**AARAV – Advanced Artificial Response and Voice**

Facial Recognition

Using CV2 for capturing video frames from the webcam and drawing rectangles around detected faces. Also used for capturing face samples.

Haar Cascade is a machine learning approach for object detection particularly to detect faces.

Integral Image: - Computing Integral images for rapid calculations of pixel sum within a rectangular area of the image.

Haar like feature: - Feature defined by the difference in the sum of pixel intensities.

Adaptive Boosting: - Select a subset of Haar like features and create a strong classifier from them.

Cascade Classifier and Cascade Detection.

Non-Maximum Suppression: - To reduce redundant detections and refine the final detection results, non-maximum suppressions can be applied.

**Code: -**

VideoCapture(0) is used to start the camera.

Cascadeclassifier, detectMultiscale, and rectangle attribute help in detecting the face.

The coordinates of the detected face are used to extract the region of interest (ROI) from the frame using array slicing (frame[y:y + h, x:x + w]).

Cam. set to set the resolution of captured frames

OS is used to make the directory for the samples to be stored. The folders will be stored based on the ID Input provided.

**Face recognition: -** The library utilizes pre-trained deep-learning models to detect faces in images or video frames. These models employ convolutional neural networks (CNNs) to identify facial features and localize faces within an image. Face Encoding and Comparison with the Samples. Face encodings capture distinctive features of a face in a high-dimensional vector space.

load\_known\_face compares the detected face encodings with the known face encoding.

**Working: -**

Based on the user ID the person has been named in the main file where AARAV calls the name when the face is recognized. It turns off if the face does not match in face recognition. So basically the work is only accessible to the members with the available samples in the sample folder. It consists of Samplegenrator.py and facerecognition.py majorly implemented in the main code itself.

Response and Voice

Pyttsx3 is a text-to-speech conversion library. It allows developers to synthesize human-like speech from text strings. It uses different speech synthesis engines and for Windows, it utilizes sapi5 speech synthesis API.

Speech\_Recognition allows developers to convert spoken language audio into text. It interfaces with cloud-based speech recognition models, such as Google speech recognition. It has highly trained data with Recurrent neural network (RNN) and convolutional neural network (CNN) to decode the spoken language into text.

**Code: -**

Pyttsx3 creates an engine for the text-speech conversion using the sapi5 API.

It retrieves voices using engine.getProperty and sets the voice based on its specific ID. It also sets the rate of speech per minute.

Engine.say passes the text audio to synthesis in speech recognition.

Enigine.runandWait waits until a speech is completed by the user.

Speech\_recognition.microphone uses the available device microphone and listens to the audio of the user's speech.

Recognize\_google converts the audio into text by specifying the language. Here en-in specifies English Indian language.

Basic Approach & Logic

The model works on understanding the query from the user's speech and then looking for a specific keyword, once the keyword gets detected by the system the system accordingly performs the conditional statement and performs that particular task assigned for the respective keyword condition. So most of the program is based on conditional statements and their execution to the specific keyword. It has a query data provided for the response to specific questions asked while the program execution.

Automation

**Call Function: -**

It is completely based on the Twilio account where we get the specific SID and an auth token, we can also request the phone number. We get a specific contact no. for the specific ID. Specifying a phone number to whom to call and call including the specific name of the contact helps in directly commanding the main model to call by name and implement the calling.

Client.calls.create is used to initiate a phone call.

The specific twilio library is available for proper implementation of the calling function.

**Keyboard Functionality: -**

A specific pynput keyboard library is used to simulate keyboard inputs.

The controller attribute is used to initialize an instance to simulate key presses.

Majorly the part of this library and file is to increase the media volume.

Keyboard.press and keyboard.release it creates a volume up and down as it presses it and then it releases the key.

**Scrolling Functionality: -**

Two libraries have a major role in this feature pyautogui and threading/thread.

When the command of scroll up is implemented the self running up becomes true and hence it scrolls up by 10 units using pyautogui.

The run function is implemented with the threading, where it simultaneously runs or executes this program even when the main program is running. It creates a different thread to run the program.

If the command is self running down then it is true and self running up is false hence this helps to scroll down by 10 units using pyautogui.

Pyautogui provides cross platform support for controlling the mouse, screen, and keyboard. It helps in automating repetitive tasks or interacting with GUI applications programmatically.

Assistant Specific Functions

**Calculator: -**

The WolfRamAlpha function takes a query as input and returns the result from the Wolfram|Alpha knowledge engine. It is a library that knows to calculate the queries provided.

WolfRamAlpha.Client initializedsobject to make queries to the API.

Requester.query send query to the API and retrieves the data.

Major role of calc function is to take the query from woeds and convert them to suitable operators (e.g., "multiply" with "\*", "plus" with "+").

**GreetMe: -**

It has a basic functionality for accessing date time and have provided a logic of how to greet the user. If the hour > 0 and hour <= 12 – speak Good morning and so on.

**Presentation Making: -**

It majorly works on two libraries as webbrowser and pyautogui. Webbrowser helps in accessing the website and pyautogui helps to execute automation.

Slidsgo is a AI tool that works in a similar way, where we have to provide a prompt/query and it develops presentations based on the provided query.  
Specific coordinates are provided for the mouse pointer for it to click and where to click to perform the automation. User has to provide the query of what presentation he wants and rest automation works on the site to directly download the presentation for us.

**Email: -**

Main tech for the operation of this function is smtplib which is a server.

It takes parameters such as sender's email, sender's password, recipient's email, subject, and content of the email.

It constructs an Emailmessage object and it sets up connection with the SMTP server, logs in using sender's credentials, sends the email message, and quits the server connection.

It takes the content as sender email, receivers email, subject and message.

587 Represents the Port of the server.

We have to access the senders gmail and generate a password for third party execution and provide the password as the sender’s password instead of the email password.

**Whatsapp: -**

The main tech used for this library is pywhatkit library. Which helps in the interaction of whatsapp with the system.  
Datetime library is used for proper time addressing. Here we have to provide the specific number in the data. Once it recognizes to whom it has to send the message to. It does it in about 90seconds and connects to whatsapp web and sends the query.  
Pywhatkit also stores a data in text file, of to whom the msg has been sent and what was the message.  
  
GPT

As we know Chat GPT is renowned and popular platform where we can ask, mostly anything and it will give us the answer. If a person has to access it, let him speak the query and command to ask gpt, it will add to the query to the prompt and execute it.  
It uses Webbrowser, Pyautogui and time to open, automate and provide a sleep time respectively.

TODO

The tech behind this functionality is simple file handling. As the user assigns task to itself, in the function. Aarav saves the task in a text file in a stack order with proper bulleting. Once the user calls the todo it simply reads the file and gets the desired output.  
It asks if the person wants to clear the last todo. And so it simply writes(f” ”) which makes the file blank. Or it simply ask to add more and waits for the query and then it writes the respective query. Counter i is placed which simply helps in bulleting and adding more ahead of the todo which was already present. The function used in file handing were file write “w” and file append “a”. While reading it simply has to open file as “r” function, which is read function and it reads the file.

Web Scrapping

DictApp: - It is an app dictionary provided that opens the respective dictionary app using .com to open the web browser. Eg: - PowerPoint, excel, etc. Otherwise, it directly runs using the os library which interacts with Windows to directly open the app listed in the query.

Dominos / Olacabs / Myntra / Amazon / Flipkart: - It accesses the site using a web browser, and it uses pyautogui to access the coordinates so that it can itself automate without human interaction. It does all the work that needs to be done by any user while looking for something on the web.

SearchNow: - provides access to Google, YouTube, and Wikipedia. It uses web browser, pywhatkit to search that specific query.  
Google, it uses the googlescrap library to search from the query.  
For YouTube, it specifically adds the query to the URL which directly opens the search query site.  
For Wikipedia, it already has a library which accesses the data and gives the result in the command prompt.