Smart SDLC – AI Enhanced Software Development Lifecycle

<u>Project Title:</u>

A Generative AI application called "Smart SDLC – AI Enhanced Software Development Lifecycle" i want to integrate ibm-granite/granite-3.3-2b-instruct model from hugging class, I want to deploy it with EastAPI so give me html and css / Gradio in google collab.With the functionality are Phase-wise AI Assistance,Generative Capabilities, Project Intelligence & Insights.

- **<u>Project Architecture for Smart SDLC System :</u>**

The Smart SDLC architecture provides a seamless integration between a user-facing interface, AI-powered backend processing, and deployment mechanisms. It focuses on improving efficiency across the entire software development lifecycle using IBM's generative AI capabilities.

<u>Pre-requisites:</u>

Python – Core language used for scripting and application logic.

FastAPI – Backend framework used for building efficient APIs.

Streamlit / Gradio – Lightweight frontend UI libraries for interaction.

IBM Watsonx AI & Granite Models – Core generative model used for intelligent SDLC assistance.

- **<u>Model Selection and Architecture:</u>**

Smart SDLC is a Generative AI application powered by the IBM Granite-3.3-2b-instruct model, developed to optimize the entire software development lifecycle (SDLC). The model is accessed through Hugging Face's Transformers library and utilized with PyTorch, allowing for high-performance text generation. Depending on system capabilities, it dynamically supports both CPU and GPU (with FP16 precision for GPUs).

Research and select - the appropriate generative AI model suitable for SDLC use cases.

Define the architecture - of the application including frontend, backend, and model integration layers.

Set up the development environment - with all necessary dependencies, GPU support, and libraries (e.g., Hugging Face Transformers, Torch, Pillow).

- **Core Functionalities Development:**

Develop the core functionalities - that allow the user to interact by selecting SDLC phases and entering task prompts.

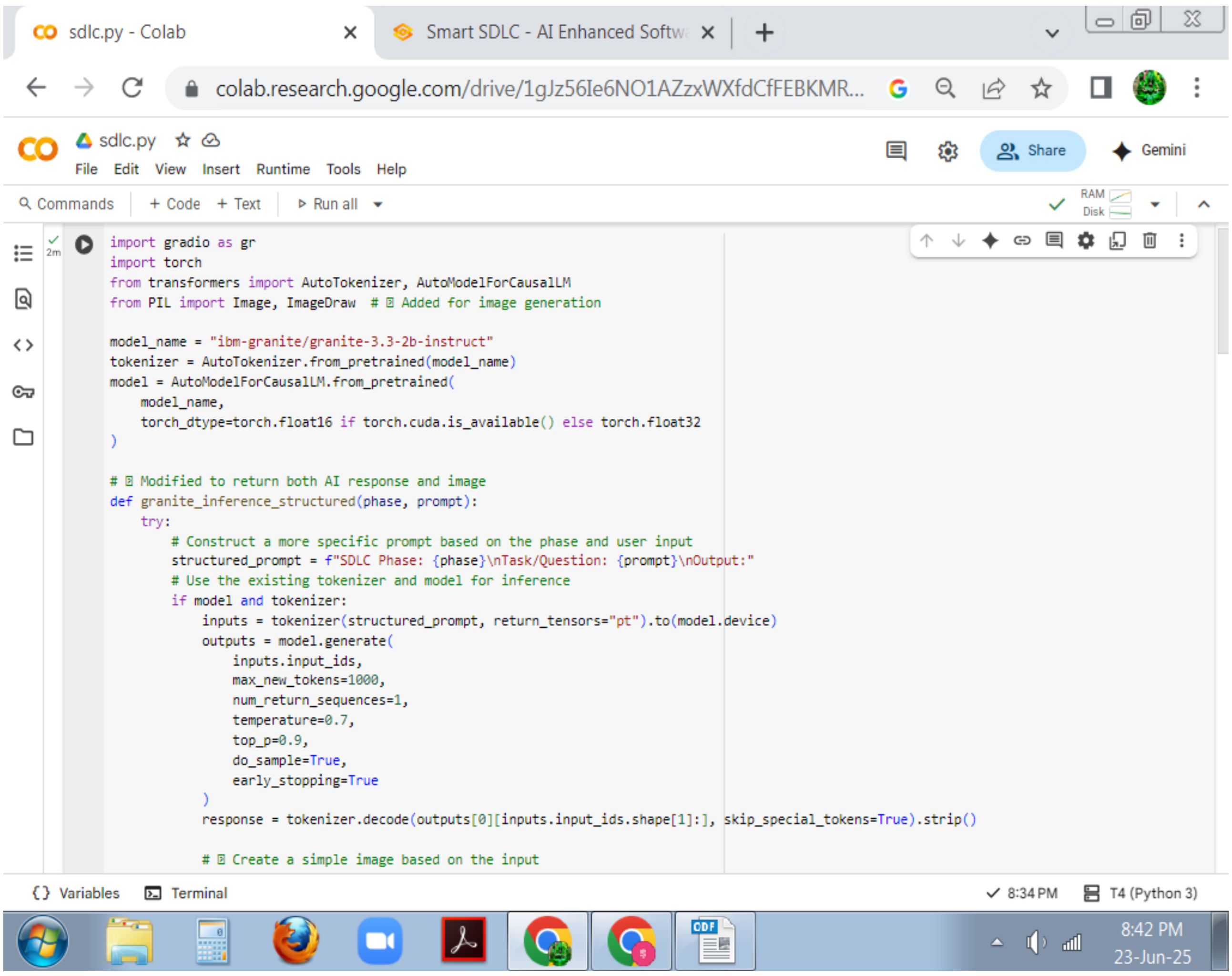
Implement the FastAPI backend - to manage routing and user input processing, ensuring smooth API interaction and prompt handling for the Granite model.

- **Main.py Development :**

The `main.py` file integrates all backend logic and AI model interaction.

Write the main application logic - including model loading, prompt formatting, AI inference, image generation, and Gradio UI logic.

This script includes all imports, interface setup, and launching logic for local testing or public sharing and the code is :

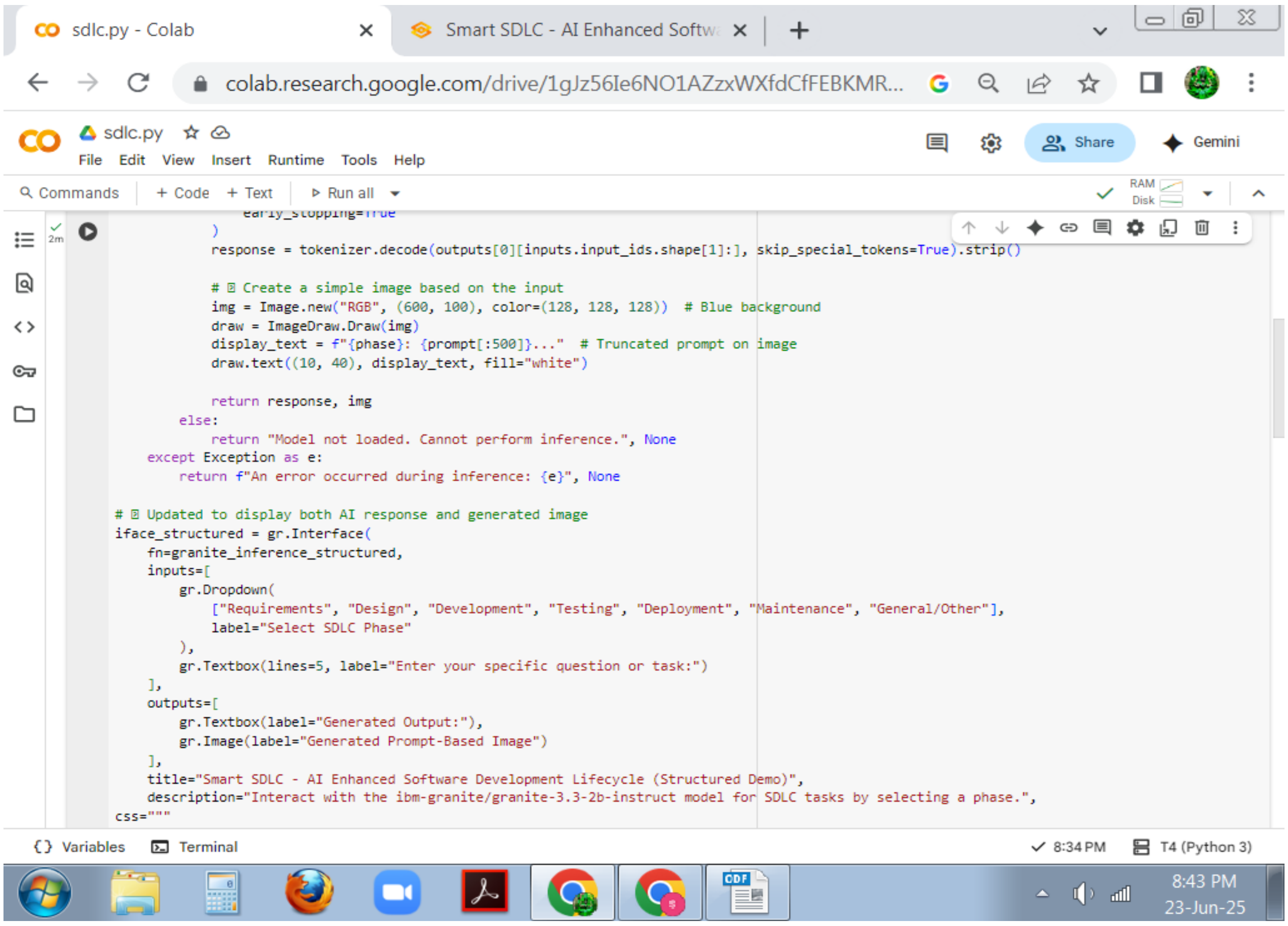

sdlc.py - Colab
Smart SDLC - AI Enhanced Softw
colab.research.google.com/drive/1gJz56Ie6NO1AZzxWXfdCfFEBKMR...
sdlc.py
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code + Text
Run all
RAM
Disk
Variables
Terminal
8:34 PM
T4 (Python 3)
8:42 PM
23-Jun-25

```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
from PIL import Image, ImageDraw  # ▯ Added for image generation

model_name = "ibm-granite/granite-3.3-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32
)

# ▯ Modified to return both AI response and image
def granite_inference_structured(phase, prompt):
    try:
        # Construct a more specific prompt based on the phase and user input
        structured_prompt = f"SDLC Phase: {phase}\nTask/Question: {prompt}\nOutput:"
        # Use the existing tokenizer and model for inference
        if model and tokenizer:
            inputs = tokenizer(structured_prompt, return_tensors="pt").to(model.device)
            outputs = model.generate(
                inputs.input_ids,
                max_new_tokens=1000,
                num_return_sequences=1,
                temperature=0.7,
                top_p=0.9,
                do_sample=True,
                early_stopping=True
            )
            response = tokenizer.decode(outputs[0][inputs.input_ids.shape[1]:], skip_special_tokens=True).strip()

            # ▯ Create a simple image based on the input
```

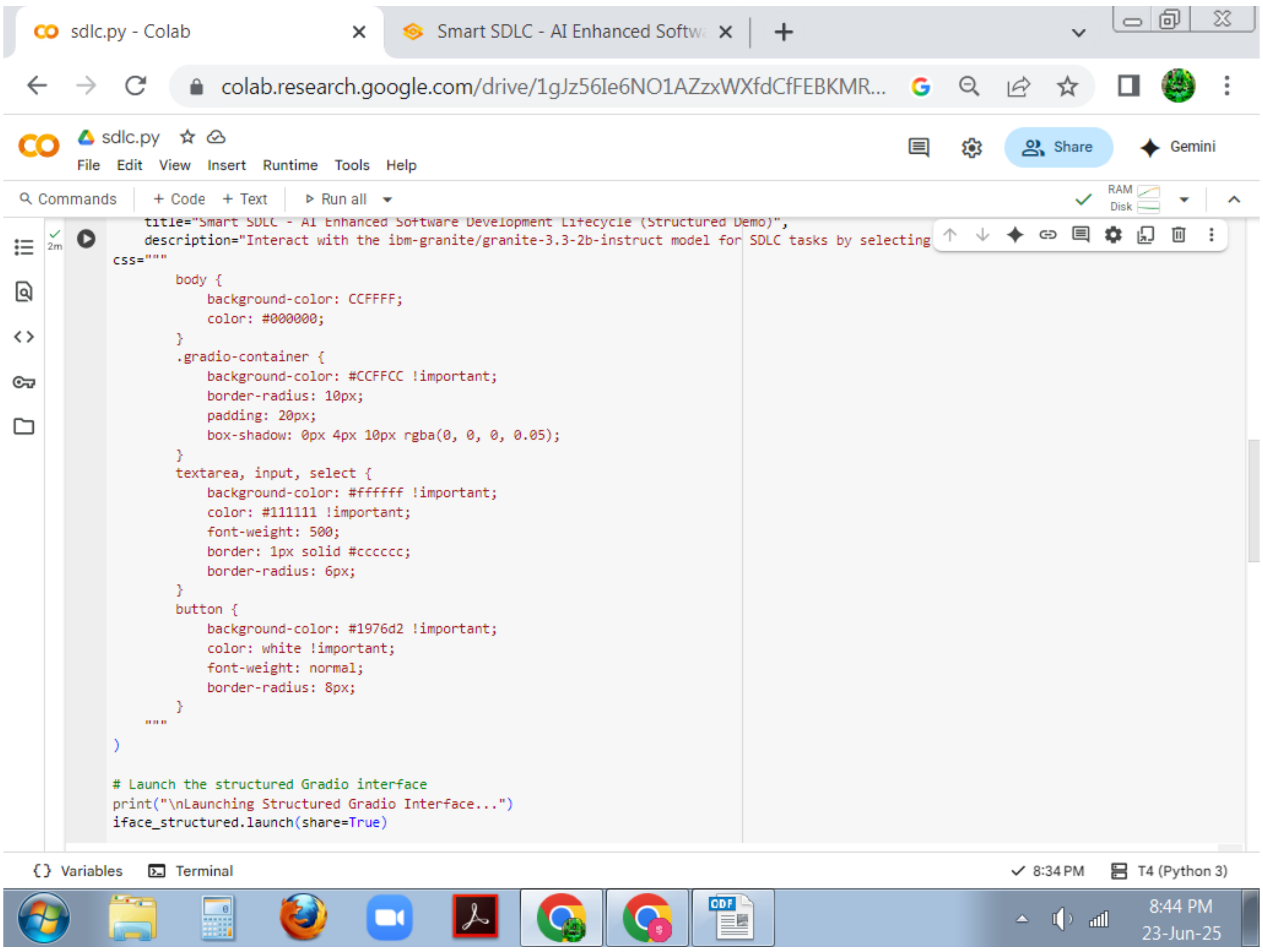

sdlc.py - Colab
Smart SDLC - AI Enhanced Softw
colab.research.google.com/drive/1gJz56Ie6NO1AZzxWXfdCfFEBKMR...
sdlc.py
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
RAM Disk
Variables
Terminal
8:34 PM
T4 (Python 3)
8:43 PM
23-Jun-25

```
            early_stopping=True
        )
        response = tokenizer.decode(outputs[0][inputs.input_ids.shape[1]:], skip_special_tokens=True).strip()

        # 🖼 Create a simple image based on the input
        img = Image.new("RGB", (600, 100), color=(128, 128, 128))  # Blue background
        draw = ImageDraw.Draw(img)
        display_text = f"{phase}: {prompt[:500]}..."  # Truncated prompt on image
        draw.text((10, 40), display_text, fill="white")

        return response, img
    else:
        return "Model not loaded. Cannot perform inference.", None
  except Exception as e:
    return f"An error occurred during inference: {e}", None

# 🖼 Updated to display both AI response and generated image
iface_structured = gr.Interface(
    fn=granite_inference_structured,
    inputs=[
        gr.Dropdown(
            ["Requirements", "Design", "Development", "Testing", "Deployment", "Maintenance", "General/Other"],
            label="Select SDLC Phase"
        ),
        gr.Textbox(lines=5, label="Enter your specific question or task:")
    ],
    outputs=[
        gr.Textbox(label="Generated Output:"),
        gr.Image(label="Generated Prompt-Based Image")
    ],
    title="Smart SDLC - AI Enhanced Software Development Lifecycle (Structured Demo)",
    description="Interact with the ibm-granite/granite-3.3-2b-instruct model for SDLC tasks by selecting a phase.",
css="""
```


sdlc.py - Colab
Smart SDLC - AI Enhanced Softw
colab.research.google.com/drive/1gJz56Ie6NO1AZzxWXfdCfFEBKMR...
sdlc.py
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
Variables
Terminal
8:34 PM
T4 (Python 3)
8:44 PM
23-Jun-25

```
    title="Smart SDLC - AI Enhanced Software Development Lifecycle (Structured Demo)",
    description="Interact with the ibm-granite/granite-3.3-2b-instruct model for SDLC tasks by selecting
css="""
        body {
            background-color: CCFFFF;
            color: #000000;
        }
        .gradio-container {
            background-color: #CCFFCC !important;
            border-radius: 10px;
            padding: 20px;
            box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.05);
        }
        textarea, input, select {
            background-color: #ffffff !important;
            color: #111111 !important;
            font-weight: 500;
            border: 1px solid #cccccc;
            border-radius: 6px;
        }
        button {
            background-color: #1976d2 !important;
            color: white !important;
            font-weight: normal;
            border-radius: 8px;
        }
    """
)

# Launch the structured Gradio interface
print("\nLaunching Structured Gradio Interface...")
iface_structured.launch(share=True)
```

- **Frontend Development :**

Design and develop the user interface - using Gradio components like dropdowns, textboxes, and output areas.

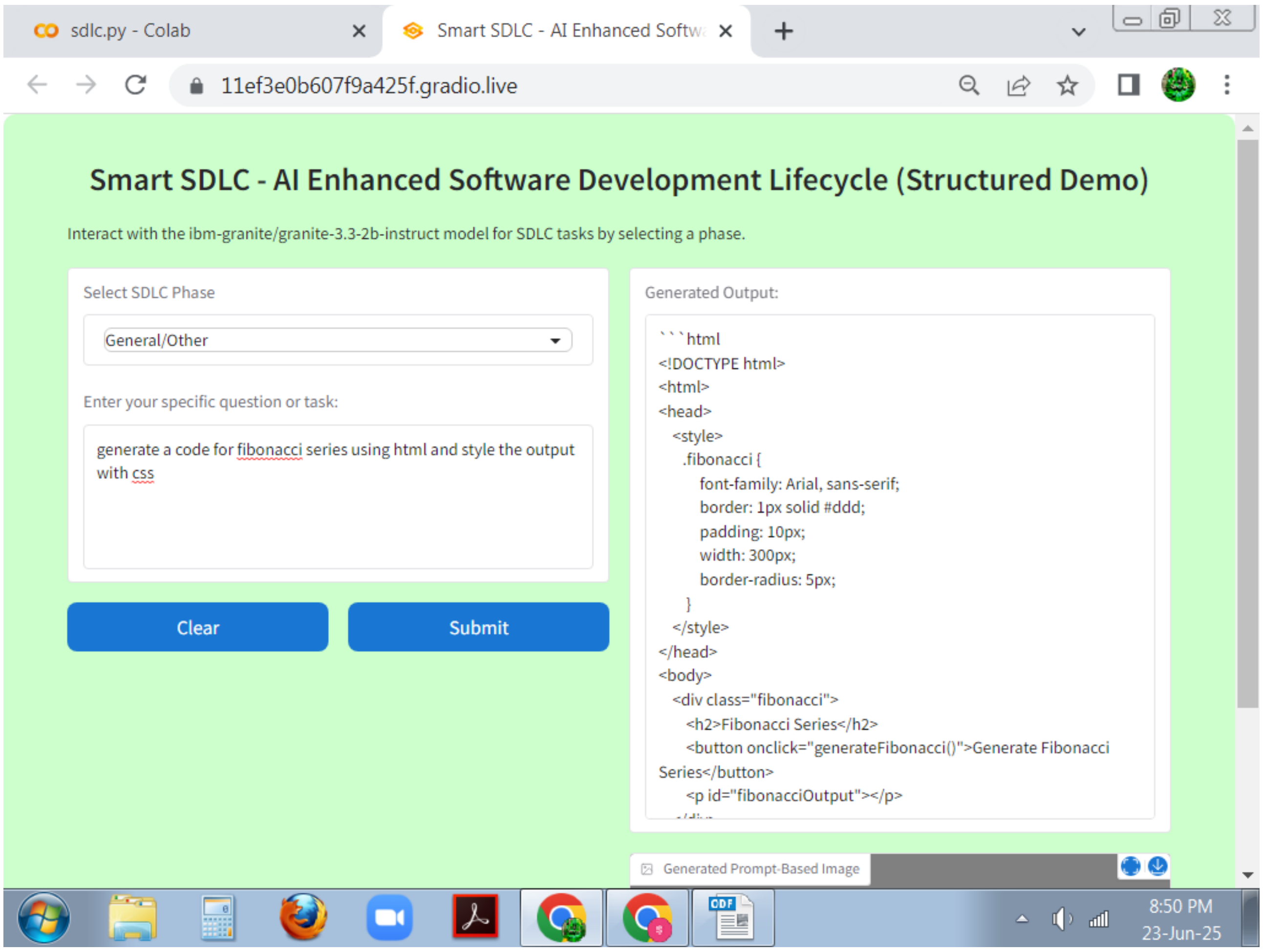
Create dynamic interaction with the backend - so that user inputs get processed in real-time, and AI outputs (text and image) are displayed seamlessly.

- **Deployment :**

Prepare the application for local deployment - by launching the Gradio interface using the `launch(share=True)` or integrating with FastAPI/Streamlit for enterprise use.

Test and verify - the deployment, ensuring all features work properly and model inference responds as expected. Additional deployment options include Hugging Face Spaces or Docker containers.

- **Output:**



sdlc.py - Colab
Smart SDLC - AI Enhanced Softw
11ef3e0b607f9a425f.gradio.live

Smart SDLC - AI Enhanced Software Development Lifecycle (Structured Demo)

Interact with the ibm-granite/granite-3.3-2b-instruct model for SDLC tasks by selecting a phase.

Select SDLC Phase

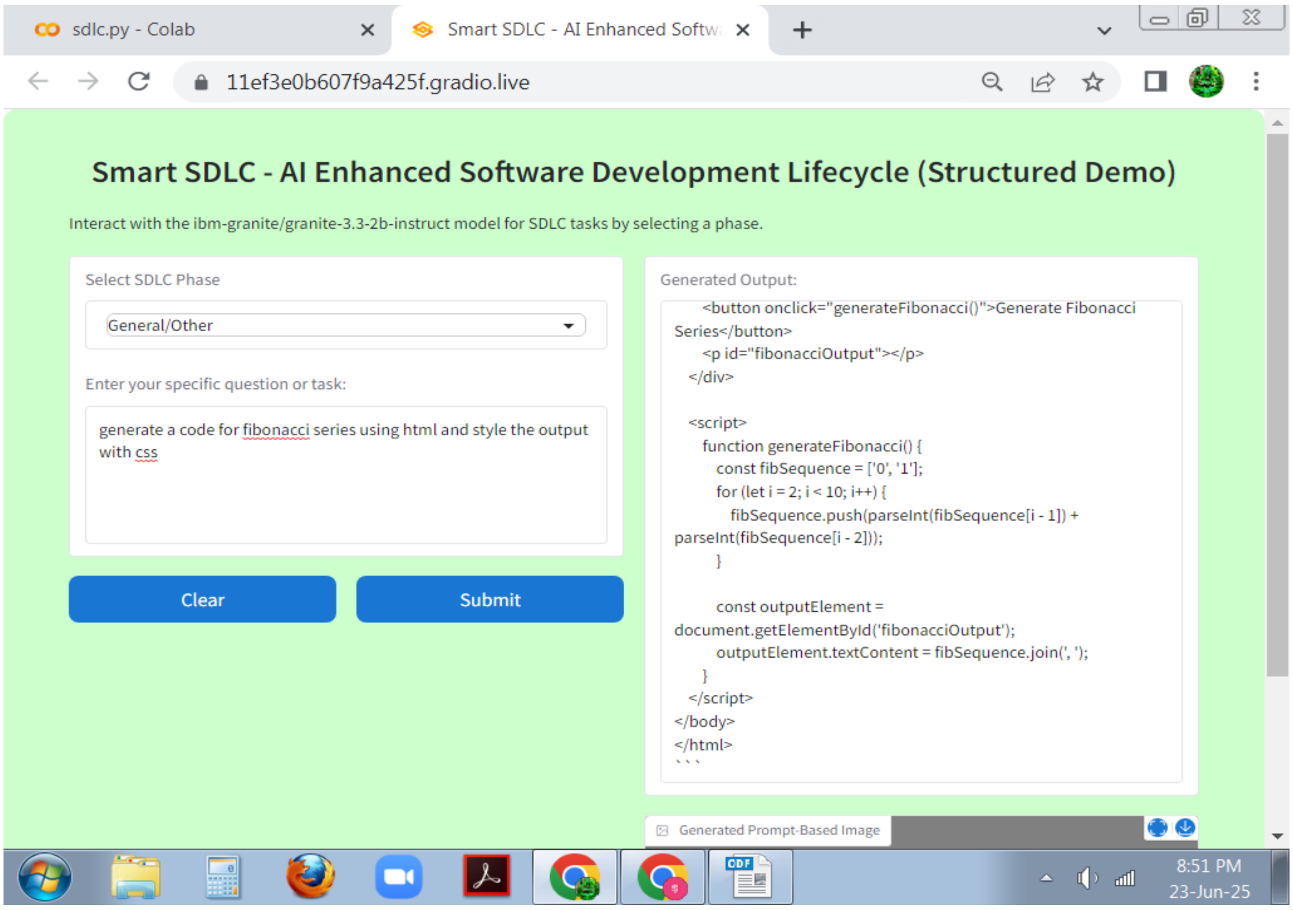
General/Other

Enter your specific question or task:

generate a code for fibonacci series using html and style the output with css

Generated Output:

~~~
```html
<!DOCTYPE html>
<html>
<head>
  <style>
   .fibonacci {
      font-family: Arial, sans-serif;
      border: 1px solid #ddd;
      padding: 10px;
      width: 300px;
      border-radius: 5px;
   }
  </style>
</head>
<body>
  <div class="fibonacci">
    <h2>Fibonacci Series</h2>
    <button onclick="generateFibonacci()">Generate Fibonacci
Series</button>
    <p id="fibonacciOutput"></p>
~~~


sdlc.py - Colab
Smart SDLC - AI Enhanced Softw
11ef3e0b607f9a425f.gradio.live

Smart SDLC - AI Enhanced Software Development Lifecycle (Structured Demo)

Interact with the ibm-granite/granite-3.3-2b-instruct model for SDLC tasks by selecting a phase.

Select SDLC Phase: General/Other

Enter your specific question or task:

generate a code for fibonacci series using html and style the output with css

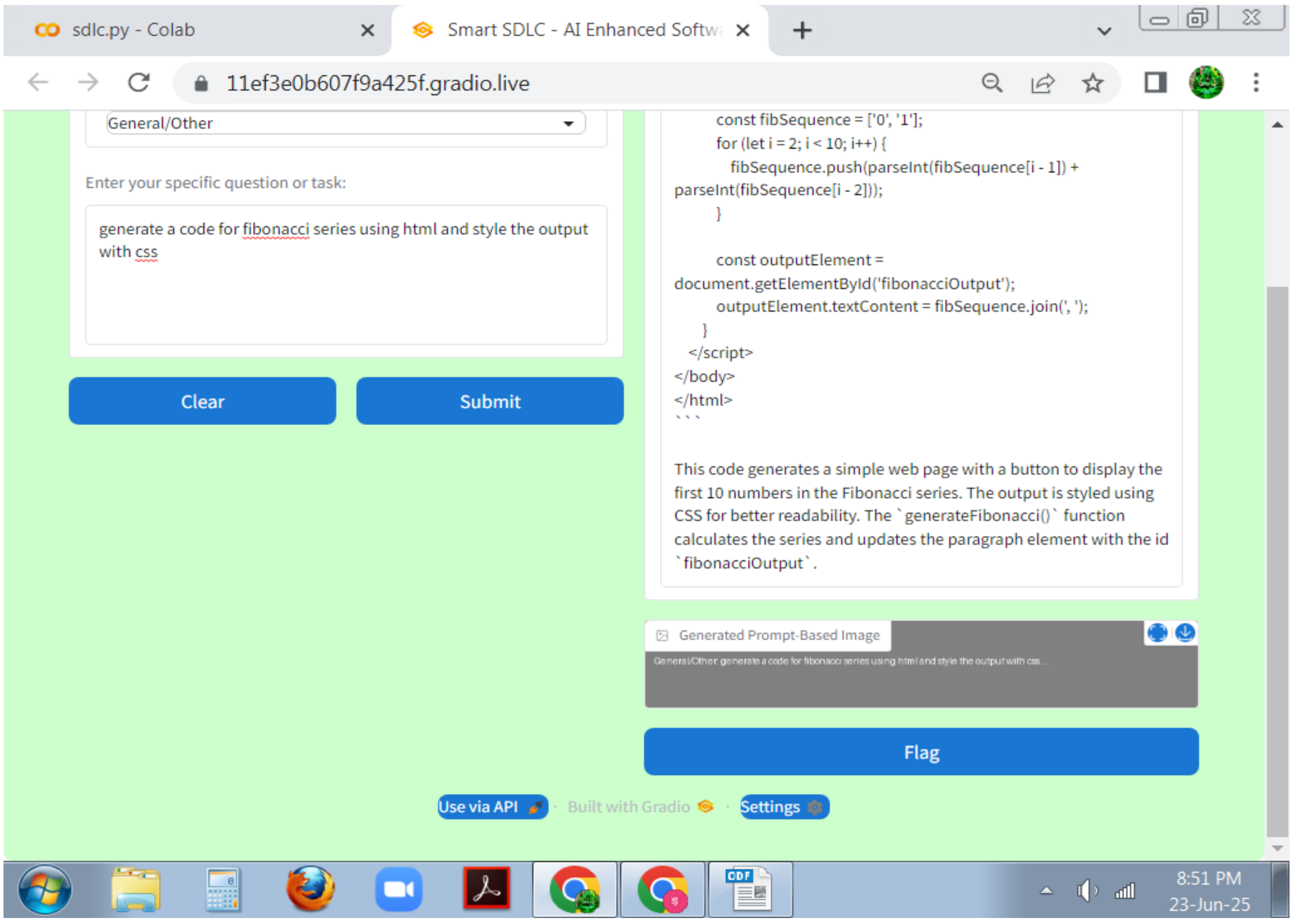
Generated Output:

```
        <button onclick="generateFibonacci()">Generate Fibonacci
Series</button>
        <p id="fibonacciOutput"></p>
    </div>

    <script>
        function generateFibonacci() {
            const fibSequence = ['0', '1'];
            for (let i = 2; i < 10; i++) {
                fibSequence.push(parseInt(fibSequence[i - 1]) +
parseInt(fibSequence[i - 2]));
            }

            const outputElement =
document.getElementById('fibonacciOutput');
            outputElement.textContent = fibSequence.join(', ');
        }
    </script>
</body>
</html>
```
` ` `

Generated Prompt-Based Image
8:51 PM
23-Jun-25


sdlc.py - Colab
Smart SDLC - AI Enhanced Softw
11ef3e0b607f9a425f.gradio.live

General/Other

Enter your specific question or task:

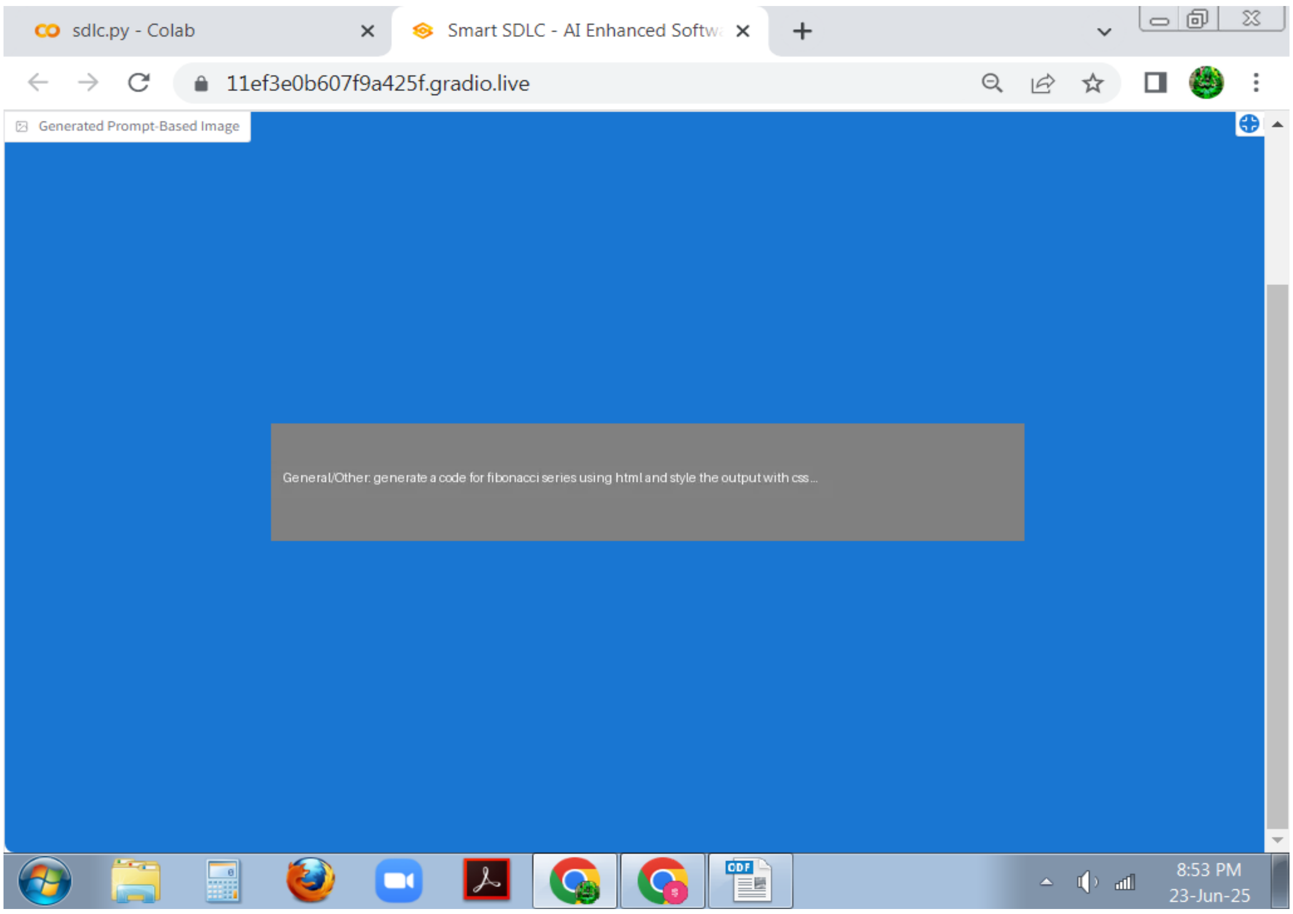
generate a code for fibonacci series using html and style the output with css

```
            const fibSequence = ['0', '1'];
            for (let i = 2; i < 10; i++) {
                fibSequence.push(parseInt(fibSequence[i - 1]) +
parseInt(fibSequence[i - 2]));
            }

            const outputElement =
document.getElementById('fibonacciOutput');
            outputElement.textContent = fibSequence.join(', ');
        }
    </script>
</body>
</html>
```
` ` `

This code generates a simple web page with a button to display the first 10 numbers in the Fibonacci series. The output is styled using CSS for better readability. The `generateFibonacci()` function calculates the series and updates the paragraph element with the id `fibonacciOutput`.

Generated Prompt-Based Image
General/Other: generate a code for fibonacci series using html and style the output with css...
Flag
Use via API
Built with Gradio
Settings
8:51 PM
23-Jun-25


sdlc.py - Colab
Smart SDLC - AI Enhanced Softw
11ef3e0b607f9a425f.gradio.live
Generated Prompt-Based Image

General/Other: generate a code for fibonacci series using html and style the output with css...

- **<u>Conclusion:</u>**

Smart SDLC leverages IBM Watsonx and Granite’s generative AI to bring intelligent automation into software development. It enhances productivity across SDLC phases—from Requirements to Maintenance—by offering phase-specific support and generating helpful textual and visual content. The combination of Python, FastAPI, Gradio, and IBM's AI model creates a highly modular and extensible development assistant.