E.C.H.O. VOICE ENABLED VIRTUAL ASSISTANT

A PROJECT REPORT
for
Mini Project (KCA353)
Session (2024-25)

Submitted by

Shivani Sahu (2300290140173)
Shristi Bhatt (2300290140178)
Vinayak Saxena (2300290140202)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Ms. Annu Yadav
Assistant Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(DECEMBER 2024)

CERTIFICATE

Certified that **Shivani Sahu (2300290140173), Srishti Bhatt (2300290140178), Vinayak Saxena (2300290140202)** have carried out the project work **E.C.H.O. Voice Enabled Virtual Assistant** (**Mini-Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/ herself, and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Ms. Annu Yadav
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head of
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

E.C.H.O Voice Enabled Virtual Assistant
ABSTRACT

Project ECHO is an advanced virtual assistant designed to seamlessly integrate into daily routines, providing users with a versatile tool for managing various tasks through voice commands. Leveraging sophisticated speech recognition and text-to-speech technologies, ECHO interprets and responds to user inputs with remarkable accuracy and efficiency. This project explores the functionalities and applications of ECHO, showcasing its capabilities in opening websites, playing music, providing the current time, and sending emails.

ECHO's core strength lies in its ability to understand and process natural language commands, making it an intuitive and user-friendly assistant. By utilizing Google's Speech Recognition API, ECHO captures and deciphers voice inputs, while the pyttsx3 library enables it to deliver clear and responsive auditory feedback. This combination of technologies ensures a smooth and interactive user experience, enhancing the practicality of the assistant in real-world scenarios.

One of the standout features of Project ECHO is its email-sending functionality. The assistant guides users through the email composition process, prompting for the subject, body, and recipients, and successfully sending the email via Gmail's SMTP server. This feature exemplifies ECHO's potential to streamline communication tasks, saving users time and effort.

Overall, Project ECHO represents a significant step forward in the development of intelligent virtual assistants. Its ability to handle a wide range of tasks with precision and ease makes it a valuable asset for users seeking to enhance their productivity and manage their daily activities more effectively. The project demonstrates the practical applications of AI-powered assistants and underscores the potential for future advancements in this field.

ACKNOWLEDGEMENT

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Annu Yadav** for her guidance, help, and encouragement throughout our project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Finally, Our sincere thanks go to our family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep our life filled with enjoyment and happiness.

Shivani Sahu

Srishti Bhatt

Vinayak Saxena

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 BACKGROUND

Virtual Assistants have evolved significantly over the years, starting from simple chatbots to sophisticated and advanced AI-powered systems. The journey began in the 1960s with ELIZA, a rudimentary chatbot created by Joseph Weizenbaum at MIT, which simulated conversations with a psychotherapist. In the 1990, the advent of personal digital assistants (PDAs) like the Apple Newton and Palm Pilot allowed users to manage personal information, setting the stage for more advanced virtual assistants.

The real breakthrough came with the introduction of voice-activated assistants. Apple's Siri, launched in 2011, brought voice recognition and AI capabilities into mainstream use, followed by Google Now in 2012 and Microsoft's Cortana in 2014, Amazon's Echo, powered by Alexa, revolutionized the smart speaker market and further integrated virtual assistants into daily life.

Let's dive deeper into the background of virtual assistants.

- **Early Innovations:**

  The concept of virtual assistants began with the development of early conversational agents. The first significant milestone was ELIZA in 1960s, ELIZA was a simple chatbot that mimicked human conversation by using pattern matching and substitution methodology, simulating a psychotherapist's interaction. Although limited in functionalities, ELIZA laid down the ground work for future advancements.

- **Evolution of PDAs:**

  In the 1990s, Personal Digital Assistants (PDAs) emerged as a new category of handheld devices. PDAs like the apple Newton and Palm Pilot enabled users to manage personal information such as contacts, calendars, and notes. These devices were the precursors to more advanced virtual assistants, providing users with a glimpse into the potential of having a digital assistant to organize and streamline their daily tasks.

- **Evolution of Voice-Activated Assistants:**

  The introduction of voice-based virtual assistants marked a significant turning point, Apple's Siri launched in 2011 revolutionized the way people interacted with their devices by allowing voice commands to perform various

tasks. This innovation was soon followed by Google Now in 2012, which utilized Google's vast data resources to provide personalized information and recommendations.

- **Modern Virtual Assistants:**

  Virtual assistants are integral parts of modern life these days. They rely on highly advanced AI techniques, such as machine learning and natural language understanding, to provide the best and context-aware assistive support. Virtual assistants are now able to support complex tasks ranging from management of schedules and sending of messages to providing real-time traffic updates and answering knowledge-based queries. Their advancement is a continuous process supported by research in AI as well as by the continuously growing demand for seamless intelligent user experiences.

1.2 NEED AND SIGNIFICANCE

Following are the needs or importances of the Virtual Assistant

- **Addressing Modern Challenges:**

  With the rapid growth of technology and the increasing complexity of daily life, there is a growing need for intelligent systems that can help manage information, streamline tasks, and improve productivity. Virtual Assistants like E.C.H.O. are designed to meet this need by providing users with a seamless and efficient way to interact with technology and access information.

- **Enhancing Productivity:**

  One of the primary benefits of virtual assistants is their ability to enhance productivity. By automating routine tasks such as setting reminders, managing schedules, and retrieving information, virtual assistants allow users to focus on more important and complex activities. This can lead to significant time saving and increased efficiency, both in personal and professional settings.

- **Accessibility and Convenience:**

  Virtual assistants make technology more accessible and convenient for a wide range of users. The voice-activated features allow people to work hands-free, which is quite helpful for people with disabilities or when working with several tasks simultaneously. Virtual assistants can also offer quick and easy access to information, making everyday tasks manageable.

- **Personalization and Context Awareness:**

  Modern virtual assistants are equipped with advanced AI capabilities that enable them to learn user preferences and provide personalized experiences. Through the understanding of the context of user interactions, virtual assistants can be more relevant and accurate in response, thus enhancing

the general user experience. This can lead to more meaningful and satisfying interactions with technology.

- **Integration and Smart Devices:**

  The proliferation of smart devices in homes and workplaces has created a demand for centralized systems that can control and manage these devices. Virtual assistants like E.C.H.O. serve as a central hub, enabling users to control their smart devices through a single interface. This integration simplifies the management of smart homes and offices, making technology more intuitive and user-friendly.

- **Future Potential:**

  It would be a significant development if assistants are considered the next advancement in the development of artificial intelligence. This area can be seen to transform human interaction with technology ov er the years. Future improvements might lead to even more sophisticated and intelligent virtual assistants who are capable of performing complex tasks and making autonomous decisions. Continued research and development in this field promise to provide a wide variety of uses in all walks of life, from health care and education to business and entertainment.

1.3 OBJECTIVE OF THE PROJECT

The main objective of the E.C.H.O. virtual assistant project is to develop an intelligent, user-friendly system that enhances productivity, accessibility, and user experience. Here are the detailed objectives.

- **Enhance Accessibility:**

  The project is designed to be accessible to a broad audience and especially to the disabled population. E.C.H.O. will have voice-controlled access, text-to-speech capabilities, and other intuitive interfaces that enable the user to interact with technology without touching it and without using complicated inputs.

- **Improve User Productivity:**

  E.C.H.O. aims at automating daily tasks and activities by providing a smooth, easy, and efficient interface in accessing schedules, reminders, and information. By streamlining daily tasks, E.C.H.O. enables the person to focus on other issues that are more complex or important, thus raising overall productivity.

- **Provide Personalized Experiences:**

  One of the most important objectives is to provide personalized, context-aware support. E.C.H.O. will utilize cutting-edge AI algorithms to understand the preferences,

habits, and patterns of users so as to deliver well-timed, context-sensitive recommendations and responses to maximize the experience.

- **Seamless Integration with Smart Devices:**

  E.C.H.O. will act as a central hub for controlling and managing smart devices within homes and offices. The project aims to enables seamless integration with a wide range of smart devices, allowing users to control their environment through a single interface. This includes managing lighting, temperature, security systems, and more.

- **Enhance Natural language Understanding:**

  A significant objective of the project is to improve natural language processing and understanding capabilities. E.C.H.O. will be designed to interpret user commands and queries with the most accuracy, thus allowing the interactions to be more natural and intuitive. This requires further development of the system's ability to understand context, intent, and sentiment.

1.4 PURPOSE

The primary goal of the E.C.H.O. virtual assistant project is to develop an intelligent, responsive, and user-friendly system designed to enhance daily productivity and accessibility by a considerable margin for its users. E.C.H.O. will automate tasks like setting reminders, scheduling appointments, and retrieving information to streamline everyday activities and let users focus on more important things. In addition, the project aims to offer personalize and context-aware support, relying on sophisticated AI algorithms to understand and adapt to user preferences to deliver customized experiences that improve user satisfaction.

E.C.H.O. also aims to integrate seamlessly with smart devices, serving as a central hub for managing smart homes and offices, thereby simplifying the control of various devices through a single, intuitive interface. The project emphasizes the advancement of AI and natura language processing capabilities to ensure accurate and natural interactions. Ensuring robust data security and privacy is another key Objective, providing users with confidence in the safety of their personal information. Finally, E.C.H.O.is designed to support continuous learning and adaptation, using machine learning techniques to improve overtime and remain effective in meeting evolving user needs. Ultimately, the project seeks to make technology more accessible, convenient, and beneficial for a diverse range of users, enhancing their interaction with digital environments.

1.5 COMPONENTS USED

- **Hardware Components**
  a. **Microcontroller/Processor:** The central processing unit that runs the virtual assistant software. Depending on the setup, this could be a personal Computer, a microcontroller capable of handling voice processing and integration with other components.
  b. **Microphone:** A crucial component for capturing voice commands from the user. It should be of high quality to ensure accurate voice recognition, ideally with noise-cancelling features to filter out background noise.

c. **Speakers:** Used to provide audible feedback to the user. These can be built-in speakers or external speakers connected to the microcontroller/processor.

- **Software Components**
  a. **Operating system:** It refers to the base software responsible for managing hardware and software resources. E.C.H.O uses Windows whereas, other OS could be macOS, Raspberry Pi or Linux.
  b. **Python:** The programming language used for developing the virtual assistant is Python because it happens to be a very simple and has more extensive libraries used in various functionalities.
  c. **Speech Recognition Library:** To open websites based on user commands.
  d. **Web Browser Library:** To open websites based on user commands.
  e. **Datetime Module:** To retrieve and provide current time information.
  f. **SMTP Library:** To send emails through the simple mail transfer protocol.

- **External Services**
  a. **Google Speech recognition API:** Used by the speech_recognition library to convert speech to text.
  b. Gmail SMTP Server: For sending emails through Gmail.

CHAPTER-2

PROBLEM STATEMENT AND LITERATURE SURVEY OF THE TECHNOLOGIES

2.1 PROBLEM STATEMENT

The advancement of artificial intelligence (AI) has made virtual assistants indispensable tools in improving human-computer interaction. However, mainstream systems like Siri, Alexa, and Google Assistant, while impressive, reveal several limitations:

1. **Lack of Customization**: These systems are often tightly coupled with their proprietary ecosystems, making it challenging to adapt them for specific user needs or use cases.
2. **High Hardware Dependency**: Many virtual assistants require sophisticated hardware setups, limiting accessibility for users with minimal configurations.
3. **Task Restrictions**: Existing assistants prioritize general-purpose tasks, with limited capability to handle desktop-specific applications or automate custom workflows.
4. **Error Management**: Challenges persist in accurately handling diverse user inputs, especially in noisy or dynamic environments.

To address these limitations, the **E.C.H.O. (Efficient Computational Human-like Oracle)** project proposes a scalable, Python-based voice assistant that focuses on:

- Enhancing **accessibility** for desktop users with basic hardware requirements.
- Providing **modularity and scalability**, enabling the addition of new features without extensive reconfiguration.
- Ensuring robust **task automation** by integrating natural language processing (NLP) and text-to-speech (TTS) technologies.
- Delivering **error management** mechanisms to handle diverse user inputs effectively.

E.C.H.O. aims to create an adaptable, lightweight, and highly practical voice-enabled assistant to enhance productivity in both personal and professional environments.

2.2 LITERATURE REVIEW

Table 2.1: Research Paper

AUTHOR	YEAR	TITLE	CONCLUSION
Sakshi R Jain, Prof. Feon Jason	2023	Personal Desktop Voice Assistant	In this paper, the benefits and shortcomings of Personal desktop assistant are discussed.
Yuqi Huang	2023	Research on the development of voice assistants in the era of AI.	This paper discusses the booming and broad development status of artificial intelligence voice assistant and how the use of voice assistant differ country to country.
Rodrigo Pereira, Claudino Lima, Et el.	2023	Virtual Assistants in industry 4.0	This paper explores the use of Vas in the industry 4.0 discussing the technical assistance design principle and identifying the characteristics of VA.
Yanchu Guan, Dong Wang, Et el.	2023	Intelligent Virtual Assistant with LLM process automation.	This paper proposes a novel LLM based Virtual assistant that can automatically perform multi-step operations within mobile app based on high level requests.

AUTHOR	YEAR	TITLE	CONCLUSION
Sitti Rachmawati Yahya, S. N. H. Sheikh Abdullah, K. Omar	2023	Design and Implementation of a VoIP PBX Integrated Vietnamese Virtual Assistant: A Case Study	Focused on integrating virtual assistants with VoIP systems, showcasing adaptability and customization for localized use cases.
Deepika Ghai	2021	Enhancing User Experience with Virtual Assistants: A Review	Analyzes techniques to improve user experience, focusing on accessibility and ease of interaction with virtual assistants.
Rajat Kumar	2020	Text Extraction and Document Image Segmentation Using Matched Wavelets and MRF Model	Explores methods to extract and process text from images, relevant to NLP tasks in virtual assistants for structured data extraction.
Shoba Natesh	2022	Restoration of Deteriorated Text Sections in Ancient Document Images Using a Tri-Level Semi-Adaptive Thresholding Technique	Demonstrates methods for processing noisy or degraded inputs, applicable for improving speech recognition accuracy in virtual assistants.

AUTHOR	YEAR	TITLE	CONCLUSION
Syed Nawaz Pasha, D. Ramesh, D.Kodhandaraman, M.D. Salauddin	2019	Research to Enhance the Old Manuscripts Resolution Using Deep Learning Mechanism	Examines the application of deep learning for data restoration, which can inspire future enhancements in NLP and TTS for virtual assistants.
Disha Bhatt	2017	Use of Adaptive Methods to Improve Degraded Document Images	Proposes adaptive approaches for enhancing data quality, useful in noise reduction for speech recognition systems in virtual assistants.
Priyanka Udaya Bhanu	2016	Segmentation of Text from Badly Degraded Document Image	Highlights techniques for parsing and segmenting text from degraded sources, relevant for NLP modules in E.C.H.O.
Anoop Joshi	2007	Enhancement of Old Manuscript Images	Discusses image enhancement techniques to improve clarity, indirectly relevant for handling noisy data in virtual assistants.

CHAPTER-3

REQUIREMENTS AND ANALYSIS

3.1 PLANNING AND SCHEDULING

The development of the E.C.H.O. virtual assistant was planned across 12 weeks, dividing the work into manageable tasks and milestones.

WEEK 1

- Forming a group of four members.
- Brainstorming ideas and selecting the theme of the project.
- Initial discussion on the scope and target functionalities of E.C.H.O.

WEEK 2

- Finalizing the project title: *E.C.H.O: Voice-Enabled Virtual Assistant*.
- Researching technologies like Python, SpeechRecognition, NLP, and TTS.
- Preparing the feasibility study for the project implementation.

WEEK 3

- Defining the objectives and expected outcomes of the project.
- Preparing and presenting the project synopsis in the first presentation.
- Outlining a basic plan for modules and functionalities.

WEEK 4

- Researching existing virtual assistants and identifying their limitations.
- Creating the architecture diagram for E.C.H.O. with modular design principles.
- Finalizing the list of tasks and their dependencies.

WEEK 5

- Setting up the development environment (Python installation, IDE setup).
- Installing required Python libraries such as SpeechRecognition, pyttsx3, and NLTK.
- Developing the initial prototype of the Speech Recognition module.

WEEK 6

- Developing the Text-to-Speech (TTS) module for audio responses.
- Writing basic scripts to capture user input and deliver responses.
- Testing the Speech Recognition and TTS modules independently.

WEEK 7

- Developing the Natural Language Processing (NLP) module to interpret commands.
- Creating logic for command parsing and extracting actionable items (e.g., recipient, message).
- Testing the NLP module with simple commands.

WEEK 8

- Integrating all modules (Speech Recognition, TTS, NLP) into a cohesive system.
- Adding task-specific functionalities like sending emails and opening applications.
- Conducting integration testing to identify bugs and improve efficiency.

WEEK 9

- Conducting user testing to evaluate performance in different environments (quiet vs. noisy).
- Fine-tuning error-handling mechanisms for incomplete or ambiguous commands.
- Preparing the second project presentation to showcase progress.

WEEK 10

- Improving the system's scalability by adding modular components.
- Adding additional features like retrieving the current date and time.
- Preparing the documentation for completed modules.

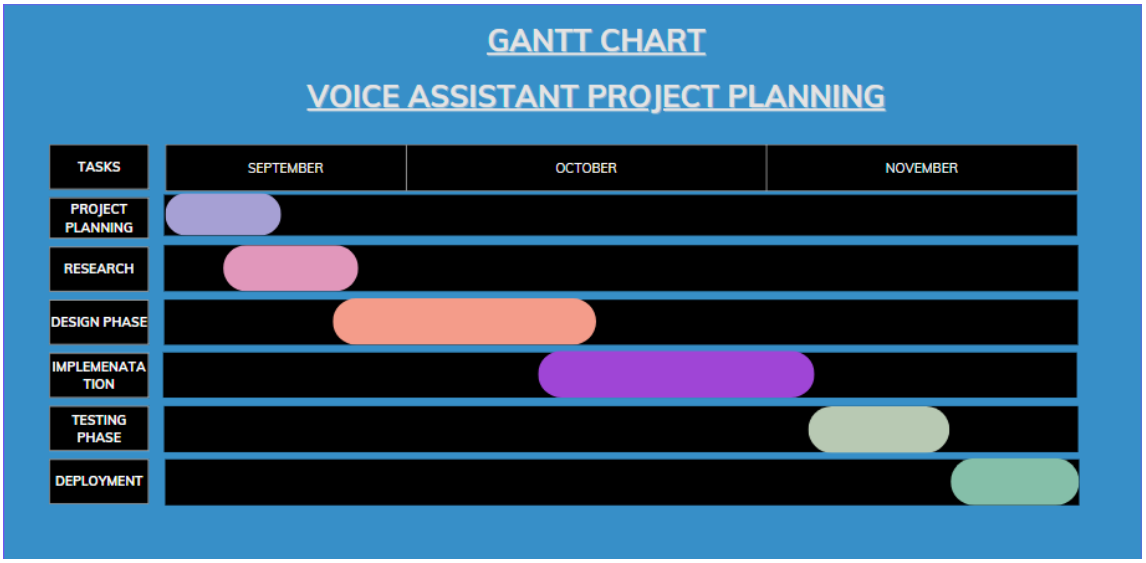
WEEK 11

- Finalizing the prototype with complete functionality.
- Preparing the project report draft, including results and challenges.
- Conducting extensive testing to ensure stability and reliability.

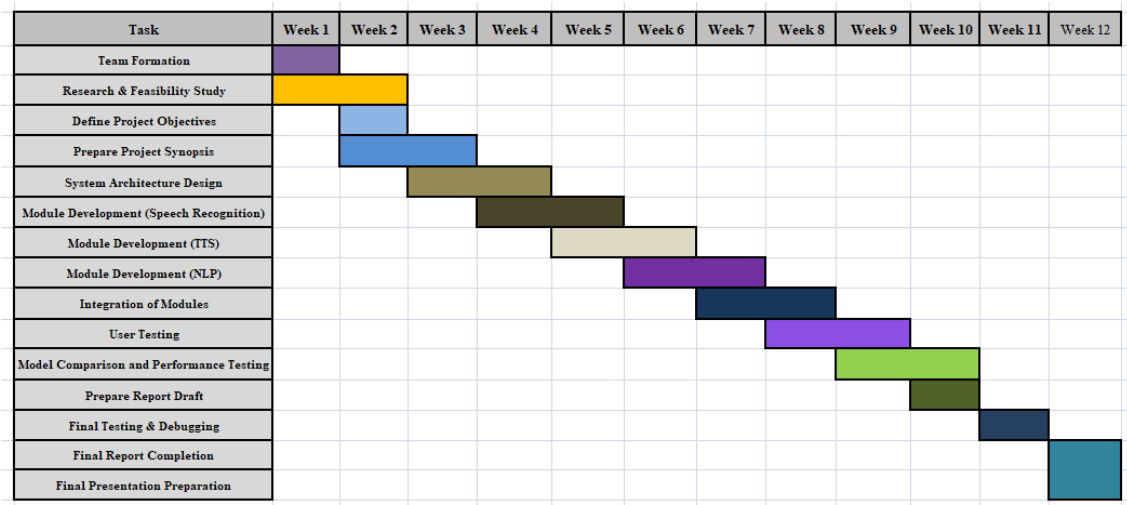
WEEK 12

- Completing the final project report and research paper.
- Reviewing and polishing the project for final submission.
- Presenting the finalized system in the concluding presentation.

3.1.1 Gantt Chart



(Figure 3.1: Gantt chart)



(Figure 3.2: Gantt chart)

3.2 SOFTWARE AND HARDWARE REQUIREMENTS

3.2.1 Hardware Requirements

The project was designed to operate efficiently on basic hardware, making it accessible to a wide range of users.

- **PC/Laptop**:
  - Processor: Intel i3 or above.
  - RAM: 4 GB or higher.
  - Storage: At least 500 MB of free space.
  - Display: Dual XGA (1024 x 768) or higher resolution.
- **Peripherals**:
  - Microphone: For capturing user voice input.
  - Speakers: For audio feedback.
- **Network**: Stable Ethernet/Wi-Fi connection for online tasks.

3.2.2 Software and Library Requirements

3.2.2.1 Software

- **Python:** Python is a versatile, high-level programming language created by Guido van Rossum in 1991. Known for its simplicity, readability, and extensive standard library, Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming.
  - **Applications**: Used in artificial intelligence, machine learning, data analysis, automation, and scripting.
  - **Strengths**: Its adaptability, large ecosystem of third-party libraries, and strong community support make it suitable for developers at all levels.
- **HTML**: HTML (HyperText Markup Language) is essential for structuring web content.
  - **Role in Project**: Though E.C.H.O. is desktop-based, HTML can be used for supplementary documentation or interfaces if the project expands to web-based functionality.
  - **Features**: Simplicity, browser compatibility, and user-friendly syntax.
- **CSS:** CSS (Cascading Style Sheets) is used to enhance the aesthetics and layout of web content.
  - **Role in Project**: Useful for designing intuitive and accessible interfaces in potential web-based extensions of E.C.H.O.
- **Flask**: Flask is a lightweight Python web framework.

- **Role in Project**: Ideal for integrating server-side functionalities, should E.C.H.O. evolve to include web-based interactions.

3.2.2.2 Libraries

- **PIP:** PIP is the default package manager for Python. It is a command-line tool that allows you to easily install, manage, and upgrade Python libraries and dependencies. It is essential for managing the third-party packages you use in your projects, and is commonly used in Python development for installing libraries from the Python Package Index (PyPI).
- **NumPy:** NumPy is the core package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. This library is essential for handling the data and performing various numerical operations, especially if your virtual assistant needs to process data (e.g., statistical analysis or large datasets).
- **Pandas:** Pandas is a powerful library used for data manipulation and analysis. It provides data structures like DataFrames and Series, which are ideal for handling and analyzing structured data (such as user commands, logs, or configurations).
- **Scikit-learn:** Scikit-learn is a popular machine learning library in Python. It provides simple and efficient tools for data analysis and modeling, such as classification, regression, clustering, and dimensionality reduction. While the main focus of E.C.H.O. is voice interaction, Scikit-learn can be used to enhance functionalities, such as predicting user actions based on prior behavior or improving decision-making processes.
- **Speech Recognition:** This library is used for converting spoken language (voice input) into text. It supports several engines, including Google Web Speech API, which helps to transcribe user speech into actionable text commands.
- **Natural Language Processing (NLP):** NLTK is used to process and interpret the text commands captured by the Speech Recognition module. It helps in tokenizing, parsing, and extracting actionable data from user input (such as identifying the action to perform: "send an email", "play music", etc.).
- **Text-to-Speech (TTS):** This library is used to convert text back into speech, allowing E.C.H.O. to speak responses or perform actions like confirming task completion to the user. It works offline and is customizable for voice pitch and rate.

If Python and pip are already installed, the following commands may be put into the command line to install these libraries –

- pip install pandas

- pip install numpy
- pip install spacy
- pip install pyttsx3
- pip install nltk
- pip install SpeechRecognition
- pip install requests
- pip install matplotlib
- pip install seaborn
- pip install scikit-learn

Once installed, you may use these libraries' features by importing them into your Python project. For instance –

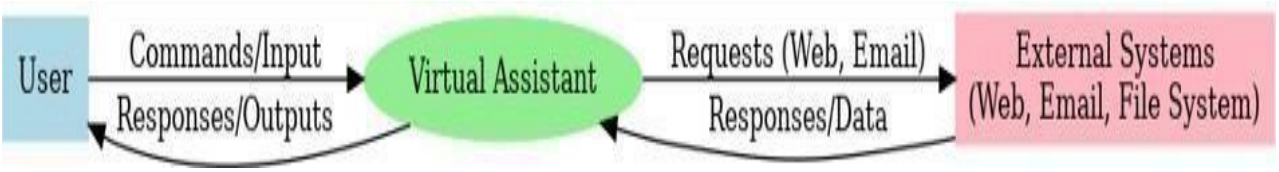
- import SpeechRecognition as sr
- import pyttsx3 import nltk
- import spacy
- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns from sklearn
- import datasets
- import requests

CHAPTER-4

SYSTEM DESIGN

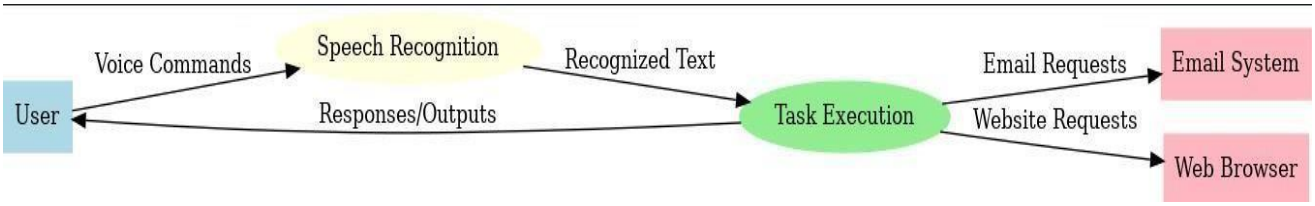
4.1 DATA FLOW DIAGRAM

This DFD is for the Speech Recognition System (ECHO), which enables users to input voice commands that the system processes and converts into actionable text. The Speech Recognition System comprises key entities like User, Speech Processor, Acoustic Model, Language Model, and Command Handler. It handles voice input, processes the audio signal, and provides actionable output in text or commands.



(Fig 4.1: DFD level 0)

This DFD is for the Speech Recognition System (ECHO), which enables users to input voice commands that the system processes and converts into actionable text. The Speech Recognition System comprises key entities like **User**, **Speech Processor**, **Acoustic Model**, **Language Model**, and **Command Handler**. It handles voice input, processes the audio signal, and provides actionable output in text or commands.



(Fig 4.2: DFD level 1)

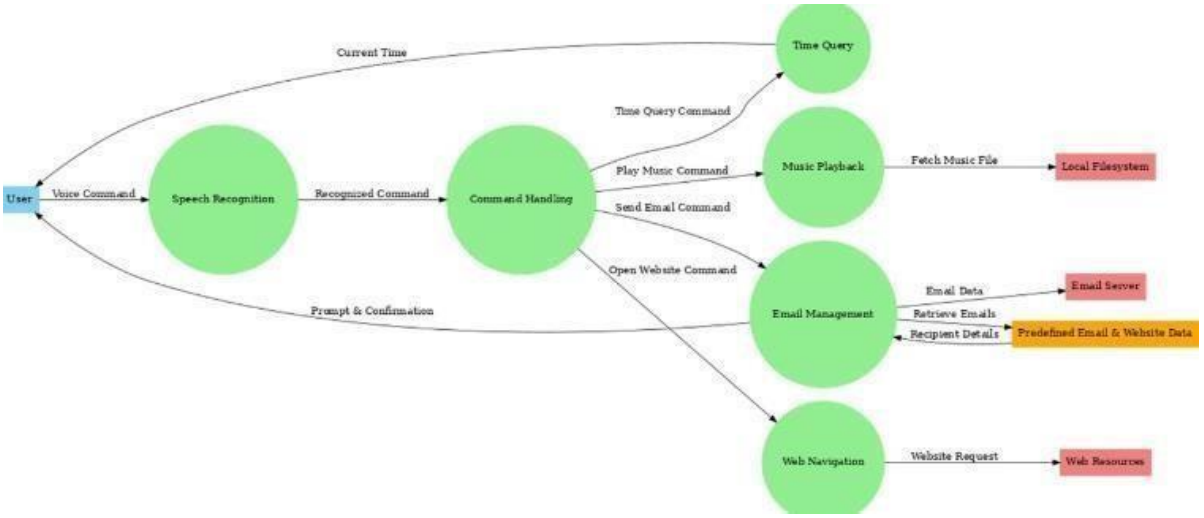
This DFD represents the internal processes of the Speech Recognition System. The input speech signal undergoes several stages:

- **Speech Analysis:** Captures and processes the input audio signal.

- **Feature Extraction:** Extracts meaningful features (e.g., frequency, pitch) from the audio data.
- **Recognition Module:** Utilizes acoustic and language models to decode the audio features into text.
- **Command Handling:** Translates recognized text into actionable commands or responses. The system interfaces with the **Acoustic Model** for sound patterns and the **Language Model** for contextual understanding.

At this level, the DFD focuses on the Speech Processing module. It involves the following steps:

- Signal Acquisition: Captures the speech signal.
- Digital Signal Processing: Prepares the audio for feature extraction.
- Feature Matching: Matches extracted features against the acoustic and language models.
- Result Generation: Provides text output or actionable commands based on recognition results.



(Fig 4.3: DFD level 2 Dashboard)

4.2 USE CASE DIAGRAM

Actors:

- User: A user is a type of participant who interacts with the crowdfunding application. They can either create campaigns to raise funds or invest in existing campaigns.
- Blockchain System: The underlying blockchain network where smart contracts are executed, transactions are processed, and campaign details are stored.

- MetaMask Wallet: is a type of cryptocurrency wallet that connects people to the blockchain through safe transactions and monetary handling.

Use Cases:

The Use Case Diagram provides a visual representation of how users interact with the Speech Recognition System (ECHO). It highlights the key functionalities and the relationships between the system components and external actors.

Actors

- User: The primary participant who interacts with the system by providing voice commands and receiving responses.
- Speech Recognition System: Processes the user's input, recognizes the speech, and performs the corresponding tasks.
- External Services:
    - Email System: Used to send emails based on user input.
    - Web Browser: Opens specified websites as per user commands.
    - Media Player: Plays predefined music files.
    - Clock/Time Management: Responds with the current time.

Use Cases

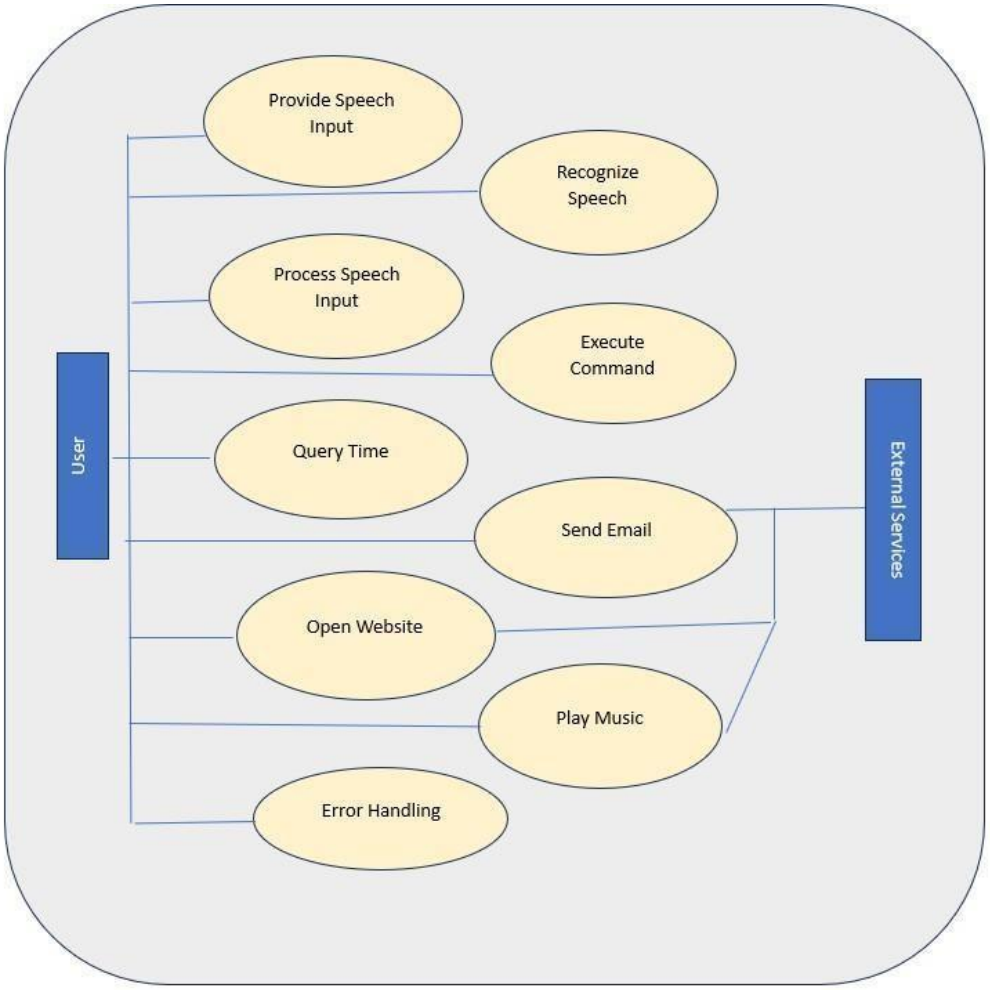
- **Provide Speech Input:**
    - Description: The user speaks into the microphone, providing input to the system.
    - Actor Involved: User.
    - Outcome: The system captures the audio signal for processing.
- **Process Speech Input:**
    - Description: The system processes the audio signal, removing noise and extracting meaningful features for analysis.
    - Actors Involved: Speech Recognition System.
    - Outcome: Pre-processed audio is ready for recognition.
- **Recognize Speech:**
    - Description: The system matches the extracted features with its trained acoustic and language models to recognize the spoken words.
    - Actors Involved: Speech Recognition System.

- Outcome: The recognized text or command is generated.

- **Execute Command:**
  - Description: Based on the recognized command, the system performs the appropriate action (e.g., opening a website, sending an email, playing music).
  - Actors Involved: Speech Recognition System, External Services.
  - Outcome: The requested action is executed successfully.
- **Query Time:**
  - Description: The user asks for the current time, and the system provides it in response.
  - Actors Involved: User, Speech Recognition System.
  - Outcome: The current time is announced to the user.
- **Send Email:**
  - Description: The user dictates an email, specifying the recipient, subject, and body. The system sends the email via the connected email service.
  - Actors Involved: User, Speech Recognition System, Email System.
  - Outcome: The email is delivered successfully.
- **Open Website:**
  - Description: The user instructs the system to open a specific website.
  - Actors Involved: User, Speech Recognition System, Web Browser.
  - Outcome: The website is opened in the browser**.**
- **Play Music:**
  - Description: The user requests the system to play music. The system opens the predefined music file.
  - Actors Involved: User, Speech Recognition System, Media Player.
  - Outcome: Music is played successfully.
- **Error Handling:**
  - Description: If the input is unclear or the system encounters an error, it prompts the user to repeat the command or correct the input.
  - Actors Involved: User, Speech Recognition System.
  - Outcome: The system retries or provides feedback on the issue.

Relationships

- The User directly interacts with the Speech Recognition System by providing input and receiving feedback or actions.
- **The Speech Recognition System:**
    - Processes user input through the Speech Processor.
    - Matches features with the Acoustic Model and Language Model to recognize the spoken words.
    - Maps recognized commands to actions through the Command Handler.
- **The Speech Recognition System communicates with External Services like:**
    - Email System for sending emails.
    - Web Browser for opening websites.
    - Media Player for playing music.



(Fig 4.4: Use Case Diagram)

4.3 CLASS DIAGRAM

The UML class diagram for the ECHO Speech Recognition System highlights three primary classes: Assistant, EmailManager, and Command. These classes collaborate to ensure the smooth functioning of the virtual assistant by enabling functionalities such as speech recognition, task execution, email management, and user interaction. The relationships between these classes illustrate the system's core structure and functionality.

Assistant Class

The Assistant class serves as the central component of the system, implementing core functionalities for user interaction and command execution. This class interacts with other components like the EmailManager and Command to handle various tasks efficiently.

- **Attributes:**
    - engine: Initializes the text-to-speech engine for voice responses.
- **Methods:**
    - say(text: str): Converts text into speech to communicate with the user.
    - take_command(prompt: str): Captures and processes the user's voice input.
    - send_email(subject, body, to_emails): Delegates email-sending tasks to the EmailManager.
    - get_email_addresses(): Collects email addresses from predefined mappings or user input.
    - open_site(site_name: str, url: str): Opens the specified website in the web browser.
    - play_music(file_path: str): Plays music from a predefined file path.
    - query_time(): Fetches and announces the current time.
    - run(): Acts as the main driver function for the assistant, enabling continuous interaction with the user.

The Assistant class is at the heart of the system, coordinating tasks and providing a seamless user experience.

EmailManager Class

The EmailManager class is responsible for managing all email-related functionalities. It ensures secure and efficient email sending by handling authentication and recipient management.

- **Attributes:**
    - from_email: Stores the sender's email address.
    - password: Stores the authentication password for the email account.

- **Methods:**
    - send_email(subject, body, to_emails): Sends emails using the SMTP protocol, incorporating the subject, body, and recipient list.
    - add_recipient(name: str, email: str): Adds new recipients to the contact list dynamically.

The EmailManager class encapsulates all email-handling logic, ensuring modularity and security in managing user communications.

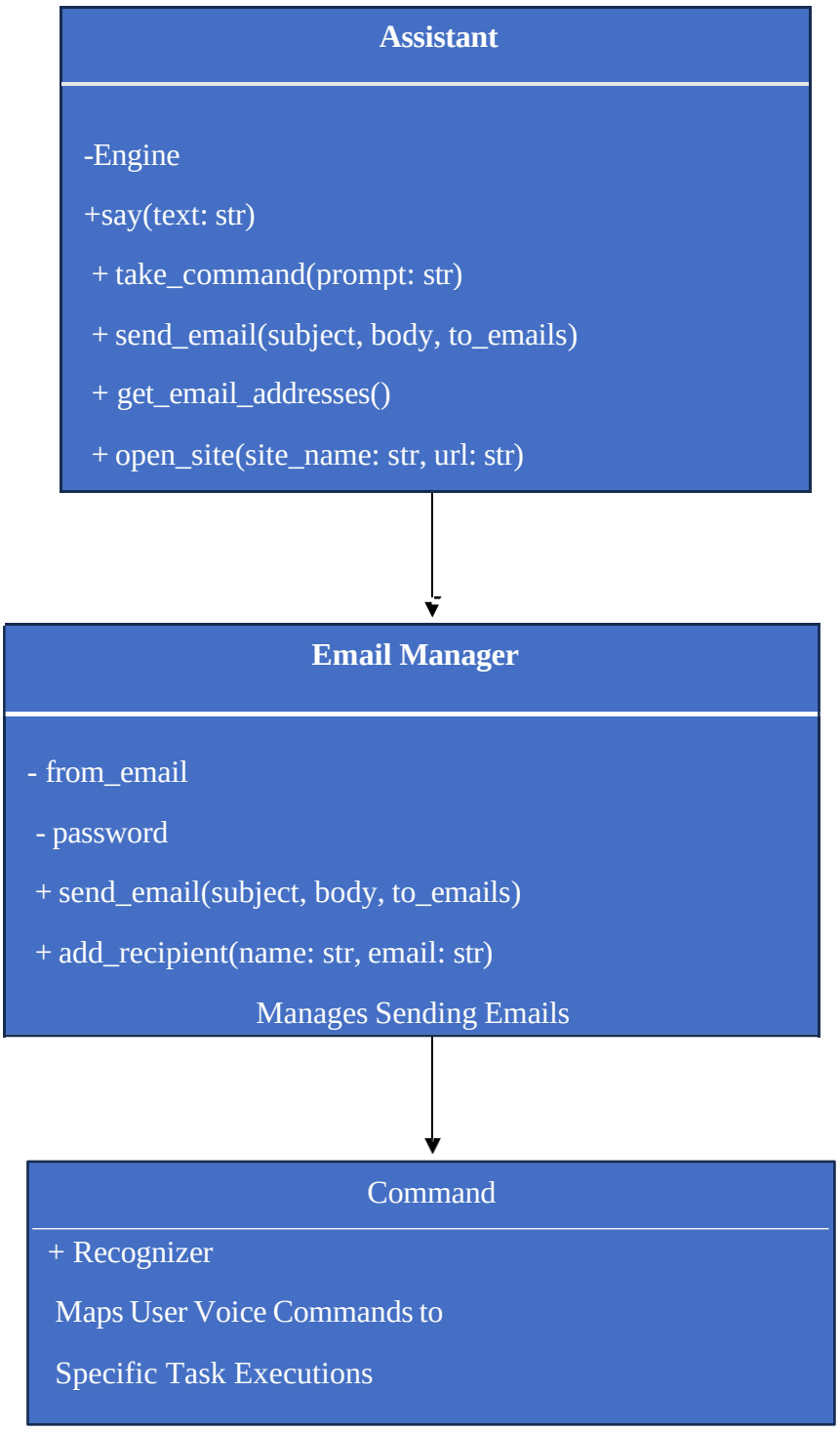
Command Class

The Command class maps user voice commands to specific task executions. It serves as the intermediary that translates recognized text into actionable tasks for the assistant.

- **Attributes:**
    - Recognizer: Manages the process of recognizing and interpreting voice commands**.**
- **Description:**
  The Command class acts as the logic layer, linking the user's input to predefined functions like opening websites, playing music, or sending emails. It ensures accurate mapping of commands to corresponding actions.

Relationships Between Classes

1. The Assistant class interacts with the EmailManager to delegate email-sending tasks, ensuring separation of concerns and modular design.
2. The Command class is used by the Assistant to process and execute user commands based on recognized speech.
3. The Assistant serves as the central point of interaction, coordinating with other classes to deliver a seamless virtual assistant experience.



(Fig 4.5: Class Diagram)

CHAPTER-5

IMPLEMENTATION AND CODING

5.1 CODING DETAILS

- **BACKEND**
  - **Speech Recognition Module:** Takes user speech as an input.

```
import speech_recognition as sr
import pyttsx3

# Initialize the pyttsx3 engine
engine = pyttsx3.init()

def say(text):
    """Converts text to speech."""
    engine.say(text)
    engine.runAndWait()

def take_command(prompt=""):
    """Captures and recognizes speech input, with a prompt for context if
provided."""
    r = sr.Recognizer()
    if prompt:
        say(prompt)
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source)
        r.pause_threshold = 0.8
        print("Listening...")
        audio = r.listen(source)
    try:
        query = r.recognize_google(audio, language="en-in")
        print(f"User said: {query}")
        return query.lower()  # Ensure recognized command is in lowercase
    except sr.UnknownValueError:
        print("Sorry, I didn't get that. Please say it again.")
        say("Sorry, I didn't get that. Please say it again.")
```

```
        return "none"
    except sr.RequestError:
        print("Sorry, my speech service is down.")
        say("Sorry, my speech service is down.")
        return "none"
```

- **Email Sending Module:** Generates email and sends it to the recipient.

```
import smtplib
import openai
from speech import say, take_command

# Set up OpenAI API key
openai.api_key = 'sk-proj-QTnqxiR72XaTO1jeBAPIa3sEXH-
0zoT8AOlEOGC1f5vrkmx-
WmTS2j478vB3noCyacNHEAHUXET3BlbkFJ3Evwlu_1nT0X5_06jvj1OSsitke7
GMMdMMQR5JtK2wo-ejXrQ4SesUaGtJRn2ONFv-2N8m9_UA'

def generate_email_content_with_openai(prompt):
    """Generates email content using OpenAI's language model based on the
provided prompt."""
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": f"Generate a professional email content based on
the following description: {prompt}"}
        ],
        max_tokens=150
    )
    return response.choices[0].message["content"].strip()

def send_email(subject, body, to_emails):
    """Sends an email with the specified subject and body to the given list of
recipient emails."""
    from_email = "ECHO.vassistant@gmail.com"
    password = "ucol xuyi kiyg lusk"

    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(from_email, password)
```

```
        message = f'Subject: {subject}\n\n{body}'
        for to_email in to_emails:
            server.sendmail(from_email, to_email, message)
        server.quit()
        say("Email sent successfully!")
    except Exception as e:
        say(f"Failed to send email: {e}")

def get_email_addresses():
    """Prompts the user to provide email recipients and returns a list of their email addresses."""
    email_address = [["Vinayak", "vinayak2002saxena@gmail.com"], ["Shivani", "shivanisahuofficial0401@gmail.com"]]
    to_emails = []
    say("Please name the recipients one by one. Say 'done' when you are finished.")
    while True:
        recipient = take_command()
        if recipient == "done":
            break
        for name, email in email_address:
            if name.lower() in recipient:
                say(f"Adding {name} to the email list.")
                to_emails.append(email)
                break
        else:
            say("Sorry, I didn't catch that. Please say the name again.")
    return to_emails
```

- **Main Module:** This Module integrates other Functionalities like playing music, opening web applications etc.

```
import os
import webbrowser
import datetime
from speech import say, take_command
from email_handler import generate_email_content_with_openai, send_email, get_email_addresses

if _name___== '_main_':
    print('VSCODE')
    say("Hi, This is ECHO, Your Everyday Assistant. How may I help you today, sir?")
```

```
    while True:
        text = take_command()
        if text != "none":
            sites = [["Youtube", "https://youtube.com"], ["Google",
"https://google.com"], ["Fast", "https://fast.com"], ["kite", "http://lms.kiet.edu"]]
            for site in sites:
                if f"open {site[0].lower()}" in text:
                    say(f"Opening {site[0]}, sir...")
                    webbrowser.open(site[1])
            if "open music" in text:
                musicpath = "C:\\Users\\ASUS\\Downloads\\Jannat Instrumental.mp3"
                say("Opening music, sir")
                os.startfile(musicpath)

            if "what is the time" in text:
                strftime = datetime.datetime.now().strftime("%H:%M:%S")
                say(f"Sir, the time is: {strftime}")

            if "send an email" in text or "send email" in text:
                while True:
                    say("What is the Subject")
                    subject = take_command()
                    if subject != "none":
                        break
                    say("No Subject is provided")
                while True:
                    say("Would you like to use OpenAI to generate Content?")
                    use_openai = take_command()
                    if use_openai == "yes":
                        while True:
                            say("Please provide the description")
                            prompt = take_command()
                            if prompt != "none":
                                body = generate_email_content_with_openai(prompt)
                                break
                            say("No description is provided")
                        break
                    elif use_openai == "no":
                        while True:
                            say("What should I say in the email")
                            body = take_command()
                            if body != "none":
                                break
```

```
                say("No description is provided")
                break
            else:
                say("Invalid input")
        while True:
            say("To whom should I send the email?")
            to_emails = get_email_addresses()
            if to_emails:
                send_email(subject, body, to_emails)
                break
            say("No valid email address is provided")

    if any(phrase in text for phrase in ["shutdown", "shut down", "go down",
"bye bye", "quit"]):
        say("Goodbye sir")
        break
```

- **FRONTEND:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Voice Assistant</title>
    <link rel="stylesheet" href="/static/CSS/style.css">
</head>
<body>
    <div class="background-image"></div>
    <!-- Navbar -->
    <nav class="navbar">

        <div class="navbar-brand"> <a href="#home">
            <img src="./tM3EEG4hR4Sf8U8H903K5A.jpg" alt="Logo" class="logo">
            </a>
            <span class="stylish-text">E.C.H.O</span>
        </div>

        <ul class="navbar-nav">
            <!-- <li><a href="#home">Home</a></li> -->
            <li><a href="#features">Features</a></li>
            <li><a href="#contact">Contact</a></li>
            <!-- <li><a href="#signup">Sign Up</a></li>
```

```
        <li><a href="#signin">Sign In</a></li> -->
        <li class="dropdown">
          <a href="javascript:void(0);" class="dropbtn">Login</a>
          <div class="dropdown-content">
            <a href="#signup">Sign Up</a>
            <a href="#signin">Sign In</a>
          </div>
        </li>
      </ul>
      <!-- <button id="theme-toggle" class="theme-btn"> </button>  -->
  </nav>

  <!-- Main Content -->
  <section id="home" class="main-section">
    <h1 style="color: azure; font-size: 3rem;" >Welcome to Your Voice
Assistant</h1>
    <button class="voice-btn" onclick="startVoiceAssistant()">Start
E.C.H.O</button>

  </section>
  <div id="loading-spinner" style="display: none;">Loading...</div>



<!-- Features Page -->
<section id="features" class="features-section">
  <h2>Features of E.C.H.O.</h2>
  <div class="features-list">
    <div class="feature-item">
      <h3>Voice Recognition</h3>
      <p>Effortlessly interpret and respond to spoken commands using advanced
voice recognition technology.</p>
    </div>
    <div class="feature-item">
      <h3>Hands-Free Operation</h3>
      <p>Perform everyday tasks without lifting a finger—just speak, and E.C.H.O.
will handle the rest.</p>
    </div>
    <div class="feature-item">
      <h3>Personalized Responses</h3>
      <p>E.C.H.O. learns your preferences to provide more relevant and helpful
responses over time.</p>
    </div>
```

```
        <div class="feature-item">
            <h3>Task Automation</h3>
            <p>Automate basic tasks like reminders, alarms, and even system shutdowns based on your voice commands.</p>
        </div>
        <div class="feature-item">
            <h3>Sending Emails</h3>
            <p>The E.C.H.O. voice-enabled assistant allows users to send emails hands-free by capturing recipient, subject, and message content through voice commands. It confirms details before securely sending the email via an SMTP server.</p>
        </div>
    </div>
</section>



    <!-- Sign Up Page -->
<section id="signup" class="form-section card">
    <div class="form-card">
        <h2>Sign Up</h2>
        <form id="signup-form">
            <input type="text" placeholder="Username" required>
            <input type="email" placeholder="Email" required>
            <input type="password" placeholder="Password" required>
            <button type="submit">Sign Up</button>
        </form>
    </div>
</section>

<!-- Sign In Page -->
<section id="signin" class="form-section card">
    <div class="form-card">
        <h2>Sign In</h2>
        <form id="signin-form">
            <input type="email" placeholder="Email" required>
            <input type="password" placeholder="Password" required>
            <button type="submit">Sign In</button>
        </form>
    </div>
</section>

 <!-- Contact Page -->
 <section id="contact" class="contact-section">
```

```
    <h2>Contact Us</h2>
    <p>If you have any questions, feedback, or need support with E.C.H.O., feel free to reach out!</p>
    <div class="contact-info">
        <p><strong>Email:</strong>  echo.vassistant@gmail.com</p>
        <p><strong>Phone:</strong> 9350307560, 9643037484</p>
        <p><strong>Address:</strong> KIET Group of Instituitions, Delhi NCR</p>
    </div>
    <form id="contact-form">
        <input type="text" placeholder="Your Name" required>
        <input type="email" placeholder="Your Email" required>
        <textarea placeholder="Your Message" rows="4" required></textarea><br>
        <button type="submit">Send Message</button>
    </form>
</section>

    <script src="/static/js/script.js"></script>
</body>
</html>
```

CHAPTER-6

SOFTWARE TESTING

6.1 TESTING STRATEGY

Our testing strategy involves a combination of manual and automated testing to thoroughly assess the functionality, performances, and reliability of our Project ECHO.

6.1.1 MANUAL TESTING

- Set Up the Environment:
  - Ensure all dependencies are installed and your environment is configured correctly.
  - Verify that microphone and internet connection are working properly.
- Perform Functional Tests:
  - Start the Assistant: Verify Initial Greeting.
  - Open a website: Test commands for opening different websites.
  - Ask for the time: Verify the time query response.
  - Play Music: Test the music is playing functionality.
  - Send an Email: Complete the email workflow, including providing the subject, body, and recipients
  - Handle Invalid Commands: Test hoe the assistant responds to unrecognized commands.
  - Shutdown the assistant: Verify the shutdown process.
- Measure Response Time:
  - Record the response time for various functions and ensure they are within acceptable limits.

6.1.2 AUTOMATATED TESTING

Automated Testing can help validate certain aspects of the code without manual intervention.

- Set Up Automated Tests:
  - Use a testing framework like unittest or pytest to create automated tests for the non-interactive components of your code.
- Write test Cases:
  - Create test cases for functions that do not involve any user interaction (like sending emails, doing web requests).
  - Use mock objects to simulate user inputs and other dependencies.

- Run Automated tests:
    - Run your automated tests and examine the results.
    - Make sure that all tests pass and detect failing tests.

6.2 TEST CASES AND OUTCOMES

6.2.1 Test Case 1: Launch the Assistant

- Step: Execute your script using the terminal or an IDE.
- Expected Result: The assistant should launch and utter, "Hi, This is ECHO, Your Everyday Assistant. How may I help you today, sir?"
- Verify: Ensure that the initial greeting will be delivered.

6.2.2 Test Case 2: Open a Webpage

- Step: Utter, "Open Google".
- Expected Result: The assistant should respond, "Opening Google, sir." and launch the Google webpage in your browser.
- Verbose: The browser opens and leads to https://google.com

6.2.3 Test Case 3: Requesting the Time

- Step: Say "What is the time?"
- Expected Result:The assistant answers by responding with the correct current time.
- Verifiability: Verify if what it speaks aligns with what shows the current time.

6.2.4 Test Case 4: Open Music

- Step: Says "Open music".
- Expected Outcome: The assistant shall speak as follows: Opening music, sir and initiate the file as requested "music."
- Verifiability: The requested music file, C:\\\\Users\\\\ASUS\\\\Downloads\\\\Jannat Instrumental.mp3 must have started playing.

6.2.5 Test Case 5: Compose Email

- Step: Say "Send an email".
- Expected Result: The assistant should open the email workflow.
    - Subject: When asked, repeat the subject of the e-mail.
    - Body: When asked, repeat the body text of the e-mail.
    - Recipient: One by one name the recipients and upon completing say "done".
    - Result: The assistant should send an email and say "E-mail sent successfully!".
- Verify: That the email has been sent to the right recipients with the right subject and body.

6.2.6 Test Case 6: Invalid Commands

- Step: Utter an unrecognized command, like "Fly to the moon."
- Expected Result: The assistant should say "Sorry, I didn't get that. Please say it again."
- Verification: Verify that the assistant responds graciously to unrecognized commands.

6.2.7 Test Case 7: Microphone Failure

- Step: Simulate a microphone failure, for example, by muting your microphone.
- Expected Result: The assistant should say "Sorry, my speech service is down."
- Verification Check if the error handling does not fail in case the microphone is not available.

6.2.8 Test Case 8: Shut Down the Assistant

- Step Say "Shutdown".
- Expected Outcome The assistant should say "Good bye, sir" and exit the script.
- Verification Verify that the script exits after giving the shutdown command.

Table 6.1: MANUAL TESTING

TEST CASE	EXPECTED OUTCOME	RESULT
Start the Assistant	The assistant should start and say, "Hi, this is ECHO, Your Everyday Assistant. How may I help you today, sir?"	PASSED
Handle Invalid Commands	The assistant should respond with "Sorry, I didn't get that. Please say it again."	PASSED

Table 6.2: AUTOMATED TESTING

TEST CASE	EXPECTED OUTCOMES	RESULT
Open A Website	The assistant should say "Opening Google, sir..." and open the Google homepage in your browser.	PASSED
Ask for the Time	The assistant should respond with the current time.	PASSED
Play Music	The assistant should say "Opening music, sir" and play the specified music file.	PASSED
Send an Email	When you say "send an email," the assistant will prompt for the subject, body, and recipients, then successfully send the email and confirm with "Email sent successfully!"	PASSED
Handle Invalid Commands	The assistant should respond with "Sorry, I didn't get that. Please say it again."	PASSED
Shutdown the Assistant	The assistant should say "Goodbye sir" and terminate the script.	PASSED

CHAPTER-7

RESULTS AND DISCUSSION

7.1 PROJECT OUTCOME

Project E.C.H.O. is a virtual assistant that performs various tasks including opening websites, playing music, providing the current time, and sending emails based in voice commands. It leverages speech recognition and text-to-speech technologies for user interaction.

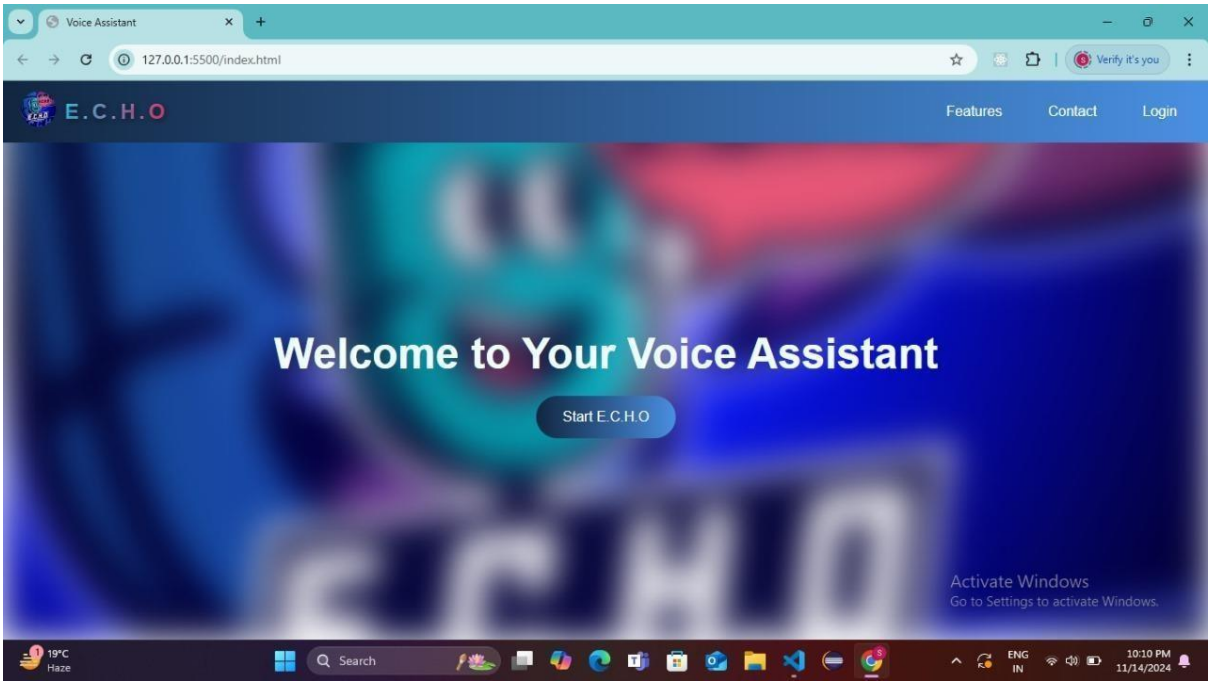
FUNCTIONALITY and FEATURES

- **Speech Recognition:**
  - **Feature**: Captures and understands voice commands using Google's Speech Recognition API.
  - **Outcome**: Accurately interprets and processes user voice commands.
- **Text-to-Speech:**
  - **Feature**: Converts text responses to spoken output using the pyttsx3 library.
  - **Outcome**: Provides clear and audible feedback to the user, enhancing interaction.
- **Web Browser:**
  - **Feature**: Opens specified websites based on user voice command.
  - **Outcome**: Successfully navigates to requested websites (e.g., Google, YouTube).
- **Music Playing:**
  - **Feature**: Plays a specific music file upon user command.
  - **Outcome**: Successfully plays the designated music file from the specified directory.
- **Time Query:**
  - **Feature**: Responds with the current time upon user request.
  - **Outcome**: Accurately provides the current time.
- **Email Sending:**
  - **Feature**: Prompts for email subject, body, and recipients, and sends emails using Gmail's SMTP server.
  - **Outcome**: Successfully sends emails and confirms with "Email sent successfully!"

- **Response Time Measurement:**
    - **Feature**: Measures and prints the response time for various functions.
    - **Outcome**: Provides numeric response times, allowing for performance evaluation and optimization.

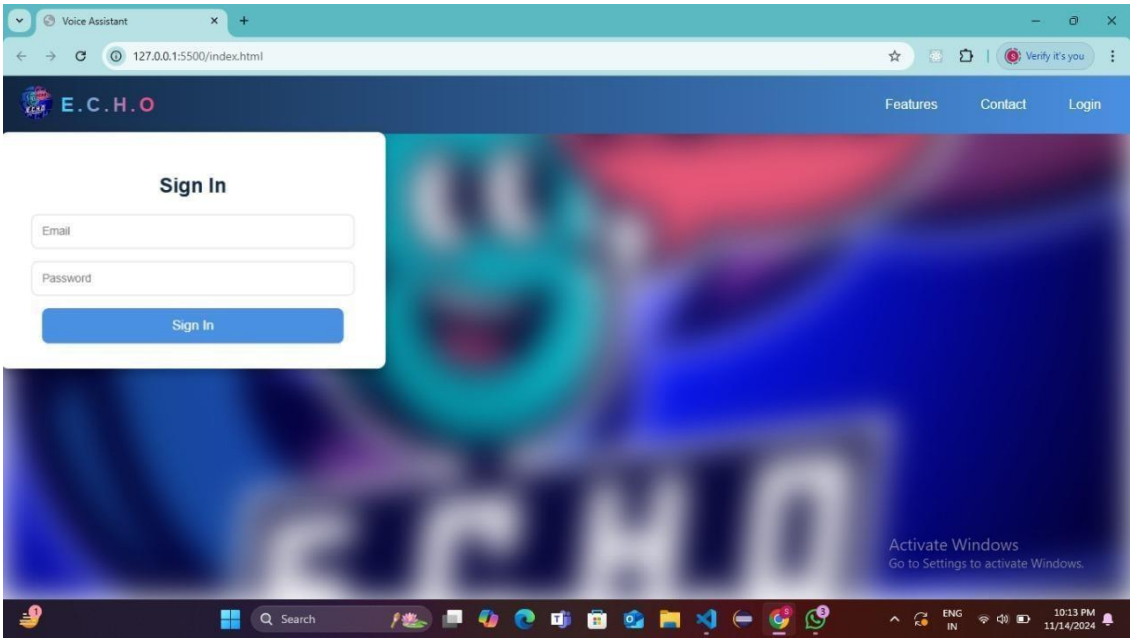
7.2 SNAPSHOT OF PROJECT

7.2.1 HOME PAGE



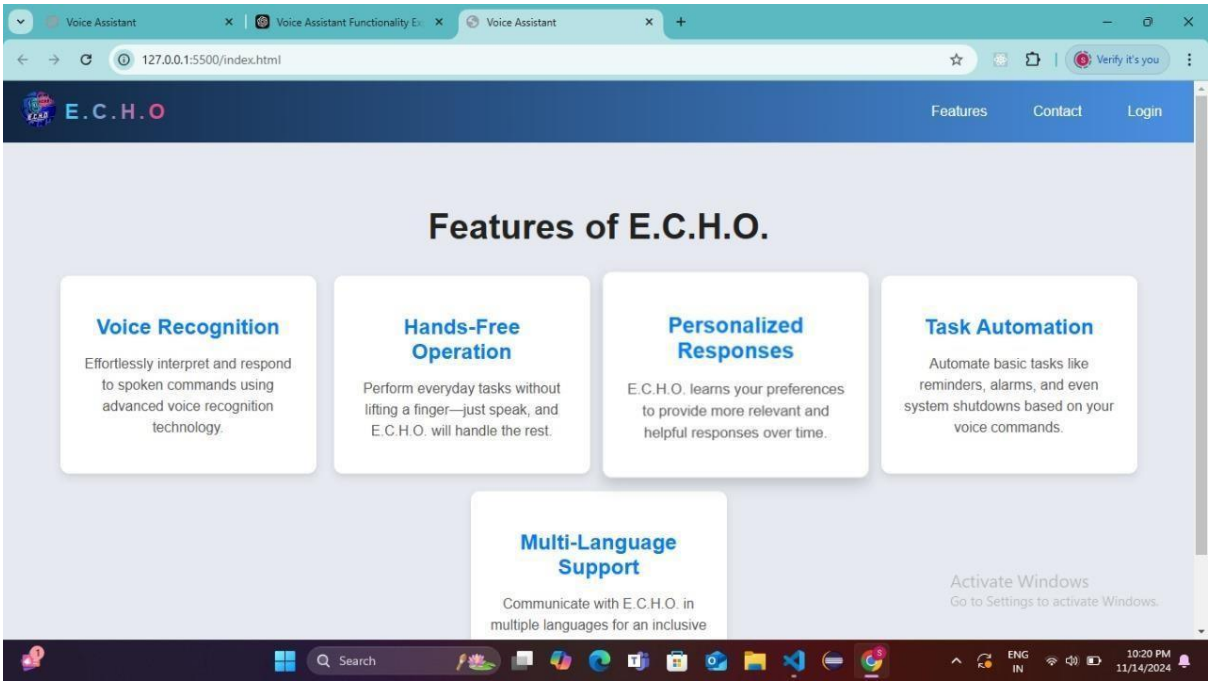
(Fig 7.1: Home Page)

7.2.2 SIGN-IN PAGE



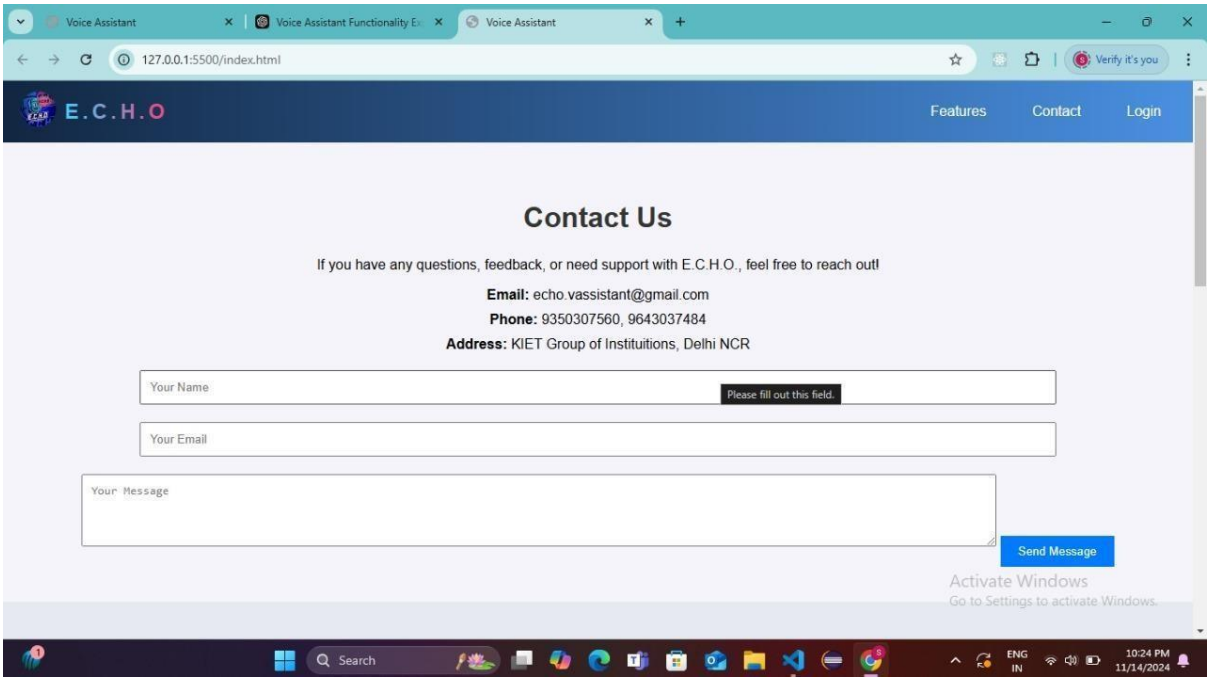
(Fig 7.2: Sign-In page)

7.2.3 FEATURE PAGE



(Fig 7.3: Feature Page)

7.2.4 CONTACT-US PAGE



(Fig 7.4: Contact-Us Page)

CHAPTER-8

CONCLUSION

Thus, Project E.C.H.O. demonstrates excellent incorporation of advanced speech recognition and text-to-speech technologies for making an extremely interactive and user-friendly virtual assistant. Robust functionality is clearly displayed as it helps users to open various websites, play music, provide the time of the day, and even send emails using voice commands. The effectiveness of this assistant in the precise interpretation and response to inputs is well demonstrated.

By incorporating response time measurement, project E.C.H.O. ensures that the system remains responsive and efficient, providing valuable insights into performance optimization. The assistant's seamless interaction and timely execution of tasks enhance the overall user experience, making it a practical and engaging tool for everyday use.

Project E.C.H.O. strengths come in the form of versatility, precision, and user-centered approach to its design, making it a very resourceful addition to anyone's toolkit for digital use. Its ability to accurately manage numerous commands with ease shows that using AI-powered assistant like E.C.H.O. can be efficient for everyday tasks and the production of work. The Conclusion, therefore brings to light a well-balanced and reliable virtual assistant capable of helping users finish tasks with efficiency and intelligently.

REFERENCES

[1] Sitti Rachmawati Yahya, S. N. H. Sheikh Abdullah, K. Omar; Review on Image Enhancement methods old manuscripts with damaged background: IEEE,August 2009, vol-10,pp.565-569.

[2] Wang yang, Pan Zhibin; Image Contrast Enhancement using local histogram equalization:Infrared physics and Technology, Nov 2017, volume-86,pp.59-65.

[3] Miao Yang, Jintong Hu, Chongyi Li, Gustavo Rohde; in-depth survey of underwater image Enhancement and Restoration: IEEE, August 2019, vol-7,pp.123638 – 123657.

[4] YuDong Zhang, LeNan Wu, Geng Wei;Color Enhancement based on HVS and PCNN: Science china Information Sciences, Sep 2019,vol-53,pp.1973-1976.

[5] Shang Jin, Yang You, Yang Huafen; A Scanned Document Image Processing Model for Information System: IEEE, Apr 2020, vol-56, pp.11563-11570.

[6] Haris Ackar, Ali Abd Almisreb, Mohd.A.saleh; A Review on Image Enhancement Techniques: Southeast Europe Journal of Soft Computing, May 2021, vol-8, issue-1, pp.42-48.

[7] P. Suganya, N. Mohanapriya, B. Kalaavathi : Satellite photo choice upgrade the utilization of multi wavelet revamp and correlation of addition systems; IEEE, 2016, vol. 25, no. 6, pp.713-724

[8] K. Kaur , N. Gupta : Execution Evaluation of Modified DBLA Using Dark Channel Prior and CLAHE ; International

Journal of Intelligent Systems and Applications, 2015, vol. 7, no. 5, pp. 48-56.

[9] W. G. Shadeed, D.I. Abu-Al-Nadi, M. J. Mismar : Street guests sign recognition in shading pics ; IEEE , Dec. 2016, vol. 2, pp. 890–893.

[10] M. Hanmandlu, D. Jha, R. Sharma: Shading picture upgrade through fluffy heightening; Pattern Recognition Letters,2018 vol. 24, no. 1-3, pp. 81-87.

[11] Hasan Demirel and Gholamreza Anbarjafari: Discrete Wavelet Transform Based SatelliteImage Resolution ; IEEE photograph preparing, May 2020, vol.20, no. 5.

[12] M. Yasmin, M. Sharif, S. Masood, M. Raza, S. Mohsin: Cerebrum Image Enhancement- A Survey; World Applied Sciences Journal, 2018, vol-17 issue-9, pp-256-262.