1. Image-Based Short Video Generation Using AI

Solution:

```
!pip install -q diffusers transformers accelerate torch torchvision safetensors
imageio imageio[ffmpeg]


from diffusers import StableVideoDiffusionPipeline
import torch
from PIL import Image
import numpy as np
import imageio
from google.colab import files  # for uploading files


print("Please upload an image file (jpg/png).")
uploaded = files.upload()
image_path = list(uploaded.keys())[0]  # get uploaded file name
image = Image.open(image_path).convert("RGB")
image = image.resize((512, 512))  # resize for model


model_id = "stabilityai/stable-video-diffusion-img2vid-xt"
pipe = StableVideoDiffusionPipeline.from_pretrained(model_id,
torch_dtype=torch.float16, variant="fp16")
pipe = pipe.to("cuda")  # use GPU

result = pipe(image, num_frames=6)  # 6 frames for low memory

frames = result.frames[0]

video_path = "/content/generated_video.mp4"
imageio.mimsave(video_path, [np.array(f) for f in frames], fps=8)

print("Video generated and saved at:", video_path)
from IPython.display import Video, display
```

```
display(Video(video_path, embed=True, width=560))
```

2. AI-Based Short Video Generation from Image and Text Inputs Using Diffusion Models

```
!pip install -q diffusers transformers accelerate torch torchvision safetensors
imageio imageio[ffmpeg]

from diffusers import StableVideoDiffusionPipeline, DiffusionPipeline
import torch, imageio, numpy as np
from PIL import Image
from google.colab import files
from IPython.display import Video, display

print("Please upload an image file (JPG or PNG):")
uploaded = files.upload()
image_path = list(uploaded.keys())[0]
print(f" Image uploaded successfully: {image_path}")

image = Image.open(image_path).convert("RGB")
image = image.resize((512, 512))


print("\n Generating video from uploaded image...")

img_model = "stabilityai/stable-video-diffusion-img2vid-xt"
img_pipe = StableVideoDiffusionPipeline.from_pretrained(
   img_model, torch_dtype=torch.float16, variant="fp16"
).to("cuda")

img_result = img_pipe(image, num_frames=6)  # generate 6 frames
img_frames = img_result.frames[0]

image_video_path = "/content/image_video.mp4"
imageio.mimsave(image_video_path, [np.array(f) for f in img_frames], fps=8)
```

```
print("\n Generating video from text prompt...")

text_model = "damo-vilab/text-to-video-ms-1.7b"
text_pipe = DiffusionPipeline.from_pretrained(
    text_model, torch_dtype=torch.float16, variant="fp16"
).to("cuda")

prompt = "A mountain landscape with clouds moving slowly."
text_result = text_pipe(prompt, num_frames=8)
text_frames = text_result.frames[0]

text_video_path = "/content/text_video.mp4"
imageio.mimsave(text_video_path, [np.array(f) for f in text_frames], fps=8)

print("\n Image-based Video:")
display(Video(image_video_path, embed=True, width=500))

print("\n Text-based Video:")
display(Video(text_video_path, embed=True, width=500))

print("\n Both videos have been generated successfully.")
```

3. Speach Generative

```
!pip install gTTS pydub -q
!apt-get install -y -qq ffmpeg

from gtts import gTTS
from IPython.display import Audio

text = """Welcome to Generative AI and Prompt engineering Tutorial"""
print("Input text:\n", text)

tts = gTTS(text, lang="en")
tts.save("story_voice.mp3")
```

```
Audio("story_voice.mp3")
```

4. **Automatic Background Music Integration for Speech Audio using Python**

```
!pip install gTTS pydub -q
!apt-get install -y -qq ffmpeg

from gtts import gTTS
text = "Welcome to MSRIT Department of MCA."
tts = gTTS(text, lang="en")
tts.save("voice.mp3")

from pydub.generators import Sine
from pydub import AudioSegment

pad = Sine(220).to_audio_segment(duration=15000).apply_gain(-10)  # 15s
ambient tone
voice =
AudioSegment.from_file("voice.mp3").set_frame_rate(44100).set_channels(2)

combined = pad.overlay(voice, position=1000)  # start after 1s
combined.export("simple_music.wav", format="wav")

from IPython.display import Audio
Audio("simple_music.wav")
```

5. **Design an AI system that can describe artworks or museum exhibits aloud when an image of an artifact is uploaded. The system should automatically analyze the image, generate a descriptive caption, and convert it into speech narration.**

```
!pip install -q gTTS transformers torch torchvision pillow –U

from google.colab import files
from PIL import Image
import torch
```

```
from transformers import BlipProcessor, BlipForConditionalGeneration

print("Please upload an image (jpg/png)")
uploaded = files.upload()
image_path = list(uploaded.keys())[0]
image = Image.open(image_path).convert("RGB")

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

inputs = processor(image, return_tensors="pt").to("cuda")
out = model.generate(**inputs, max_length=30)
caption = processor.decode(out[0], skip_special_tokens=True)

print("\n Generated Description:")
print(caption)

from gtts import gTTS
tts = gTTS(caption, lang="en")
tts.save("image_speech.mp3")

from IPython.display import Audio, display
print("\n Image analyzed and converted to speech successfully!")
display(Audio("image_speech.mp3"))
```