```
!pip install -q transformers pillow torch
```

```
import torch
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image

# Load the model (this may take a minute the first time)
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base").to("cuda")

def generate_prompt(image_path):
    raw_image = Image.open(image_path).convert('RGB')

    # Process image for the model
    inputs = processor(raw_image, return_tensors="pt").to("cuda")

    # Generate caption
    out = model.generate(**inputs)
    caption = processor.decode(out[0], skip_special_tokens=True)

    # Add your "Prompt Optimization" keywords
    optimized_prompt = f"{caption}, highly detailed, cinematic lighting, 8k, masterpiece style"
    negative_prompt = "blurry, distorted, low quality, watermark, extra fingers"

    return optimized_prompt, negative_prompt
```

```
Using a slow image processor as `use_fast` is unset and a slow processor was saved with this model. `use_fast=True` will be th
/usr/local/lib/python3.12/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 i
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%                                   287/287 [00:00<00:00, 26.6kB/s]
tokenizer_config.json: 100%                                      506/506 [00:00<00:00, 55.1kB/s]
vocab.txt:      232k/? [00:00<00:00, 18.0MB/s]
tokenizer.json:       711k/? [00:00<00:00, 37.9MB/s]
special_tokens_map.json: 100%                                    125/125 [00:00<00:00, 14.2kB/s]
config.json:       4.56k/? [00:00<00:00, 340kB/s]
pytorch_model.bin: 100%                                          990M/990M [00:07<00:00, 286MB/s]
```

```
# Replace 'WhatsApp Image 2025-12-24 at 8.09.30 AM' with the name of the file you uploaded
opt_p, neg_p = generate_prompt('WhatsApp Image 2025-12-24 at 8.09.30 AM')

print(f"🔥 Optimized Prompt: {opt_p}")
print(f"🚫 Negative Prompt: {neg_p}")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-2713777489.py in <cell line: 0>()
      1 # Replace 'WhatsApp Image 2025-12-24 at 8.09.30 AM' with the name of the file you uploaded
----> 2 opt_p, neg_p = generate_prompt('WhatsApp Image 2025-12-24 at 8.09.30 AM')
      3 
      4 print(f"🔥 Optimized Prompt: {opt_p}")
      5 print(f"🚫 Negative Prompt: {neg_p}")

                                    1 frames
/usr/local/lib/python3.12/dist-packages/PIL/Image.py in open(fp, mode, formats)
   3511     if is_path(fp):
   3512         filename = os.fspath(fp)
-> 3513         fp = builtins.open(filename, "rb")
   3514         exclusive_fp = True
   3515     else:

FileNotFoundError: [Errno 2] No such file or directory: 'WhatsApp Image 2025-12-24 at 8.09.30 AM'
```

```
# Replace 'your_image.jpg' with the name of the file you uploaded
opt_p, neg_p = generate_prompt('test.jpg')

print(f"🔥 Optimized Prompt: {opt_p}")
print(f"🚫 Negative Prompt: {neg_p}")
```

```
🔥 Optimized Prompt: a cat laying in the grass, highly detailed, cinematic lighting, 8k, masterpiece style
🚫 Negative Prompt: blurry, distorted, low quality, watermark, extra fingers
```

```
def generate_prompt(image_path, style="Realistic"):
    raw_image = Image.open(image_path).convert('RGB')
    inputs = processor(raw_image, return_tensors="pt").to("cuda")
    out = model.generate(**inputs)
    caption = processor.decode(out[0], skip_special_tokens=True)

    # Style Dictionary
    styles = {
        "Realistic": "highly detailed, 8k resolution, cinematic lighting, masterpiece, photorealistic",
        "Anime": "anime style, studio ghibli, vibrant colors, clean lines, high quality digital art",
        "Cyberpunk": "neon lights, futuristic, synthwave aesthetic, dark city background, high contrast"
    }

    selected_style = styles.get(style, styles["Realistic"])

    optimized_prompt = f"{caption}, {selected_style}"
    negative_prompt = "blurry, low quality, distorted, watermark, text, grainy, extra limbs"

    return optimized_prompt, negative_prompt

# To run it now:
opt_p, neg_p = generate_prompt('test.jpg', style="Anime")
print(f"🔥 Optimized Prompt: {opt_p}")
```

```
🔥 Optimized Prompt: a cat laying in the grass, anime style, studio ghibli, vibrant colors, clean lines, high quality digital
```

```
!pip install -q streamlit
!npm install -g localtunnel
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.0/9.0 MB 102.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 135.5 MB/s eta 0:00:00
⠙⠹⠸⠼⠴⠦⠧⠇⠏⠋⠙⠹⠸⠼⠴⠦
added 22 packages in 2s
⠦
⠦3 packages are looking for funding
⠦  run `npm fund` for details
⠦
```

```
import urllib
print("Password/Endpoint IP for localtunnel is:", urllib.request.urlopen('https://ipv4.icanhazip.com').read().decode('utf8').st
```

```
Password/Endpoint IP for localtunnel is: 34.125.101.94
```

```
import subprocess
import sys

# This forces the installation if something is missing
try:
    import transformers
    import torch
except ImportError:
    subprocess.check_call([sys.executable, "-m", "pip", "install", "transformers", "torch", "Pillow"])
```

```
%%writefile app.py
import streamlit as st
import torch
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image

# 1. Page Configuration for a professional look
st.set_page_config(page_title="Vision-Prompt AI", page_icon="🎨", layout="wide")

# 2. Custom CSS for colors and styling
st.markdown("""
    <style>
    .main {
        background-color: #f0f2f6;
    }
    .stButton>button {
        width: 100%;
        border-radius: 20px;
        background-color: #4CAF50;
        color: white;
    }
    </style>
    """, unsafe_allow_html=True) # <--- MAKE SURE THIS SAYS html

# 3. Sidebar for Project Info
with st.sidebar:
    st.title("🧑‍💻 Project Info")
    st.info("**System:** Vision-Language Model")
    st.info("**Model:** BLIP (Salesforce)")
    st.info("**Framework:** Streamlit + PyTorch")
    st.divider()
    st.write("Created by: [Your Name]")

# 4. Main Header
st.title("✨ AI Image to Prompt Generator")
st.write("Transform your images into high-quality descriptive prompts for Stable Diffusion & Midjourney.")

# 5. Layout Columns
col1, col2 = st.columns([1, 1])

with col1:
    st.subheader("📷 Input Image")
    uploaded_file = st.file_uploader("Drag and drop your image here", type=["jpg", "png", "jpeg"])
    if uploaded_file:
        image = Image.open(uploaded_file).convert('RGB')
        st.image(image, use_container_width=True)

with col2:
    st.subheader("📝 Generated Output")
    if uploaded_file:
        if st.button('🚀 Analyze & Generate Prompt'):
            with st.spinner('AI is analyzing features...'):
                # Load Model
                processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
                model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base").to("cuda")

                # Process
                inputs = processor(image, return_tensors="pt").to("cuda")
                out = model.generate(**inputs)
                caption = processor.decode(out[0], skip_special_tokens=True)

                # Display Results in attractive boxes
                st.markdown("### 🔥 Optimized Prompt")
                st.success(f"{caption}, highly detailed, 8k resolution, cinematic lighting, masterpiece, sharp focus, digital

                st.markdown("### 🚫 Negative Prompt")
                st.warning("blurry, low quality, distorted, watermark, text, grainy, extra fingers, out of frame")
    else:
        st.write("Please upload an image to see the generated results.")
```

```
Overwriting app.py
```

```
!streamlit run app.py & npx localtunnel --port 8501
```

```
⠙
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

⠙⠹⠸⠼⠴⠦⠧⠇⠏⠋⠙⠹⠸your url is: https://cute-pots-attack.loca.lt

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://34.125.101.94:8501

  Stopping...
^C
```

```
!streamlit run app.py & npx localtunnel --port 8501
```

```
!streamlit run app.py & npx localtunnel --port 8501

⠙
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

⠹⠸⠼⠴⠦⠧⠇⠏⠋⠙⠹⠸your url is: https://small-eagles-pay.loca.lt

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://34.125.101.94:8501

2025-12-24 06:22:28.994324: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:467] Unable to register cuFFT factory: A
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1766557349.028584    8167 cuda_dnn.cc:8579] Unable to register cuDNN factory: Attempting to register factory for p
E0000 00:00:1766557349.039131    8167 cuda_blas.cc:1407] Unable to register cuBLAS factory: Attempting to register factory for
W0000 00:00:1766557349.064917    8167 computation_placer.cc:177] computation placer already registered. Please check linkage a
W0000 00:00:1766557349.064953    8167 computation_placer.cc:177] computation placer already registered. Please check linkage a
W0000 00:00:1766557349.064961    8167 computation_placer.cc:177] computation placer already registered. Please check linkage a
W0000 00:00:1766557349.064967    8167 computation_placer.cc:177] computation placer already registered. Please check linkage a
2025-12-24 06:22:29.072478: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use av
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler
2025-12-24 06:25:08.801 Please replace `use_container_width` with `width`.

`use_container_width` will be removed after 2025-12-31.

For `use_container_width=True`, use `width='stretch'`. For `use_container_width=False`, use `width='content'`.
2025-12-24 06:25:22.427 Please replace `use_container_width` with `width`.

`use_container_width` will be removed after 2025-12-31.

For `use_container_width=True`, use `width='stretch'`. For `use_container_width=False`, use `width='content'`.
Using a slow image processor as `use_fast` is unset and a slow processor was saved with this model. `use_fast=True` will be th
preprocessor_config.json: 100% 287/287 [00:00<00:00, 2.65MB/s]
tokenizer_config.json: 100% 506/506 [00:00<00:00, 4.98MB/s]
vocab.txt: 232kB [00:00, 18.3MB/s]
tokenizer.json: 711kB [00:00, 57.2MB/s]
special_tokens_map.json: 100% 125/125 [00:00<00:00, 1.46MB/s]
config.json: 4.56kB [00:00, 28.5MB/s]
pytorch_model.bin: 100% 990M/990M [00:07<00:00, 134MB/s]
model.safetensors: 100% 990M/990M [00:08<00:00, 111MB/s]
  Stopping...
^C
```