# **AI Mentee - Resume Maker**

**Step-by-Step Process**

**1. Setting Up Dependencies**

The script begins by importing necessary libraries:

* speech\_recognition for capturing and transcribing audio.
* google.generativeai for analyzing and categorizing the transcribed text.
* os for environment variable handling (if needed).
* googletrans for translating the analyzed text.

#code

import speech\_recognition as sr

import google.generativeai as genai

import os

from googletrans import Translator

**2. Configuring Google Cloud Credentials**

The script sets up the credentials for Google Cloud to use the Generative AI API. This involves configuring the API key.

#code

genai.configure(api\_key='YOUR\_API\_KEY\_HERE')

model = genai.GenerativeModel('gemini-pro')

**3. Initializing Components**

Three key components are initialized:

* recognizer for capturing and recognizing speech.
* model for interacting with the Gemini Pro API.
* translator for translating text.

code

recognizer = sr.Recognizer()

translator = Translator()

**4. Recording Audio**

The record\_audio function captures audio input from the microphone.

#code

def record\_audio():

with sr.Microphone() as source:

print("Speak now...")

audio = recognizer.listen(source)

return audio

**5. Transcribing Audio**

The transcribe\_audio function transcribes the recorded audio to text using Google’s speech recognition service.

#code

def transcribe\_audio(audio):

try:

text = recognizer.recognize\_google(audio)

return text

except sr.UnknownValueError:

return "Speech recognition could not understand the audio"

except sr.RequestError as e:

return f"Could not request results from speech recognition service; {e}"

**6. Analyzing Text**

The analyze\_text function sends the transcribed text to the Gemini Pro model to categorize it into sections typically found in a resume.

#code

def analyze\_text(text):

prompt = f"""

Analyze the following text and categorize it into different sections of an ATS resume:

{text}

Categories should include:

1. Name

2. Email

3. Contact

4. Career Objective

5. Education

6. Project/Internship

7. Skills

8. Achievements

For each category, provide the relevant information from the text without using stars (\*) in the headings.

"""

response = model.generate\_content(prompt)

return response.text

**7. Translating Text**

The translate\_text function translates the analyzed and categorized text into the selected language.

#code

def translate\_text(text, lang\_code):

try:

translated = translator.translate(text, dest=lang\_code)

return translated.text

except Exception as e:

return f"Translation error: {e}"

**8. Selecting Language**

The get\_language\_choice function allows the user to select a language for the translated resume content.

#code

def get\_language\_choice():

print("Select language for the resume content:")

print("1. English")

print("2. Telugu")

print("3. Hindi")

choice = input("Enter the number corresponding to your choice: ")

if choice == '1':

return 'en'

elif choice == '2':

return 'te'

elif choice == '3':

return 'hi'

else:

print("Invalid choice. Defaulting to English.")

return 'en'

**9. Main Function**

The main function ties all the components together, performing the entire process from recording to translating.

#code

def main():

audio = record\_audio()

transcribed\_text = transcribe\_audio(audio)

print("Transcribed text:", transcribed\_text)

analysis = analyze\_text(transcribed\_text)

print("\nAnalysis and categorization:")

print(analysis)

lang\_code = get\_language\_choice()

translated\_analysis = translate\_text(analysis, lang\_code)

print("\nTranslated Resume Content:")

print(translated\_analysis)

**10. Entry Point**

The script entry point ensures the main function is called when the script is executed.

#code

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Model Used In Working**

**Speech Recognition Model**

The script uses the speech\_recognition library, which interfaces with Google’s Web Speech API to convert spoken language into text. This service provides highly accurate speech-to-text capabilities.

**Gemini Pro Model**

The core of the text analysis and categorization is handled by the Gemini Pro model from Google’s Generative AI suite. The model is designed for generating and understanding complex content.

#code

model = genai.GenerativeModel('gemini-pro')

The analyze\_text function sends a prompt to the Gemini Pro model, asking it to categorize the provided text into specific resume sections. The model processes the input and returns a structured response based on the provided categories.

**Translation Model**

For translation, the script uses the googletrans library, which interacts with the Google Translate API. This allows the script to convert the analyzed resume content into various languages based on user preference.

#code

translator = Translator()

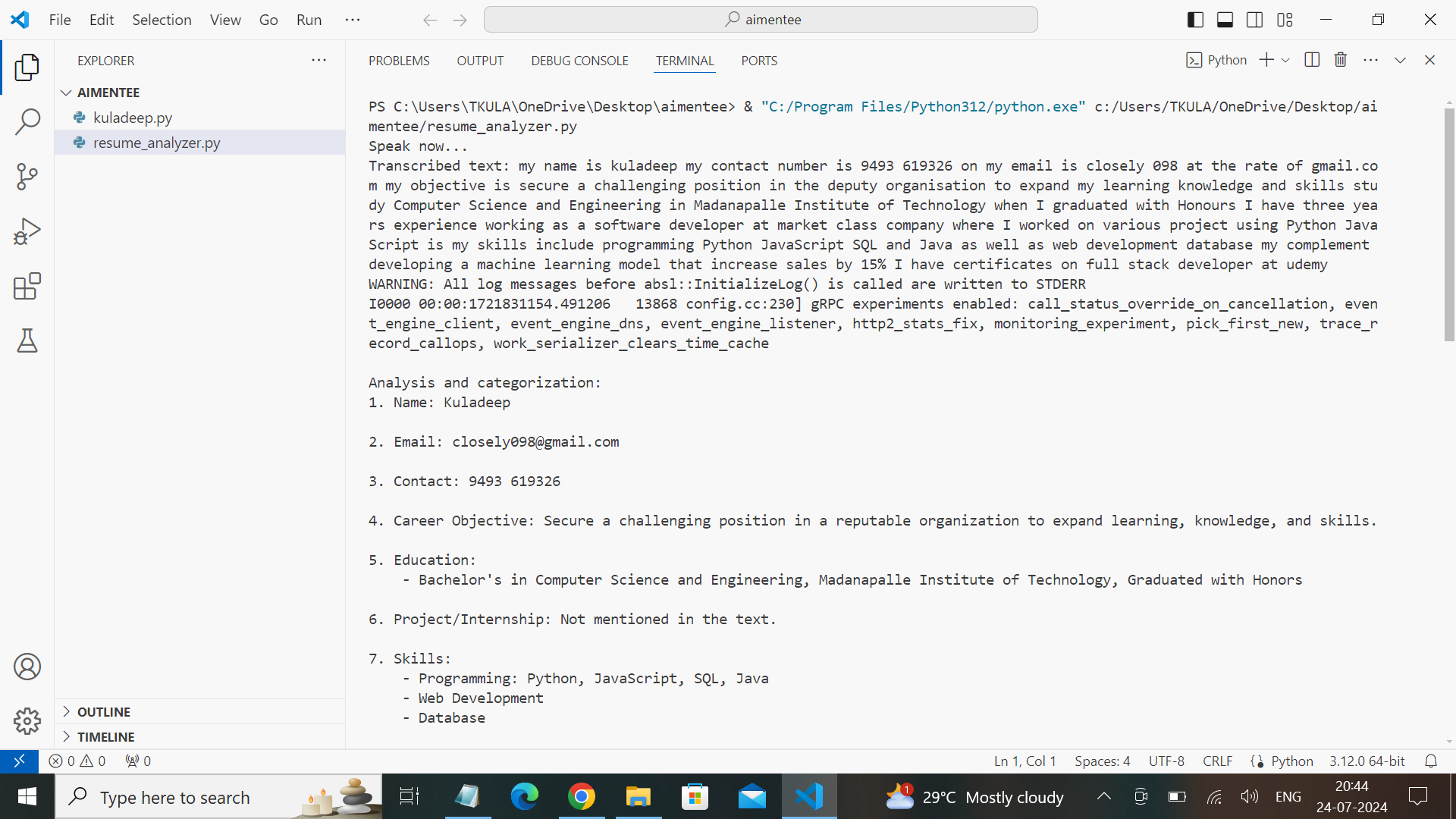
**Integration of Models**

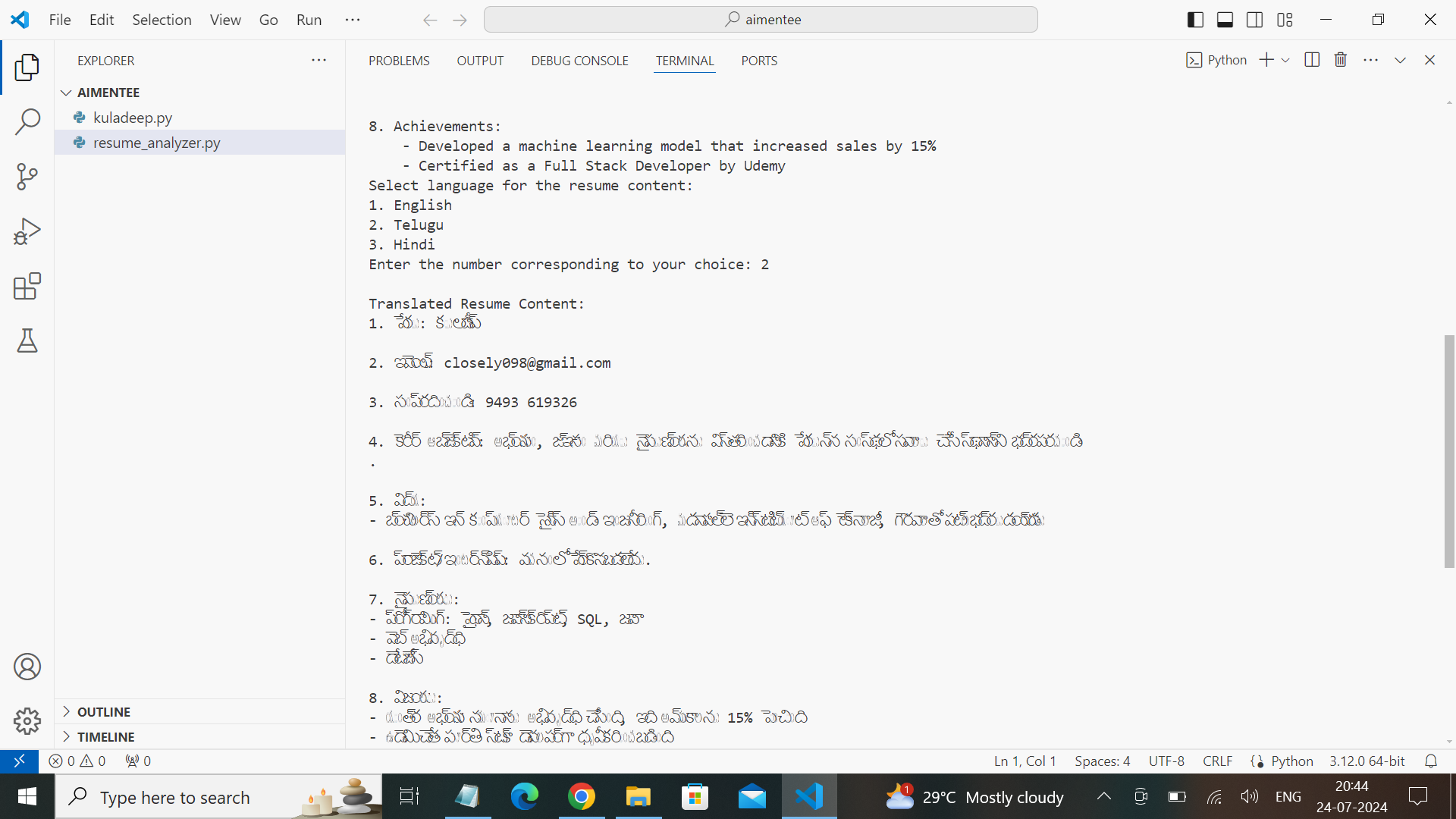
The combination of these models allows the script to achieve its functionality:

1. Speech recognition captures and converts spoken input into text.
2. The Gemini Pro model analyzes and categorizes the transcribed text.
3. The Google Translate API translates the categorized text into the desired language.

By leveraging these advanced models, the script automates the process of creating a structured and translated resume from spoken input, providing a seamless and efficient workflow.

**Outputs:**

****

****