International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 05 | May - 2023 Impact Factor: 8.176 ISSN: 2582-3930

IRIS – Your Personal Desktop Assistant

[1]Sparsh Goyal, [2]Yash Bhargav, [3]Nazia Shamim, [4]Prof. Bipin Pandey
[1,2,3,4]*Departement of CSE, Dronacharya Group of Institutions, Greater Noida, India*

Abstract **- With ever-emerging technology, the meaning of an artist is also changing. These days, with the help of handy technology and interactive operating systems, anyone can become an artist using their phones and laptops or PCs. Moreover, these technologies on their own have become diverse in the ways we, Humans, interact with them. One of the ways which have gained quite a lot of popularity is through Voice control. Many mobile devices these days come pre-loaded with at least one voice assistant. These voice assistants on their own are "smart" and are capable of handling the whole mobile device. However, the market around "voice assistants" for PCs and Laptops seems confusingly empty. One of the main competitors in this area was Microsoft's Cortana, which was also made optional with Microsoft's Windows 11. We came across this news during our research phase and immediately decided to tackle this situation. We have designed a voice assistant capable of handling tasks presented to it and is capable of doing that while running on any operating system that the user desires. Furthermore, we have designed this assistant to be capable of understanding any command given by the user even if the syntax of the language used by the user is incorrect. This will allow the voice assistant to be accessible by a wide range of users from any linguistic backgrounds allowing everyone to be a creator and a proud business owner.**

Index Terms **– Artificial Desktop Assistant, Python, Virtual Assistant, Voice recognition, Database, Text-o-Speech**

I. INTRODUCTION

With more people accessing the internet, information has become easy to learn and more affordable. More people are now learning and applying skills that were inaccessible previously, resulting in new and innovative businesses emerging every day. We can say that a new generation of business owners is rising where people are their bosses instead of slaving their skills off to some big multinational corporation. This age is recognized as the age of creators, developers and social influencers. These individuals, each having some or the other form of skill, show off their skills on various social media and video sharing sites such as YouTube, Instagram and Facebook, earning hundreds of thousands of followers who respect them and their art. In the latter part of the last decade, social media artists earned much of their fame and devised a new way of earning money, i.e., through social media.

Though today's smartphones have become insanely smart regarding how much computation power they have, most creators still consider PCs and Laptops as their main computational resource.

Now, to access technology, we need to access technological devices too. In the last decade or so, the input system to various devices has changed drastically. We now have voice assistants embedded in our systems that take input from the user as a voice command, processes it using various AI and Machine Learning algorithms and present an output.

Although many voice assistants are available in the market already for mobile devices running on android or IOS, such as Google Assistant, SIRI, Bixby, etc., the market for desktop voice assistants is still very much available. A strong competitor earlier in the shape of Microsoft's Cortana, which was made compulsory with Windows 10 and 10.1, has been made optional with Microsoft's Windows 11 in the middle of 2021.

This step was taken after Cortana received heavy criticism on both technical and non-technical fronts for its unwillingness to provide the users with flexible options and its inability to understand basic commands provided by the user using amateur syntax. Another limitation was Cortana's cross-platform access as there was none of it.

We worked on these sets of problems only. We designed a virtual assistant which was not only cross platform accessible but also understand a wide variety of questions and commands given by the user irrespective of the syntax of the command/query language. This allows a wide variety of users from different cliques of society, culture, country of origin and linguistic background to access the assistant and perform 'n' number of tasks. Moreover, we have linked the assistant with a local-to-user system database, which will allow users to add any number of functionalities they want in the system without actually needing to interpret the assistant's core code or program. This will encourage more developers and creators from non-technical backgrounds to use the assistant without hesitation. Because our assistant is an individual software made using Python language, which uses UNIX file and command assorting method our assistant will be completely cross platform accessible working on any Operating system. Other functionalities offered by the assistant include: -

- Name and voice change of the assistant
- Shutdown, Restart and Log-off
- Play music (Online and available in system directory)
- Web Scraping
- Getting news updates
- Setting a reminder and alarm
- Open any application and software on the system
- Turn Bluetooth and WIFI on or off
- Increase and decrease system volume

- Access E-mail and check for updates

Thus, our system will solve approximately every problem Microsoft's Cortana poses.

II. PROBLEM FORMULATION

As engineers, we tend to design our machines to be, in some sort, extensions of a human body so that it is not cumbersome for them to use them or learn how to use them. As developers we intend to design our software as intuitive as possible keeping the real-world design in our brains. This approach is kept in mind from not a business mindset, but from a customer standpoint too, where we try to think like the end user who will use our systems to avail their day-to-day tasks.

This allows us to design hardware and software capable of understanding and (maybe) solving every possible humane problem with the least amount of latency and error possible.

However, sometimes even some fascinating and sophisticated pieces of technologies can have some really basic bugs, which can cause extreme user dissatisfaction and unrest. One of the major examples of this is Microsoft's Cortana, which, although it created much news when introduced, made many users unhappy later. One of the reasons for this was its inability to do exactly what it was supposed to do, i.e., make systems hands free.

Thus, the problem that we are dealing with is simple, we need to develop a voice assistant which, anyone, ranging from a child to an average user to a non-technical professional to a technical professional can use to access their respective PCs and laptops. A voice assistant with the least amount of latency and error and the capability to make these machines easier to use.

More specifically, some of the problems that we learnt needed to be solved included: -

- Platform Agnostic software, i.e., run on multiple operating systems without losing any functionality.
- Amateur language syntax understandability i.e., the ability to understand improper sentences, mispronunciations, accents and chorus or local terms used by the user.
- Variable command understandability, i.e., the ability to perform a task even if the user's command is not the same syntactically and semantically every time.
- Learning ability of the assistant i.e., to learn from the user and the environment, to perform task based on their needs.
- Multi-tasking i.e., assistant's tasks and performance does not interrupt or affect the user's tasks and performance and vice-verse.

III. PROPOSED APPROACH

The approach to the problems mentioned above are kept simple too. This is done because we are designing this assistant for all sorts of users and not only for the technical professionals.

To understand the approach more efficiently, we first need to understand the technology used in the design: -

Technology stack

1) *Python*: Python is an Object-Oriented Programming language which has recently become one of the most popular and widely used programming language. The applications of python include Data Science, Data Analytics, Data Visualization, Machine learning, Deep learning, Natural Language Processing, Artificial Intelligence etc. This is majorly because of its highly intuitive syntax with 33 keywords and more than 150,000 libraries available free of cost. The open-source availability of python also makes it one of the most developer friendly language with a community of more than 100,000 contributors on GitHub and Stack overflow.

   This project is made solely using python i.e., both the backend and frontend of this assistant are coded using python only.

Some of the modules and libraries of Python that we used are: -

a) *Pyttsx3*: Python text-to-speech or pyttsx3 is a text-to-speech library in python compatible with python 2 and python 3. Unlike other text-to-speech libraries, it can function even without internet access. Some of the additional functionalities that this library offers include the control over rate of the voice, volume of the voice and most interestingly gender of the voice which can either be male or female.

b) *PyAutoGUI*: PyAutoGUI is a python library that lets python code control the device's input peripherals, i.e., mouse and keyboard to automate interaction with other applications. It's compatibility does not vary with different versions of Python and works on Windows, macOS and Linux operating systems. This module is preferred due to its less dependencies.

c) *SQLIite3*: SQLite3 is a language that provides a lightweight database that does not require a separate server and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

d) *HTTP. Server*: This module defines classes for implementing HTTP servers.

e) *Speech Recognition*: this library is the standard Python library used to perform speech recognition online and offline. It supports multiple engines and APIs such as Google Speech recognition, Google Speech recognition API,

Microsoft Bing voice recognition and IBM Speech to text API.

f) *Urllib. Requests*: this module defines functions and classes that help open URLs (mostly HTTP).

g) *Random*: This module generates pseudo-random characters. There is the functionality to perform random operations by selecting an integer or a float value from a given or user generated list of items.

h) *OpenCV*: OpenCV is an open-source library used for computer vision applications such as video analysis, CCTV footage analysis and image analysis. OpenCV is written by C++ and has more than 2,500 optimized algorithms.

i) *Datetime*: In Python, date and time is a library file can be imported as a module named **datetime** that works with both dates and time. **Python Datetime module** comes built into Python3 and Python2, so there is no need to install it explicitely. Date and time are objects in Python, so when you manipulate them, you are manipulating objects and not string or timestamps.

j) *Tkinter*: Tkinter is the standard GUI library for Python3 and Python2. Python when combined with Tkinter provides a fast and easy way to create GUI applications. There are various features that are provided by Tkinter such as buttons, labels and text boxes used in GUI application. These are called widgets.

We used tkinter in this project to design the interface or the front-end of this assistant. Since, tkinter is a pre-installed library that comes with python 3 installation package, it is easy to implement and cross platform accessible. We created the front-end using tkinter because it allows the software to be light and occupies less memory space on both primary and secondary memory of the system, thus, making it least cumbersome GUI toolkit for the processor.

2) *SQLite*3: SQLite3 is a library that creates a serverless, transactional SQL database engine. SQLite3 engine is not a standalone process like other databases, you can link it statically or dynamically as per user requirements with the platform. SQLite accesses its storage files directly. This allows its integration into other systems and software more easily as there is no need to develop transaction access system to access data items and data files from the database. SQLite was integrated into this project to allow us to create a relational database. This database will hold all the data related to various aspects of the assistant's backend design both temporarily and permanently. This data may or may not include assistant's name and voice type, creator's log and licensing details, permission requirement and other dependencies, command syntax, keywords, file locations etc. Since, it is be relational database, all the data items are related to one another.

3) DB Browser: It is an open-source, fast, reliable and visual software which is used to create, design and edit databases using SQL and SQLite. It is visually appealing software that provides user with a simple spreadsheet like interface and the functionality to edit and update databases without using or learning complex SQL commands.

We used DB browser specifically because it allows remote update through the process of query cross-relation and referencing where another software can also access the database using some other tools or in our case some other language and command.

Now, that we understand and are familiar with the technology used, we need to understand how the data flows through the system, this will allow us to understand how different requests, commands and queries will be handled by the system.

Command Flow Chart

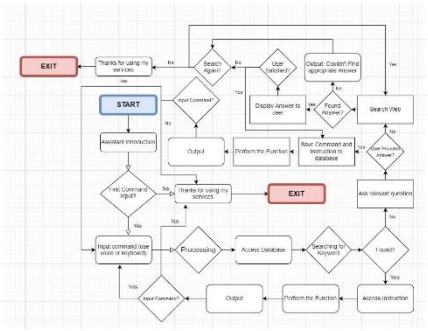

Fig. 1 Command Flow Chart

Points that we can infer from the data flow diagram are:

1) Assistant allows the user to choose whether they need to enter the command using voice recognition or by typing it using the keyboard.
2) The assistant will provide the user with a greetings message and an introduction message every time it is booted again. Introduction functionality allows user to know their assistant and its other functionalities inside out.
3) The database contains each and every command and the instruction corresponding to it, which means it will be accessed every time a new command is given to the assistant. This allows the database to be constantly updated.
4) The database is accessed by the assistant for both reading the data and retrieving the data. Moreover, it can also write the database if and when required.
5) A continuous chain of commands is presents that allows the assistant to be constantly active and keep listening to the user.
6) Manual self-learning is integrated in the system where, the assistant will ask user relevant questions if the command provided by the user does not match any keyword, instruction or statement already present in the database. These questions are also integrated manually resulting in less processing time.

7) If the user provides the assistant with relevant and logical answers to the questions asked the assistant will not only perform the task at that very moment but also remember it by storing the command and the instruction in the database for future references.
8) If the user is unable or unwilling to provide answers, the assistant is also capable of searching the web for the answers. If something relevant is found, the data is presented to the user for their approval, if approval received, task is performed and stored simultaneously, if not then the user is asked again to provide for the answers.
9) this loop continues, until and unless the task is performed or user exits the loop voluntarily.
10) If no relevant information regarding the command is found, the assistant will ask the user for the information, if provided, it will be executed and stored, if not the assistant will exit the loop.

Now, that we understand how different queries and commands are being executed, we will learn how specific problems will be dealt with.

Problem Solution and Technology Implementation

1) Platform Agnostic Software: To create a software which is accessible on any operating system without losing any functionality, there are certain pre-requisites that need to be fulfilled, these are:

a) The technologies used must be platform agnostic too. This will allow least number of sub-modules to be integrated.

b) The credentials of the technology must be original and malware free. Thus, making the software safe and secure.

c) The system and software requirements of the technologies must be simple and not over the top to allow the software to run on any device irrespective of era of manufacturing and/or hardware/software age.

d) The number of dependencies must be reduced to reduce the clutter of folders, sub-folders, by-folders and meta-data.

e) Permission-requirements must be defined clearly in the README file to allow user to take discretionary decisions.

These pre-requisites are resolved and taken care of in the following ways:

a) The use of Python 3 resolves most of the pre-requisites in the way that—

i. It is platform agnostic.
ii. It is available open-source.
iii. The credentials are certified from the Python organization itself.
iv. The language module is malware free, virus free and bloatware free.
v. The original package from the certified website comes pre-loaded with most libraries and modules used for the development.

b) Besides Python, SQLite is used for the design, development and maintenance of the database. This is because, A database in SQLite is a single disk file, and the file format is cross-platform i.e., a database created on one machine can be copied and used on different machine with a different architecture. SQLite databases are portable across 32-bit and 64-bit machines thus, reducing the number of dependencies. Moreover, like Python, SQLite is also available to download and install from its official website, which resolves certification and security requisites.

2) Amateur language syntax understandability: There are two ways in which this functionality can be integrated in the software, these are: -

a) Integration of a machine learning algorithm such as natural language processing and natural language understanding based on various models such as **BERT**, GPT2, XLNet, RoBERTa etc. But there are certain caveats related to these algorithms and models, such as—

i. Availability of training data.
ii. The time required to train the model.
iii. Phrasing ambiguities.
iv. Innate biases in the training dataset and training model itself.
v. Keeping the conversation moving, etc.

Apart from these there are system and machine-based caveats such as—

i. Processing time.
ii. Processing power.
iii. Processor engagement.
iv. Device power consumption.
v. Memory management and memory requirements, etc.

Which makes AI and machine learning model integration an erroneous choice.

b) Model based on manual machine learning using the database based on hit and try, QNA and questionnaire system. This system, although require more time to code on the backend, provides better results in the future because of following features—

i. Training data is basically the conversation that the user will have with the software. Therefore, we can say that it is available locally.
ii. Time required to train the model is variable and based on user.
iii. Zero to least ambiguities in phrasing as the sentences are formed according to language, syntax, and methodologies employed by the user.
iv. Since, there is no explicitly defined training model employed, biases reduce close to null.
v. Conversation keeps moving forward as long as user wants it to continue.

Furthermore, machine-based caveats are also reduced as—

i. Algorithm used is simple and less complex, thus reducing processing time and power.
ii. Processor engagement is reduced following the simple architecture.
iii. While memory requirement increases, memory management is relatively better by many folds.
iv. constant engagement might increase device power usage, it is balanced by the less usage of power by the processor.

All these advantages led to the integration of this model in the software.

3) Variable command understandability: This problem is resolved using the model employed to resolve the Amateur language syntax understandability problem only as the basis to solve the problem are similar. To understand this more efficiently consider this: -

A) Variable commands are a type of syntax error only, as the keywords used by the user does not match any keyword already available in the database.

B) Thus, the keywords are added following a simple QNA session with user, where the system asks, in reply of the variable command, what does it mean and the user replies with the right keywords.

Therefore, no other extra model is employed to resolve this problem.

4) Learning ability of the assistant: This problem or we can say feature is partially resolved/added using the same model which is used to resolve Amateur language syntax understandability and Variable command understandability problems. This is because: -

A) We anticipated a case or scenario where; the user might not provide or is incapable of providing the system with an appropriate solution or answer to the current problem or question.

B) In this scenario the software must be capable of searching external sources for the answer or solution such as the internet.

C) If the solution or answer is founded, it is presented to the user for approval and performed and stored if approval received, else it is either searched again or the loops is ended based on user will and discretion.

Thus, two models are employed to allow the assistant to learn.

5) Multi-tasking compatibility: multi-tasking is integrated in the system in such a way that if a task is given to the user while the processor and the operating system is already engaged in doing some other tasks provided by the user, the performance or actions of the assistant's system will not affect or impede with the performance or the actions of the other task provided by the user.

This particular functionality is integrated using the concept of a virtual environments. A virtual environment is a functionality that creates an isolated and segregated environment within the main system only, where all the required libraries, paths, modules and database of the implemented python script is stored. This environment being segregated from all other files on the system do not hamper with the natural functioning and services of the system. This also allows the script to run and, send and receive requests to and from the processor distinctively. Thus, increasing the processor efficiency by intuitive processor organization. This module and functionality are available with Python version 3.0 and above and is compatible with Windows, macOS and Linux operating systems.

Therefore, we can say that the approach to the problem, although kept as simple as possible, is still reasonably complicated in terms of implementation and optimization.

IV. TEST CASES

TEST ID	TEST DESCRIPTION	EXPECTED OUTCOME	ACTUAL RESULT	PASS/FAIL
TC_01	Checking cross platform accessibility of the assistant	Accessible on Windows and MacOS	The Assistant worked well with both OS, with some minor modifications	PASS = 1
TC_02	Amateur language syntax understandability	Assistant can understand mispronunciation and Indian Accent	A command with inappropriate syntax was given to the assistant which was performed successfully. See fig. 2	PASS = 1
TC_03	Variable command understandability	Assistant can understand variable commands	A task was provided 3 times to the assistant using 3 different syntaxes and it performed the task each time. See fig. 3	PASS = 1
TC_04	Learning ability of the assistant	Assistant learnt a new command by cross-questioning the User	A new task was provided to the assistant and it learnt it using the cross-questioning method. See fig. 4	PASS = 1

TC_05	Multi-tasking	The assistant was capable of doing task separately without affecting the performance of the system	The assistant was performing well with other tasks ongoing	PASS = 1

TABLE I. Test Cases

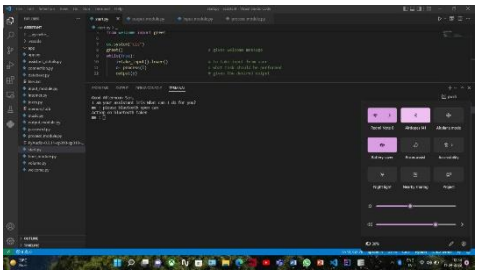

Fig. 2 TC_02

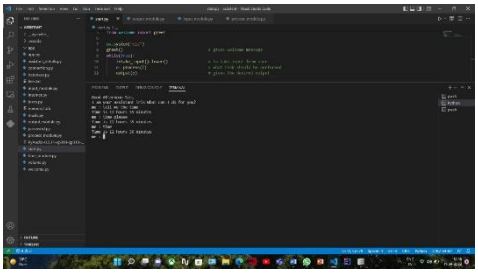

Fig. 3 TC_03

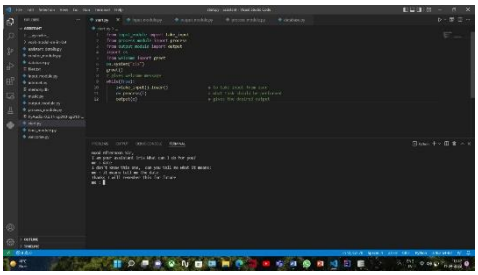

Fig. 4 TC_04

V. RESULT AND ANALYSIS

The aim of this project at the time of commencement was to make various simple and complex tasks that a user might need or want to perform on an immobile or a partially immobile desktop device such as a PC or a Laptop completely hands free and the solution that we discovered was to develop a Voice assistant. A voice assistant is a software which works by taking an input from the user in the form of a voice command, processes the command and provides the user with an output. Now, while moving forward with the research we came across several other works that have already been done in this particular department by massive multinational corporations such as Microsoft, Google, Samsung and Amazon, out of which we discovered that Microsoft was one of the very first contributor with its very own voice assistant called Cortana. As we researched further, we came to understand that Cortana was definitely not the best solution as it had many flaws. Our aim now was to work on these particular flaws only.

Now, consider the test cases discussed in preceding section, it is easy to infer that, we were able to resolve and present result to all of the problems that were discussed earlier in the Problem Formulation section [II]. There were some problems that were resolved using one model only. There were some problems that require the implication of some libraries and modules explicitly. These libraries and modules and some other pieces of technology which were not available with the standard package will be mentioned under the dependencies section in the README file of the project. The certified links to these dependencies will be provided to the user with the process to resolve them, also mentioned under the README file.

Analyzing the test cases, we came across certain observations such as: -

1) The backend and the frontend of the software is coded on the Visual Studio Code software, which is an open-source Interactive Development Environment, developed by the Microsoft Corporation.
2) The testing is done on a PC device which is running on Windows 11 operating system.
3) The testing is done inside the terminal window of the Windows 11 operating system knows as the Command Prompt.
4) It is also evident that the assistant presents the solution and the statements in not only the audio format but also in the textual format where each task is acquainted to the user in both visual and audio format before it is being performed.
5) The frontend of the software is kept relatively simpler with only fundamental buttons at dispose and a minimalistic color scheme.

Therefore, we can claim that the desired results were achieved in the manner they were perceived to achieve in the first place.

VI. CONCLUSION

In this age of Creators, designers and developers we are using technology to move out and reach the most isolated parts of the world. This not only brought the world closer in terms of sharing culture and community but also art and talent as creative people are now their own boss. In this age we are now in need of machine that not only do the job for us, but also do it as efficiently as they can and this is where our research come in.

During our research we discovered that the voice assistant for desktops department was in dire need of an update after the massive failure of Microsoft's Cortana. Some of the major flaws that we concluded needed fix were the software's ability to be cross-platform accessible, understand amateur syntax, mispronunciations, accents and other speech errors, understand

commands that were given using a different syntax, learn rapidly and adjust its functionalities according to user needs, etc.

These issues were dealt with individually using various different algorithms and models. The language that we used to develop this assistant was Python, both the backend and the frontend, and the Database is developed using SQLite. Through our rigorous testing we also concluded that all of the functionalities were working fine and the assistant seemed to be functioning perfectly. These problems, although, seem to have been eradicated completely still need regular check and maintenance and updating, if and when deemed necessary.

We would also like to acknowledge that, although, we tried our best to resolve every major design, architecture, functionality and implementation flaw that we discovered and came across, there is still scope of development and improvement in the technology in terms of model and algorithms used in the backend for the training purpose of the assistant, the architecture of the database used and/or the design language and style for the frontend.

VII. REFERENCES

[1] Harshit Agrawal, Nivedita Singh, Gaurav Kumar, Dr. Diwakar Yagyasen, Mr. Surya Vikram Singh, "Voice Assistant Using Python," International Journal of Innovative Research in Technology, Volume 8, Issue 2, Page(s): 419 – 423, 2021.

[2] George Terzopoulos, Maya Satratzemi, "Voice Assistants and Smart Speakers in Everyday Life and in Education," Informatics in Education, Vol. 19, No. 3, 473–490, 2020.

[3] Abhay Dekate, Chaitanya Kulkarni, Rohan Kiledar, "Study of Voice Controlled Personal Assistant Device," International Journal of Computer Trends and Technology (IJCTT), Volume 42 Number 1, 2016.

[4] Deepak Shende, Ria Umahiya, Monika Raghorte, Aishwarya Bhisikar, Anup Bhange, "AI Based Voice Assistant Using Python," Journal of Emerging Technologies and Innovative Research, Volume 6 Issue 2, 2019.

[5] Vishal Kumar Dhanraj, Lokeshkriplani, Semal Mahajan, "Research Paper onDesktop Voice Assistant," International Journal of Research in Engineering and Science, Volume 10 Issue 2, 2022, PP. 15-20.