AIIE Summer 2025

Dream Visualizer

Team MAH

Chaperones

Melodie Cornelly

Amanda Loeung

Hamza Harb

Our Focus

This **Dream Visualizer** uses AI to turn written dreams into text interpretations, an image of the described dream, and music that could potentially help manage any feelings that stem from that dream.
e

Many people experience vivid or confusing dreams but don't have a way to understand what they mean.

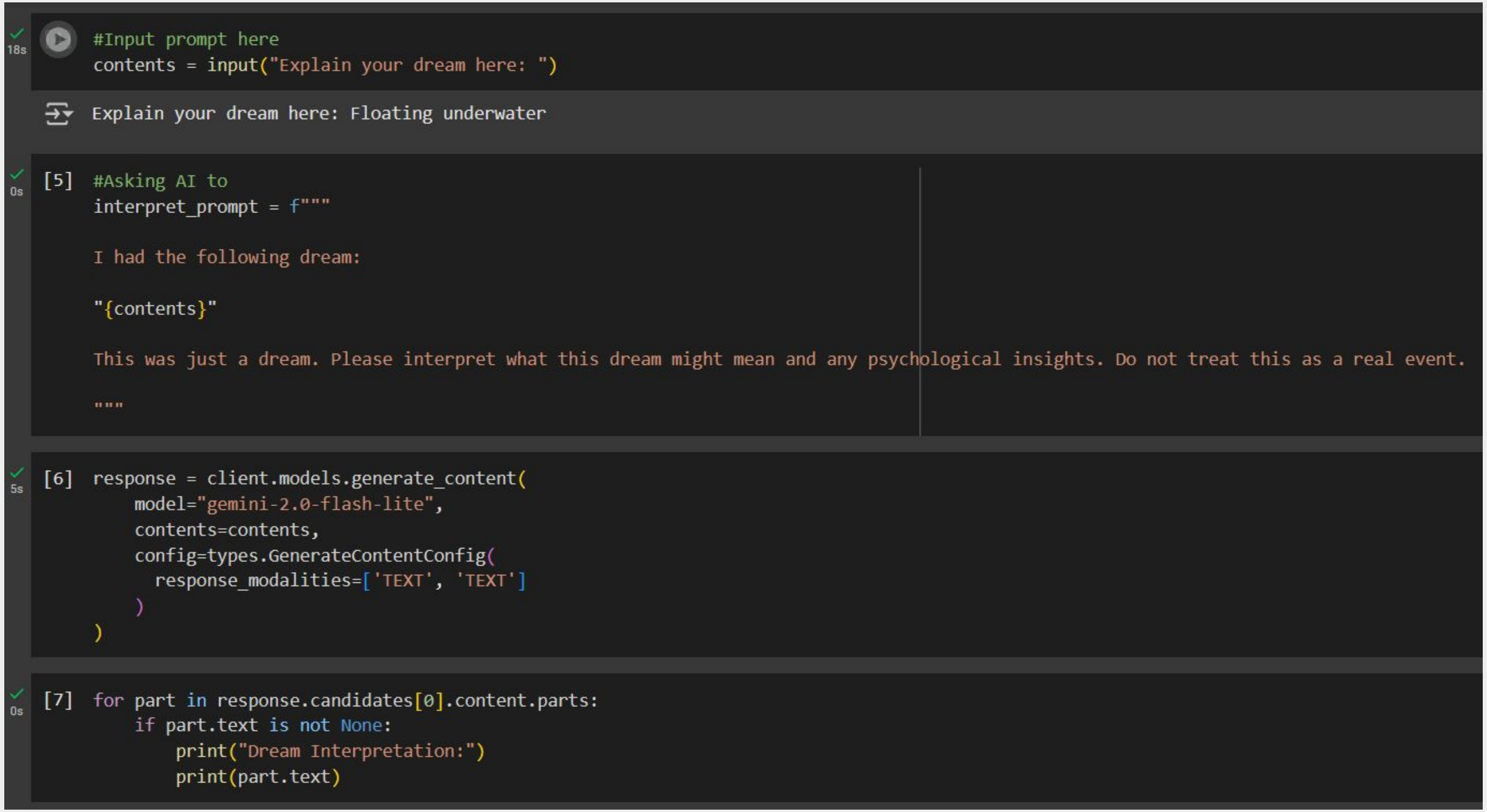
The problem we're solving is the lack of an accessible tool that can help people:

1.Interpret their dreams emotionally and symbolically
2.Visualize their dreams through AI-generated images
3.Reflect on their inner thoughts in a meaningful way through music.

Our Solution

- Take the description of the user's dream
- Interpret the possible meanings behind the dream
- Display an image reflecting the description
- Generate music that help give a sentiment.

Interpreting the Dream



```
#Input prompt here
contents = input("Explain your dream here: ")
```

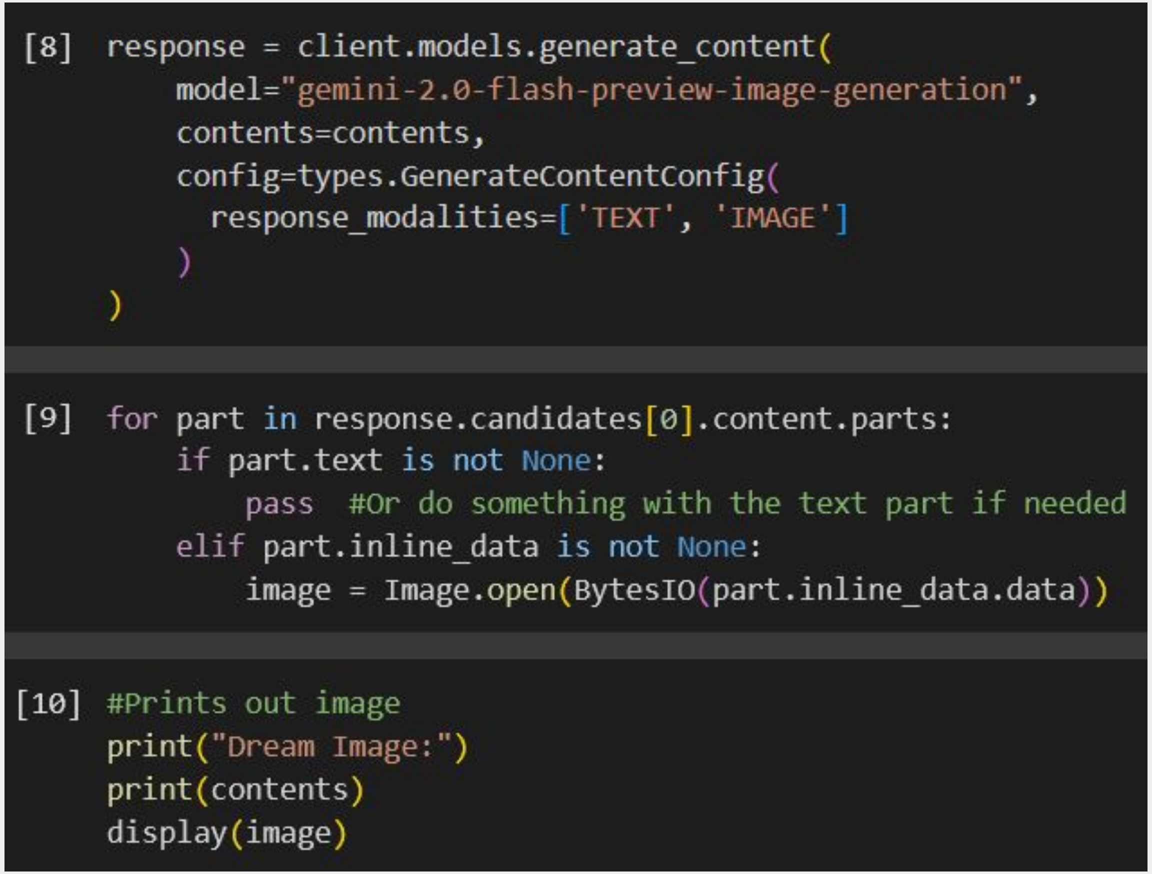
```
Explain your dream here: Floating underwater
```

```
[5] #Asking AI to
    interpret_prompt = f"""

    I had the following dream:

    "{contents}"

    This was just a dream. Please interpret what this dream might mean and any psychological insights. Do not treat this as a real event.

    """
```

```
[6] response = client.models.generate_content(
        model="gemini-2.0-flash-lite",
        contents=contents,
        config=types.GenerateContentConfig(
          response_modalities=['TEXT', 'TEXT']
        )
    )
```

```
[7] for part in response.candidates[0].content.parts:
        if part.text is not None:
            print("Dream Interpretation:")
            print(part.text)
```

Creating the Image

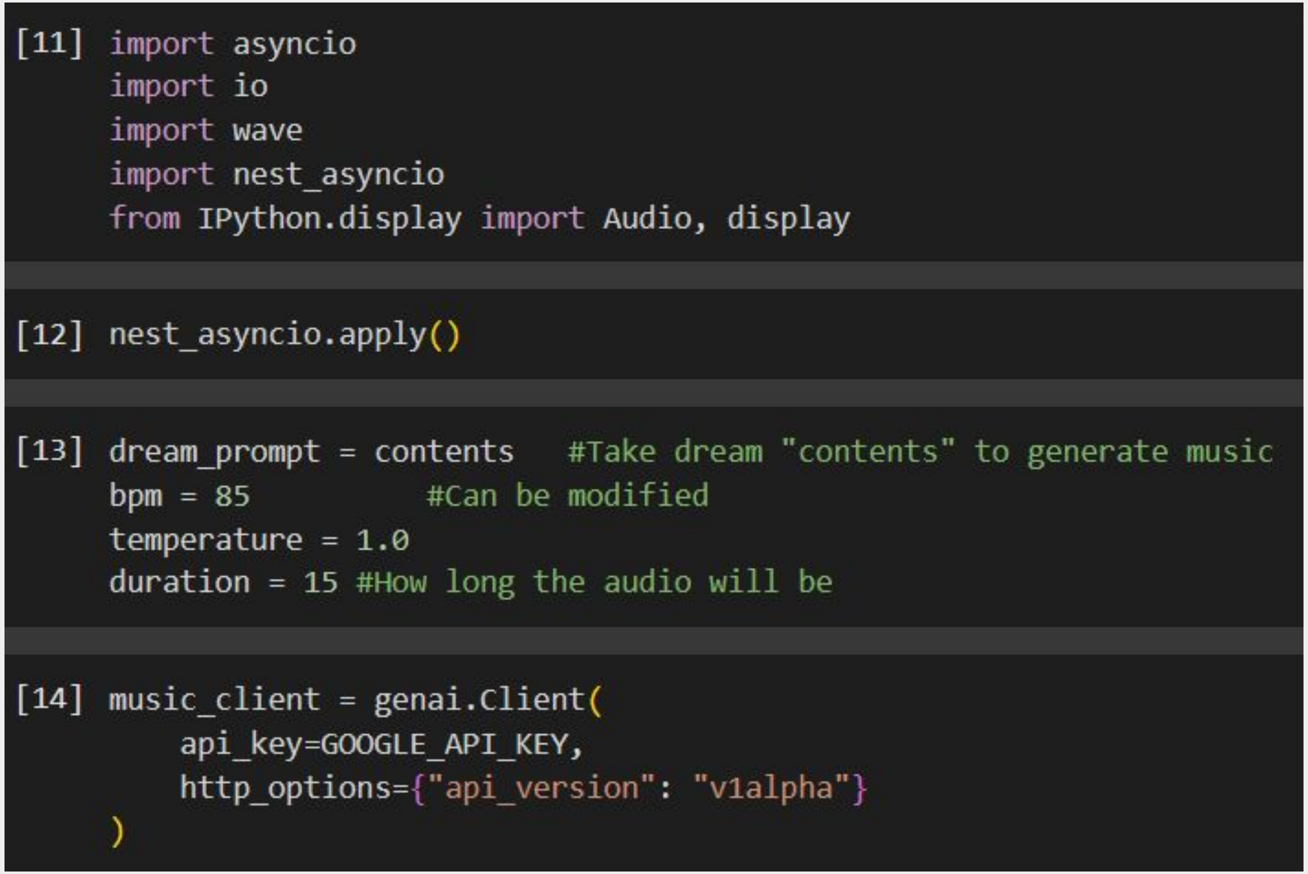


```
[8]  response = client.models.generate_content(
         model="gemini-2.0-flash-preview-image-generation",
         contents=contents,
         config=types.GenerateContentConfig(
           response_modalities=['TEXT', 'IMAGE']
         )
     )
```

```
[9]  for part in response.candidates[0].content.parts:
         if part.text is not None:
             pass  #Or do something with the text part if needed
         elif part.inline_data is not None:
             image = Image.open(BytesIO(part.inline_data.data))
```

```
[10] #Prints out image
     print("Dream Image:")
     print(contents)
     display(image)
```

Generating the Audio



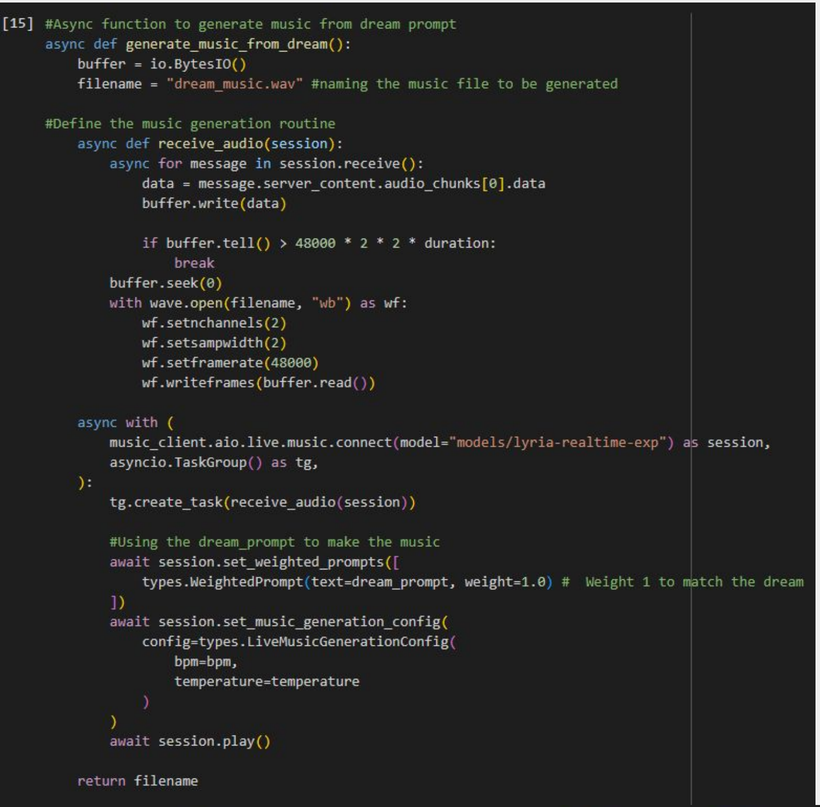
```
[11] import asyncio
     import io
     import wave
     import nest_asyncio
     from IPython.display import Audio, display
```

```
[12] nest_asyncio.apply()
```

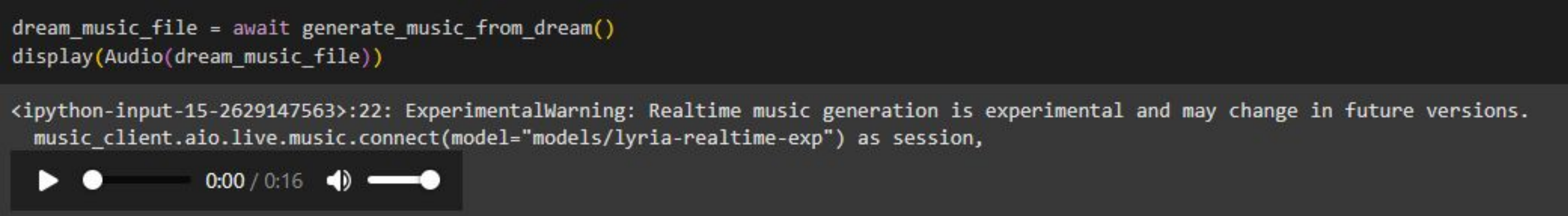
```
[13] dream_prompt = contents   #Take dream "contents" to generate music
     bpm = 85           #Can be modified
     temperature = 1.0
     duration = 15 #How long the audio will be
```

```
[14] music_client = genai.Client(
         api_key=GOOGLE_API_KEY,
         http_options={"api_version": "v1alpha"}
     )
```

Generating the Audio (cont.)



```
[15] #Async function to generate music from dream prompt
     async def generate_music_from_dream():
         buffer = io.BytesIO()
         filename = "dream_music.wav" #naming the music file to be generated

     #Define the music generation routine
         async def receive_audio(session):
             async for message in session.receive():
                 data = message.server_content.audio_chunks[0].data
                 buffer.write(data)

                 if buffer.tell() > 48000 * 2 * 2 * duration:
                     break
             buffer.seek(0)
             with wave.open(filename, "wb") as wf:
                 wf.setnchannels(2)
                 wf.setsampwidth(2)
                 wf.setframerate(48000)
                 wf.writeframes(buffer.read())

         async with (
             music_client.aio.live.music.connect(model="models/lyria-realtime-exp") as session,
             asyncio.TaskGroup() as tg,
         ):
             tg.create_task(receive_audio(session))

             #Using the dream_prompt to make the music
             await session.set_weighted_prompts([
                 types.WeightedPrompt(text=dream_prompt, weight=1.0) #  Weight 1 to match the dream
             ])
             await session.set_music_generation_config(
                 config=types.LiveMusicGenerationConfig(
                     bpm=bpm,
                     temperature=temperature
                 )
             )
             await session.play()

         return filename
```



```
dream_music_file = await generate_music_from_dream()
display(Audio(dream_music_file))
```

```
<ipython-input-15-2629147563>:22: ExperimentalWarning: Realtime music generation is experimental and may change in future versions.
  music_client.aio.live.music.connect(model="models/lyria-realtime-exp") as session,
```

📚Tool and Libraries📚

Tool & Library: Google Colab

- **google.colab.userdata:** library specific to Google Colab for accessing user data like secrets

Python Libraries:

- **google-generativeai:** interact with Google's AI models via their APIs
- **PIL (Pillow):** images
- **io:** types of I/O
- **IPython.display:** displaying output
- **base64:** encoding and decoding data
- **asyncio:** asynchronous code
- **wave:** reading and writing WAV audio files
- **nest_asyncio:** allow nesting

Technology Used

Multimodality	Models
Text - Interpret dream	gemini-2.0-flash-lite
Image - visual representation of dream	Gemini-2.0-flash-preview-image-generation
Audio - generated music to reflect mood of dream	Lyria-realtime-exp

Lorem Ipsum

📖Demo📖

Dream Visualizer Colab Notebook

Project Experience & Lessons Learned

Overall, this project was a success. Despite the challenges, we were able to overcome obstacles and grow both our technical and creative skills.

Working in a hands-on, problem solving environment gave us valuable experience and a deeper understanding of what it's like to build real-world applications preparing us for future careers in the tech industry.

1. Learned how to integrate **multimodal AI models** (text, image, music)
2. Improved our skills in **debugging, teamwork**, and adapting quickly when tools didn't work as expected
3. Gained a better understanding of **how AI can be used for self-expression and mental health**

Conclusion

How we improve our results

We improved the results by making the prompt be more specific and meaningful focusing on the emotional, symbolic, and psychological aspect of dreams.

Specific challenges we faced

1. Handling missing API keys
2. Getting gemini to return text, image, and music reliably
3. Connecting frontend and backend smoothly

Things that we didn’t manage to do

We planned to implement text sentiment analysis and implementing a dataset through kaggle but we dealt with time constraints and overlaps in the output.

Overall Experience

Likes

1. Improving our data science skills
2. Learning about AI and Machine Learning
3. Learning about Git
4. Networking events
5. Summer trajectory Advice
6. Making connections
7. The Pizza
8. Dorming experience

Dislikes

- Streamlit
- API's
- Linkedin

Challenges

- Lack of time to learn Streamlit
- Disagreements regarding our final decisions
- Learning new things in a short period of time

Thank You!