**User will store there api key in our db and based on user id system should get the key and perform the below functions. So don’t hard code and replace the hardcode way in the below code.**

**Serverless function**

**Function**

Attached is the python for object detection model. Where user can train and use to detect objects in the video.

Training video <https://www.youtube.com/watch?v=mf2f4Cx-JNs>

Scanning video <https://youtu.be/bKomEJpsLWM>

I am looking for a customization:

I need one model where only authorized user can train the model by feeding images.

Second each user can create their own custom model using their own images and save it privately and select the model to scan the objects, user can select multiple models to scan.

Identified images should be displayed in the page immediately and convert into album.

All identified images should be converted into zip and available for user to download.

**Function**

Python code that uses the SpeechRecognition library to listen to the user's microphone, then detects the language using the langid library. The recognized language and the user's input are then sent to the ChatGPT API using the OpenAI library, and the response is displayed. The conversation continues until the user says "bye Kumar".

To run this code, you'll need to install the following dependencies:

- SpeechRecognition: `pip install SpeechRecognition`

- langid: `pip install langid`

- OpenAI: `pip install openai`

```python

import speech\_recognition as sr

import langid

import openai

# Set up the OpenAI API credentials

openai.api\_key = 'YOUR\_API\_KEY'

def recognize\_language(text):

# Use langid to detect the language of the input text

lang, \_ = langid.classify(text)

return lang

def get\_chatgpt\_response(text, lang):

# Use the OpenAI ChatGPT API to get a response

response = openai.Completion.create(

engine="text-davinci-003",

prompt=text,

max\_tokens=50,

temperature=0.7,

n=1,

stop=None,

temperature=0.7,

language=lang,

model="davinci",

)

return response.choices[0].text.strip()

# Initialize the speech recognizer

recognizer = sr.Recognizer()

# Start listening to the microphone

with sr.Microphone() as source:

print("Listening...")

while True:

try:

audio = recognizer.listen(source)

# Convert speech to text

user\_input = recognizer.recognize\_google(audio)

print("You:", user\_input)

# Detect the language of the user input

language = recognize\_language(user\_input)

# Send the user input to ChatGPT and get a response

response = get\_chatgpt\_response(user\_input, language)

print("ChatGPT:", response)

# Check if the user wants to end the conversation

if "bye Kumar" in response:

break

except sr.UnknownValueError:

print("Could not understand audio")

except sr.RequestError as e:

print("Could not request results; {0}".format(e))

```

Please note that you need to replace `'YOUR\_API\_KEY'` with your actual OpenAI API key, which you can obtain from the OpenAI website. Additionally, you may need to adjust the OpenAI API parameters according to your requirements.

Remember to handle any exceptions and errors that may occur during speech recognition or API calls.

OR

Sure! Here's an example Python project that uses the SpeechRecognition library to listen to the user's microphone, detects the language of the speech using the langdetect library, sends the recognized language and user input to ChatGPT, and responds with the answer in the detected language. The conversation stops when the user says a predefined keyword.

Please note that you'll need to install the required libraries before running this code. You can install them using pip by running `pip install SpeechRecognition langdetect openai`.

```python

import speech\_recognition as sr

from langdetect import detect

import openai

# Set up your OpenAI API credentials

openai.api\_key = 'YOUR\_API\_KEY'

# Define the keyword to stop the conversation

STOP\_KEYWORD = 'stop'

# Function to get the response from ChatGPT

def get\_chat\_response(input\_text):

response = openai.Completion.create(

engine='text-davinci-003',

prompt=input\_text,

max\_tokens=50,

temperature=0.7,

n=1,

stop=None,

temperature=0.7

)

return response.choices[0].text.strip()

# Function to detect the language of the input text

def detect\_language(input\_text):

return detect(input\_text)

# Function to listen to user's microphone and process speech

def listen\_and\_chat():

recognizer = sr.Recognizer()

microphone = sr.Microphone()

with microphone as source:

print("Speak something...")

recognizer.adjust\_for\_ambient\_noise(source)

audio = recognizer.listen(source)

try:

user\_input = recognizer.recognize\_google(audio)

print("You said:", user\_input)

# Detect the language

language = detect\_language(user\_input)

print("Detected language:", language)

while True:

if user\_input.lower() == STOP\_KEYWORD:

print("Conversation stopped.")

break

# Send the user input to ChatGPT and get the response

response = get\_chat\_response(user\_input)

# Print the response in the detected language

print("ChatGPT:", response)

# Listen for the next user input

print("Speak something...")

audio = recognizer.listen(source)

user\_input = recognizer.recognize\_google(audio)

print("You said:", user\_input)

# Detect the language of the new input

language = detect\_language(user\_input)

print("Detected language:", language)

except sr.UnknownValueError:

print("Could not understand audio.")

except sr.RequestError as e:

print("Error: {0}".format(e))

# Run the chat program

listen\_and\_chat()

```

Make sure to replace `'YOUR\_API\_KEY'` with your actual OpenAI API key. You can obtain an API key by signing up for the OpenAI GPT-3 API.

Note that the response from ChatGPT may not always be in the same language as the user input due to the nature of the language model. You may need to use additional libraries or APIs for language translation if you want the response to always match the detected language.

**Function 4:**

Certainly! Here's an example of a Python AI project that uses natural language processing (NLP) to generate an image based on user input. In this example, we'll be using the OpenAI GPT-3.5 language model and the OpenAI DALL-E image generation model or stable diffusion or other best to run on our local server.

To get started, make sure you have the OpenAI Python package installed. You can install it using pip:

```shell

pip install openai

```

Once you have the package installed, you'll need an API key from OpenAI. You can follow their documentation on how to obtain an API key.

Here's the code for the project:

```python

import openai

import requests

from PIL import Image

from io import BytesIO

# Set up OpenAI API credentials

openai.api\_key = 'YOUR\_API\_KEY'

# Generate image based on NLP input

def generate\_image\_from\_nlp(input\_text):

# Generate image description using OpenAI GPT-3.5

response = openai.Completion.create(

engine='text-davinci-003',

prompt=f"Generate an image based on the following text: {input\_text}",

max\_tokens=50,

temperature=0.7,

n = 1,

stop=None,

timeout=10,

log\_level="info",

logprobs=0

)

# Extract the image description from the API response

choices = response.choices[0].text.strip().split('\n')

image\_description = choices[0]

# Generate image using OpenAI DALL-E

response = openai.Completion.create(

engine='davinci',

prompt=f"Image description: {image\_description}",

max\_tokens=150,

temperature=0.7,

n = 1,

stop=None,

timeout=10,

log\_level="info",

logprobs=0

)

# Extract the image URL from the API response

image\_url = response.choices[0].text.strip().split('\n')[0]

# Download and display the generated image

response = requests.get(image\_url)

image = Image.open(BytesIO(response.content))

image.show()

# Accept user input and generate image

user\_input = input("Enter a text description: ")

generate\_image\_from\_nlp(user\_input)

```

Make sure to replace `'YOUR\_API\_KEY'` with your actual OpenAI API key.

This code takes user input as a text description and uses the GPT-3.5 model to generate a more concise image description. It then passes this image description to the DALL-E model, which generates an image based on the description. Finally, the generated image is downloaded and displayed.

Please note that using OpenAI models may incur costs, so make sure to review the pricing details on the OpenAI website.

Remember to use this code responsibly and comply with OpenAI's usage policies.

Python project that uses visual input to detect objects and respond with their names. To achieve this, we'll use the OpenCV library for image processing and the pre-trained MobileNetSSD model for object detection.

Before we begin, make sure you have OpenCV and the MobileNetSSD model installed. You can install OpenCV using pip:

```

pip install opencv-python

```

Now let's write the code:

```python

import cv2

def detect\_objects(image\_path):

# Load the pre-trained MobileNetSSD model

net = cv2.dnn.readNetFromCaffe('MobileNetSSD\_deploy.prototxt.txt', 'MobileNetSSD\_deploy.caffemodel')

# Load the input image

image = cv2.imread(image\_path)

# Resize the image for processing (optional, modify as per your requirements)

image\_resized = cv2.resize(image, (300, 300))

# Create a blob from the image

blob = cv2.dnn.blobFromImage(image\_resized, 0.007843, (300, 300), 127.5)

# Set the blob as input to the network

net.setInput(blob)

# Perform object detection

detections = net.forward()

# Iterate over the detections and extract object names

objects = []

for i in range(detections.shape[2]):

confidence = detections[0, 0, i, 2]

# Minimum confidence threshold (adjust as per your requirements)

if confidence > 0.5:

class\_id = int(detections[0, 0, i, 1])

class\_name = classNames[class\_id]

objects.append(class\_name)

return objects

# List of class names for MobileNetSSD

classNames = {0: 'background', 1: 'aeroplane', 2: 'bicycle', 3: 'bird', 4: 'boat', 5: 'bottle', 6: 'bus',

7: 'car', 8: 'cat', 9: 'chair', 10: 'cow', 11: 'diningtable', 12: 'dog', 13: 'horse',

14: 'motorbike', 15: 'person', 16: 'pottedplant', 17: 'sheep', 18: 'sofa', 19: 'train',

20: 'tvmonitor'}

# Path to the input image

image\_path = 'input\_image.jpg'

# Detect objects in the image

detected\_objects = detect\_objects(image\_path)

# Print the detected object names

if len(detected\_objects) > 0:

print('Detected objects:')

for object\_name in detected\_objects:

print(object\_name)

else:

print('No objects detected.')

```

Make sure you have the MobileNetSSD model files `MobileNetSSD\_deploy.prototxt.txt` and `MobileNetSSD\_deploy.caffemodel` in the same directory as your Python script. These files can be downloaded from the OpenCV GitHub repository: [MobileNetSSD](https://github.com/opencv/opencv/blob/master/samples/dnn/mobilenet\_ssd/MobileNetSSD\_deploy.prototxt.txt) and [MobileNetSSD model](https://github.com/opencv/opencv\_3rdparty/blob/dnn\_samples\_face\_detector\_20170830/res10\_300x300\_ssd\_iter\_140000.caffemodel).

Replace `'input\_image.jpg'` with the path to your desired input image.

When you run this script, it will detect objects in the input image using

the MobileNetSSD model and display the names of the detected objects. You can adjust the confidence threshold and resize the image as per your requirements.

Note: Object detection accuracy may vary depending on the complexity and quality of the input image, as well as the capabilities of the MobileNetSSD model.