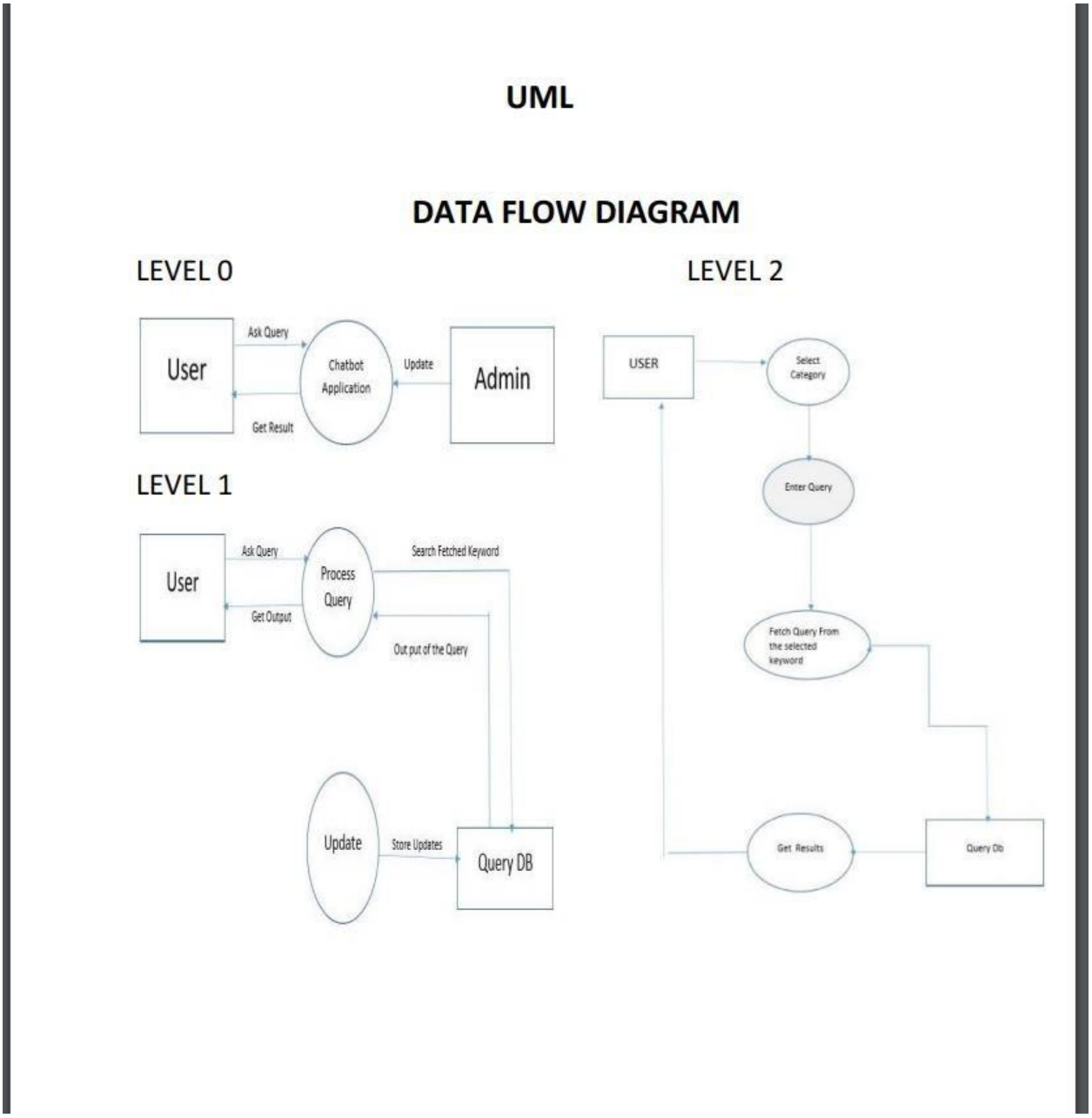
To understand the differences between a physical and logical DFD, we need to know what DFD is. A DFD stands for data flow diagram and it helps in representing graphically the flow of data in an organization, particularly its information system. A DFD enables a user to know where information comes in, where it goes inside the organization and how it finally leaves the organization. DFD does give information about whether the processing of information takes place sequentially or if it is processed in a parallel fashion. There are two types of DFD's known as physical and logical DFD. Though both serve the same purpose of representing data flow, there are some differences between the two that will be discussed in this article.

Any DFD begins with an overview DFD that describes in a nutshell the system to be designed. A logical data flow diagram, as the name indicates concentrates on the business and tells about the events that take place in a business and the data generated from each such event. A physical DFD, on the other hand is more concerned with how the flow of information is to be represented. It is a usual practice to use DFD's for representation of logical data flow and processing of data. However, it is prudent to evolve a logical DFD after first developing a physical DFD that reflects all the persons in the organization performing various operations and how data flows between all these persons.

What is the difference between Physical DFD and Logical DFD?

While there is no restraint on the developer to depict how the system is constructed in the case of logical DFD, it is necessary to show how the system has been constructed. There are certain features of logical DFD that make it popular among organizations. A logical DFD makes it easier to communicate for the employees of an organization, leads to more stable systems, allows for better understanding of the system by analysts, is flexible and easy to maintain, and allows the user to remove redundancies easily. On the other hand, a physical DFD is clear on division between manual and automated processes, gives detailed description of processes, identifies temporary data stores, and adds more efficiency.

CHATBOT DFDs:



METHODOLOGY

PLATFORMS/ TOOLS USED

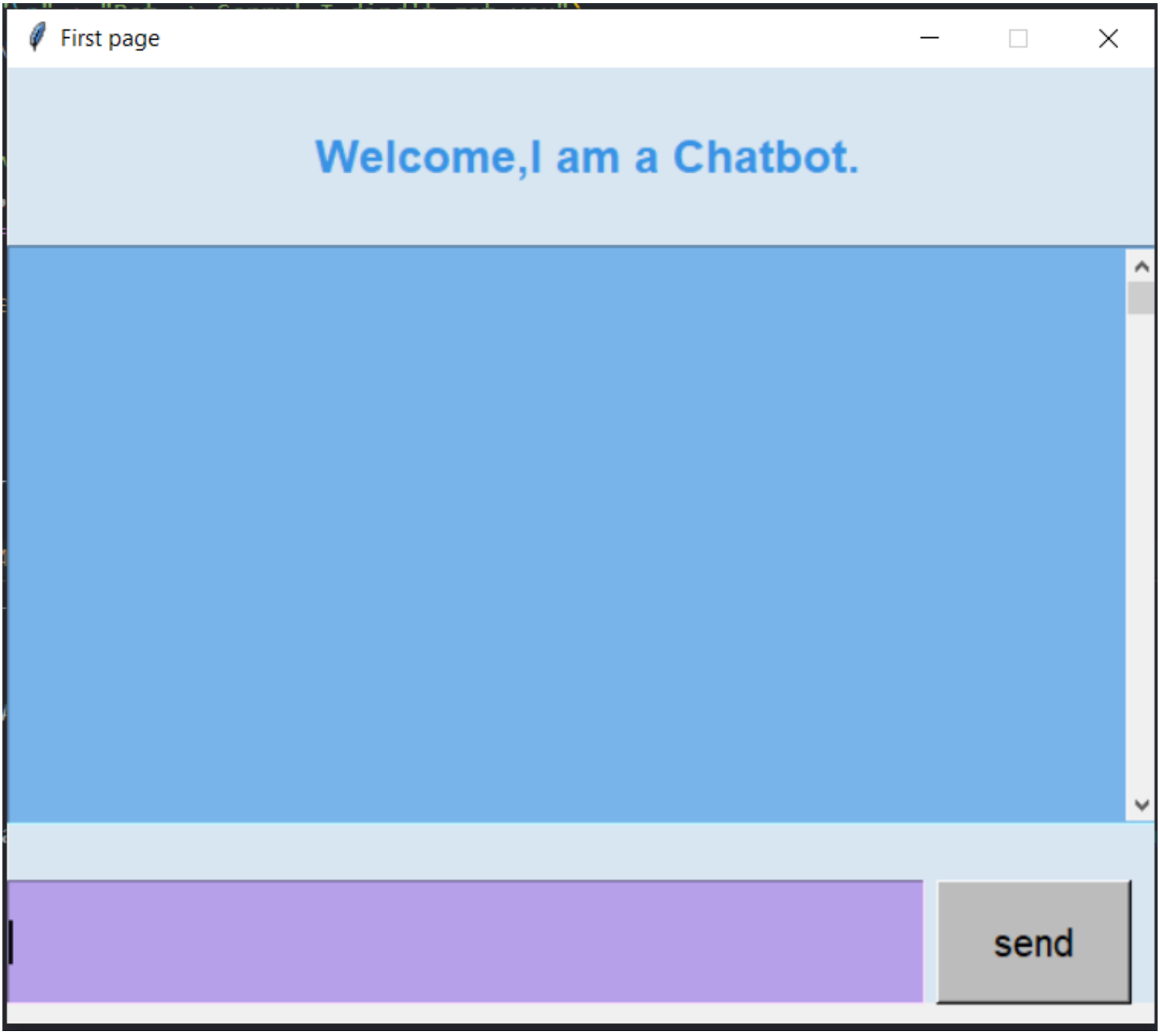
- **Hardware Platform:**

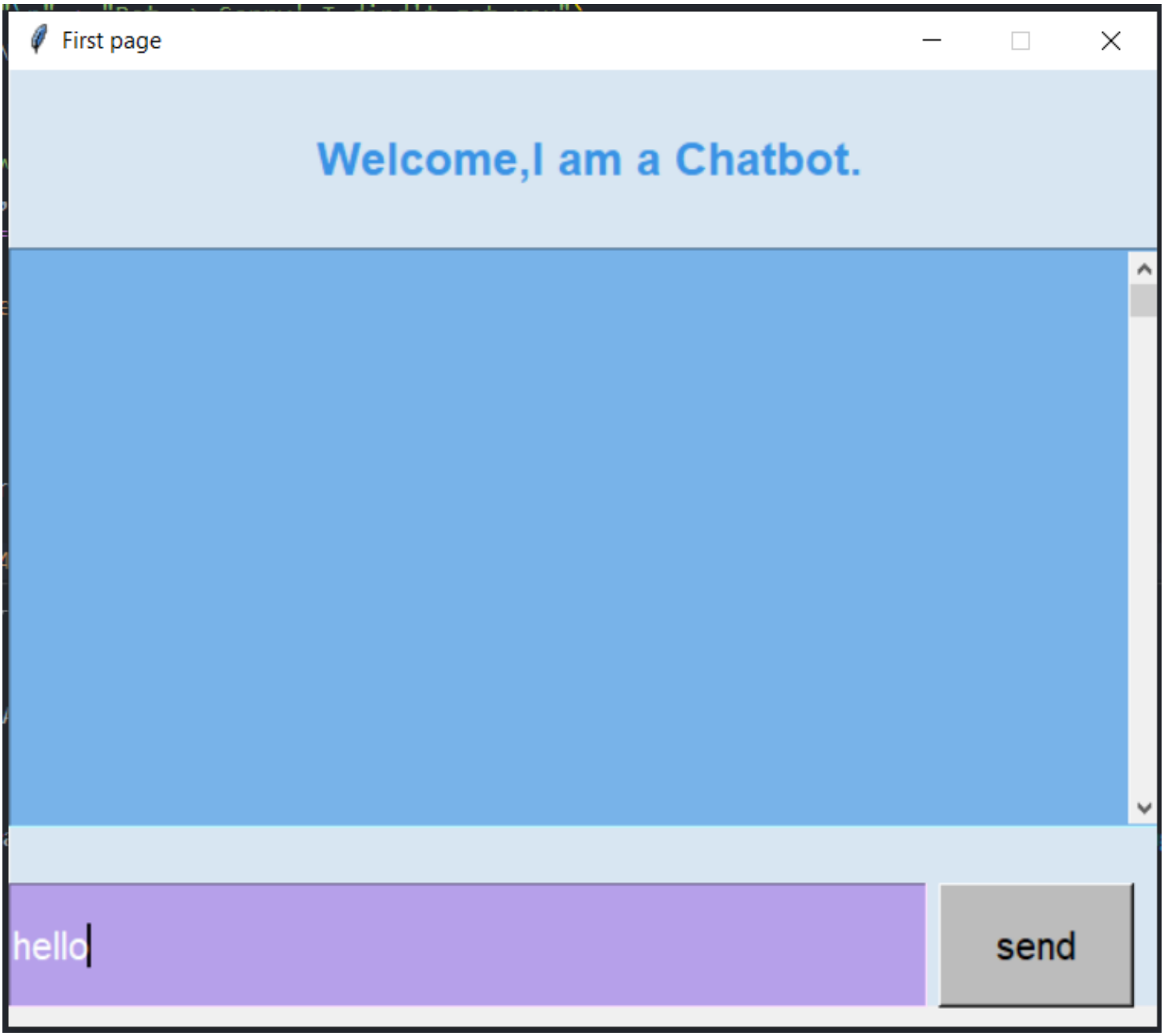
Processor:	i3 or above
Processor speed:	2.00GHz CPU
RAM:	4GB or above
Hard disk:	50GB or above
Internet Connection:	Yes
Microphone:	Optional
Speaker:	Optional
Webcam:	Optional

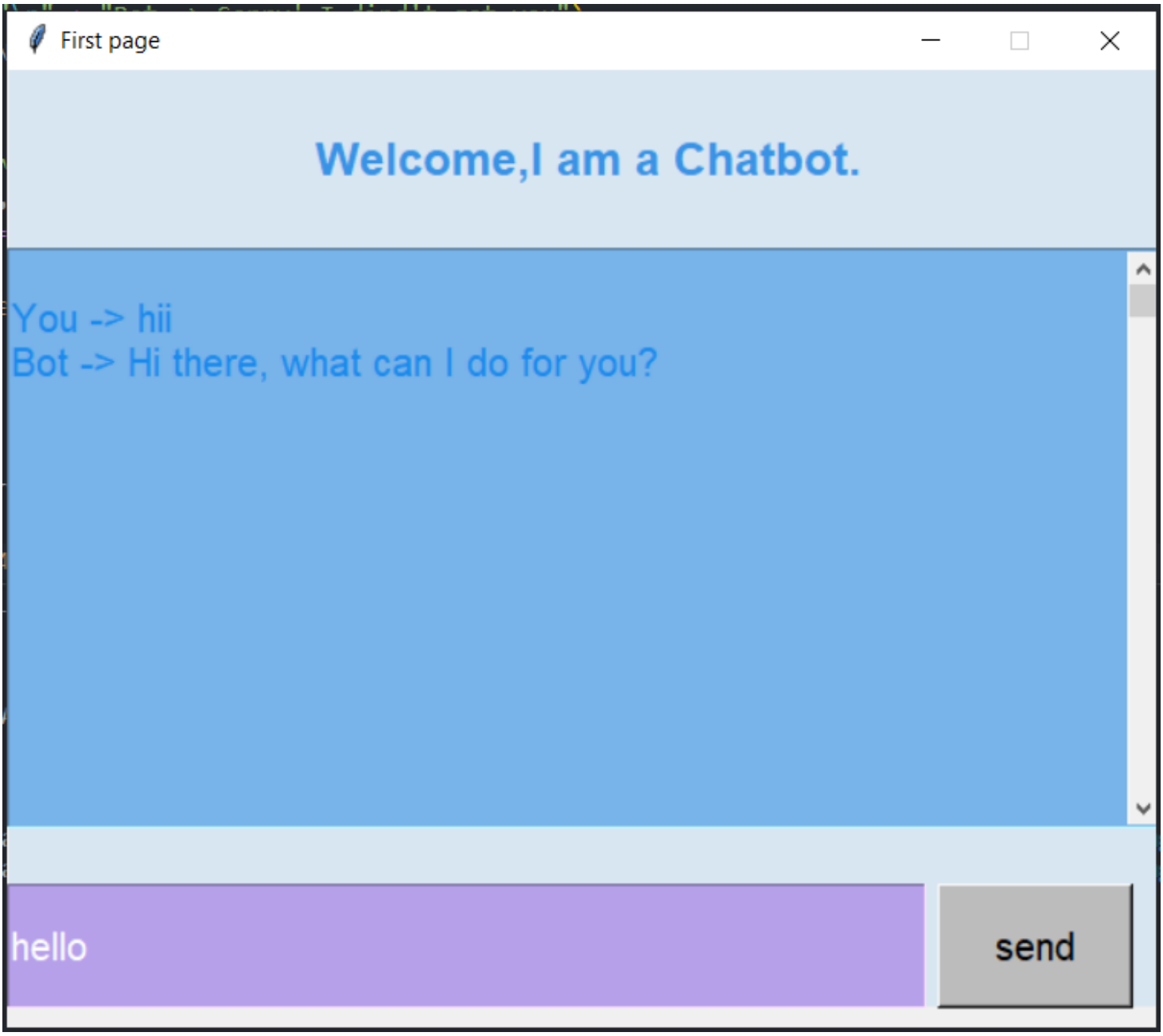
- **Software Platform:**

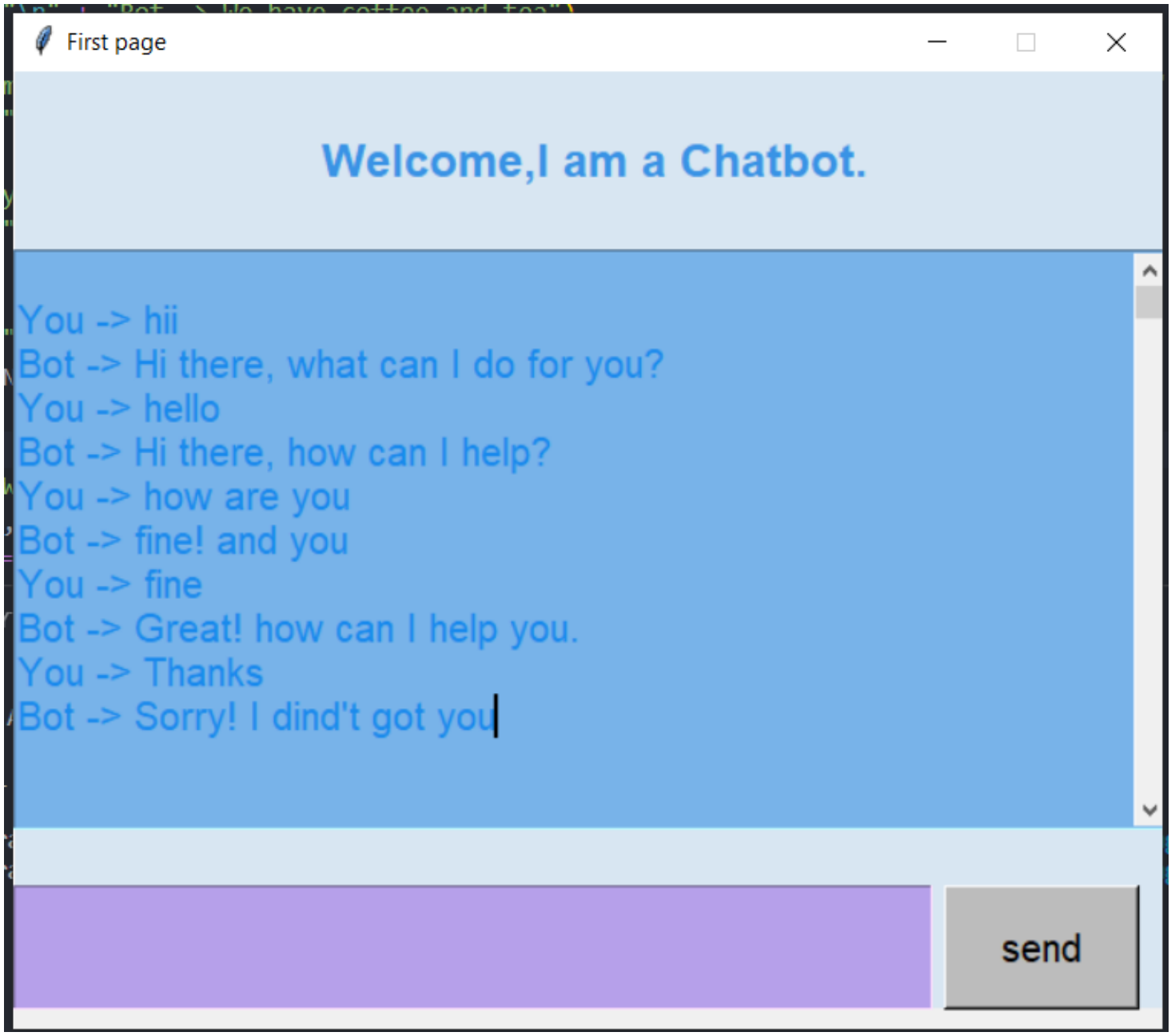
GUI:	Python Tkinter
Back End:	Python
Operation System:	Windows or any equivalent

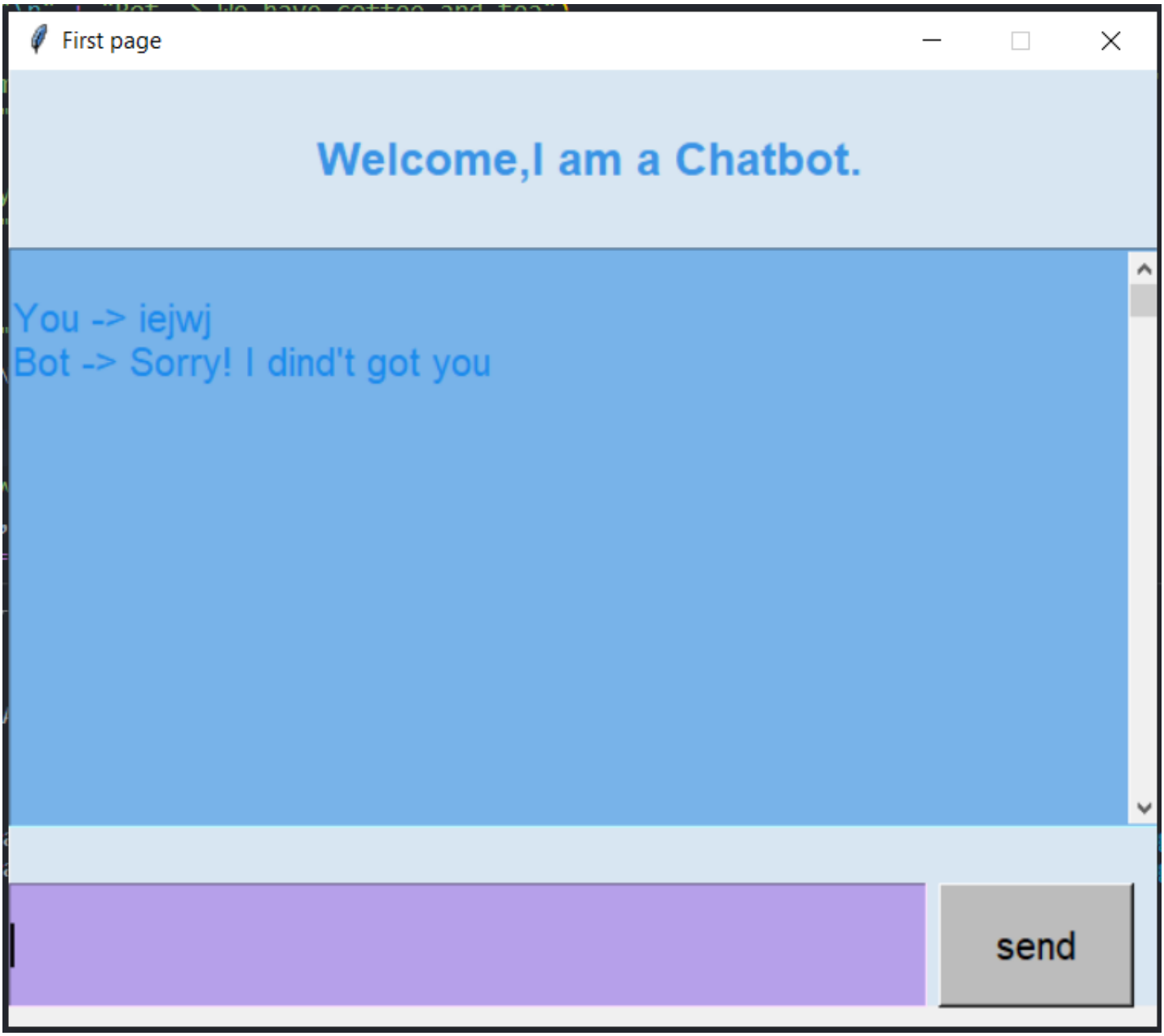
OUTPUTS




First page
Welcome,I am a Chatbot.
hello
send


First page
Welcome,I am a Chatbot.
You -> hii
Bot -> Hi there, what can I do for you?
hello
send


First page
Welcome,I am a Chatbot.
You -> hii
Bot -> Hi there, what can I do for you?
You -> hello
Bot -> Hi there, how can I help?
You -> how are you
Bot -> fine! and you
You -> fine
Bot -> Great! how can I help you.
You -> Thanks
Bot -> Sorry! I dind't got you
send


First page
Welcome,I am a Chatbot.
You -> iejwj
Bot -> Sorry! I dind't got you
send

CONCLUSION

While developing the system a conscious effort has been made to create and develop a software package, making use of available tools, techniques and resources – that would generate a proper system. While making the system, an eye has been kept on making it as user-friendly. As such one may hope that the system will be acceptable to any user and will adequately meet his/her needs. As in case of any system development process where there are a number of short comings, there have been some shortcomings in the development of this system also. There are some of the areas of improvement which couldn't be implemented due to time constraints.

BIBLIOGRAPHY:

- www.en.m.wikipedia.org
- www.geeksforgeeks.org/css-introduction/
- www.w3schools.com
- www.quackit.com/tutorial/
- www.tutorialrepublic.com
- www.developer.mozilla.org
- www.javatpoint.com/python-tutorial