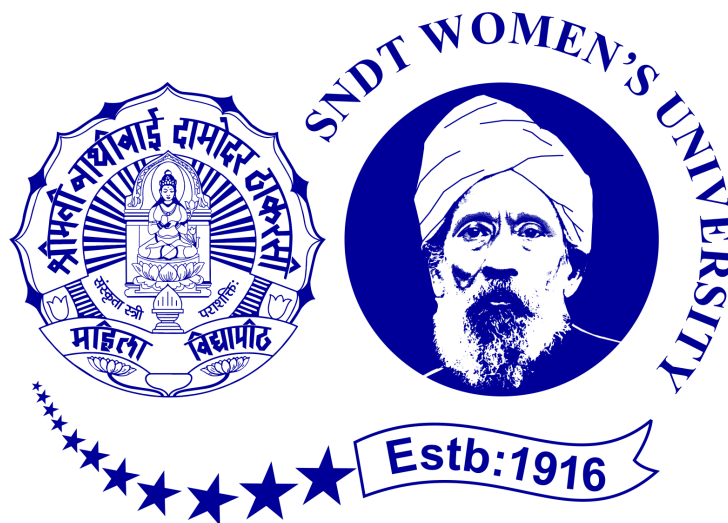


**A MINI-PROJECT REPORT ON**  
**EVA: PYTHON-BASED VIRTUAL AI ASSISTANT**

Submitted in partial fulfillment for the  
degree of Bachelor of Technology in  
Computer Science and Technology

Submitted by  
**Heli Makwana (40)**  
**Nisha Mandal (41)**  
**Tanisha Sankhe (60)**

**Group No. 05**  
**Under the guidance of**  
**Prof. Prajakta Gotarne**



**USHA MITTAL INSTITUTE OF TECHNOLOGY**  
**S.N.D.T WOMEN'S UNIVERSITY**  
**MUMBAI-400049**  
**2024- 2025**

# CERTIFICATE

This is to certify that **Heli Makwana (40), Nisha Mandal (41), Tanisha Sankhe (60)** (Group No. 05), has successfully completed Mini-Project phase-2 project work on “**Eva: Python-Based Virtual AI Assistant**” in partial fulfillment of the B.Tech. degree in **Computer Science and Technology** during the year 2024-25 as prescribed by SNDT Women’s University.

(Project Guide)

**PROF. PRAJAKTA GOTARNE**

(Head of Department)

**PROF. KUMUD WASNIK**

Principal

**DR.YOGESH NERKAR**

**EXAMINAR 1**

**EXAMINAR 2**

## **Declaration**

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

(Name of the student)

(Roll No)

Date

## **ACKNOWLEDGEMENT**

We would like to express our sincere appreciation to everyone who gave their invaluable assistance and support in this project. We are highly indebted to Prof. Prajakta Gotarne for her guidance and constant supervision as well as for providing necessary information regarding the project also for her support in this project. Without her help, this project would not have been possible. Prof. Prajakta Gotarne.

We would also like to thank our Head of Department Prof.Kumud Wasnik for her in sights in shaping the direction and content of this report. We would also like to extend our gratitude to our classmates who provided valuable feedback and encouragement throughout the process.

Thank you all for your contributions, support, and encouragement.

# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Introduction . . . . .	9
1.2	Problem Statement . . . . .	9
<b>2</b>	<b>Literature Review</b>	<b>11</b>
2.1	Technical Paper . . . . .	11
2.2	Literature Survey . . . . .	11
<b>3</b>	<b>Existing System</b>	<b>13</b>
<b>4</b>	<b>Proposed System</b>	<b>14</b>
4.1	Data Flow Diagram . . . . .	14
<b>5</b>	<b>Architectural Overview</b>	<b>15</b>
<b>6</b>	<b>Hardware And Software requirements</b>	<b>17</b>
6.1	Hardware Requirements . . . . .	17
6.2	Software Requirements . . . . .	17
<b>7</b>	<b>Implementation details</b>	<b>18</b>
7.1	Key Windows Tasks . . . . .	18
7.2	Automation Tasks . . . . .	18
7.3	Personalization Tasks . . . . .	19
7.4	Implementation . . . . .	19
7.5	Result . . . . .	21
<b>8</b>	<b>Applications</b>	<b>22</b>
<b>9</b>	<b>Scope / Future Work</b>	<b>23</b>
<b>10</b>	<b>Conclusion</b>	<b>24</b>
	<b>References</b>	<b>25</b>

# List of Figures

3.1	Existing System Architecture . . . . .	13
4.1	Data Flow Diagram . . . . .	14
5.1	Proposed System Architecture . . . . .	16
7.1	Screenshots of Implementaions . . . . .	20
7.2	Screenshots of various functionalities . . . . .	21

# List of Tables

2.1	Table of Literature Survey . . . . .	12
-----	--------------------------------------	----

## Abstract

In recent years, virtual assistants have become increasingly prevalent in various domains, offering users an intuitive and efficient interface for interacting with technology. Python, with its versatility and extensive libraries, has emerged as a popular choice for developing virtual assistants due to its capabilities in AI integration, voice recognition, and intelligent task execution. The widespread adoption of voice-activated technologies is drastically changing the face of our society. Modern devices have voice assistant features built right in, from desktop computers to cell phones. The brains behind this technology include an AI-based voice recognition system that can understand human speech, turn it into text, and use services like Google Text-to-Speech.

The ability to create a more convenient and imaginative living is made possible by the current advancements in computing and manufacturing. AI-powered desktop assistants that integrate both software and hardware components are being developed with the goal of providing quick computation and a deep comprehension of user requirements. These assistants attempt to smoothly incorporate technology into ordinary human interactions by using hardware such as microphones to record and process audio requests and respond in a voice that is appropriate for a human. This paper explores strategies for developing Python-based virtual assistants, focusing on the integration of artificial intelligence (AI) technologies, such as natural language processing and machine learning, to enhance user interactions. We discuss the importance of voice recognition systems in enabling hands-free operation and facilitating seamless communication between users and virtual assistants.

Additionally, we delve into techniques for implementing intelligent task execution, including automation of repetitive tasks and integration with external services and APIs. Through a combination of theoretical insights and practical examples, this paper provides developers with valuable guidance for creating robust and user-friendly Python-based virtual assistants that meet the evolving needs of modern technology users. The Virtual AI Assistant aims to enhance user productivity, convenience, and overall quality of life by serving as a knowledgeable and reliable digital companion that adapts to individual needs and preferences. With ongoing advancements in artificial intelligence and natural language understanding, the capabilities of virtual assistants continue to evolve, promising even more sophisticated and personalized experiences in the future.

### Keywords:

ABBREVIATIONS	EXPANSION
IOT	Internet of Things
AI	Artificial Intelligence
COM	Communication Port
OOPS	Object Oriented Programming
API	Application Programming Interface
TTS	Text to Speech
STT	Speech to Text
RAD	Rapid Application Development
UIDs	Unique Identifiers
IP	Internet Protocol



# Chapter 1

## Introduction

### 1.1 Introduction

In today's digital world, virtual assistants have become essential tools, helping users manage tasks and improve productivity through voice commands. Popular assistants like Siri, Alexa, and Cortana have simplified daily activities such as sending messages, setting reminders, and controlling devices. However, many still have limitations in accuracy, personalization, and seamless integration with other platforms, making them less effective for more complex tasks. EVA, a Python-based virtual AI assistant, is designed to address these challenges by offering advanced speech recognition and automation capabilities. Built using Python's robust libraries, EVA ensures accurate command execution, providing a reliable and efficient experience for users. With EVA, tasks such as sending messages, making video calls, and browsing the web can be performed effortlessly through voice commands, making it more intuitive and user-friendly than traditional assistants.

EVA also goes beyond basic functions, offering automation for desktop tasks such as managing applications, controlling system settings like locking or shutting down, and providing real-time updates like weather forecasts. Furthermore, EVA's integration with popular communication platforms, such as WhatsApp, allows users to initiate video calls and send messages hands-free, streamlining communication. With its personalized features, seamless interface, and powerful automation, EVA is designed to offer users a smarter, more efficient way to interact with technology, enhancing productivity and simplifying everyday routines.

### 1.2 Problem Statement

The problem statement highlights the current limitations of virtual AI assistants, focusing on several key areas that require improvement:

- **Inaccurate Speech Recognition and Conversion:** Existing AI assistants often misinterpret user speech, leading to errors in converting spoken commands into text. This results in frequent misunderstandings, incorrect responses, and diminished user trust.
- **Limited Desktop Task Automation:** Many virtual assistants lack robust capabilities for automating essential desktop tasks such as file management, web browsing, and communication. This restricts their utility, forcing users to rely on manual efforts for tasks that should be streamlined.
- **Generic User Experience:** The lack of personalization in most virtual assistants results in a one-size-fits-all approach, which fails to cater to individual user preferences, behavior patterns, and unique workflows. This creates a disconnect between the assistant and the user.

- **Complex User Interfaces:** Virtual assistants often feature unintuitive or cluttered interfaces, making them difficult to use for a broad range of users, especially those with limited technical expertise.
- **Inefficient Communication Workflows:** Existing solutions do not offer direct integration with popular communication platforms like WhatsApp, making actions such as initiating video calls or sending messages cumbersome and time-consuming. This disrupts user productivity and workflow.

## Proposed Solution

To address these challenges, the proposed system aims to enhance virtual AI assistants by introducing the following improvements:

1. **Enhanced Speech Recognition and Conversion:** The system will feature advanced speech recognition technology to accurately convert speech to text, allowing for more reliable voice command interaction.
2. **Expanded Desktop Actions and Automations:** The assistant will support a wide range of desktop actions and automations, including:
  - **Utility Actions:** File management and system settings.
  - **Chrome Automation:** Web browsing and bookmark management.
  - **Communication Actions:** Managing communications and messages.
  - **Entertainment Actions:** Music playback and media control.
  - **Volume Control:** Adjusting system volume.
  - **Notepad Automation:** Automating tasks in Paint and Notepad.
3. **Personalized User Experience:** EVA moves beyond the one-size-fits-all approach by offering deep personalization features. It adapts to each user's behavior patterns, preferences, and unique workflows. This customization allows EVA to deliver more relevant suggestions and actions based on individual needs, enhancing the overall interaction and engagement with the assistant. By learning from user interactions over time, EVA becomes increasingly efficient and aligned with personal preferences.
4. **Intuitive User Interface:** EVA boasts a user-friendly, clean, and intuitive interface designed to cater to a broad range of users, including those with limited technical knowledge. With a simple, easy-to-navigate design, EVA ensures that tasks such as managing applications, controlling system functions, and setting reminders are effortless. The interface is organized and minimizes clutter, providing a smooth and efficient user experience for both novice and advanced users.
5. **Seamless Communication Integration:** EVA integrates directly with popular communication platforms, such as WhatsApp, allowing users to initiate video calls, send messages, and interact with their contacts hands-free through voice commands. This eliminates the need for multiple steps or switching between apps, streamlining communication and improving productivity. By integrating seamlessly with existing platforms, EVA ensures that users can communicate efficiently without disrupting their workflow.

## Chapter 2

# Literature Review

### 2.1 Technical Paper

A technical paper is a document that presents research findings, innovations, developments, or solutions related to a specific technical field or topic. It's drafted for a specialized audience, such as researchers, engineers, scientists, or professionals. They often follow a structured format and include detailed descriptions of methodologies, results, analysis, and conclusions. They focus on various aspects of virtual AI assistants, including their architecture, algorithms, user interaction, etc. In our case, it proved to be highly informative, providing valuable insights into the problem and guiding us in devising an efficient solution. The table below offers a summary of the literature survey conducted:

### 2.2 Literature Survey

- (a) In 2023, Sri Meenakshi Pandey, S. Sindu, and Ms. C. A. Daphine Desona Clemency provided an overview of AI-based virtual assistants, exploring their current state, challenges, and opportunities. They also offered recommendations for future research to advance the field and address emerging issues in intelligent personal assistants.
- (b) The development of a Virtual Assistant (VPA) using AI and Python, as presented in March 2020, offers significant benefits for modern life. However, acknowledging its limitations, future improvements could focus on refining natural language processing, expanding knowledge bases, and enhancing contextual understanding for more seamless assistance.
- (c) In May 2020, Tushar Bansal, Ritik Karnwal, Vishal Singh, and Hardik Bansal spearheaded the development of Genesis, a Python-based virtual assistant. They reflect on its evolution, noting improvements in performance over time. Additionally, they contemplate potential enhancements to further elevate Genesis's capabilities.
- (d) In July 2023, N. Umapathi, G. Karthick, N. Venkateswaran, R. Jegadeesan, and Dava Srinivas explored voice-activated virtual assistants in Python. They noted that these virtual assistants are becoming smarter and more effective for organizing schedules.
- (e) In 2023, Aditi Bombe, Atharva Kale, Atharva Nayak, Anushka Darure, and Yash Chavan focused on advancing the capabilities and usability of AI-based voice assistants through their project on a Personal Desktop AI Assistant using Python. They emphasized the importance of addressing ongoing challenges and ethical considerations within the development and deployment of such assistants.

<b>Paper Title, Year of Publishment</b>	<b>Authors</b>	<b>Methodology and Technologies</b>	<b>Observations/Findings and Remarks</b>
AI BASED VIRTUAL ASSISTANT, (Year: 2023)	Sri Meenakshi Pandey, S Sindu, Ms C A Daphine Desona Clemency	Comprehensive overview of the current state of research and development in intelligent personal assistants and potential implications.	Identifies challenges, opportunities, and offers recommendations for future research and practice.
Virtual Assistant Using AI and Python (Year March-2020)	A. Sudhakar Reddy M, Vyshnavi, C. Raju Kumar, Saumya	Development of VPA using AI and Python, significance in modern life. Acknowledges limitations, suggests areas for future improvement.	Acknowledges limitations, suggests areas for improvement.
Genesis - The Digital Assistant (Python) (Year May 2020)	Tushar Bansal, Ritik Karnwal, Vishal Singh, Hardik Bansal	Development of Genesis, a Python-based virtual assistant.	Reflects on evolution, improvement in performance, potential enhancements.
Desktop Virtual Assistant Using Python (Year July 2023)	N Umapathi, G Karthick, N Venkateswaran, R Jegadeesan, Dava Srinivas	Exploration of voice-activated virtual assistants in Python.	Virtual assistants becoming smarter, effective for organizing schedules.
Personal Desktop AI Assistant Using Python (Year: 2023)	Aditi Bombe, Atharva Kale, Atharva Nayak, Anushka Darure, Yash Chavan	Advancing the capabilities and usability of AI-based voice assistants.	Highlights challenges and ethical considerations.

Table 2.1: Table of Literature Survey

## Chapter 3

# Existing System

Several existing systems of AI assistants have been developed by various companies and organizations, each offering unique features and functionalities. Here are some notable examples:

- **Amazon Alexa:** Developed by Amazon, Alexa is a cloud-based voice service that powers devices like the Amazon Echo. Alexa can perform tasks such as answering questions, playing music, setting alarms, controlling smart home devices, and providing weather updates.
- **Apple Siri:** Siri is Apple's virtual assistant, available on iOS devices, macOS, watchOS, and HomePod. Siri can perform tasks such as sending messages, making calls, setting reminders, providing recommendations, and controlling smart home devices.
- **Google Assistant:** Google Assistant is a virtual assistant developed by Google, available on Android devices, iOS, smart speakers, and other Google products. It can perform tasks such as answering questions, sending messages, making reservations, playing music, and controlling smart home devices.
- **Microsoft Cortana:** Cortana is a virtual assistant developed by Microsoft, available on Windows devices, Xbox, and other Microsoft products. Cortana can perform tasks such as setting reminders, searching the web, sending emails, and providing personalized recommendations.
- **Samsung Bixby:** Bixby is Samsung's virtual assistant, available on Samsung Galaxy smartphones and other devices. Bixby can perform tasks such as controlling device settings, making calls, sending messages, and providing personalized recommendations.

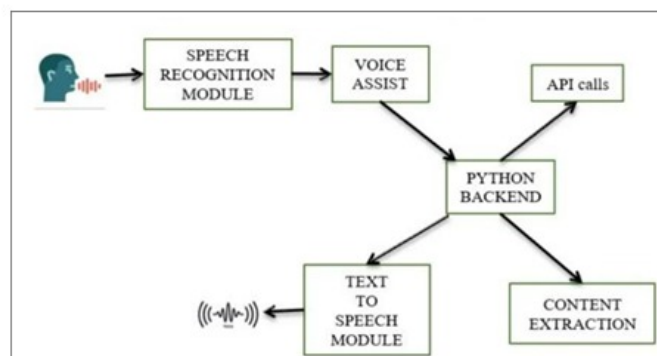


Figure 3.1: Existing System Architecture

## Chapter 4

# Proposed System

- (a) **Speech Recognition and Conversion:** Your assistant can accurately recognize and convert speech to text, allowing users to issue commands through speech input.
- (b) **Text to Speech Conversion:** The assistant can convert text to speech, enabling it to provide output in both speech and text formats.
- (c) **Desktop Actions and Automations:**
  - **Utility Actions:** Your assistant can perform various utility actions such as retrieving the current time, shutting down or restarting the system, opening applications, etc.
  - **Chrome Automations:** It can interact with the Chrome browser, perform navigation actions, manage tabs, and perform searches.
  - **Communication Actions:** Your assistant can engage in basic communication actions such as greeting, expressing gratitude, and introducing itself.
  - **Entertainment Actions:** It can play music or songs and open YouTube.
  - **Volume Control Actions:** The assistant can adjust the system's volume settings like volume up and down, and muting the volume.
  - **WhatsApp Automation:** It can perform actions such as making video calls and sending automated messages on WhatsApp.
  - **NotePad Automation:** Your assistant can assist in note-making tasks using Notepad.

### 4.1 Data Flow Diagram

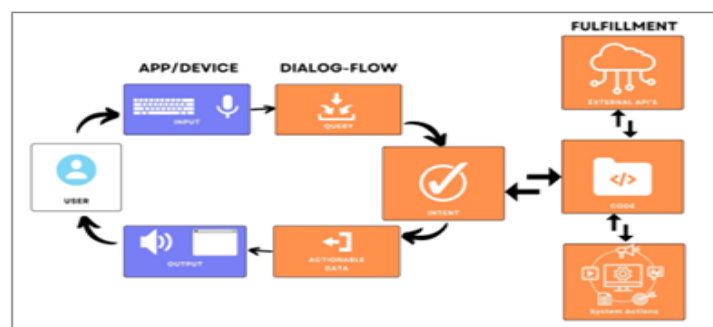


Figure 4.1: Data Flow Diagram

## Chapter 5

# Architectural Overview

- (a) The workflow of a virtual AI assistant revolves around understanding user input, performing tasks or providing information based on that input, and delivering appropriate responses to the user, all while continuously learning and adapting to improve the user experience.
- (b) It starts with a signal word. Users say the names of their voice assistants for the same reason. They might say, “Hey Siri!” or simply, “Alexa!” After the voice assistant hears its signal word, it starts to listen. The device waits for a pause to know you’ve finished your request. The voice assistant then sends our request over to its source code. Once in the source code, our request is compared to other requests. It’s split into separate commands that our voice assistant can understand. The source code then sends these commands back to the voice assistant. Once it receives the commands, the voice assistant knows what to do next.
- (c) If it understands, the voice assistant will carry out the task we asked for. For example, “Hey EVA! What’s the weather?” EVA reports back to us in seconds.
- (d) The user gives the voice input through microphone and the assistant is triggered by the wake up word and performs the STT (Speech to Text) and converts it into a text and understands the Voice input and further performs the task said by the user repeatedly and delivers it via TTS (Text to Speech) module via AI Voice. These are the important features of the voice assistant but other than this, we can do an plenty of things with the assistant.
- (e) The virtual assistant’s architecture encompasses several key functionalities:
  - Greetings: Eva greets you based on the time of day (Good Morning/Afternoon/Evening).
  - Voice: Eva uses a female voice for a more friendly interaction.
  - Personalized Note-taking: Eva can create and open a note file for you.
  - System Control:
    - Shutdown: Shut down your system.
    - Lock: Lock your computer screen.
    - Screenshot: Take a screenshot.
    - Volume Control: Increase, decrease, or mute the volume.
  - Web Browsing:
    - Open Google/YouTube/Chrome: Open specified websites.
    - Search Google/YouTube/Wikipedia: Search for information on Google, YouTube, or Wikipedia.
    - Music and Videos: Opening the music and videos.
    - Play Music on Spotify: Opens Spotify and starts playback.
    - Play Movie on Netflix: Searches for a movie on Netflix.
    - Play YouTube Video: Plays a YouTube video based on your request.

-Control YouTube: Pause, resume, fullscreen, skip, go back, mute, previous video, next video.

App Management:

-Open Notepad: Opens Notepad.

-Close Applications: Close specific applications (Chrome, Notepad, Word, Excel, PowerPoint, Command Prompt).

System Information:

-WhatsApp Automation: It can perform actions such as making video calls and sending automated messages on WhatsApp. -Time: Tells you the current time.

Other:

-Ignite Window: Brings you to the desktop.

-Open Downloads: Opens the Downloads folder.

-Open Google History: Opens your Google activity history.

Weather:

-Weather Search: Get the weather information for a specific location.

Gratitude:

-Eva expresses gratitude with phrases like "You're welcome!" or "I'm happy to help!"

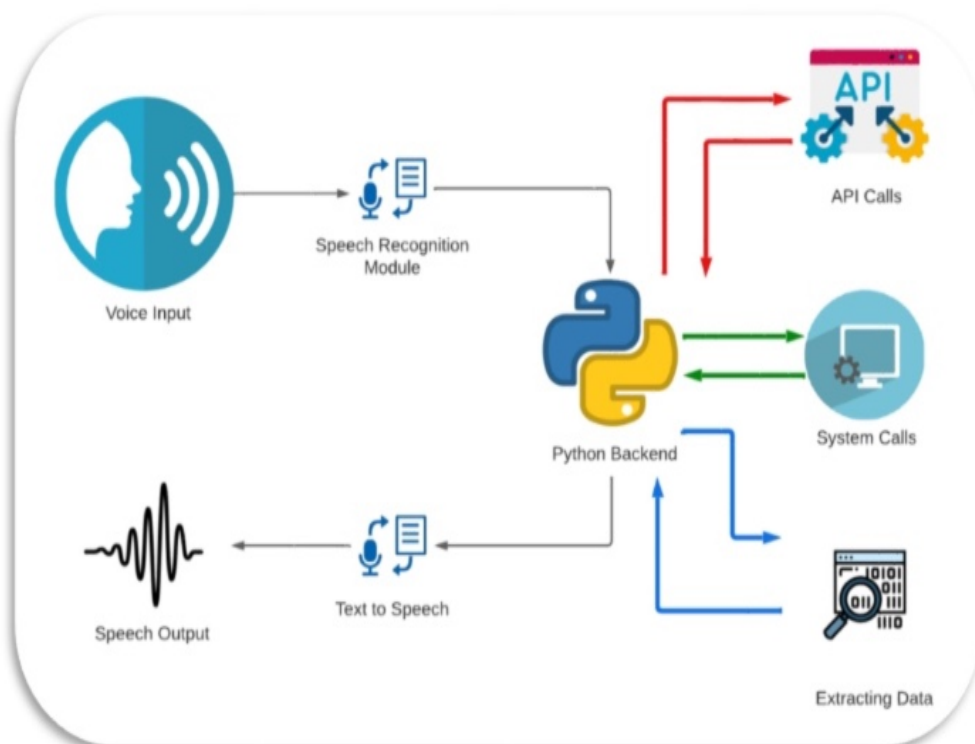


Figure 5.1: Proposed System Architecture



## Chapter 6

# Hardware And Software requirements

### 6.1 Hardware Requirements

- **Processor:**  
Recommended: Intel Core i5 or higher  
Minimum: Intel Core i3 or equivalent
- **Speed:**  
Recommended: 2.0 GHz or higher  
Minimum: 1.5 GHz
- **RAM:**  
Recommended: 4 GB or higher  
Minimum: 2 GB
- **Hard Disk:**  
Recommended: SSD with at least 128 GB storage  
Minimum: HDD with 80 GB storage
- **Input Devices:**  
Standard keyboard compatible with the system  
Multi-button mouse for navigation and interaction

### 6.2 Software Requirements

- **Operating System:**  
Windows 10 (64-bit) or the latest version of macOS or a Linux distribution compatible with Python development
- **Python Development Environment:**  
Python 3.9 installed
- **Integrated Development Environment (IDE):**  
Recommended: PyCharm, Jupyter Notebook, or Visual Studio Code for Python coding and development

## Chapter 7

# Implementation details

### 7.1 Key Windows Tasks

- **Time Display:** Display the current time.
- **Open Google:** Launch the Google homepage.
- **Open YouTube:** Open the YouTube website.
- **Play Music:** Access the Gaana website to play music.
- **Weather:** Provide weather information.
- **Shutdown:** Initiate system shutdown.
- **Lock System:** Lock the user's system.
- **Close Applications:** Close Command Prompt, Notepad, Microsoft Word, Microsoft Excel, and Microsoft PowerPoint.
- **Open Chrome:** Launch Google Chrome browser.
- **Maximize/Minimize/Close Windows:** Manipulate window states (maximize, minimize, close).
- **Google Search:** Perform a Google search.
- **Open Application:** Open specified applications.
- **Close Chrome:** Close the Google Chrome browser.
- **Take a Screenshot:** Capture and save screenshots.
- **Open Downloads:** Opens the Downloads folder.
- **Open Google History:** Opens your Google activity history.
- **Ignite Window:** Brings you to the desktop.

### 7.2 Automation Tasks

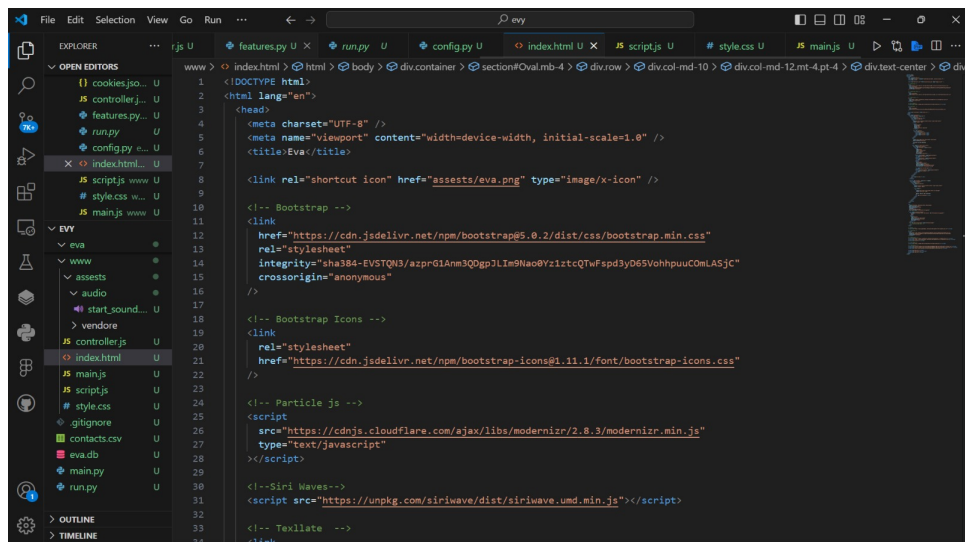
- **Chrome Automation:** Opening, managing, and terminating Chrome browser sessions.
- **YouTube Automation:** Navigating to YouTube and playing music or videos.
- **Spotify Automation:** Play music on spotify.

- **Netflix Automation:** Play and searches movie on Netflix.
- **Notepad Automation:** Creating notes based on user input.
- **WhatsApp Automation:** Making a video call and sending messages.

## 7.3 Personalization Tasks

- **Greet User:** Eva greets you based on the time of day (Good Morning/Afternoon/Evening).
- **Thank User:** Acknowledge user gratitude.
- **Voice:** Eva uses a female voice for a more friendly interaction.
- **Personalized Note-taking:** Eva can create and open a note file for you.

## 7.4 Implementation



```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Eva</title>

  <link rel="shortcut icon" href="assets/eva.png" type="image/x-icon" />

  <!-- Bootstrap -->
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgp3LlIm9Nao0Yz1ttcQTWf3pd30D6VohhhpucOoLAsjC"
    crossorigin="anonymous"
  />

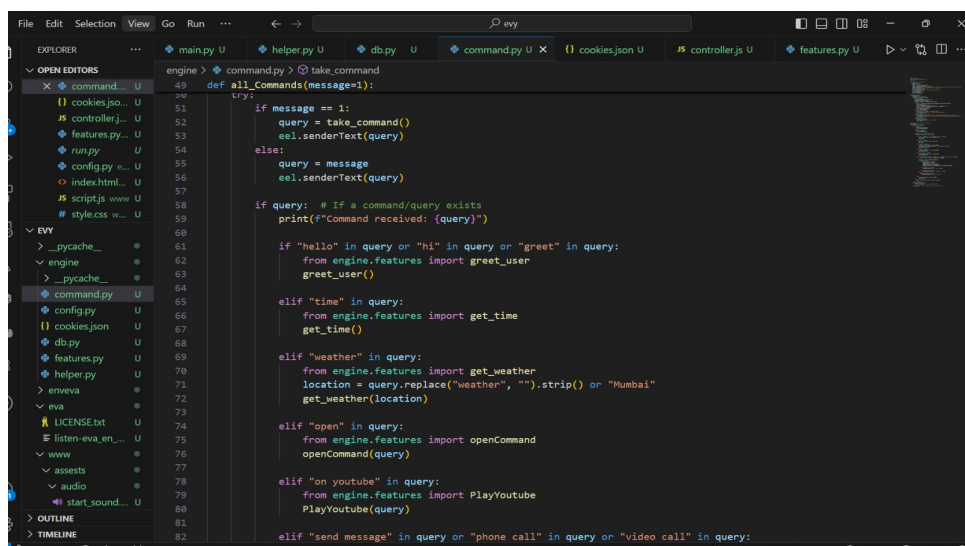
  <!-- Bootstrap Icons -->
  <link
    rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.1/font/bootstrap-icons.css"
  />

  <!-- Particle.js -->
  <script
    src="https://cdnjs.cloudflare.com/ajax/libs/modernizr/2.8.3/modernizr.min.js"
    type="text/javascript"
  ></script>

  <!-- Siri Waves -->
  <script src="https://unpkg.com/siriwave/dist/siriwave.umd.min.js"></script>

  <!-- TeXtlate -->
  <link

```



```

def take_command():
    try:
        if message == 1:
            query = take_command()
            eel.senderText(query)
        else:
            query = message
            eel.senderText(query)

    if query: # If a command/query exists
        print(f"Command received: {query}")

        if "hello" in query or "hi" in query or "greet" in query:
            from engine.features import greet_user
            greet_user()

        elif "time" in query:
            from engine.features import get_time
            get_time()

        elif "weather" in query:
            from engine.features import get_weather
            location = query.replace("weather", "").strip() or "Mumbai"
            get_weather(location)

        elif "open" in query:
            from engine.features import openCommand
            openCommand(query)

        elif "on youtube" in query:
            from engine.features import PlayYoutube
            PlayYoutube(query)

        elif "send message" in query or "phone call" in query or "video call" in query:

```

```

engine > features.py > chatBot
180 def findContact(query):
181     speak(not exist in contacts)
182     return 0, 0
183
200 def whatsapp(mobile_no, message, flag, name):
201
202     if flag == 'message':
203         target_tab = 12
204         computer_message = "message send successfully to "+name
205
206     elif flag == 'call':
207         target_tab = 7
208         message = ''
209         computer_message = "calling to "+name
210
211     else:
212         target_tab = 6
213         message = ''
214         jarvis_message = "staring video call with "+name
215
216     # Encode the message for URL
217     encoded_message = quote(message)
218
219     # Construct the URL
220     whatsapp_url = f"whatsapp://send?phone={mobile_no}&text={encoded_message}"
221
222     # Construct the full command
223     full_command = f'start "" "{whatsapp_url}"'
224
225     # Open WhatsApp with the constructed URL using cmd.exe
226     subprocess.run(full_command, shell=True)
227     time.sleep(5)
228     subprocess.run(full_command, shell=True)
229

```

```

run.py > ...
1 import multiprocessing
2 from main import start # Import the start function from main
3 from engine.features import hotword # Import the hotword function
4
5 # To run Eva
6 def startEva():
7     print("Process 1 (Eva) is running.")
8     start() # Start the main application
9
10 # To run hotword
11 def listenHotword():
12     print("Process 2 (Hotword Listener) is running.")
13     hotword() # Start the hotword detection
14
15 # Start both processes
16 if __name__ == '__main__':
17     p1 = multiprocessing.Process(target=startEva)
18     p2 = multiprocessing.Process(target=listenHotword)
19
20 # Start both processes
21 p1.start()
22 p2.start()
23
24 # Wait for the Eva process to finish
25 p1.join()
26
27 # If the hotword listener is still alive, terminate it
28 if p2.is_alive():
29     p2.terminate()
30     p2.join()
31
32 print("System stopped.")
33

```

```

run.py > ...
1 import multiprocessing
2 from main import start # Import the start function from main
3 from engine.features import hotword # Import the hotword function
4
5 # To run Eva
6 def startEva():
7     print("Process 1 (Eva) is running.")
8     start() # Start the main application
9
10 # To run hotword
11 def listenHotword():
12     print("Process 2 (Hotword Listener) is running.")
13     hotword() # Start the hotword detection
14
15 # Start both processes

```

Listening...

Recognizing...

User said: tell me about Shri Ram in two lines

Command received: tell me about shri ram in two lines

Shri Ram, also known as Lord Rama, is a major deity in Hinduism and is revered as the seventh avatar of Vishnu. He is known for his virtues, including righteousness and bravery, and his story is detailed in the epic Ramayana.

Speaking: Shri Ram, also known as Lord Rama, is a major deity in Hinduism and is revered as the seventh avatar of Vishnu. He is known for his virtues, including righteousness and bravery, and his story is detailed in the epic Ramayana.

hotword detected

Playing sound...

Sound played successfully.

Listening...

Recognizing...

User said: open canva

Command received: open canva

Speaking: Opening canva

Command received: send message to tanisha

70202 62293

Figure 7.1: Screenshots of Implementaions

## 7.5 Result

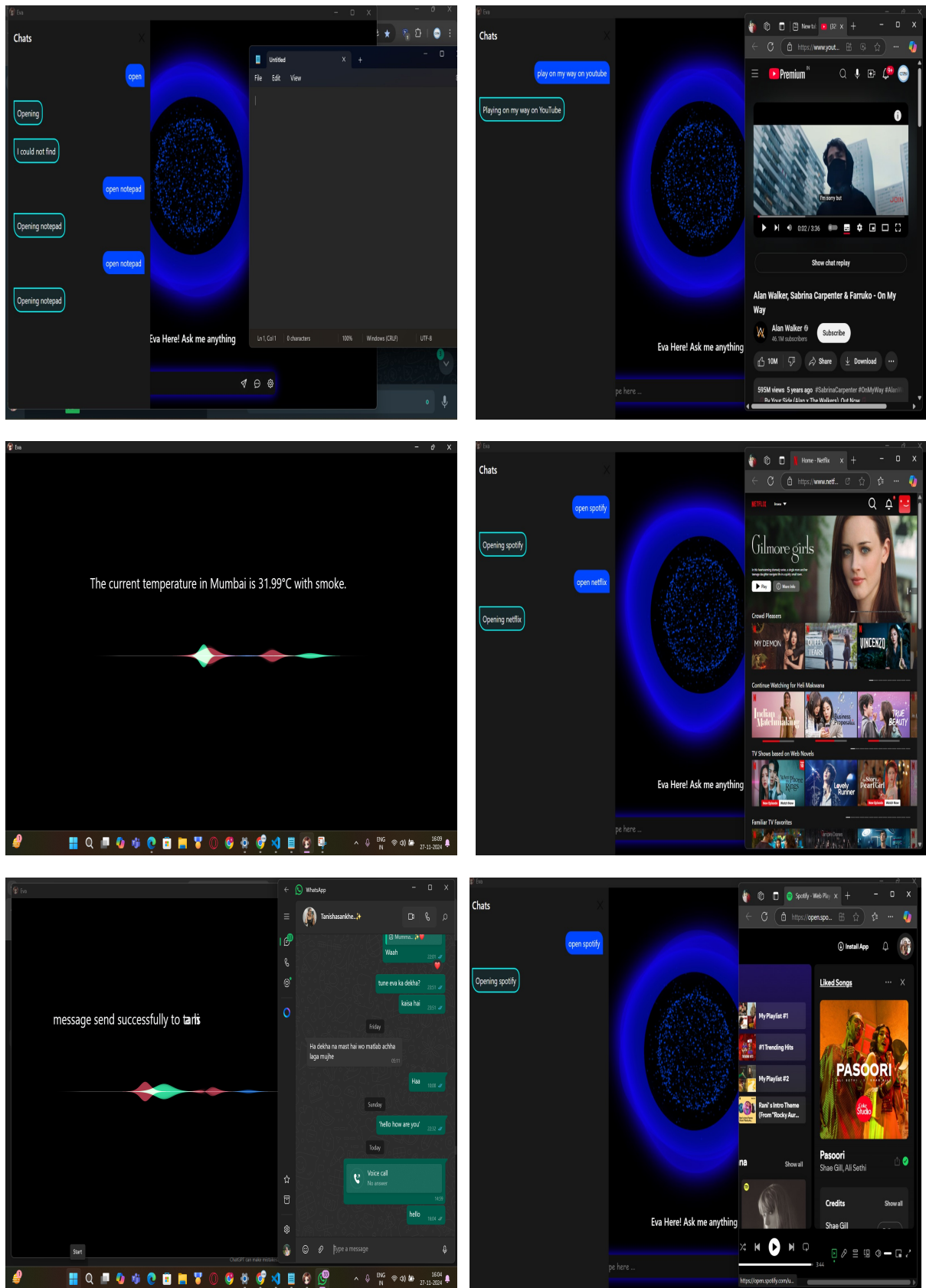


Figure 7.2: Screenshots of various functionalities

## Chapter 8

# Applications

- **Personal Assistants:** Python-based virtual assistants can serve as personal productivity tools, helping users manage their schedules, set reminders, organize tasks, and provide relevant information. Users can interact with the assistant using natural language commands, making it easy to delegate tasks and access information hands-free.
- **Customer Service Chatbots:** Python-based virtual assistants can be deployed as chatbots on websites, mobile apps, and messaging platforms. Using AI-driven natural language processing, the assistant can understand queries, provide solutions, and offer customer support and assistance, improving customer satisfaction and reducing the workload on human support agents.
- **Voice-Activated Applications:** Python-based virtual assistants can be integrated into various applications to enable voice-controlled interaction and hands-free operation. This includes voice-activated navigation systems, voice-controlled media players, voice-activated gaming experiences, and voice-controlled vehicle interfaces.
- **Educational Tools:** Python-based virtual assistants can be utilized as educational tools to provide personalized learning experiences and assistance with homework, research, and studying. The assistant can answer questions, explain concepts, recommend educational resources, and adapt its responses based on the user's learning progress and preferences.
- **Healthcare Applications:** In the healthcare industry, Python-based virtual assistants can assist healthcare professionals with tasks such as patient record management, appointment scheduling, medication reminders, and medical research. Additionally, they can provide patients with information, answer medical questions, and offer guidance on healthy living practices.
- **Business Process Automation:** Python-based virtual assistants can streamline business workflows by automating repetitive tasks, generating reports, scheduling meetings, managing emails, and providing data analysis insights. This improves efficiency, reduces errors, and frees up employees to focus on more strategic tasks.
- **Accessibility Solutions:** Python-based virtual assistants can improve accessibility for individuals with disabilities by offering voice-controlled interfaces, screen reader features, and personalized assistance like speech-to-text and text-to-speech. With AI integration and voice recognition, these assistants provide versatile solutions. As AI technology advances, the potential for innovative, impactful applications of Python-powered virtual assistants continues to grow.

## Chapter 9

# Scope / Future Work

The major milestone that this project tries to achieve is to increase the accuracy of the speech-to-text software.

- **Enhanced Security Protocols:** Integrate AI-driven 2FA methods that adapt based on user behavior and context, improving security while minimizing user friction.
- **Voice Detection:** Enhancing voice detection capabilities enables the assistant to accurately recognize and differentiate between different users' voices. This enables personalized interactions and tailored responses based on the user's voice profile. Implementing speaker recognition algorithms can help achieve this functionality.
- **Agricultural Domain Ideas:**
  - Crop Monitoring:** Use AI to analyze data from sensors and drones to monitor crop health, soil conditions, and weather patterns, providing farmers with actionable insights.
  - Precision Farming:** Implement AI algorithms to optimize planting, watering, and harvesting schedules based on real-time data, enhancing yield and resource efficiency.
  - Pest and Disease Detection:** Develop AI systems that utilize image recognition to identify pests and diseases in crops, allowing for timely intervention and reducing chemical use.
- **Facial Recognition for Secure Access:** In our AI assistant project, we plan to implement facial recognition technology to enhance security. This feature will ensure that only authorized users can access the system. When a user registers their face, the AI will securely store the facial data and use it solely for authentication purposes. This means that the AI will not share or disclose any facial information to others, ensuring user privacy and data protection. By utilizing facial recognition, we aim to provide a seamless and secure access experience, reinforcing trust in our AI assistant.
- **Integration with External Services** Strengthening integration with external services and APIs enables the assistant to access a broader range of functionalities and information from third-party platforms. This includes integrating with popular services such as email providers, calendars, social media platforms, e-commerce websites, and content streaming services.

## Chapter 10

# Conclusion

The development of a Python-based virtual assistant represents a significant advancement in leveraging artificial intelligence (AI) integration, voice recognition, and intelligent task execution to create a versatile and user-friendly digital companion. Through the utilization of cutting-edge technologies and programming techniques, such as natural language processing (NLP) and machine learning (ML), Python-based virtual assistants offer a wide array of functionalities and capabilities to users.

Voice recognition plays a crucial role in enhancing user experience by allowing users to interact with the virtual assistant through spoken commands. By integrating speech recognition libraries like *SpeechRecognition* or using cloud-based APIs such as Google Cloud Speech-to-Text or Amazon Transcribe, developers can enable real-time speech-to-text conversion, enabling seamless communication between users and the virtual assistant.

Intelligent task execution enables the virtual assistant to perform a wide range of tasks on behalf of the user, ranging from simple inquiries to complex operations. Through the implementation of intelligent algorithms and decision-making mechanisms, the assistant can automate repetitive tasks, manage schedules, fetch information from external sources, and execute commands across various applications and platforms.

In conclusion, the development of a Python-based virtual assistant offers a promising solution for enhancing productivity, efficiency, and user experience in both personal and professional contexts. By harnessing the power of AI integration, voice recognition, and intelligent task execution, developers can create sophisticated and adaptable assistants capable of addressing diverse user needs and preferences. As technology continues to advance, Python-based virtual assistants are poised to become indispensable tools for streamlining daily tasks, accessing information, and facilitating seamless communication in the digital age.



# References

- [1] Sri Meenakshi Pandey, S Sindu, Ms C A Daphine Desona Clemency, *AI Based Virtual Assistant*, Volume 8, Issue 3, ISSN: 2456-3315, IJRTI, 2023.
- [2] Vyshnavi, Sudhakar Reddy, Raju Kumar, Saumya, Snehal Nitin Patil, *Virtual Assistant Using Artificial Intelligence and Python*, International Journal of Emerging Technologies and Innovative Research, ISSN:2349-5162, Vol.7, Issue 3, pp.1116-1119, March 2020.
- [3] Umapathi, G Karthick, N Venkateswaran, R Jegadeesan, Dava Srinivas, *Desktop Virtual Assistant Using Python*, Research Gate, July 2023.
- [4] Bansal, Ritik Karnwal, Vishal Singh, Hardik Bansal, *Genesis-The Digital Assistant (Python)*, International Journal of Engineering Applied Sciences and Technology, Vol. 5, Issue 1, ISSN No. 2455-2143, Published Online May 2020 in IJEAST.
- [5] Aditi Bombe, Atharva Kale, Atharva Nayak, Anushka Darure, Yash Chavan, Ashutosh Deorukhkar, *Personal Desktop AI Assistant Using Python*, International Research Journal of Engineering and Technology, Volume: 10, Issue: 09, September 2023.