L3: Image generation app 🎨

Load your HF API key and relevant Python libraries

In [ ]:

```
import os
import io
import IPython.display
from PIL import Image
import base64
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
hf_api_key = os.environ['HF_API_KEY']
```

In [ ]:

```
# Helper function
import requests, json

#Text-to-image endpoint
def get_completion(inputs, parameters=None, ENDPOINT_URL=os.environ['HF_API_TTI_BA
    headers = {
      "Authorization": f"Bearer {hf_api_key}",
      "Content-Type": "application/json"
    }
    data = { "inputs": inputs }
    if parameters is not None:
        data.update({"parameters": parameters})
    response = requests.request("POST",
                                ENDPOINT_URL,
                                headers=headers,
                                data=json.dumps(data))
    return json.loads(response.content.decode("utf-8"))
```

Building an image generation app

Here we are going to run `runwayml/stable-diffusion-v1-5` using the 🧨 `diffusers` library.

How about running it locally?

The code would look very similar if you were running it locally instead of from an API.

```
from diffusers import DiffusionPipeline

pipeline = DiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-
5")

def get_completion(prompt):
    return pipeline(prompt).images[0]
```

In [ ]:

```
prompt = "a dog in a park"

result = get_completion(prompt)
IPython.display.HTML(f'<img src="{result}" />')
```

Generating with `gr.Interface()`

In [ ]:

```
import gradio as gr

#A helper function to convert the PIL image to base64
#so you can send it to the API
def base64_to_pil(img_base64):
    base64_decoded = base64.b64decode(img_base64)
    byte_stream = io.BytesIO(base64_decoded)
    pil_image = Image.open(byte_stream)
    return pil_image

def generate(prompt):
    output = get_completion(prompt)
    result_image = base64_to_pil(output)
    return result_image

gr.close_all()
demo = gr.Interface(fn=generate,
                    inputs=[gr.Textbox(label="Your prompt")],
                    outputs=[gr.Image(label="Result")],
                    title="Image Generation with Stable Diffusion",
                    description="Generate any image with Stable Diffusion",
                    allow_flagging="never",
                    examples=["the spirit of a tamagotchi wandering in the city of

demo.launch(share=True, server_port=int(os.environ['PORT1']))
```

In [ ]:

```
demo.close()
```

Building a more advanced interface

In [ ]:

```
import gradio as gr

#A helper function to convert the PIL image to base64
# so you can send it to the API
def base64_to_pil(img_base64):
    base64_decoded = base64.b64decode(img_base64)
    byte_stream = io.BytesIO(base64_decoded)
    pil_image = Image.open(byte_stream)
    return pil_image

def generate(prompt, negative_prompt, steps, guidance, width, height):
    params = {
        "negative_prompt": negative_prompt,
        "num_inference_steps": steps,
        "guidance_scale": guidance,
        "width": width,
        "height": height
    }

    output = get_completion(prompt, params)
    pil_image = base64_to_pil(output)
    return pil_image

gr.close_all()
demo = gr.Interface(fn=generate,
                    inputs=[
                        gr.Textbox(label="Your prompt"),
                        gr.Textbox(label="Negative prompt"),
                        gr.Slider(label="Inference Steps", minimum=1, maximum=100,
                                  info="In how many steps will the denoiser denoise
                        gr.Slider(label="Guidance Scale", minimum=1, maximum=20, v
                                  info="Controls how much the text prompt influenc
                        gr.Slider(label="Width", minimum=64, maximum=512, step=64,
                        gr.Slider(label="Height", minimum=64, maximum=512, step=64
                    ],
                    outputs=[gr.Image(label="Result")],
                    title="Image Generation with Stable Diffusion",
                    description="Generate any image with Stable Diffusion",
                    allow_flagging="never"
                    )

demo.launch(share=True, server_port=int(os.environ['PORT2']))
```

In [ ]:

```
demo.close()
```

`gr.Blocks()` to the rescue!

In [ ]:

```
with gr.Blocks() as demo:
    gr.Markdown("# Image Generation with Stable Diffusion")
    prompt = gr.Textbox(label="Your prompt")
    with gr.Row():
        with gr.Column():
            negative_prompt = gr.Textbox(label="Negative prompt")
            steps = gr.Slider(label="Inference Steps", minimum=1, maximum=100, val
                      info="In many steps will the denoiser denoise the image?")
            guidance = gr.Slider(label="Guidance Scale", minimum=1, maximum=20, va
                      info="Controls how much the text prompt influences the resul
            width = gr.Slider(label="Width", minimum=64, maximum=512, step=64, val
            height = gr.Slider(label="Height", minimum=64, maximum=512, step=64, v
            btn = gr.Button("Submit")
        with gr.Column():
            output = gr.Image(label="Result")

    btn.click(fn=generate, inputs=[prompt,negative_prompt,steps,guidance,width,hei
gr.close_all()
demo.launch(share=True, server_port=int(os.environ['PORT3']))
```

In [ ]:

```
with gr.Blocks() as demo:
    gr.Markdown("# Image Generation with Stable Diffusion")
    with gr.Row():
        with gr.Column(scale=4):
            prompt = gr.Textbox(label="Your prompt") #Give prompt some real estate
        with gr.Column(scale=1, min_width=50):
            btn = gr.Button("Submit") #Submit button side by side!
    with gr.Accordion("Advanced options", open=False): #Let's hide the advanced op
            negative_prompt = gr.Textbox(label="Negative prompt")
            with gr.Row():
                with gr.Column():
                    steps = gr.Slider(label="Inference Steps", minimum=1, maximum=
                      info="In many steps will the denoiser denoise the image?")
                    guidance = gr.Slider(label="Guidance Scale", minimum=1, maximu
                      info="Controls how much the text prompt influences the resul
                with gr.Column():
                    width = gr.Slider(label="Width", minimum=64, maximum=512, step
                    height = gr.Slider(label="Height", minimum=64, maximum=512, st
    output = gr.Image(label="Result") #Move the output up too

    btn.click(fn=generate, inputs=[prompt,negative_prompt,steps,guidance,width,hei

gr.close_all()
demo.launch(share=True, server_port=int(os.environ['PORT4']))
```

In [ ]:

```
gr.close_all()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```