Task -1

```
!pip -q install pytesseract transformers
!pip -q install pdfplumber PyMuPDF pymupdf
!pip -q install langchain-community
```

```
from PIL import Image
import pytesseract
from transformers import CLIPProcessor, CLIPModel
import torch

# Load logo
image = Image.open("sample_image.png")

# OCR to extract text
ocr_text = pytesseract.image_to_string(image)

# Use CLIP for visual embedding
processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")

inputs = processor(images=image, return_tensors="pt")
with torch.no_grad():
    image_features = model.get_image_features(**inputs)

image_features = image_features / image_features.norm(dim=-1, keepdim=True)
```

```
Using a slow image processor as `use_fast` is unset and a slow processor was saved with this model. `use_fast=True` will be the defa
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
preprocessor_config.json: 100%          316/316 [00:00<00:00, 16.8kB/s]
tokenizer_config.json: 100%             592/592 [00:00<00:00, 44.8kB/s]
vocab.json: 100%                        862k/862k [00:00<00:00, 5.29MB/s]
merges.txt: 100%                        525k/525k [00:00<00:00, 8.18MB/s]
tokenizer.json: 100%                    2.22M/2.22M [00:00<00:00, 17.8MB/s]
special_tokens_map.json: 100%           389/389 [00:00<00:00, 23.0kB/s]
config.json: 100%                       4.19k/4.19k [00:00<00:00, 359kB/s]
pytorch_model.bin: 100%                 605M/605M [00:03<00:00, 125MB/s]
model.safetensors:   2%                 10.5M/605M [00:00<00:22, 26.0MB/s]
```

```
image_features
```

```
tensor([[ 2.5519e-02, -1.1114e-02, -1.3579e-02, -4.5351e-03,  2.8519e-03,
         -1.5786e-02,  1.2823e-02,  3.6550e-02, -4.7057e-02,  3.0452e-02,
         -1.0689e-02, -8.5247e-03, -6.5664e-03, -5.9172e-02, -1.7665e-02,
         -1.7372e-03, -5.3519e-02,  1.9526e-02, -1.3455e-02,  2.7507e-02,
          2.3673e-02,  1.0063e-02, -3.1868e-02, -3.3355e-02,  4.0332e-02,
         -2.5852e-03,  1.7108e-02,  2.5900e-02, -4.0285e-03, -1.0737e-02,
         -3.9632e-02,  1.3385e-02, -1.3765e-02,  2.5334e-02,  1.0374e-02,
          1.0276e-02,  1.2495e-03, -3.2030e-03, -2.5184e-02,  5.0658e-03,
          2.3535e-02,  2.1341e-02,  1.7921e-02, -2.4038e-02,  2.4659e-02,
         -4.8051e-02, -2.9449e-02,  4.8353e-02,  1.3849e-02, -2.0298e-02,
          1.8593e-02,  4.2167e-02, -3.0245e-03, -3.0518e-02, -1.6168e-03,
         -1.7304e-02,  1.3302e-02,  2.2900e-02, -1.1510e-02,  1.6507e-02,
         -6.1334e-02,  9.3461e-03, -9.3433e-03, -5.0401e-03, -8.7980e-03,
          2.4645e-02, -9.9170e-03, -2.3846e-02, -9.0936e-03,  2.3542e-02,
          8.8370e-04, -1.6780e-02,  7.4740e-03, -1.9636e-02,  9.5290e-04,
          2.6817e-03, -2.9583e-02, -2.2816e-02, -2.0897e-02, -2.6198e-02,
          2.9381e-02, -3.9887e-03, -1.0359e-02,  1.9464e-02, -2.5479e-02,
         -1.1079e-02, -8.1007e-02, -7.4993e-03,  1.5620e-02, -1.9340e-02,
          6.5425e-03, -3.8757e-02, -7.0047e-01,  8.6590e-03, -2.3385e-03,
          3.9947e-03,  1.7094e-02,  2.5841e-02, -8.0459e-03, -1.5003e-01,
         -1.5397e-02, -3.1707e-03,  1.2637e-03, -1.1419e-02,  7.1068e-02,
          1.2016e-02, -1.0402e-02,  5.2495e-02,  1.6313e-02,  1.6215e-02,
          8.7162e-03,  7.7563e-03, -5.1079e-02, -8.8477e-03,  8.7708e-03,
         -1.6996e-02, -2.8059e-02, -1.2292e-02, -6.7356e-03, -6.1008e-03,
          2.5683e-02, -6.7511e-03, -5.6203e-03, -3.0527e-03, -6.5042e-03,
          1.5079e-02, -1.6155e-03, -2.9505e-02,  3.0662e-02,  2.7249e-02,
         -1.6570e-02,  1.5257e-02, -3.5479e-02,  8.9708e-02,  1.0306e-02,
         -4.6791e-03, -4.0394e-03, -6.7923e-02,  3.7135e-02, -3.0569e-02,
          2.0091e-02,  1.7411e-02,  1.3112e-02, -1.0037e-02,  8.4228e-03,
         -2.2102e-02,  1.0044e-02, -1.2146e-02, -1.0443e-02,  5.4293e-04,
          7.5151e-03, -8.3462e-03,  6.7400e-02, -4.6766e-02,  1.8373e-02,
         -4.7313e-03, -1.1123e-02,  2.4041e-03,  3.9111e-02, -2.5795e-02,
          1.8637e-02,  1.6821e-03,  2.2563e-02, -2.7228e-02,  2.9442e-02,
         -1.5701e-02, -7.0898e-03,  4.1106e-02, -2.4416e-02, -7.1117e-03,
          2.7092e-03,  1.3999e-02,  2.2418e-02,  3.3166e-02,  7.2585e-03,
          1.1048e-02, -2.0473e-02,  1.1957e-02, -7.2472e-02,  2.4266e-02,
          8.4740e-03,  1.6235e-02, -4.8646e-02, -2.4992e-02, -2.1755e-02,
         -1.0885e-02,  3.6625e-02,  7.6480e-03, -4.9306e-02,  1.2447e-02,
          5.5848e-03,  4.5531e-02,  3.9081e-02, -1.9220e-03, -7.2571e-02,
          1.5796e-02, -9.7407e-03,  5.1425e-02,  4.0224e-02, -5.8815e-03,
         -3.8145e-02,  1.2267e-02,  1.0354e-02,  1.2456e-02, -1.9603e-02,
          1.3390e-02, -6.0406e-03, -1.8850e-02, -3.2147e-03,  2.7656e-02,
         -2.6026e-02, -3.8071e-02, -3.7901e-02, -1.2622e-04, -3.2299e-02,
         -6.0678e-03, -2.2976e-02,  1.8841e-02,  8.7083e-02, -2.1715e-02,
         -1.8655e-02,  4.2174e-02, -3.6790e-02, -3.2955e-02,  3.1577e-03,
          3.8712e-02, -4.5527e-02, -1.0407e-02, -2.1498e-02,  2.6348e-02,
          2.4987e-02, -1.1190e-02, -3.2222e-03,  6.3683e-02,  7.3417e-03,
          7.2701e-02, -3.1123e-02,  8.4051e-03,  1.6805e-02,  2.9512e-02,
         -9.8815e-03, -9.5176e-03, -4.4571e-04,  5.6544e-03, -5.6002e-02,
          3.2706e-02, -3.8281e-02,  1.7043e-02, -1.4705e-02,  4.6489e-02,
         -2.0826e-02, -3.3353e-03,  1.1230e-02, -3.4515e-02,  1.6882e-02,
          2.2294e-02, -1.9153e-02,  1.8465e-02,  6.7714e-02,  1.8325e-02,
         -5.4093e-02,  9.2072e-03, -5.8645e-02, -2.0085e-02, -6.8993e-03,
         -3.2398e-02, -5.4884e-02,  2.4563e-02,  7.0592e-04, -1.6432e-02,
         -1.0755e-02,  2.7805e-02,  4.1162e-02,  4.9763e-02,  1.9423e-02,
          5.9274e-03, -1.1828e-03, -2.3576e-02, -1.3063e-02,  1.1856e-02,
          2.9036e-02,  8.7146e-03,  4.4350e-03,  5.1019e-03, -9.8939e-04,
          3.1762e-02, -9.9671e-02, -1.6800e-02, -1.6408e-02,  1.8042e-02,
```

Start coding or generate with AI.

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.1/24.1 MB 77.3 MB/s eta 0:00:00
```

```python
import pdfplumber
import fitz  # PyMuPDF

pdf_text = ""
with pdfplumber.open("document.pdf") as pdf:
    for page in pdf.pages:
        pdf_text += page.extract_text()

# Extract images and colors (advanced)
doc = fitz.open("document.pdf")
colors = []
for page in doc:
    for img in page.get_images(full=True):
        pix = fitz.Pixmap(doc, img[0])
        if pix.n >= 4:  # RGBA
            pix = fitz.Pixmap(fitz.csRGB, pix)
        colors.append(pix.get_pixmap().samples)
        pix = None
```

```python
from transformers import pipeline

# Use zero-shot classifier or summarizer
persona_text = """
John is a 34-year-old eco-conscious startup founder. He values minimalism, clarity, and nature-inspired themes.
"""

summary = pipeline("summarization")(persona_text, max_length=60, min_length=20, do_sample=False)
```

```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 and revision a4f8f3e (https://huggingface.co/sshleifer/distilbart
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json:  100%                                  1.80k/1.80k  [00:00<00:00,  25.7kB/s]
pytorch_model.bin:  100%                                1.22G/1.22G  [00:17<00:00,  110MB/s]
model.safetensors:  100%                                1.22G/1.22G  [00:13<00:00,  124MB/s]
tokenizer_config.json:  100%                              26.0/26.0  [00:00<00:00,  406B/s]
vocab.json:  100%                                   899k/899k  [00:00<00:00,  8.48MB/s]
merges.txt:  100%                                   456k/456k  [00:00<00:00,  7.03MB/s]
Device set to use cpu
Your max_length is set to 60, but your input_length is only 32. Since this is a summarization task, where outputs shorter than the i
```

```
import os
os.environ["OPENAI_API_KEY"] = "sk-proj-REzVRJL0cRKtU5NlknjiBpFVYiAHYgv_prBtrsSa5n3F8XyR8-OqmsjK0crfZo-43v2PFQKlqWT3BlbkFJmxRg78oI6RYoU
```

```
from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate

template = """
Based on the following data:
Image OCR Text: {ocr_text}
Image Embedding: {image_description}
PDF Text: {pdf_text}
Persona Summary: {persona}

Create a structured identity vector with keys:
- Brand Tone
- Visual Themes
- Colors / Fonts
- Personality Traits
- Style Keywords
"""

prompt = ChatPromptTemplate.from_template(template)
chat = ChatOpenAI(temperature=0.2)

identity_vector = chat.invoke(prompt.format_messages(
    ocr_text=ocr_text,
    image_description="(describe from CLIP if needed)",
    pdf_text=pdf_text[:1000],  # Truncate if large
    persona=summary[0]['summary_text']
))
```

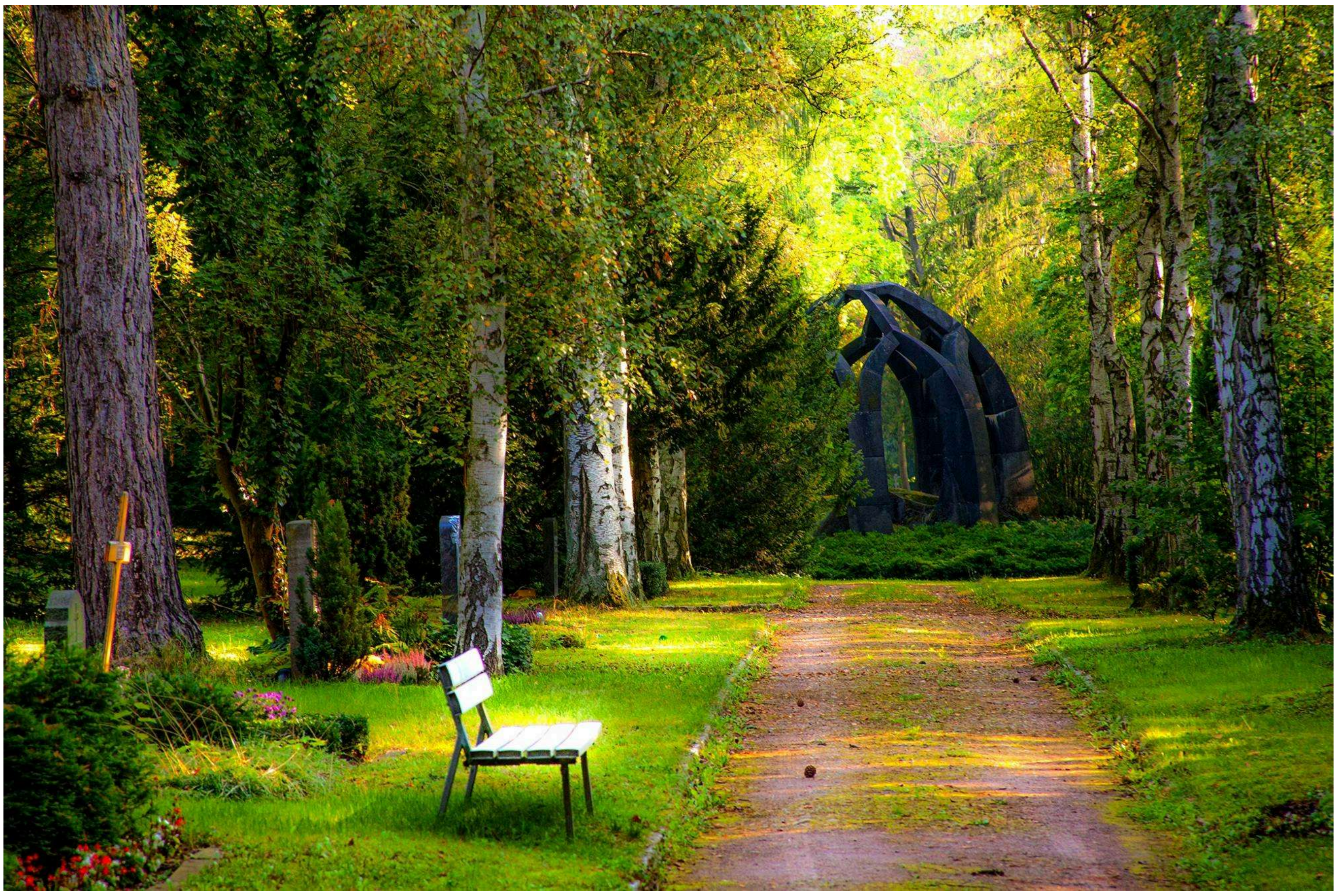
Task -2

```
!pip -q install transformers Pillow matplotlib colorthief openai
```

```
from PIL import Image
from IPython.display import display

# Upload image
uploaded_path = "/content/sample_image.png"  # Upload via Colab sidebar
theme_prompt = "Futuristic neon vibe for interactive UI"

img = Image.open(uploaded_path)
display(img)
```



```
from transformers import BlipProcessor, BlipForConditionalGeneration
import torch

processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base").eval()

inputs = processor(images=img, return_tensors="pt")
with torch.no_grad():
    out = model.generate(**inputs)
caption = processor.decode(out[0], skip_special_tokens=True)

print("Image Caption:", caption)
```

```
preprocessor_config.json: 100%                    287/287 [00:00<00:00, 7.74kB/s]
tokenizer_config.json: 100%                       506/506 [00:00<00:00, 7.13kB/s]
vocab.txt: 100%                                   232k/232k [00:00<00:00, 2.02MB/s]
tokenizer.json: 100%                              711k/711k [00:00<00:00, 8.40MB/s]
special_tokens_map.json: 100%                     125/125 [00:00<00:00, 3.38kB/s]
config.json: 100%                                 4.56k/4.56k [00:00<00:00, 78.1kB/s]
pytorch_model.bin: 100%                           990M/990M [00:12<00:00, 110MB/s]
model.safetensors: 100%                           990M/990M [00:14<00:00, 80.5MB/s]
Image Caption: a bench sits in the middle of a forest
```

```
from colorthief import ColorThief

color_thief = ColorThief(uploaded_path)
palette = color_thief.get_palette(color_count=5)
hex_colors = ['#%02x%02x%02x' % color for color in palette]

print("Dominant Colors:", hex_colors)
```

```
Dominant Colors: ['#323d13', '#d1c151', '#887d32', '#90b71d', '#878295']
```

```
import openai
from openai import OpenAI
import os

api_key = "sk-proj-REzVRJL0cRKtU5NlknjiBpFVYiAHYgv_prBtrsSa5n3F8XyR8-OqmsjK0crfZo-43v2PFQKlqWT3BlbkFJmxRg78oI6RYoUNOnCH8Hm7L2jG8OOhaU-7vV

prompt = f"""
You are a UI theme config generator. Given:
- A user theme description: "{theme_prompt}"
- An image caption: "{caption}"
- A color palette: {hex_colors}

Generate a JSON configuration with the following keys:
- primary_colors
- mood
- font_style
- background
- ui_elements_style (buttons, cards, navbar)
"""
client = OpenAI(api_key=api_key)
response = client.chat.completions.create(
    model="gpt-4",
    messages=[{"role": "user", "content": prompt}],
)

print(response.choices[0].message.content)

import json
config_json = response.choices[0].message.content
print(config_json)
```

```
---------------------------------------------------------------------------
NotFoundError                             Traceback (most recent call last)
/tmp/ipython-input-30-4128620956.py in <cell line: 0>()
     19 """
     20 client = OpenAI(api_key=api_key)
---> 21 response = client.chat.completions.create(
     22     model="gpt-4",
     23     messages=[{"role": "user", "content": prompt}],

  ─────────────────────────────── 3 frames ───────────────────────────────
/usr/local/lib/python3.11/dist-packages/openai/_base_client.py in request(self, cast_to, options, stream, stream_cls)
```