**PART 2**

**Step 1: Initial Research via Internet Search**

**Key Concepts from the Video:**

1. **MoviePy for Video Processing:** The video demonstrates how to use MoviePy to manipulate images, create video clips, and combine them with audio files. It walks through how to:
   * Load image files using ImageClip.
   * Set the duration for each image in the video.
   * Add audio tracks, like voiceovers, to video clips.
   * Concatenate multiple clips to create a final video output.
2. **gTTS for Voiceover Generation:** The video explains how to use the Google Text-to-Speech (gTTS) library to generate audio files from text. It shows how to:
   * Convert text into speech.
   * Save the speech as an audio file in MP3 format.
   * Integrate these audio files as the soundtrack for video clips.
3. **Combining Image and Audio into a Video:** One of the central parts of the video is how to combine image clips with their corresponding voiceover audio, then merge these clips into a single video. The tutorial illustrates the use of MoviePy's concatenate\_videoclips function to achieve this.

**How I Used the Video to Generate My Code:**

1. **Image-to-Video Conversion with Audio:** I applied the knowledge from the video to convert a series of images into video clips using the MoviePy library. I followed the video’s method to:
   * Load image files using the ImageClip class from MoviePy.
   * Set a specific duration for each image clip.

Here’s a snippet of how I used MoviePy based on the video:

img\_clip = mp.ImageClip(image\_path).set\_duration(duration)

1. **Generating Voiceovers with gTTS:** I also used the tutorial’s method for generating voiceovers using the gTTS library. The video explained how to convert text into speech, save it as an MP3 file, and then associate it with an image clip in the video. This helped me implement the following part of the code:

tts = gtts.gTTS(text, lang='en')

tts.save(f"./data/voiceover\_{i}.mp3")

img\_clip = img\_clip.set\_audio(voiceover\_clips[i])

1. **Merging Image and Audio into Final Video:** Finally, the video provided insights on how to combine all the individual clips (images with voiceovers) into a final video file. I followed its guidance to use the concatenate\_videoclips function to merge everything:

final\_clip = mp.concatenate\_videoclips(image\_clips)

final\_clip.write\_videofile(output\_file\_path, fps=24)

ChatGPT explained how to attach an audio file to a video clip using the set\_audio function in MoviePy. It provided me with the following code:

python

Copy code

img\_clip = mp.ImageClip(image\_path).set\_duration(duration)

img\_clip = img\_clip.set\_audio(voiceover\_clips[i])

**Step 2: Asking ChatGPT for Additional Help**

ChatGPT suggested using the Pillow library to draw text on images. It provided me with a detailed explanation and code for creating a captioned image, including handling font size and positioning the text properly on the image. Here’s the code ChatGPT generated:

python

Copy code

from PIL import Image, ImageDraw, ImageFont

def add\_caption\_to\_image(image\_path, caption\_text, font\_size=30):

img = Image.open(image\_path)

draw = ImageDraw.Draw(img)

font = ImageFont.truetype("arial.ttf", font\_size)

text\_width, text\_height = draw.textsize(caption\_text, font=font)

text\_x = (img.width - text\_width) // 2

text\_y = img.height - text\_height - 20

draw.text((text\_x, text\_y), caption\_text, font=font, fill="white")

img.save('captioned\_image.png')

return 'captioned\_image.png'

**Final Code Implementation:**

Here’s the final Python code I used to add captions to my video:

import moviepy.editor as mp

import gtts

from PIL import Image, ImageDraw, ImageFont

def add\_caption\_to\_image(image\_path, caption\_text, font\_size=24):

img = Image.open(image\_path)

draw = ImageDraw.Draw(img)

font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans-Bold.ttf", font\_size)

bbox = draw.textbbox((0, 0), caption\_text, font=font)

text\_width = bbox[2] - bbox[0]

text\_height = bbox[3] - bbox[1]

text\_x = (img.width - text\_width) // 2

text\_y = img.height - text\_height - 10

draw.text((text\_x, text\_y), caption\_text, fill="white", font=font)

img\_with\_caption\_path = './captioned\_image.png'

img.save(img\_with\_caption\_path)

return img\_with\_caption\_path

images = [

("./data/image1.png", 7),

("./data/image2.png", 6),

("./data/image3.png", 8),

("./data/image4.png", 5),

("./data/image5.png", 7)

]

voiceover\_texts = [

"A bright yellow taxi speeds through the busy streets of New York City.",

"A barista pours frothy milk into a cup of coffee at a cozy cafe.",

"A street artist paints a vibrant mural on a city wall under the afternoon sun.",

"People walk across a crowded crosswalk in front of towering skyscrapers.",

"A food truck serves tacos to a long line of customers in a city park."

]

# Generate voiceover using gTTS

voiceover\_clips = []

for i, text in enumerate(voiceover\_texts):

tts = gtts.gTTS(text, lang='en')

tts.save(f"./data/voiceover\_{i}.mp3")

voiceover\_clips.append(mp.AudioFileClip(f"./data/voiceover\_{i}.mp3"))

# Create a video clip for each image

image\_clips = []

for i, (image\_path, duration) in enumerate(images):

captioned\_image\_path = add\_caption\_to\_image(image\_path, captions[i], font\_size=40)

img\_clip = mp.ImageClip(captioned\_image\_path).set\_duration(duration)

img\_clip = img\_clip.set\_audio(voiceover\_clips[i])

image\_clips.append(img\_clip)

final\_clip = mp.concatenate\_videoclips(image\_clips)

final\_clip.write\_videofile("./final\_video\_with\_captions.mp4", fps=24)