```
import streamlit as st
from openai import OpenAI
import torch
from diffusers import StableDiffusionPipeline
from PIL import Image
import base64
import io

# App Configuration
st.set_page_config(
    page_title="LogoSynth AI - AI Logo Generator",
    page_icon="✨",
    layout="wide"
)

# CSS
st.markdown("""
<style>
    :root {
        --primary: #6d28d9;
        --primary-light: #8b5cf6;
        --primary-dark: #5b21b6;
        --secondary: #ec4899;
        --accent-1: #06b6d4;
        --accent-2: #f59e0b;
        --accent-3: #10b981;
        --dark: #0f172a;
        --light: #f8fafc;
        --gray: #64748b;
        --bg: #0e1117;
        --card-bg: #1e293b;
        --border: #334155;
    }

    body {
        background-color: var(--bg);
        color: var(--light);
        font-family: 'Poppins', 'Segoe UI', Tahoma, Geneva, Verdana, sans-
serif;
        line-height: 1.6;
    }

    .stApp {
        background-color: var(--bg);
    }

    .header {
```

```
        background: linear-gradient(120deg, var(--primary), var(--secondary),
var(--accent-1));
        background-size: 200% 200%;
        animation: gradient-shift 10s ease infinite;
        color: white;
        padding: 4rem 0;
        text-align: center;
        position: relative;
        overflow: hidden;
        box-shadow: 0 4px 20px rgba(0, 0, 0, 0.2);
        border-bottom-left-radius: 30px;
        border-bottom-right-radius: 30px;
        margin-bottom: 3rem;
    }

    @keyframes gradient-shift {
        0% {background-position: 0% 50%;}
        50% {background-position: 100% 50%;}
        100% {background-position: 0% 50%;}
    }

    .logo-title {
        font-size: 3.5rem;
        font-weight: 800;
        margin-bottom: 1rem;
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);
        letter-spacing: -1px;
    }

    .logo-title span {
        color: var(--accent-2);
        background: white;
        padding: 0 10px;
        border-radius: 8px;
        color: var(--primary);
    }

    .tagline {
        font-size: 1.4rem;
        margin-bottom: 2rem;
        max-width: 800px;
        margin-left: auto;
        margin-right: auto;
    }

    .generator-card {
        background: linear-gradient(135deg, var(--card-bg), #1a2333);
        border-radius: 20px;
```

```
        padding: 2.5rem;
        box-shadow: 0 15px 35px rgba(0, 0, 0, 0.1);
        margin-bottom: 3rem;
        position: relative;
        overflow: hidden;
        border: 1px solid var(--border);
    }

    .generator-title {
        display: flex;
        align-items: center;
        margin-bottom: 2rem;
    }

    .generator-title h2 {
        font-size: 2rem;
        font-weight: 700;
        margin-left: 15px;
        background: linear-gradient(90deg, var(--primary), var(--secondary));
        -webkit-background-clip: text;
        background-clip: text;
        -webkit-text-fill-color: transparent;
        color: transparent;
    }

    .generator-title .icon {
        font-size: 2.5rem;
        color: var(--primary);
    }

    .stTextInput>div>div>input, .stSelectbox>div>div>select,
.stTextArea>div>div>textarea {
        background-color: var(--card-bg);
        color: var(--light);
        border-color: var(--border);
        border-radius: 10px;
        padding: 0.9rem 1.2rem;
        font-size: 1rem;
        transition: all 0.3s;
        box-shadow: 0 2px 5px rgba(0, 0, 0, 0.05);
    }

    .stTextInput>div>div>input:focus, .stSelectbox>div>div>select:focus,
.stTextArea>div>div>textarea:focus {
        outline: none;
        border-color: var(--primary-light);
        box-shadow: 0 0 0 3px rgba(139, 92, 246, 0.2);
    }
```

```
    .stSelectbox>div>div>select {
        appearance: none;
        background-image: url("data:image/svg+xml;charset=UTF-8,%3csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 24 24' fill='none'
stroke='%238b5cf6' stroke-width='2' stroke-linecap='round' stroke-
linejoin='round'%3e%3cpolyline points='6 9 12 15 18
9'%3e%3c/polyline%3e%3c/svg%3e");
        background-repeat: no-repeat;
        background-position: right 1rem center;
        background-size: 1.2em;
        padding-right: 3rem;
    }

    .stButton>button {
        background: linear-gradient(90deg, var(--primary), var(--secondary));
        color: white;
        border: none;
        padding: 1rem 2rem;
        border-radius: 10px;
        font-size: 1.1rem;
        font-weight: 600;
        cursor: pointer;
        transition: all 0.3s;
        box-shadow: 0 4px 15px rgba(109, 40, 217, 0.4);
        width: 100%;
    }

    .stButton>button:hover {
        transform: translateY(-2px);
        box-shadow: 0 6px 18px rgba(109, 40, 217, 0.5);
    }

    .stButton>button:active {
        transform: translateY(0);
    }

    .logo-card {
        background-color: var(--card-bg);
        border-radius: 16px;
        padding: 15px;
        border: 1px solid var(--border);
        margin-bottom: 20px;
        box-shadow: 0 10px 25px rgba(0, 0, 0, 0.08);
        transition: transform 0.3s, box-shadow 0.3s;
    }

    .logo-card:hover {
```

```
        transform: translateY(-10px);
        box-shadow: 0 15px 35px rgba(0, 0, 0, 0.12);
    }

    .spinner {
        width: 70px;
        height: 70px;
        border-radius: 50%;
        background: conic-gradient(transparent 50%, var(--primary));
        -webkit-mask: radial-gradient(farthest-side, transparent calc(100% -
10px), #fff 0);
        mask: radial-gradient(farthest-side, transparent calc(100% - 10px),
#fff 0);
        animation: spin 1s infinite linear;
        margin: 0 auto 1rem;
    }

    @keyframes spin {
        to { transform: rotate(360deg); }
    }

    .footer {
        background-color: var(--dark);
        color: white;
        padding: 2rem 0;
        border-top-left-radius: 30px;
        border-top-right-radius: 30px;
        margin-top: 3rem;
    }

    .footer-bottom {
        text-align: center;
        padding-top: 2rem;
        border-top: 1px solid rgba(255, 255, 255, 0.1);
        font-size: 0.9rem;
        color: rgba(255, 255, 255, 0.5);
    }

    @media (max-width: 768px) {
        .logo-title {
            font-size: 2.5rem;
        }

        .tagline {
            font-size: 1.1rem;
        }
    }
</style>
```

```
""", unsafe_allow_html=True)

# Initialize OpenAI client
client = OpenAI(
    base_url="https://openrouter.ai/api/v1",
    api_key="sk-or-v1-
5255972cb1e5e683cdf1376142b8d657f061205b4ed71c9ca1aa5fb61020c1f3",  # Replace
with your actual API key
)

# Initialize Stable Diffusion pipeline
@st.cache_resource
def load_model():
    return StableDiffusionPipeline.from_pretrained(
        "runwayml/stable-diffusion-v1-5",
        torch_dtype=torch.float16,
        safety_checker=None
    ).to("cuda")

pipe = load_model()

def refine_description(raw_description, brand_name, industry, style,
color_scheme):
    # First validate all inputs exist and are strings
    if not all([isinstance(x, str) and x.strip() for x in [raw_description,
brand_name, industry, style, color_scheme]]):
        return raw_description  # Return original if any parameter is invalid

    try:
        response = client.chat.completions.create(
            model="deepseek/deepseek-chat-v3-0324:free",
            messages=[
                {"role": "system", "content": "Refine this logo
description..."},
                {"role": "user", "content": raw_description}
            ]
        )

        # Simple, robust response extraction
        return getattr(getattr(getattr(response, 'choices', [{}])[0],
'message', {}), 'content', raw_description)

    except Exception:
        return raw_description  # Silent fallback to original

def create_prompt(refined_description, brand_name, industry, style,
color_scheme):
    # Basic validation
```

```
    if not isinstance(refined_description, str) or not
refined_description.strip():
        refined_description = f"{brand_name} {industry} logo"

    try:
        response = client.chat.completions.create(
            model="deepseek/deepseek-chat-v3-0324:free",
            messages=[
                {"role": "system", "content": "Create logo prompt..."},
                {"role": "user", "content": refined_description}
            ]
        )

        # Simple response extraction with fallback
        return getattr(getattr(getattr(response, 'choices', [{}])[0],
'message', {}), 'content',
                       f"Professional {style} logo for {brand_name} with
{color_scheme} colors")

    except Exception:
        return f"Professional {style} logo for {brand_name} with
{color_scheme} colors"

# Function to generate logos
def generate_logos(final_prompt):
    logos = []
    with st.spinner("Generating unique logo concepts..."):
        for i in range(3):  # Generate 3 variants
            image = pipe(
                final_prompt,
                num_inference_steps=50,
                guidance_scale=7.5
            ).images[0]

            buffered = io.BytesIO()
            image.save(buffered, format="PNG")
            img_bytes = base64.b64encode(buffered.getvalue()).decode()
            logos.append(img_bytes)
    return logos

# App Header
st.markdown("""
<div class="header">
    <div class="container">
        <div class="header-content">
            <div class="logo-title">Logo<span>Synth AI</span></div>
            <div class="tagline">Where Imagination Meets Algorithm</div>
        </div>
```

```
    </div>
</div>
""", unsafe_allow_html=True)

# Main content container
with st.container():
    # Generator card
    st.markdown("""
    <div class="generator-card">
        <div class="generator-title">
            <div class="icon">✨</div>
            <h2>Create Your Dream Logo</h2>
        </div>
    """, unsafe_allow_html=True)

    # Input form
    with st.form("logo_form"):
        col1, col2 = st.columns(2)
        with col1:
            brand_name = st.text_input("Brand Name", placeholder="Your company
name")
        with col2:
            industry = st.selectbox("Industry", [
                "Technology", "Food & Beverage", "Health & Wellness",
                "Finance", "Education", "Fashion", "Creative Arts",
                "Sports", "Real Estate", "Other"
            ])

        # Additional style options
        style = st.selectbox("Style", [
            "Modern & Minimalist", "Vintage & Retro", "Playful & Colorful",
            "Geometric", "Handcrafted", "Luxury", "Futuristic",
            "Organic", "Abstract", "Corporate"
        ])

        # Color scheme selection
        color_scheme = st.selectbox("Color Scheme", [
            "Cool Blues", "Warm Reds & Oranges", "Nature Greens",
            "Vibrant Rainbow", "Monochromatic", "Pastel Tones",
            "Neon Brights", "Earth Tones", "Metallic Shades",
            "Dark & Moody"
        ])

        # Description field
        description = st.text_area(
            "Describe your ideal logo",
            placeholder="e.g., I want a fox mascot logo with geometric shapes
that looks modern but friendly...",
```

```
            height=100
        )

        generate_col, regenerate_col = st.columns(2)
        with generate_col:
            generate_clicked = st.form_submit_button("✨ Generate Logos",
use_container_width=True)
        with regenerate_col:
            regenerate_clicked = st.form_submit_button("🔄 Regenerate",
use_container_width=True,
                                                       disabled=not
st.session_state.get('logos'))

    st.markdown("</div>", unsafe_allow_html=True)  # Close generator-card

# In the generation logic section, update the code to properly handle
color_scheme:

# In your generation logic:
if generate_clicked or regenerate_clicked:
    if not brand_name or not description:
        st.warning("Please fill in both brand name and description")
    else:
        with st.spinner("Creating your logos..."):
            # Get all inputs with defaults
            brand = str(brand_name) if brand_name else "Brand"
            desc = str(description) if description else "Logo"
            industry_type = str(industry) if industry else "Business"
            logo_style = str(style) if style else "Modern"
            colors = str(color_scheme) if color_scheme else "vibrant colors"

            # Safe generation process
            try:
                refined = refine_description(desc, brand, industry_type,
logo_style, colors)
                prompt = create_prompt(refined, brand, industry_type,
logo_style, colors)

                if prompt:  # Only generate if we got a valid prompt
                    st.session_state.logos = generate_logos(prompt)
                    st.session_state.final_prompt = prompt
                    st.session_state.refined_description = refined
                else:
                    st.warning("Couldn't create prompt - using default")
                    st.session_state.logos = generate_logos(
                        f"Professional {logo_style} logo for {brand} with
{colors}"
                    )
```

```
            except Exception as e:
                st.error("Logo generation encountered an issue")
                st.session_state.logos = generate_logos(
                    f"Simple {logo_style} logo for {brand}"
                )

# Display results
if 'logos' in st.session_state:
    st.subheader("Your Generated Logos")

    cols = st.columns(3)
    for idx, img_bytes in enumerate(st.session_state.logos):
        with cols[idx]:
            st.markdown(f'<div class="logo-card">', unsafe_allow_html=True)
            st.image(f"{img_bytes}",
use_container_width=True)
            st.download_button(
                label="Download Logo",
                data=base64.b64decode(img_bytes),
                file_name=f"{brand_name.replace(' ', '_')}_logo_{idx+1}.png",
                mime="image/png",
                use_container_width=True
            )
            st.markdown(f'</div>', unsafe_allow_html=True)

# Simplified Footer
st.markdown("""
<div class="footer">
    <div class="container">
        <div class="footer-bottom">
            <p>Made by Parth Chauhan, Aryan Makwana, Sumit Solanki</p>
        </div>
    </div>
</div>
""", unsafe_allow_html=True)
```