

1. INTRODUCTION

1.1. General Introduction

The world is moving at a very high pace in terms of technology. We humans have aspired a lot about the system capabilities,

1.1.1. Artificial intelligence

Artificial intelligence is the intelligence display by machines unlike the natural intelligence that is demonstrated by animals including humans. the term artificial intelligence had previously been used to describe machines that mimic and display human cognitive skills that are associated with human mind such as learning and problem solving. The definition had since been rejected by major AI researchers who now describe AI in terms of rationality and asking rationally which does not limit how intelligence can be articulated.

1.1.2. Goals of Artificial Intelligence

The general problem all stimulating intelligence has been broken down into sub problems. These consists of traits or capability that researchers expect an intelligent system to display. The traits described below have received the most attention.

i. Reasoning, problem-solving

Early researchers developed algorithms that imitated step-by-step reasoning that humans use when they solve puzzles or make logical deductions. By the late 1980s and 1990s, AI research had developed methods for dealing with Uncertain or incomplete information, employing concepts from probability and economics.

ii. Knowledge representation

Knowledge representation and knowledge engineering allow AI programs to answer questions intelligently and make deductions about real-world facts.

A representation of "what exists" is an ontology: the set of objects, relations, concepts, and properties formally described so that software agents can interpret them. The most general ontologies are called upper ontologies, which attempt to provide a foundation for all other knowledge and act as mediators between domain ontologies that cover specific knowledge about a particular knowledge domain (field of interest or area of concern). A truly intelligent program would also need access to common sense knowledge; the set of facts that an average person knows. The semantics of an ontology is typically represented in description logic, such as the Web Ontology Language.

iii. Planning

An intelligent agent that can plan makes a representation of the state of the world, makes predictions about how their actions will change it and make choices that maximize the utility (or "value") of the available choices. In classical planning problems, the agent can assume that it is the only system acting in the world, allowing the agent to be certain of the consequences of its actions. However, if the agent is not the only actor, then it requires that the agent reason under uncertainty, and continuously re-assess its environment and adapt. Multi-agent planning uses the cooperation and competition of many agents to achieve a given goal. Emergent behaviour such as this is used by evolutionary algorithms and swarm intelligence.

iv. Learning

Machine learning (ML), a fundamental concept of AI research since the field's inception, is the study of computer algorithms that improve automatically through experience.

Unsupervised learning finds patterns in a stream of input. Supervised learning requires a human to label the input data first, and comes in two main varieties: classification and numerical regression. Classification is used to determine what category something belongs in—the program sees a number of examples of things from several categories and will learn to classify new inputs. Regression is the attempt to produce a function that describes the relationship between inputs and outputs and predicts how the outputs should change as the inputs change. Both classifiers and regression learners can be viewed as "function approximators" trying to learn an unknown (possibly implicit) function; for example, a spam classifier can be viewed as learning a function that maps from the text of an email to one of two categories, "spam" or "not spam". In reinforcement learning the agent is rewarded for good responses and punished for bad ones. The agent classifies its responses to form a strategy for operating in its problem space. Transfer learning is when the knowledge gained from one problem is applied to a new problem. Computational learning theory can assess learners by computational complexity, by sample complexity (how much data is required), or by other notions of optimization.

v. Natural language Processing

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of NLP include information retrieval, question answering and machine translation.

Symbolic AI used formal syntax to translate the deep structure of sentences into logic. This failed to produce useful applications, due to the intractability of logic and the breadth of common sense knowledge. Modern statistical techniques include co-occurrence frequencies (how often one word appears near another), "Keyword spotting" (searching for a particular word to retrieve information), transformer-based deep learning (which finds patterns in text), and others. They have achieved acceptable accuracy at the page or paragraph level, and, by 2019, could generate coherent text.

vi. Perception

Machine perception is the ability to use input from sensors (such as cameras, microphones, wireless signals, and active lidar, sonar, radar, and tactile sensors) to deduce aspects of the world. Applications include speech recognition, facial recognition, and object recognition. Computer vision is the ability to analyse visual input.

vii. Motion and manipulation

AI is heavily used in robotics. Localization is how a robot knows its location and maps its environment. When given a small, static, and visible environment, this is easy; however, dynamic environments, such as (in endoscopy) the interior of a patient's breathing body, pose a greater challenge.

Motion planning is the process of breaking down a movement task into "primitives" such as individual joint movements. Such movement often involves compliant motion, a process where movement requires maintaining physical contact with an object. Robots can learn from experience how to move efficiently despite the presence of friction and gear slippage.

viii. Social Intelligence

Affective computing is an interdisciplinary umbrella that comprises systems that recognize, interpret, process or simulate human feeling, emotion and mood. For

example, some virtual assistants are programmed to speak conversationally or even to banter humorously; it makes them appear more sensitive to the emotional dynamics of human interaction, or to otherwise facilitate human–computer interaction. However, this tends to give naïve users an unrealistic conception of how intelligent existing computer agents actually are.

Moderate successes related to affective computing include textual sentiment analysis and, more recently, multimodal sentiment analysis), wherein AI classifies the affects displayed by a videotaped subject.

ix. General Intelligence

A machine with general intelligence can solve a wide variety of problems with breadth and versatility similar to human intelligence. There are several competing ideas about how to develop artificial general intelligence. Hans Moravec and Marvin Minsky argue that work in different individual domains can be incorporated into an advanced multi-agent system or cognitive architecture with general intelligence. Pedro Domingos hopes that there is a conceptually straightforward, but mathematically difficult, "master algorithm" that could lead to AGI. Others believe that anthropomorphic features like an artificial brain or simulated child development will someday reach a critical point where general intelligence emerges.

1.1.3. Types of Artificial Intelligence

Artificial intelligence is broadly divided into two categories viz., AI type-1 and AI type-2. The AI type-1 categorisation is based on the capabilities of AI whereas the AI type-2 categorisation is based on the functionalities of the AI system.

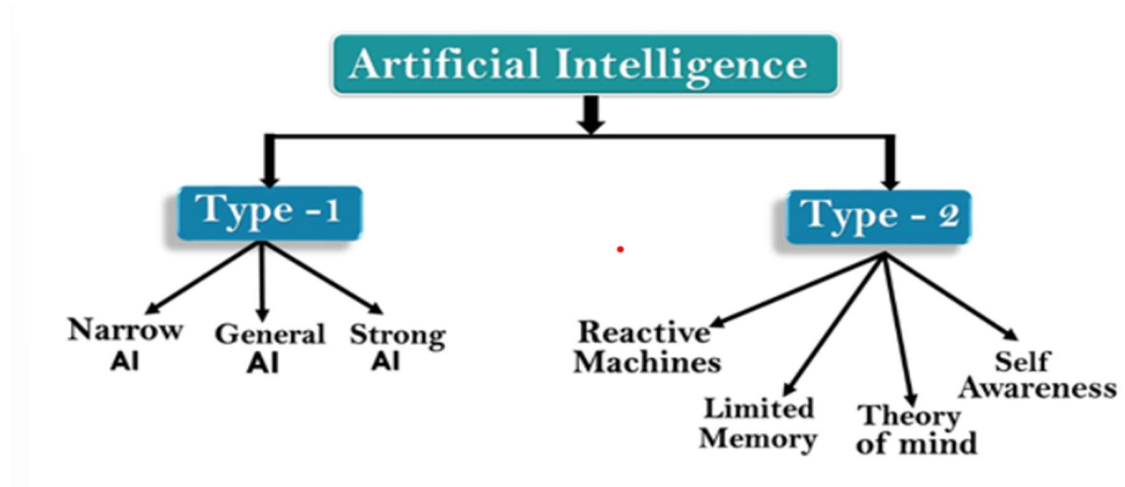


Figure 1.1.3

1.1.3.1. AI type-1: Based on Capabilities

AI type-1 are divided on the basis of capabilities, taking into considerations the tasks or the number of tasks an AI can perform efficiently. Based on this categorisation they are divided onto three sub-categories.

i. Weak AI or Narrow AI:

Weak AI or narrow AI is a type of AI that can perform a dedicated task with intelligence. It is one of the most common and widely used in the world of Artificial Intelligence.

A Narrow AI cannot perform any task that lies beyond its field or limitations, it is only trained for one specific task. Therefore, it is termed as Weak AI.

Apple Siri, Google assistant, Microsoft Cortana and Amazon Alexa are some of the most common examples of Weak AIs those work with a limited pre-defined range of functions.

Some examples of Narrow AI are chess playing, purchasing suggestions on e-commerce websites, self-driving cars, speech recognition and image recognition.

ii. General AI

General AI is capable of performing any intellectual task with efficiency like a human. The idea behind the General AI is to make such a system which could be smarter and think like a human by its own.

Currently, there is no such system or programme capable enough to mimic human thinking capabilities. The researchers are focused on developing machines with General AI, and it will take lot of efforts and time to develop such systems.

iii. Super AI

Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence and can perform any task better than human with cognitive properties. It is an outcome of general AI. Some key characteristics of strong AI include capability include the ability to think, to reason, solve the puzzle, make judgments, plan, learn, and communicate by its own. Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.

1.1.3.2. AI Type-2: Based on functionality

i. Reactive machines

Purely reactive machines are the most basic types of Artificial Intelligence. Such AI systems do not store memories or past experiences for future actions. These machines only focus on current scenarios and react on it as per possible best action. IBM's Deep Blue system is an example of reactive machines. Google's AlphaGo is also an example of reactive machines.

ii. Limited memory

Limited memory machines can store past experiences or some data for a short period of time. These machines can use stored data for a limited time period only. Self-driving cars are one of the best examples of Limited Memory

systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road.

iii. Theory of Mind

Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.

This type of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.

iv. Self-Awareness

Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.

These machines will be smarter than human mind.

Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

1.2. Project overview

Auxiliator: One's personal assistant, is a reactive artificial intelligence programmed using python and its multiple libraries. This program works on the principles of Natural language processing, speech recognition, speech to text and text to speech. The main motive behind developing this project was to develop a reaction based artificial intelligence system that is capable of taking user inputs in the form of voice commands, process and understand the voice command and react accordingly to the user either by executing tasks or by giving a vocal response.

In this project the speech recognition is used, speech recognition, or speech-to-text, is the ability of a machine or program to identify words spoken aloud and convert them into readable text. Rudimentary speech recognition software has a limited vocabulary and may only identify words and phrases when spoken clearly. More sophisticated software can handle natural speech, different accents and various languages. Speech recognition uses a broad array of research in computer science, linguistics and computer engineering. Many modern devices and text-focused programs have speech recognition functions in them to allow for easier or hands-free use of a device.

Natural language processing (NLP) is the ability of a computer program to understand human language as it is spoken and written -- referred to as natural language. It is a component of artificial intelligence (AI). NLP enables computers to understand natural language as humans do. Whether the language is spoken or written, natural language processing uses artificial intelligence to take real-world input, process it, and make sense of it in a way a computer can understand. Just as humans have different sensors -- such as ears to hear and eyes to see -- computers have programs to read and microphones to collect audio. And just as humans have a brain to process that input, computers have a program to process their respective inputs. At some point in processing, the input is converted to code that the computer can understand.

1.3. Structure

Front end: A terminal based executable file

Backend: Python

The program is completely structured on the Python programming language and the package contains a set of other system files that support the execution of the application.

1.4. Software Requirements

- Operating system: Microsoft Windows 7 or above
- Python Version: Python 9 or above

- Libraries required: Pyaudio, Flask, Pywhatkit, Speech Recognition, Win wi-fi.

1.5. Hardware Requirements

- A working Laptop or Personal computer
- A well-integrated or peripheral microphone
- A speaker system
- Internet Connectivity

1.6. Feasibility study

As the name implies, a feasibility analysis is used to determine the viability of an idea, such as ensuring a project is legally and technically feasible as well as economically justifiable. It tells us whether a project is worth the investment—in some cases, a project may not be doable. There can be many reasons for this, including requiring too many resources, which not only prevents those resources from performing other tasks but also may cost more than an organization would earn back by taking on a project that isn't profitable.

Feasibility consideration: In our feasibility, we have considered the three main aspects of feasibility study viz., Technical Feasibility, Operational Feasibility and Economic Feasibility.

1.6.1. Technical Feasibility

This assessment focuses on the technical resources available. It helps to determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. Our proposed system works on Python and all of its technical requirements are met in nearly all the computer systems available in the market. Moreover, the modules and the libraries required can be easily downloaded from the internet free of cost and does not require any special configuration.

1.6.2. Economic Feasibility

This assessment typically involves a cost/ benefits analysis of the project, helps to determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility—helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide. The total development cost of our proposed project is free of cost. All the softwares used namely Pycharm community edition, Visual Studio Code as well as all the libraries and the files included or imported are open source and cost-free.

1.6.3. Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether—and how well—the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

2. LITERATURE SURVEY

2.1. Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

2.1.1. History of Python

Python was conceived in the late 1980s[41] by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL,[42] capable of exception handling and interfacing with the Amoeba operating system.

Python programming language is being updated regularly with new features and supports. There are lots of update in Python versions, started from 1994 to current release.

Python Timeline

- Python 1.0 – January 1994
- Python 1.5 – December 1997
- Python 1.6 – September 2000
- Python 2.0 – October 2000
- Python 2.1 – April 2001
- Python 2.2 – December 2001
- Python 2.3 – July 2003
- Python 2.4 – November 2004
- Python 2.5 – September 2006
- Python 2.6 – October 2008
- Python 2.7 –July 2010
- Python 3.0 – December 2008
- Python 3.1 – June 2009

- Python 3.2 –February 2011
- Python 3.3 – September 2012
- Python 3.4 – March 2014
- Python 3.5 – September 2015
- Python 3.6 – December 2016
- Python 3.7 – June 2018
- Python 3.8 – October 2019

Later versions of python were also released in the following years and now the most recent version is Python 3.10.4. is being used.

2.2. Speech Recognition

Speech recognition is an interdisciplinary subfield of computer science and computational linguistics that develops methodologies and technologies that enable the recognition and translation of spoken language into text by computers with the main benefit of searchability. It is also known as automatic speech recognition (ASR), computer speech recognition or speech to text (STT). It incorporates knowledge and research in the computer science, linguistics and computer engineering fields. The reverse process is speech synthesis.

Some speech recognition systems require "training" (also called "enrollment") where an individual speaker reads text or isolated vocabulary into the system. The system analyzes the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased accuracy. Systems that do not use training are called "speaker-independent" systems. Systems that use training are called "speaker dependent".

Speech recognition applications include voice user interfaces such as voice dialing (e.g. "call home"), call routing (e.g. "I would like to make a collect call"), domotic appliance control, search key words (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured

documents (e.g. a radiology report), determining speaker characteristics, speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed direct voice input).

The term voice recognition or speaker identification refers to identifying the speaker, rather than what they are saying. Recognizing the speaker can simplify the task of translating speech in systems that have been trained on a specific person's voice or it can be used to authenticate or verify the identity of a speaker as part of a security process.

From the technology perspective, speech recognition has a long history with several waves of major innovations. Most recently, the field has benefited from advances in deep learning and big data. The advances are evidenced not only by the surge of academic papers published in the field, but more importantly by the worldwide industry adoption of a variety of deep learning methods in designing and deploying speech recognition systems.

2.3. Natural language Processing (NLP)

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modelling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. There’s a good chance you’ve interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots,

and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

2.3.1. Natural Language Processing Tasks

Human language is filled with ambiguities that make it incredibly difficult to write software that accurately determines the intended meaning of text or voice data. Homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, variations in sentence structure—these just a few of the irregularities of human language that take humans years to learn, but that programmers must teach natural language-driven applications to recognize and understand accurately from the start, if those applications are going to be useful.

Several NLP tasks break down human text and voice data in ways that help the computer make sense of what it's ingesting. Some of these tasks include the following:

i. Speech recognition

Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.

ii. Parts of speech tagging

Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies 'make' as a verb in 'I can make a paper plane,' and as a noun in 'What make of car do you own?'

iii. Word sense disambiguation

Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).

iv. Named entity recognition

Named entity recognition, or NEM, identifies words or phrases as useful entities. NEM identifies 'Kentucky' as a location or 'Fred' as a man's name.

v. Co-reference resolution

Co-reference resolution is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).

vi. Sentient analysis

Sentiment analysis attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.

vii. Natural language generation

Natural language generation is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

2.3.2. Natural Language Processing Applications

Natural language processing is the driving force behind machine intelligence in many modern real-world applications. Here are a few examples:

- i. Spam detection: You may not think of spam detection as an NLP solution, but the best spam detection technologies use NLP's text classification capabilities to scan emails for language that often indicates spam or phishing. These indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names, and more. Spam detection is one of a handful of NLP problems that experts consider 'mostly solved' (although you may argue that this doesn't match your email experience).
- ii. Machine translation: Google Translate is an example of widely available NLP technology at work. Truly useful machine translation involves more than replacing words in one language with words of another. Effective translation has to capture accurately the meaning and tone of the input language and translate it to text with the same meaning and desired impact in the output language. Machine translation tools are making good progress in terms of accuracy. A great way to test any machine translation tool is to translate text to one language and then back to the original. An oft-cited classic example: Not long ago, translating "The spirit is willing but the flesh is weak" from English to Russian and back yielded "The vodka is good but the meat is rotten." Today, the result is "The spirit desires, but the flesh is weak," which isn't perfect, but inspires much more confidence in the English-to-Russian translation.
- iii. Virtual agents and chatbots: Virtual agents such as Apple's Siri and Amazon's Alexa use speech recognition to recognize patterns in voice commands and natural language generation to respond with appropriate action or helpful comments. Chatbots perform the same magic in response to typed text entries. The best of these also learn to recognize contextual clues about human requests and use them to provide even better responses or options over time. The next enhancement for these applications is question answering, the ability to respond to

our questions—anticipated or not—with relevant and helpful answers in their own words.

- iv. Social media sentiment analysis: NLP has become an essential business tool for uncovering hidden data insights from social media channels. Sentiment analysis can analyse language used in social media posts, responses, reviews, and more to extract attitudes and emotions in response to products, promotions, and events—information companies can use in product designs, advertising campaigns, and more.
- v. Text summarization: Text summarization uses NLP techniques to digest huge volumes of digital text and create summaries and synopses for indexes, research databases, or busy readers who don't have time to read full text. The best text summarization applications use semantic reasoning and natural language generation (NLG) to add useful context and conclusions to summaries.

2.4. Previous works

Kumaran N., Rangaraj V., Siva Sharan S., Dhanalakshmi R. (2020), The proposed system executes commands given by the user. Thus, it highly depends on just the voice commands given by the user to complete his job. Designing a voice-controlled computer interface not only makes the execution of a command easier, but it also helps the disabled individuals to control a computer. Hand-free computing is possible where a user can interact with the computer without the use of their hands. Speech recognition can be trained to recognize various voice commands. Disabled persons may find the hand-free computer is vital in their life. This system is designed to recognize the speech and execute with full capability Synthesizing means it converts text to speech. The user is asked to provide voice command by using a microphone. The microphone shall take the speech as a command, and the analogue signals are converted to digital in the internal circuits. Then digital signals are processed as an acoustic model. Once the particular applications are identified by the system, it then opens the application. When the

application is found, the system shall prompt the user to create a new application in the current working directory. The system would ask for various operation such as edit, read, paste, copy, and other similar operations that shall be performed inside it once the application opens. The system uses the ferment synthesis, a type of speech synthesizer for responding to the user through voice command.

Iannizzotto Giancarlo, Bello Lucia Lo, Nucita Andrea, Grasso Giorgio Mario (2018),

Recent developments in smart assistants and smart home automation are lately attracting the interest and curiosity of consumers and researchers. Speech enabled virtual assistants (often named smart speakers) offer a wide variety of network oriented services and, in some cases, can connect to smart environments, thus enhancing them with new and effective user interfaces. However, such devices also reveal new needs and some weaknesses. In particular, they represent faceless and blind assistants, unable to show a face, and therefore an emotion, and unable to 'see' the user. As a consequence, the interaction is impaired and, in some cases, ineffective. Moreover, most of those devices heavily rely on cloud-based services, thus transmitting potentially sensitive data to remote servers. To overcome such issues, in this paper we combine some of the most advanced techniques in computer vision, deep learning, speech generation and recognition, and artificial intelligence, into a virtual assistant architecture for smart home automation systems. The proposed assistant is effective and resource-efficient, interactive and customizable, and the realized prototype runs on a low-cost, small sized, Raspberry PI 3 device. For testing purposes, the system was integrated with an open-source home automation environment and ran for several days, while people were encouraged to interact with it, and proved to be accurate, reliable and appealing.

3. PROPOSED METHODOLOGY

3.1. Problem Definition

There are many intelligent personal assistants namely “Siri”, “Cortana” or “OK google” but all of them have certain limitations like browsing local directories, opening and closing local applications on the basis of voice commands.

The proposed system will give the user access to its personal assistant by voice command, it will let the user to access his local files and applications, perform tasks like Wikipedia search, Google search, WhatsApp message sending, email generations and delivery over the internet.

3.2. Project Objectives

Auxiliartor fulfill the following objectives: -

- Let the user access the local files saved in the system.
- Let the user access the tools or open the third-party app just by voice command.
- Let the user send mail as well as WhatsApp messages vocally.
- Enables the user to browse, search on Wikipedia or surf over YouTube.
- Can play local songs, video or open local pictures.
- It updates the user with the latest news
- Can open user defined directories
- Hands free shutdown

3.3. Methodology Used

Spiral Model is a risk-driven software development process model. It is a combination of waterfall model and iterative model. Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.

Each phase of spiral model in software engineering begins with a design goal and ends with the client reviewing the progress. The spiral model in software engineering was first mentioned by Barry Boehm in his 1986 paper.

The development process in Spiral model in SDLC, starts with a small set of requirement and goes through each development phase for those set of requirements. The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase.

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are: -

1. Objective's determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

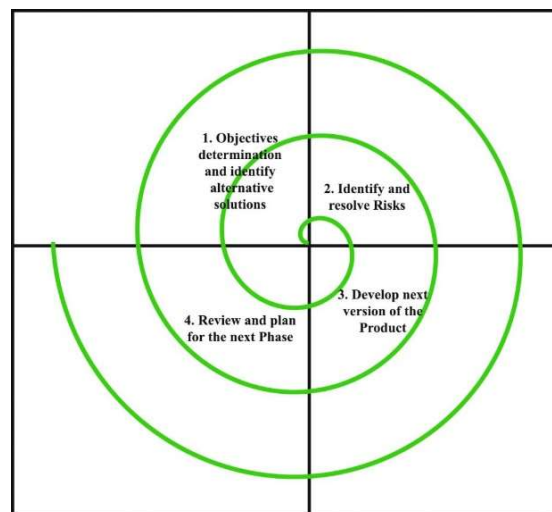


Figure 3.3

3.3.1. Reason to choose spiral model

- Highly flexible model
- Fast and cost-effective development
- Well-suited for large scale projects and mission-critical developments
- Works well for complex projects
- Monitoring is easy and effective
- Strong emphasis on client approval
- Focus on documentation control
- Potential for additional post-project functionality
- Software is produced early on in the project lifecycle
- Risk analysis helps to eliminate and avoid risk
- Changed requirements are accommodated during the project lifespan
- The end product can be highly customized

3.3.2. Phases of Spiral Model

- **Objective's determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. It includes estimating the cost, schedule and resources for the iteration. It also involves understanding the system requirements for continuous communication between the system analyst and the customer. Then alternative solutions possible for the phase are proposed in this phase.
- **Identify and resolve Risks:** During the second phase, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

- **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified. It includes testing, coding and deploying software at the customer site. At the end of the third phase, the next version of the software is available.
- **Review and plan for the next Phase:** In the evaluation phase, the current project output is reviewed and evaluated before the project moves on to the next spiral. Any critical issues will be identified here, and the necessary steps will be taken to deal with them.

3.4. Implementation

3.4.1. Module Description

- **Pytttsx3:** pytttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the pytttsx3.init() factory function to get a reference to a pytttsx3.
- **Datetime:** It's a combination of date and time along with the attribute's year, month, day, hour, minute, second, microsecond, and tzinfo.
- **Speech_Recognition:** Speech Recognition Module is a compact easy control speaking recognition board. It is a speaker-dependent module and supports up to 80 voice commands. Any sound could be trained as command. Users need to train the module first before recognizing any voice command.
- **Wikipedia:** Wikipedia is a multilingual online encyclopedia created and maintained as an open collaboration project by a community of volunteer editors using a wiki-based editing system. Python provides the Wikipedia module (or API) to scrap the data from the Wikipedia pages. This module allows us to get and parse the information from Wikipedia.

- **OS:** The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.
- **Random:** Python Random module is an in-built module of Python which is used to generate random numbers. These are pseudo-random numbers means these are not truly random. This module can be used to perform random actions such as generating random numbers, print random a value for a list or string, etc.
- **Web browser:** The web browser module provides a high-level interface to allow displaying web-based documents to users. Under most circumstances, simply calling the `open ()` function from this module will do the right thing.
- **Smtplib:** Python provides a smtplib module, which defines the SMTP client session object used to send emails to an internet machine. For this purpose, we have to import the smtplib module using the import statement. The SMTP object is used for the email transfer. The following syntax is used to create the smtplib object.
- **JSON:** JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network. It's used by lots of APIs and Databases, and it's easy for both humans and machines to read. JSON represents objects as name/value pairs, just like a Python dictionary.
- **Win wi-fi:** A Wi-Fi CLI tool for Windows. It allows you to scan Wi-Fi Access Points without being an admin to disable and enable the Wi-Fi interface.
- **Requests:** Requests library is one of the integral parts of Python for making HTTP requests to a specified URL. Whether it be REST APIs or Web Scrapping, requests is must to be learned for proceeding further with these technologies. When one makes a request to a URI, it returns a response. Python requests provides inbuilt functionalities for managing both the request and response.

- **Pyjokes:** Python supports creation of random jokes using one of its libraries. Let us explore it a little more, Pyjokes is a python library that is used to create one-line jokes for programmers. Informally, it can also be referred as a fun python library which is simple to use.
- **Pywhatkit:** Python offers numerous inbuilt libraries to ease our work. Among them pywhatkit is a Python library for sending WhatsApp messages at a certain time, it has several features like send WhatsApp messages, play a YouTube video, perform a google search, get information on a particular topic etc. The pywhatkit module can also be used for converting text into handwritten text images.

3.4.2. Codes Snapshots and Procedure

3.4.2.1. Procedure

i. Installing all the important modules and packages

- a. Firstly, we have to download the **pyaudio** library from the internet and install it on the command prompt using the python package installer.

Great caution should be taken while downloading the pyaudio library as the cp version and build architecture of the pyaudio file should match with the installed python version and the architecture.

```
C:\Users\khali\Downloads>pip install PyAudio-0.2.11-cp310-cp310-win_amd64.whl
```

Figure 3.4.2.1 (a)

- b. Installing the pytsx3 on the system using the pip installer.

```
pip install pytsx3
```

Figure 3.4.2.1 (b)

- c. Installing the speech recognition package.

```
pip install speechrecognition_
```

Figure 3.4.2.1 (c)

- d. Installing the flask package.

```
pip install flask_
```

Figure 3.4.2.1 (d)

- e. Installing the pywhatkit package on the system using pip installer.

```
pip install pywhatkit_
```

Figure 3.4.2.1 (e)

- f. Installing winwifi, pyjokes and Wikipedia using pip installer.

```
pip install pyjokes_
```

```
pip install winwifi
```

```
pip install wikipedia _
```

Figure 3.4.2.1 (f)

ii. Importing the modules

All the installed modules other than the pyaudio should be imported in the main file. These include some of the pre-installed packages like os, datetime, random etc.

```
import pyttsx3
import datetime
import speech_recognition as sr
import wikipedia
import os
import random
import webbrowser
import smtplib
import json
import re
import time
import winwifi
import requests
import pyjokes
from word2number import w2n
import pywhatkit as kit
from functools import lru_cache
```

Figure 3.4.2.1 (g)

iii. Setting up the pyttsx3 engine and configuring it.

```
engine = pyttsx3.init("sapi5")
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
engine.setProperty('rate', 120)
di = {"Abdullah": "khalidabdullah512@gmail.com",
      "user": "userdependend000@gmail.com",
      "aryaman": "aryamanyugveer@gmail.com"}
song = {}
contacts = {"Aryaman": "+917033592833",
            "harry": "+917491843847"
            }
```

Figure 3.4.2.1 (h)

iv. Setting up the speak and command functions.

```
def speak(audio):  
    engine.say(audio)  
    engine.runAndWait()
```

```
def command():  
    r = sr.Recognizer()  
    with sr.Microphone() as source:  
        print("listening...")  
        r.pause_threshold = 1  
        r.energy_threshold = 1000  
        audio = r.listen(source)  
    try:  
        print("Recognizing...")  
        query = r.recognize_google(audio, language="en-in")  
        print(f"u said {query}")  
    except Exception as e:  
        print("Say it again")  
        return "none"  
    return query
```

Figure 3.4.2.1 (i)

3.4.2.2. Code

```
import pyttsx3
import datetime
import speech_recognition as sr
import wikipedia
import os
import random
import webbrowser
import smtplib
import json
import re
import time
import winwifi
import requests
import pyjokes
from word2number import w2n
import pywhatkit as kit
from functools import lru_cache

engine = pyttsx3.init("sapi5")
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
engine.setProperty('rate', 120)
```

```
di={"aman":"amanshukla5614@gmail.com","user":"khalidabdullah512@gmail.com"}
```

```
doc = {}
```

```
song = {}
```

```
op = {}
```

```
video = {}
```

```
contacts = {"Aryaman": "+917033592833",  
            "harry": "+917491843847"  
            }
```

```
def speak(audio):  
    engine.say(audio)  
    engine.runAndWait()
```

```
def ffile(file):  
    for i in range(0, len(file)):  
        file[i] = file[i].replace("\n", "")  
        file[i] = file[i].replace("\\"", "\"")  
        file[i] = file[i].replace("\\"", "\"")
```

```
f = open("media.txt", "r")
file = f.readlines()
f.close()
if (len(file) == 0):
    s = open("media.txt", "a")
    print("hello")
    a = input("enter the music location")
    s.write(f"{a}\n")
    a = input("enter the pictures location")
    s.write(f"{a}\n")
    a = input("enter the document location")
    s.write(f"{a}\n")
    a = input("enter the video location")
    s.write(f"{a}\n")
    a = input("enter the wifi name")
    s.write(f"{a}\n")
    s.close()
    f = open("media.txt", "r")
    file = f.readlines()
    f.close()
    ffile(file)
else:
    ffile(file)
#print(file)
winwifi.WinWiFi.connect(f"{file[4]}")
```

```
f1 = open("location.txt", "r")
file1 = f1.readlines()
f1.close()
if (len(file1) == 0):
    t = open("location.txt", "a")
    b = input("enter Vs Code location: ")
    t.write(f'{b}\n')
    b = input("enter the Pycharm location: ")
    t.write(f'{b}\n')
    b = input("enter the Webex location: ")
    t.write(f'{b}\n')
    b = input("enter the chrome location: ")
    t.write(f'{b}\n')
    b = input("enter the skype location: ")
    t.write(f'{b}\n')
    b = input("enter the zoom location: ")
    t.write(f'{b}\n')
    b = input("enter the paint location: ")
    t.write(f'{b}\n')
    t.close()
    f1 = open("location.txt", "r")
    file1 = f1.readlines()
    f1.close()
    ffile(file1)
else:
    ffile(file1)
#print(file1)
```



```
def setdocument():  
    mudir = file[2]  
    u = os.listdir(mudir)  
    i = 0  
    for items in u:  
        v = items.split(".")[0].lower()  
        doc.update({v: i})  
        i += 1  
    print(doc)  
    return doc
```

```
setdocument()
```

```
def setsongs():  
    mudir = file[0]  
    songs = os.listdir(mudir)  
    i = 0  
    for items in songs:  
        v = items.split(".")[0].lower()  
        song.update({v: i})  
        i += 1  
    print(song)  
    return song
```

```
def video_set():  
    mudir = file[3]  
    picu = os.listdir(mudir)  
    i = 0  
    for items in picu:  
        v = items.split(".")[0].lower()  
        video.update({v: i})  
        i += 1  
    print(video)  
    return video
```

```
video_set()
```

```
def setpicture():  
    mudir = file[1]  
    picture = os.listdir(mudir)  
    i = 0  
    for items in picture:  
        v = items.split(".")[0].lower()  
        op.update({v: i})  
        i += 1  
    print(op)  
    return song
```

```
setpicture()
```

```
def wishme():
    joke = pyjokes.get_joke()
    print(joke)
    speak(joke)
    hour = int(datetime.datetime.now().hour)
    if hour < 12:
        speak("Good morning sir..")
    elif hour > 12 and hour < 16:
        speak("Good Afternoon sir...")
    elif hour >= 16 and hour <= 24:
        speak("Good Evening sir..")
    speak(f" i am alex....how may i help you")

def command():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("listening...")
        r.pause_threshold = 1
        r.energy_threshold = 1000
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language="en-in")
        print(f"u said {query}")
    except Exception as e:
        print("Say it again")
        return "none"
    return query

lru_cache(maxsize=10)
```

```
def playsongs():
    try:
        mudir = file[0]
        songs = os.listdir(mudir)
        speak("what would you like to hear sir...")
        song_name = command().lower()
        if "rehane do" in song_name or "nothing" in song_name:
            return
        elif "koi bhi" in song_name or "any one" in query:
            v = len(songs)
            i = random.randint(0, v - 1)
        elif song_name == "none":
            speak("which song sir..")
        else:
            i = song[song_name]
        speak("wait sir")
        os.startfile(os.path.join(mudir, songs[i]))
    except Exception as e:
        speak("this song is not exist please search a valid song")
```

```
def document():
    try:
        mud = file[2]
        l = os.listdir(mud)
        speak("which document sir..")
        dir = command().lower()
        if "rahane do" in dir or "rehane do" in dir:
            return
        elif "koi bhi" in dir or "any file":
            v = len(l)
            i = random.randint(0, v - 1)
        elif dir == "none":
            speak("which document sir..")
        else:
            i = doc[dir]
            speak("wait sir")
            os.startfile(os.path.join(mud, l[i]))
    except Exception as e:
        speak("this document is not exist please search a valid song")
```

```
def video():
    try:
        mud = file[3]
        l = os.listdir(mud)
        speak("which video sir..")
        dir = command().lower()
        if "nothing" in dir or "rehane do" in dir:
            return
        elif "koi bhi" in dir:
            v = len(l)
            i = random.randint(0, v - 1)
        elif dir == "none":
            speak("which video sir..")
        else:
            i = doc[dir]
            speak("wait sir")
            os.startfile(os.path.join(mud, l[i]))
    except Exception as e:
        speak("this video is not exist please search a valid song")
```

```
def picture_set():
    try:
        mudiru = file[1]
        pic = os.listdir(mudiru)
        speak("which picture sir..")
        pic_name = command().lower()
        if "nothing" in pic_name or "rehane do" in pic_name:
            return
        elif "koi bhi" in pic_name:
            v = len(pic)
            i = random.randint(0, v - 1)
        elif pic_name == "none":
            speak("which picture sir..")
        else:
            i = op[pic_name]
            speak("wait sir")
            os.startfile(os.path.join(mudiru, pic[i]))
    except Exception as e:
        speak("this picture is not exist please search a valid song")

def sendemail(content, to):
    server = smtplib.SMTP("smtp.gmail.com", 587)
    server.ehlo()
    server.starttls()
    fr = "alextheauxiliator@gmail.com"
    server.login(fr, "aleypass")
    server.sendmail(fr, to, content)
    server.close()
```

```
def confirm():
    ch = "y"
    i = 1
    while ch == "y" and i <= 5:
        speak("sir please enter password")
        pas = input("enter your password\n")
        with open("password.text") as f:
            a = f.read()
            if a == pas:
                break
            else:
                if i == 5:
                    exit()
                else:
                    i += 1

def shurukartehain():
    query = command().lower()
    while not (
        "start all task" in query or "lets start" in query or "Ready" in query):
        query = command().lower()
    speak(" all systems are online, initiating all tasks. Let's do something stormy
today ")
```



```
def daily():  
    hour = int(datetime.datetime.now().hour)  
    min = int(datetime.datetime.now().minute)  
    if hour == 16 and min == 30:  
        speak("sorry for disturbing you sir..but this is your coaching time")  
    if hour == 20 and min == 0:  
        speak("sorry for disturbing you sir..but this is your english speaking course  
time")  
    if hour == 22 and min == 30:  
        speak("sorry for disturbing you sir..but this is your aptitude practice time")  
    if hour == 12 and min == 30:  
        speak("sorry for disturbing you sir..but this is your django study time")  
    return  
  
if __name__ == '__main__':  
    setsongs()  
    confirm()  
    wishme()  
    shurukartehain()  
  
    ch = "y"  
    while ch == "y":  
        daily()  
        query = command().lower()
```

```
if "wikipedia" in query and "alex" in query:
    try:
        speak("searching sir...")
        query = query.replace("search ", "")
        query = query.replace("alex ", "")
        query = query.replace("about ", "")
        query = query.replace(" in wikipedia", "")
        result = wikipedia.summary(query, sentences=3)
        speak("According to wikipedia...")
        speak(result)
    except Exception as e:
        speak(f"sorry sir there is no information about {query} in wikipedia")
elif "document" in query and "alex" in query:
    document()

elif "google" in query and "alex" in query:
    speak("wait sir")
    speak("what should i search sir")
    search = command()
    while search == "none":
        speak("what should i search")
        search = command()
    kit.search(search)
```

```
elif "youtube" in query and "alex" in query:
```

```
    speak("what should i search")
```

```
    search = command()
```

```
    while search == "none":
```

```
        speak("what should i search")
```

```
        search = command()
```

```
    search.replace("search", "")
```

```
    speak("wait sir.")
```

```
    kit.playonyt(search)
```

```
elif "open hackerrank" in query and "alex" in query:
```

```
    webbrowser.open("hackerrank.com")
```

```
elif "open gallery" in query and "alex" in query:
```

```
    picture_set()
```

```
elif "play music" in query or "play songs" in query or "play song" in query  
or "play a song" in query or "please a song" in query or "please song" in query  
and "alex" in query:
```

```
    playsongs()
```

```
elif "time" in query and "alex" in query:
```

```
    time = datetime.datetime.now().strftime("%H:%M:%S")
```

```
    speak(time)
```

```
elif "open visual code" in query and "alex" in query:
```

```
    speak("Opening visual code")
```

```
    os.startfile(file1[0])
```

elif "close visual code" in query and "alex" in query:

```
    speak("closing visual code")  
    os.system("taskkill /f /im Code.exe")
```

elif " open pycharm" in query and "alex" in query:

```
    speak("opening pycharm")  
    os.startfile(file1[1])
```

elif "close pycharm" in query and "alex" in query:

```
    speak("Closing pycharm")  
    os.system("taskkill /f /im pycharm64.exe")
```

elif "open command prompt" in query and "alex" in query:

```
    speak("Opening command prompt")  
    os.startfile("C:\\Windows\\System32\\cmd.exe")
```

elif "close command prompt" in query and "alex" in query:

```
    speak("Closing command prompt")  
    os.system("taskkill /f /im cmd.exe")
```

elif "open webex" in query and "alex" in query:

```
    speak("Opening")  
    os.startfile(file1[2])
```

elif "close webex" in query and "alex" in query:

```
    speak("Closing")  
    os.system("taskkill /f /im CiscoCollabHost.exe")
```

```
elif "open word" in query and "alex" in query:  
    speak("Opening")  
    os.startfile("C:\\Program Files\\Microsoft  
Office\\root\\Office16\\WINWORD.EXE")
```

```
elif "close word" in query and "alex" in query:  
    speak("Closing")  
    os.system("taskkill /f /im WINWORD.EXE")
```

```
elif "open file explorer" in query and "alex" in query:  
    speak("Opening")  
    os.startfile("C:\\Windows\\explorer.exe")
```

```
elif "close file explorer" in query and "alex" in query:  
    speak("Closing")  
    os.system("taskkill /f /im explorer.exe")
```

```
elif "open calendar" in query and "alex" in query:  
    speak("Opening")  
    os.startfile("C:\\Program Files\\WindowsApps\\microsoft.windowscommunicationsapps_16005.14326.2085  
8.0_x64__8wekyb3d8bbwe\\HxCalendarAppImm.exe")
```

elif "close calendar" in query and "alex" in query:

speak("Closing")

os.system("taskkill /f /im HxCalendarAppImm.exe")

elif "open excel" in query and "alex" in query:

speak("Opening")

**os.startfile("C:\\ProgramFiles\\Microsoft
Office\\root\\Office16\\EXCEL.exe")**

elif "close excel" in query and "alex" in query:

speak("Closing")

os.system("taskkill /f /im EXCEL.exe")

elif "open screen keyboard" in query and "alex" in query:

speak("Opening")

os.startfile("C:\\Windows\\System32\\osk.exe")

elif "close screen keyboard" in query and "alex" in query:

speak("Closing")

os.system("taskkill /f /im osk.exe")

elif "open powerpoint" in query and "alex" in query:

speak("Opening")

**os.startfile("C:\\ProgramFiles\\Microsoft
Office\\root\\Office16\\POWERPNT.EXE")**

elif "close powerpoint" in query and "alex" in query:

speak("Closing")

os.system("taskkill /f /im POWERPNT.EXE")

elif "open chrome" in query and "alex" in query:

speak("Opening")

os.startfile(file1[3])

elif "close chrome" in query and "alex" in query:

speak("Closing")

os.system("taskkill /f /im chrome.exe")

elif "open skype" in query and "alex" in query:

speak("Opening")

os.startfile(file1[4])

elif "close skype" in query and "alex" in query:

speak("Closing")

os.system("taskkill /f /im Skype.exe")

elif "open zoom" in query and "alex" in query:

speak("Opening")

os.startfile(file1[5])

elif "close zoom" in query and "alex" in query:

speak("Closing")

os.system("taskkill /f /im zoom.exe")

elif "open music" in query:

speak("executing")

os.startfile("D:\\Music")

elif "open document" in query:

speak("executing")

os.startfile("D:\\Documents")

elif "open download" in query:

speak("executing")

os.startfile("C:\\Users\\khali\\Downloads")

elif "open videos" in query:

speak("executing")

os.startfile("D:\\Videos")

elif "open media" in query:

speak("executing")

os.startfile("C:\\Program Files\\Windows Media Player\\wmplayer.exe")


```
elif "send mail" in query or "send email" in query:
    try:
        speak("are u confirm")
        confirm = command().lower()
        if confirm == "none":
            confirm = command().lower()
        if "rahane do" in confirm or "rehane do" in confirm or "no" in
confirm or "nahi" in confirm:
            pass
            speak("what should i say")
            content = command()
            speak("are u confirm")
            confirm = command().lower()
            print(confirm)
            if confirm == "none":
                confirm = command().lower()
            if "rahane do" in confirm or "rehane do" in confirm or "no" in
confirm:
                pass
            elif content == None:
                speak("say it again")
                content = command()
            elif content != "" and "rahane do" not in confirm:
                speak("To whom sir ")
                item = command().lower()
                print(item)
                s = item.split(" ")
                print(s)
                for i in s:
                    for item in di.keys():
                        if item == i:
```

```
        to = i
        print(to)
        too = di[to]
        print(too)
        sendemail(content, too)
        speak("email is send")
    except Exception as e:
        speak("sry sir i m not able to send your email")

    elif "change song" in query or "next song" in query or "change music" in
query or "next music" in query and "alex" in query:
        playsongs()

    elif "who invented you" in query and "alex" in query:
        speak("Aryaman and Abdullah invented me.")

    elif "tell me something about yourself" in query and "alex" in query:
        speak("I am Alex an Artificial intelligence programme, developed by
Aryaman and Abdullah as a their Bachelors of technology project under
Auxiliator:One's personal Artificial Assistant I am developed using python
programming language and can perform multiple tasks, such as playing "
        "music for you, opening your files etc.")
        speak("my task is to perform functions as per the user's voice commands.
I am here to assist you.")
        print("1. Today's news alex\n")
        print("2. Tell me the time alex\n")
        print("3. Google search alex\n")
```

```
    speak("Here is a list of some functions i can perform for you. Give it a try  
    !!!")
```

```
    elif "aj ki news" in query or "today's news" in query or "news sunao" in  
    query and "alex" in query:
```

```
        url=requests.get("http://newsapi.org/v2/top-  
        headlines?country=in&category=sports&apiKey=657de6952ac3443e9d6ba9b27a  
        126d13").text
```

```
        content = json.loads(url)
```

```
        con = content['articles']
```

```
        speak("Todays news is..")
```

```
        c = 1
```

```
        ran = ["Moving on to the next news", "Next news", "next is",  
        "continuing on to the next news", "listen carefully"]
```

```
        for items in con:
```

```
            print(items['title'])
```

```
            speak(items['title'])
```

```
            speak(random.choice(ran))
```

```
            if c == 10:
```

```
                break
```

```
            c += 1
```

```
    elif "open paint" in query and "alex" in query:
```

```
        speak("opening paint")
```

```
        os.startfile(file1[6])
```

```
    elif "close paint" in query and "alex" in query:
```

```
        speak("closing paint")
```

```
os.system("taskkill /f /im CiscoCollabHost.exe")
```

```
elif "say" in query and "alex" in query:
```

```
    query = query.replace("say", "")
```

```
    query = query.replace("alex", "")
```

```
    speak(query)
```

```
    print(query)
```

```
elif "joke" in query and "alex" in query:
```

```
    joke = pyjokes.get_joke()
```

```
    print(joke)
```

```
    speak(joke)
```

```
elif "how are you" in query and "alex" in query:
```

```
    hru = ["Fine, and you?", "Good, how about yourself?", "Doing fine, and  
you?", "Good, how about you?"]
```

```
    speak(random.choice(hru))
```

```
elif "whatsapp" in query and "alex" in query:
    try:
        speak("what is the msg sir")
        msg = command()
        b = 0
        while msg == "none":
            speak("what is the message sir")
            msg = command().lower()
        speak("send to whom")
        for items in contacts.keys():
            print(items)
        whom = command().lower()
        while whom == "none":
            speak("send to whom")
            whom = command().lower()
        if whom == "leave it" or whom == "rehana do" or whom == "rahana
do":
            break
        else:
            s = whom.split(" ")
            print(s)
            for i in s:
                for items in contacts.keys():
                    if items == i:
                        reciever = contacts[items]
                        print(reciever)
            hour = int(datetime.datetime.now().hour)
            min = int(datetime.datetime.now().minute) + 2
            kit.sendwhatmsg(reciever, msg, hour, min, 55)
    except Exception as e:
```

```
    speak("this contact is not exist")
```

```
elif "shut down" in query and "alex" in query:
```

```
    time = int(datetime.datetime.now().hour)
```

```
    speak("Are you confirm sir")
```

```
    msg = command().lower()
```

```
    while msg == "none":
```

```
        speak(" confirm your command ")
```

```
        msg = command().lower()
```

```
    if "yes" in msg:
```

```
        kit.shutdown(time)
```

```
    else:
```

```
        pass
```

```
elif "enough" in query and "alex" in query:
```

```
    speak("sorry for inconvenience sir.")
```

```
elif "hey alex" in query:
```

```
    speak("hello sir..")
```

```
elif "what are you doing" in query and "alex" in query:
```

```
    speak("waiting for your command sir..")
```

```
elif "see you later" in query and "alex" in query:
```

```
    speak("Bye sir...take care and have a good day")
```

```
    ch = "n"
```

```
elif "Sleep" in query or "let's have some rest" in query:
```

```
    speak("by what minutes shall i be back sir ")
```

```
    Qry = command().lower()
```

```
    Qry = Qry.replace("minutes", "")
```

```
    Qry = Qry.replace("minute", "")
```

```
    print(Qry)
```

```
    local_time = int(Qry)
```

```
    local_time = local_time * 60
```

```
    time.sleep(local_time)
```

```
    speak("Hey there! i m back at your service ")
```

```
elif "stop" in query:
```

```
    speak("terminating the system now. we'll meet again soon!!!")
```

```
    print("terminating the system now. we'll meet again soon!!!")
```

```
    exit()
```

3.4.2.3. Snapshots

Here are the screenshots of the working of the application over the command line interface.

i. Demonstration of opening of chrome application

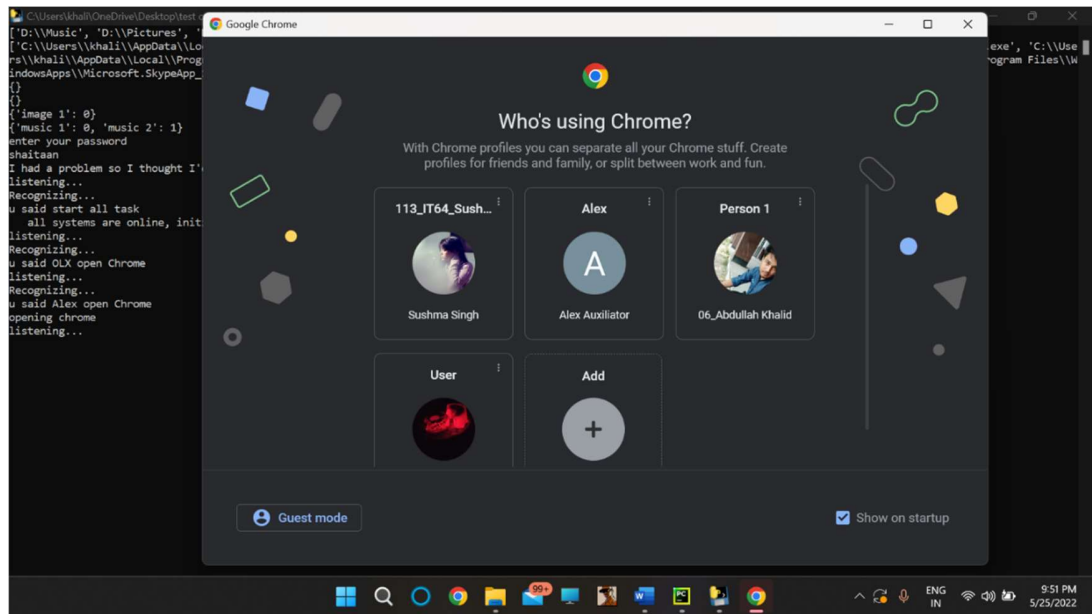


Figure 3.4.2.3 (a)

ii. Demonstration of opening and closing of zoom application

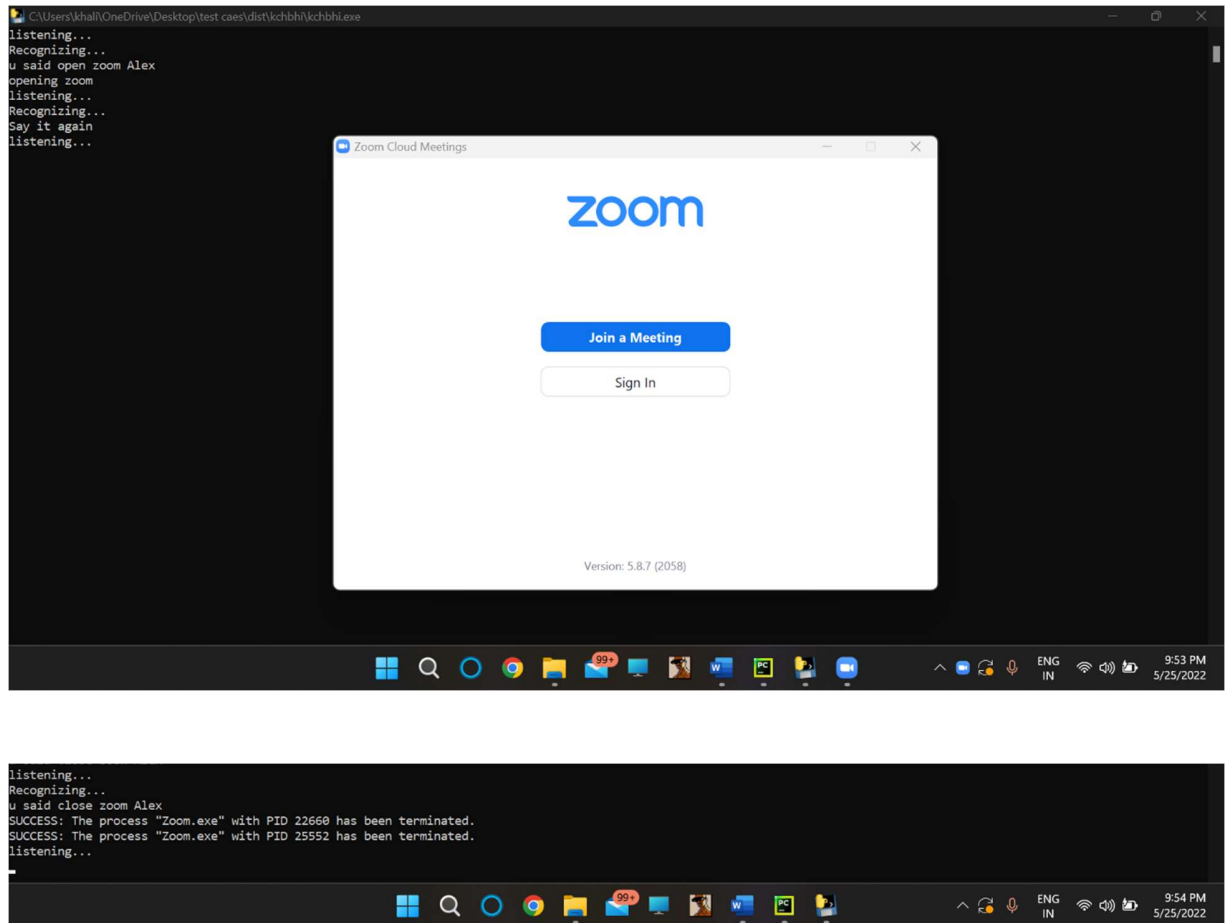


Figure 3.4.2.3 (b)

iii. Demonstration of opening of downloads file

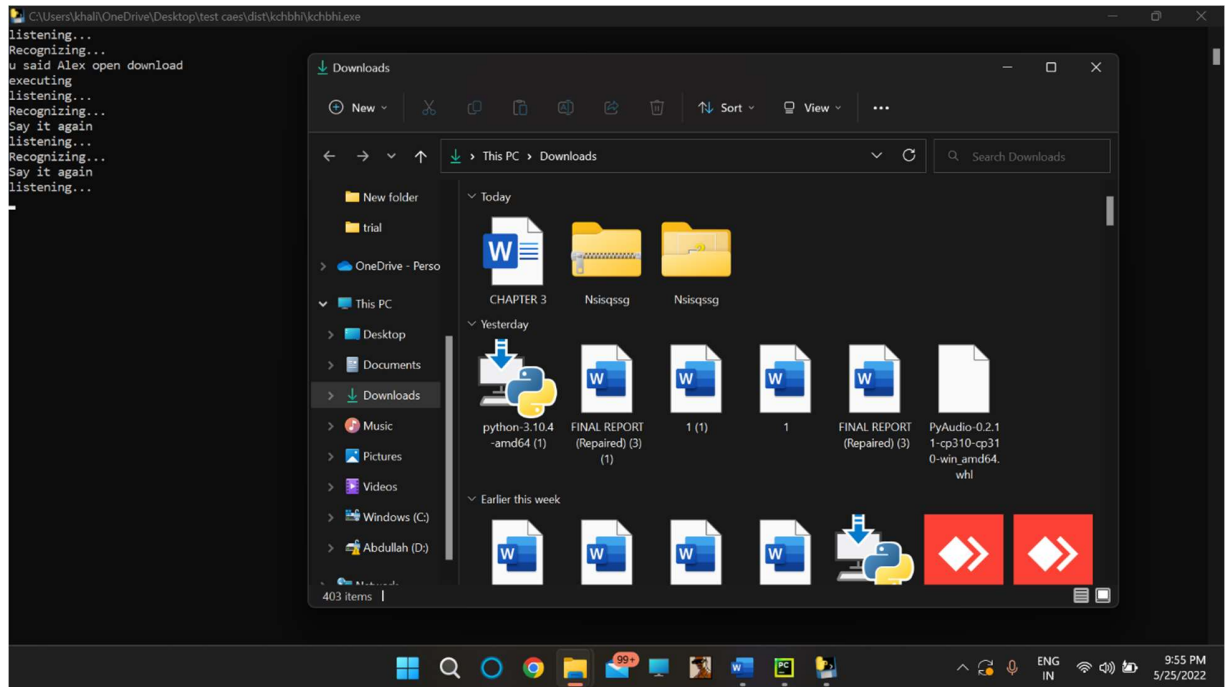


Figure 3.4.2.3 (c)

iv. Demonstration of opening and closing of word application

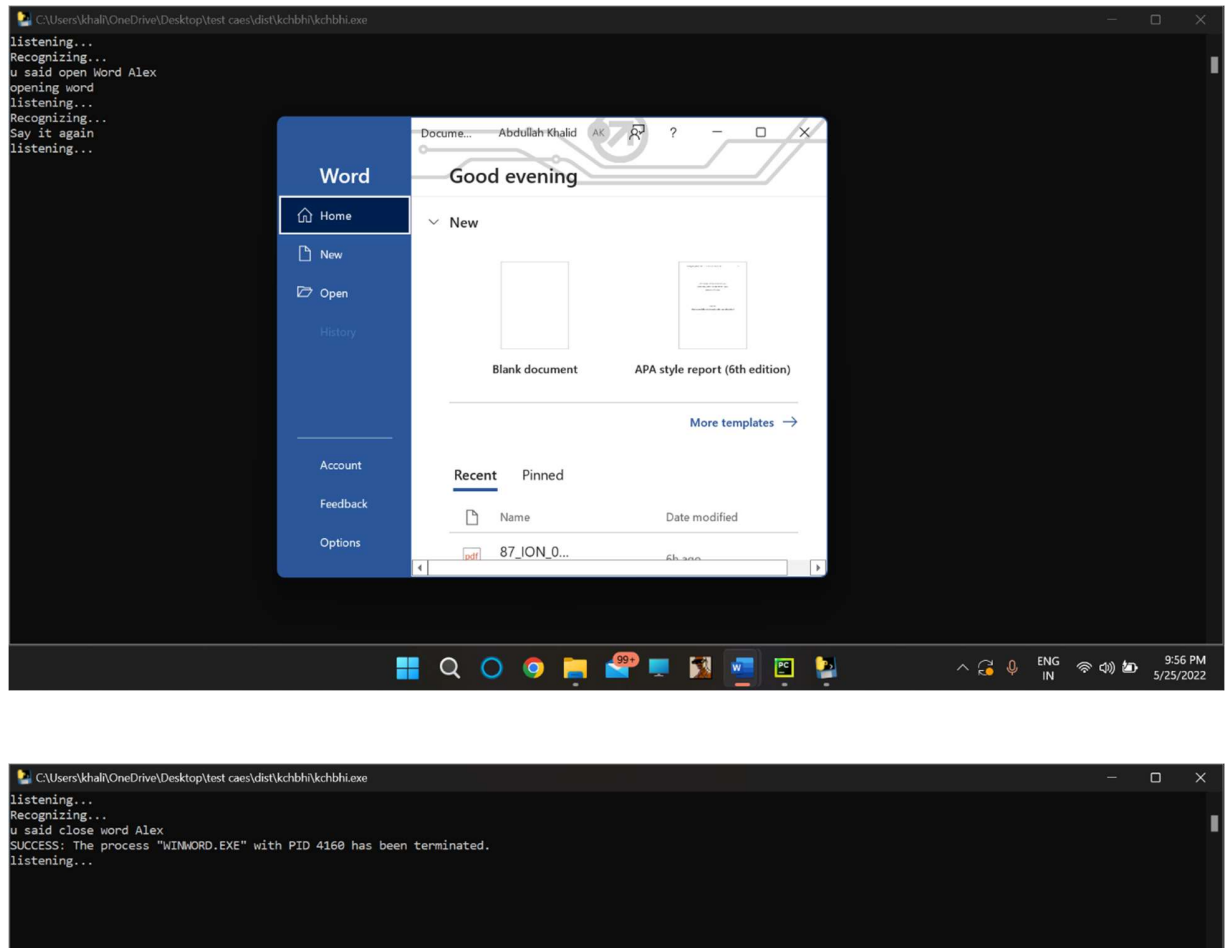


Figure 3.4.2.3 (d)

v. Demonstration of opening a picture by telling the picture name

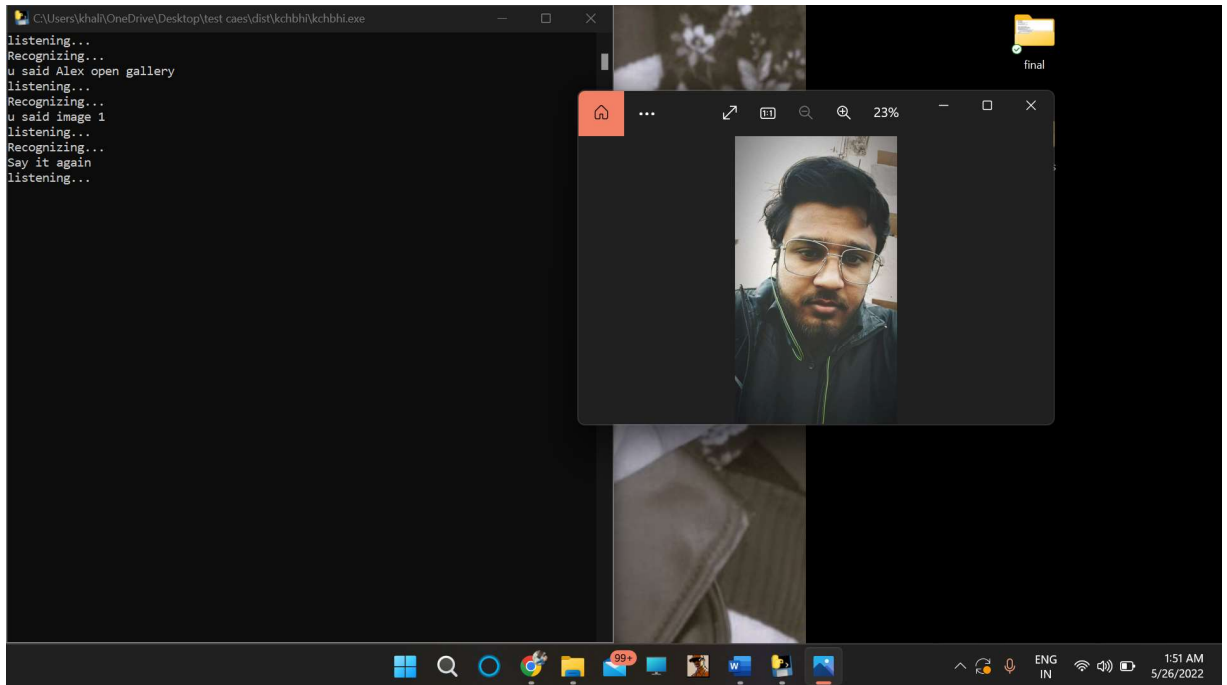


Figure 3.4.2.3 (e)

vi. Demonstration of playing music offline by saying the name

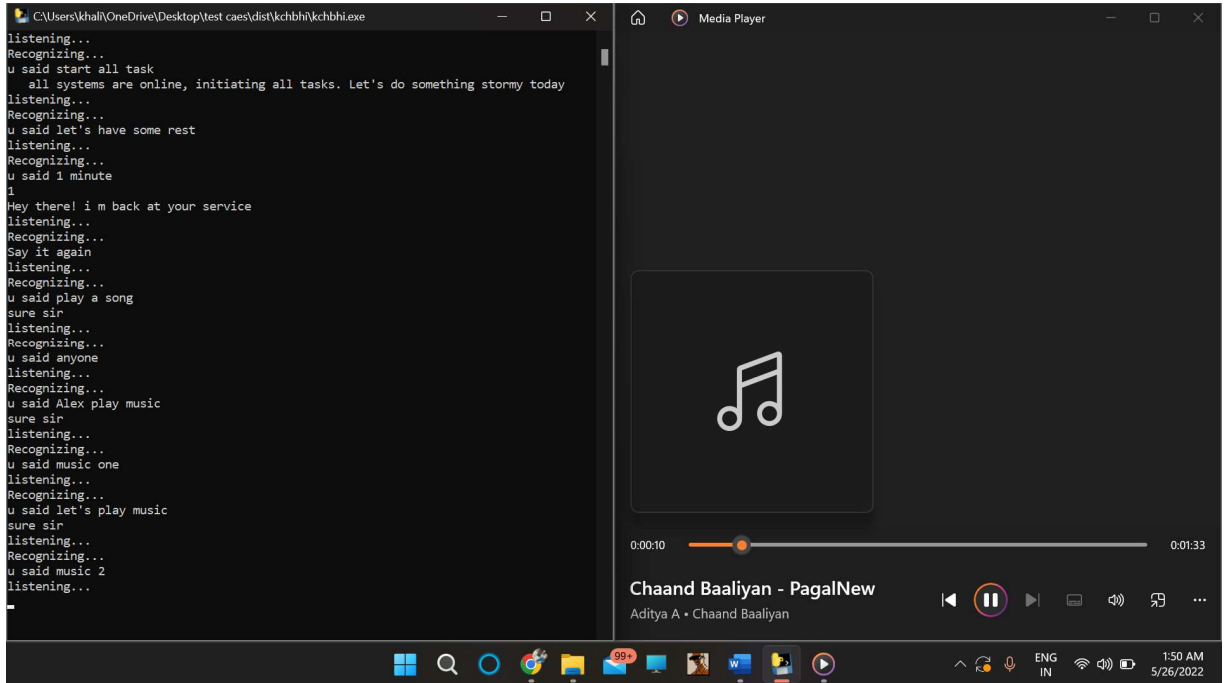
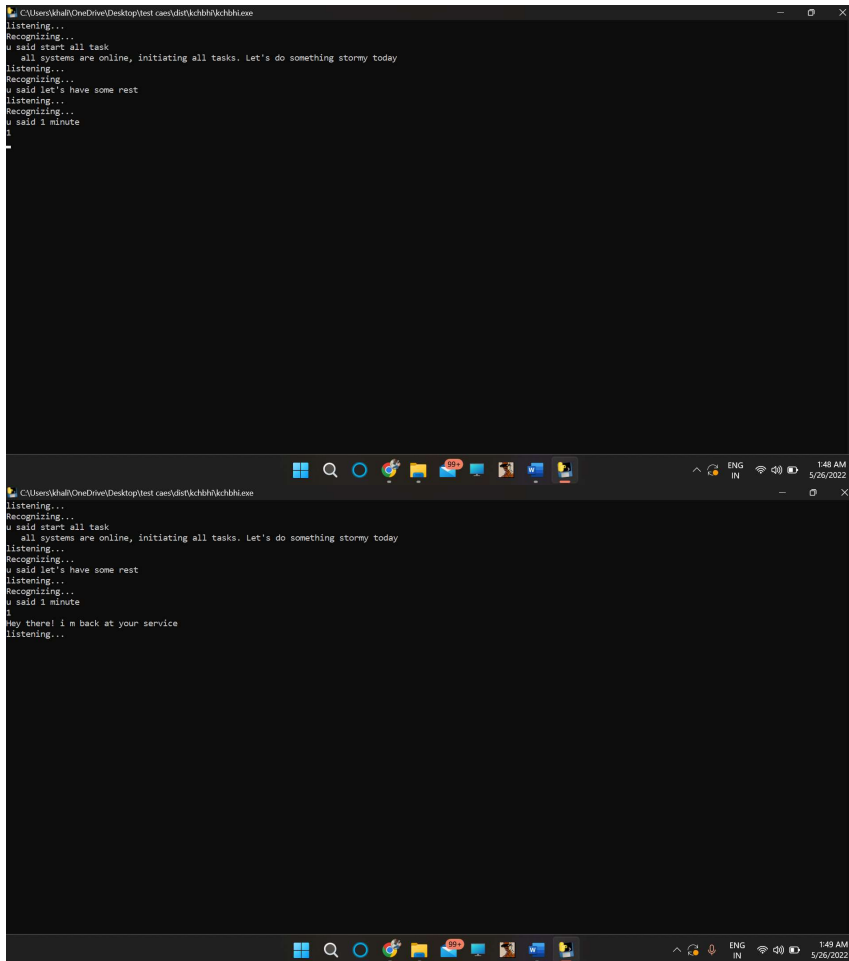


Figure 3.4.2.3 (f)

vii. Demonstration of Sleep function



```
C:\Users\khal\OneDrive\Desktop\test case\dist\kchbb\kchbb.exe
listening...
recognizing...
u said start all task
  all systems are online, initiating all tasks. Let's do something stormy today
listening...
recognizing...
u said let's have some rest
listening...
recognizing...
u said i minute
i
hey there! i'm back at your service
listening...
```

Figure 3.4.2.3 (g)

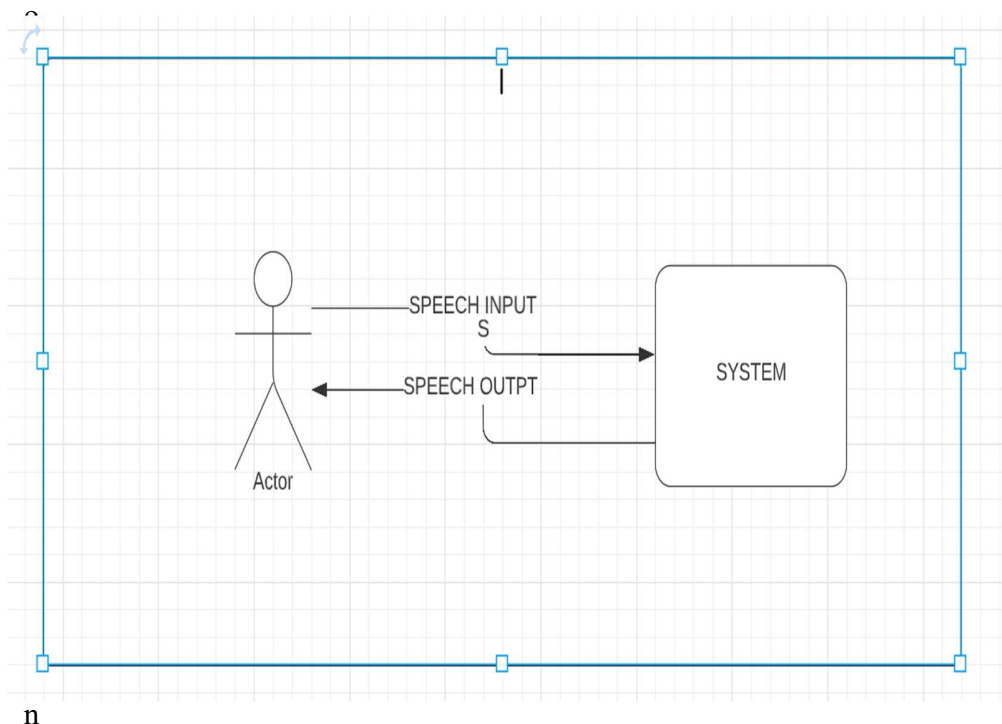
3.5. Diagram of the model

3.5.1. Data flow Diagrams (DFD)

In Software engineering DFD(data flow diagram) can be drawn to represent the system of different levels of abstraction. Higher-level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2 or beyond.

i. 0-level DFD or Context Diagram

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and



dictated by incoming/outgoing arrows.

Figure 3.5.1 (a)

ii. **1-level DFD**

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

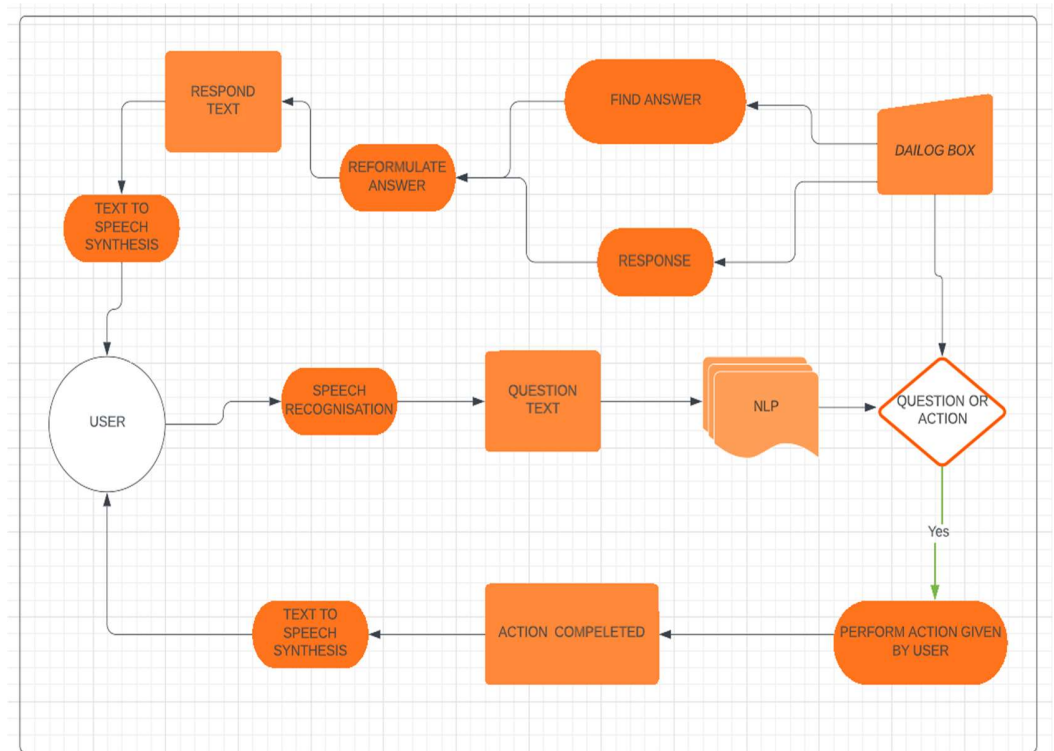


Figure 3.5.1 (b)

3.5.2. Use Case Diagram

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

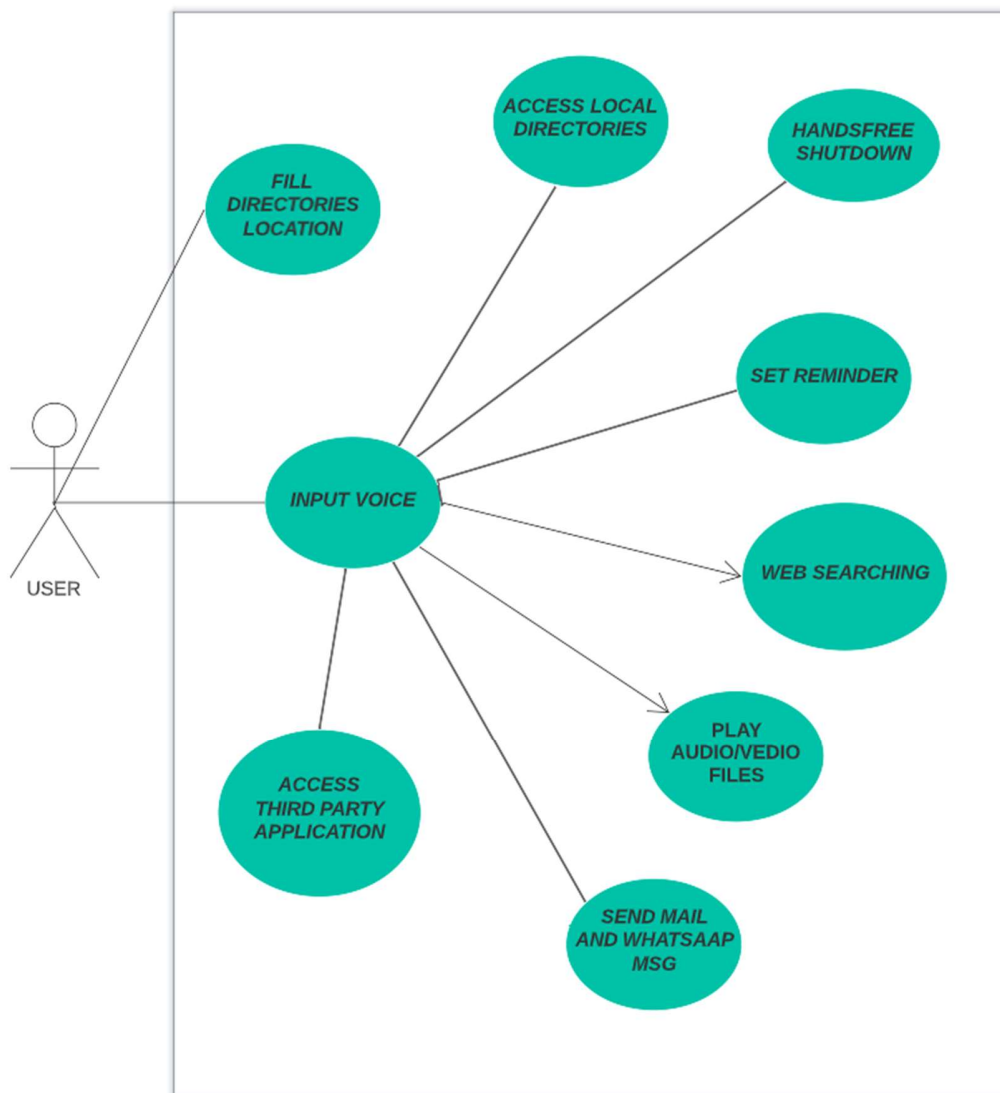


Figure 3.5.2

3.5.3. Block Diagram

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams.

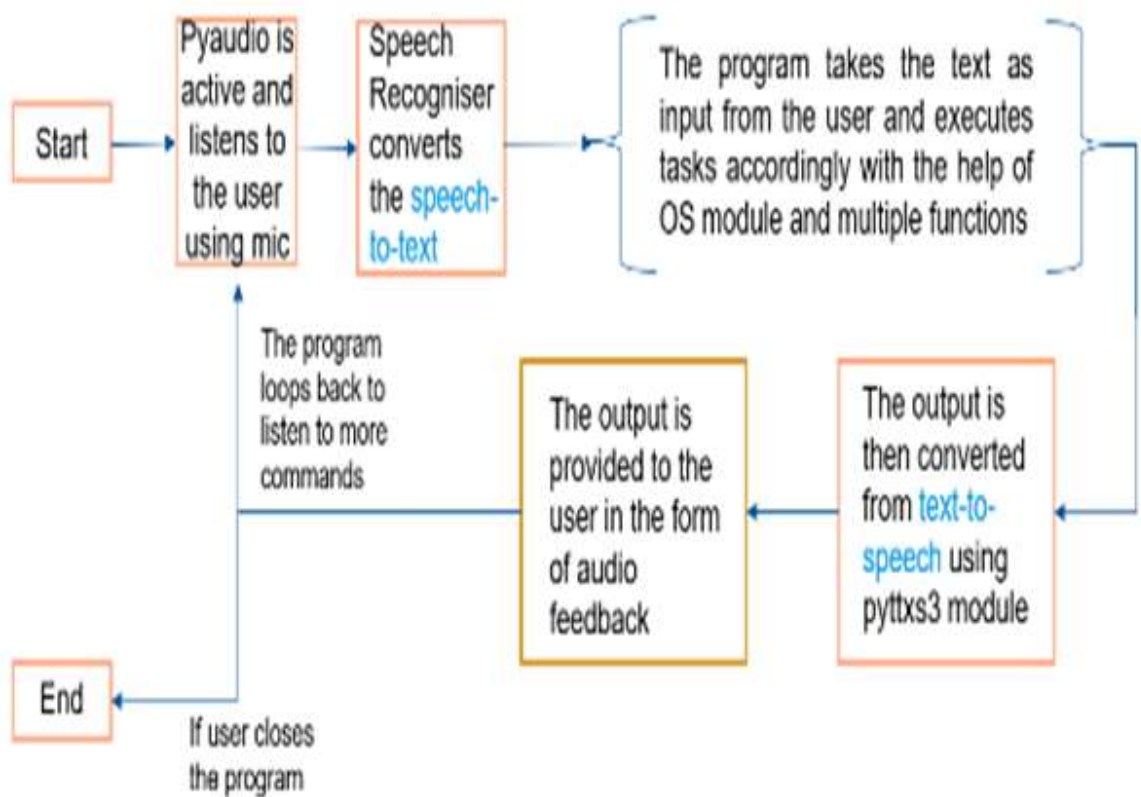


Figure 3.5.3

4. RESULT ANALYSIS AND DISCUSSION

4.1 Testing

Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Here are important testing strategies:

Unit Testing: Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff. The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Unit testing is an important step in the development process, because if done correctly, it can help detect early flaws in code which may be more difficult to find in later testing stages.

In the testing of this program we have tested each and every function of the application individually for their working and efficiency before implementing it into the final software.

Integration testing: It focuses on the construction and design of the software. You need to see that the integrated units are working without errors or not. It is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers.

The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

System testing: In this method, your software is compiled as a whole and then tested as a whole. This testing strategy checks the functionality, security, portability, amongst others. It is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

4.2 Test Case

Testing Cases are classified into three categories.

i. Functional Testing:

Functional testing is the process through which QAs determine if a piece of software is acting in accordance with pre-determined requirements. It uses black-box testing techniques, in which the tester has no knowledge of the internal system logic.

In the proposed software all the functions and methods are tested individually, firstly in the ideal condition setup and then in the real world scenarios.

ii. Performance Testing:

Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload. Organizations will run performance tests in order to identify performance-related bottlenecks.

In the test run both the program file as well as the executable file proved to be working fine and were stable on the system it was tested upon.

iii. Maintenance:

Software maintenance is the process of changing, modifying, and updating software to keep up with customer needs. Software maintenance is done after the product has launched for several reasons including improving the software overall, correcting its use or bugs, to boost performance, and more.

4.3 Applications

- i Personal Assistant
- ii Can be used in the field of education as a student's learning assistant or a teacher's teaching assistant.
- iii The proposed system can be implemented over user interactive kiosks.
- iv The proposed application can be configured over any python enabled microprocessor and customizations can be performed accordingly.

4.4 Advantages

- i Access local files saved in the systems
- ii Required Internet only to execute the online part of the program
- iii Easy to customize according to the requirements of the user
- iv Bypasses some of the background noise.
- v This program be implemented on any Integrated Development environment.
- vi The proposed application uses internet services for functionality and not for uploading any kind of user data over the internet clouds or anything in account of the security of the user data.
- vii Real world interactions.
- viii All the commands are associated with a wake word so the system does not take any arbitrary user conversation with any entity as an input command.

4.5 Limitations

- i The application requires a strong internet connectivity for some of its features to work properly.
- ii Efficiency decreases at noisy places.
- iii There is some chance of misinterpretation of voice commands if they are not pronounced appropriately.
- iv The directory path can be implemented only in the first run of the program and the path cannot be modified later.
- v Some of the important packages are to be downloaded by the user from the internet

5. CONCLUSION

Working on this project was a great pleasure for us , we had a good experience and got to learn a lot about the Python programming language. We got a real life practical experience about implementation of different modules and packages of Python into a program as well as we got to work with the text to speech recognition and the natural language processing algorithms and methodologies. We got to learn the methodologies an implementation of file management through programming language. We got to learn about the artificial intelligence system as well Which is an emerging technology and can help us in our future endeavors. We were able to develop A reaction based artificial intelligence system Using python programming language We got to know about different artificial intelligence applications and algorithms as well as methodologies. All these deductions and knowledge will provide us with a better understanding of Artificial intelligence projects and other projects while working on them in future independently.

Some of the deductions that we find while working on this project are as follows:

1. Python has advantages over other programming languages while working with artificial intelligence projects.
2. Python is so easy to work language with its support to all kinds of programming paradigms It is quite easy to implement any kind of program in python and run them over any integrated development environments.
3. In this project we are using python, speech recognition, natural language processing and artificial intelligence as a platform to solve day to day issues of the user while working with and handling computer systems.

6. FUTURE SCOPE OF THE PROJECT

Our project has a great good future scope as we are trying, we will be trying to implement a lot of other features in functionalities to the project as the time goes by. Some of the future functionalities and implementations are as follows:

- i. We will be adding functionalities so that our program can work with multiple interaction or vocal languages.
- ii. We will be trying to implement some of the complexity and analysis and emotional analysis Algorithms which will be able to judge and assess the user's mood and emotions by Axis listening to their Voices.
- iii. A more interactive graphical user interface can be implemented in the future.
- iv. We will try to proceed the A I forward to the native AI in the direction of general AI development by implementing multiple Artificial intelligence algorithms so that the program will be a full-fledged artificial intelligence system that can implement account Account and be responsible for a lot of programming capabilities as well as identifying user users needs on its own and modifying the program core by itself

APPENDIX-A

LIST OF FIGURES

<u>S. No.</u>	<u>Figure No.</u>	<u>Description</u>	<u>Page No.</u>
1.	Figure 1.1.3	AI type-1 and type-2	
2.	Figure 3.3	Spiral model	
3.	Figure 3.4.2.1 (a)	Installing pyaudio on command prompt	
4.	Figure 3.4.2.1 (b)	Installing pyttsx3 on command prompt	
5.	Figure 3.4.2.1 (c)	Installing Speech recognition on command prompt	
6.	Figure 3.4.2.1 (d)	Installing flask on command prompt	
7.	Figure 3.4.2.1 (e)	Installing pywhatkit on command prompt	
8.	Figure 3.4.2.1 (f)	Installing pyjokes, winwifi and wikipedia on command prompt	
9.	Figure 3.4.2.1 (g)	Importing all the modules and packages in the main file of the program	
10.	Figure 3.4.2.1 (h)	Pyttsx3 engine setup and configuration in the main program file	
11.	Figure 3.4.2.1 (i)	Speak and command function integration in main program file	
12.	Figure 3.4.2.3 (a)	Working of program to open Chrome application	
13.	Figure 3.4.2.3 (b)	Working of program to open and close Zoom application	
14.	Figure 3.4.2.3 (c)	Working of program to open Downloads directories	

15.	Figure 3.4.2.3 (d)	Working of program to open and close word application	
16.	Figure 3.4.2.3 (e)	Working of program to open an image file	
17.	Figure 3.4.2.3 (f)	Working of program to play music from the locally stored music files	
18.	Figure 3.4.2.3 (g)	Working of sleep function of the program	
19.	Figure 3.5.1 (a)	Data flow diagram level-0: showing the basic interaction of program with the user entity	
20.	Figure 3.5.1 (b)	Data flow diagram level-1: Highlighting the main function of the program and breakdown of the high -level processes	
21.	Figure 3.5.2	Use case diagram: describing the high-level function and scope of the application and identification of actors between the program and its actors	
22.	Figure 3.5.3	Block diagram: demonstrating the blocks of the programs and showing the relationships between them	

LIST OF ABRIVIATION

<u>Abbreviation</u>	<u>Description</u>
AI	Artificial intelligence
NLP	Natural Language processing
DFD	Data flow diagram
pip	Preferred installer program

APPENDIX -B

REFERENCE & BIBLIOGRAPHY

- Intelligent Personal Assistant - Implementing Voice Commands enabling Speech Recognition
Kumaran N.;Rangaraj V.;Siva Sharan S.;Dhanalakshmi R.; (2020).
- Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)
Kepuska, Veton; Bohouta, Gamal (2018). [IEEE 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC) - Las Vegas, NV, USA (2018.1.8-2018.1.10)] 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC) .
- A Vision and Speech Enabled, Customizable, Virtual Assistant for Smart Environments.
Iannizzotto, Giancarlo; Bello, Lucia Lo; Nucita, Andrea; Grasso, Giorgio Mario (2018). [IEEE 2018 11th International Conference on Human System Interaction (HSI) - Gdansk, Poland (2018.7.4-2018.7.6)] 2018 11th International Conference on Human System Interaction (HSI)
- A Study on Automatic Speech Recognition Systems.
Ibrahim, Habib; Varol, Asaf (2020). [IEEE 2020 8th International Symposium on Digital Forensics and Security (ISDFS) - Beirut, Lebanon (2020.6.1-2020.6.2)] 2020 8th International Symposium on Digital Forensics and Security (ISDFS).

- An Overview of Speech Recognition Technology
Zhang, Xinman; Peng, Yurui; Xu, Xuebin (2019). [IEEE 2019 4th International Conference on Control, Robotics and Cybernetics (CRC) - Tokyo, Japan (2019.9.27-2019.9.30)] 2019 4th International Conference on Control, Robotics and Cybernetics (CRC) .
- Artificial Intelligence in the 21st Century. IEEE Access
Liu, Jiaying; Kong, Xiangjie; Xia, Feng; Bai, Xiaomei; Wang, Lei; Qing, Qing; Lee, Ivan (2018).
- Natural Language Processing and Its Applications in Machine Translation: A Diachronic Review .
Kai Jiang;Xi Lu; (2020).