**SMART SDLC PROJECT-SOURCE CODE**

App.py

Libraries

!pip install transformers torch gradio -q

**Coding :**

import gradio as gr

import torch

from transformers import AutoTokenizer, AutoModelForCausalLM

import PyPDF2

import io

# Load model and tokenizer

model\_name = "ibm-granite/granite-3.2-2b-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,

    device\_map="auto" if torch.cuda.is\_available() else None

)

if tokenizer.pad\_token is None:

  tokenizer.pad\_token = tokenizer.eos\_token

def generate\_response(prompt,max\_length=512):

  inputs = tokenizer(prompt, return\_tensors="pt", truncation=True, max\_length=512)

  if torch.cuda.is\_available():

    inputs = {k: v.to(model.device) for k, v in inputs.items()}

  with torch.no\_grad():

    outputs = model.generate(

        \*\*inputs,

        max\_length=max\_length,

        temperature=0.7,

        do\_sample=True,

        pad\_token\_id=tokenizer.eos\_token\_id

    )

    response = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

    response = response.replace(prompt, "").strip()

    return response

def extract\_text\_from\_pdf(pdf\_file):

  if pdf\_file is None:

    return""

  try:

    pdf\_reader = PyPDF2.PdfReader(pdf\_file)

    text = ""

    for page in pdf\_reader.pages:

      text += page.extract\_text() + "\n"

    return text

  except Exception as e:

    return f"Error reading PDF: {str(e)}"

def requirement\_analysis(pdf\_file, prompt\_text):

  # Get text from PDF or prompt

  if pdf\_file is not None:

    content =  extract\_text\_from\_pdf(pdf\_file)

    analysis\_prompt = f"Analyze the following document and extract key software requirements. organize them into functional  requirements, non-functional requirements: {content}"

  else:

    analysis\_prompt = f"Analyze the following requirements and organize them into functional requirements, non-functional requirements, and technical specific requirements: {prompt\_text}"

  return generate\_response(analysis\_prompt, max\_length=1200)

def code\_generation(prompt, language):

  code\_prompt = f"generate{language} code for the following requirement:\n\n{prompt}\n\nCode:"

  return generate\_response(code\_prompt, max\_length=1200)

# Create  gradio interface

with gr.Blocks() as app:

  gr.Markdown("# Smart Code Analysis & Generation Tool")

  with gr.Tabs():

    with gr.TabItem("Code Analysis"):

      with gr.Row():

        with gr.Column():

          pdf\_upload = gr.File(label="Upload PDF", file\_types=[".pdf"])

          prompt\_input = gr.Textbox(

              label="Or write requirements here",

              placeholder="Describe your software requirements..." ,

              lines=5

          )

          analyze\_btn = gr.Button("Analyze")

        with gr.Column():

          analysis\_output = gr.Textbox(label="Requirements  Analysis", lines=20)

      analyze\_btn.click(requirement\_analysis, inputs=[pdf\_upload, prompt\_input], outputs=analysis\_output)

    with gr.TabItem("Code Generation"):

      with gr.Row():

        with gr.Column():

          code\_prompt = gr.Textbox(

              label="Code Requirements",

              placeholder="Describe what code you want to generate...",

              lines=5

          )

          language\_dropdown = gr.Dropdown(

              choices=["Python", "Javascript", "Java", "C++", "C#", "PHP","Go", "Rust"],

              label="Programming Language",

              value="Python"

          )

          generate\_btn = gr.Button("Generate Code")

        with gr.Column():

          code\_output = gr.Textbox(label="Generate Code", lines=20)

    generate\_btn.click(code\_generation, inputs=[code\_prompt, language\_dropdown], outputs=code\_output)

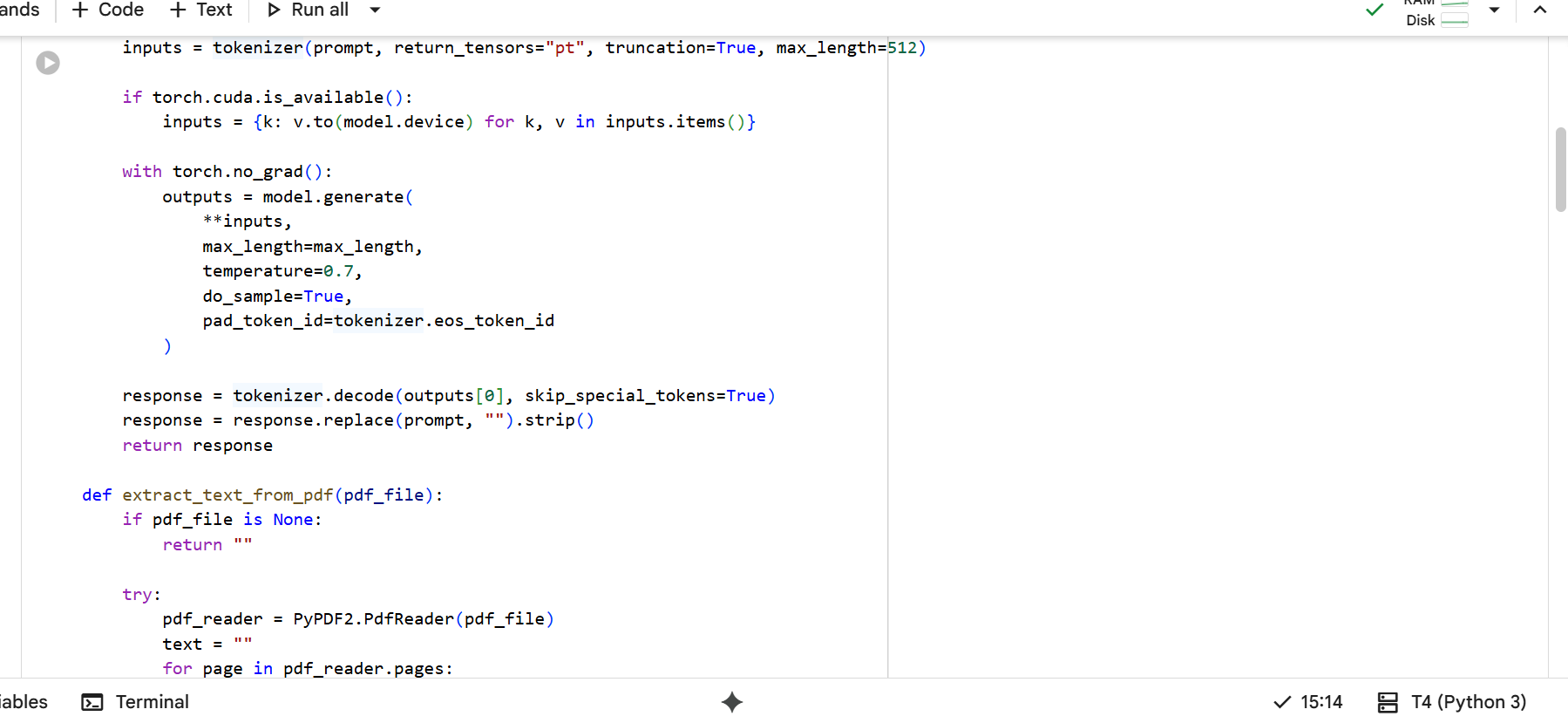
app.launch(share=True)

**step 1 : Edit the code in Google colab**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step 2:**

****

**Step 3:**

****

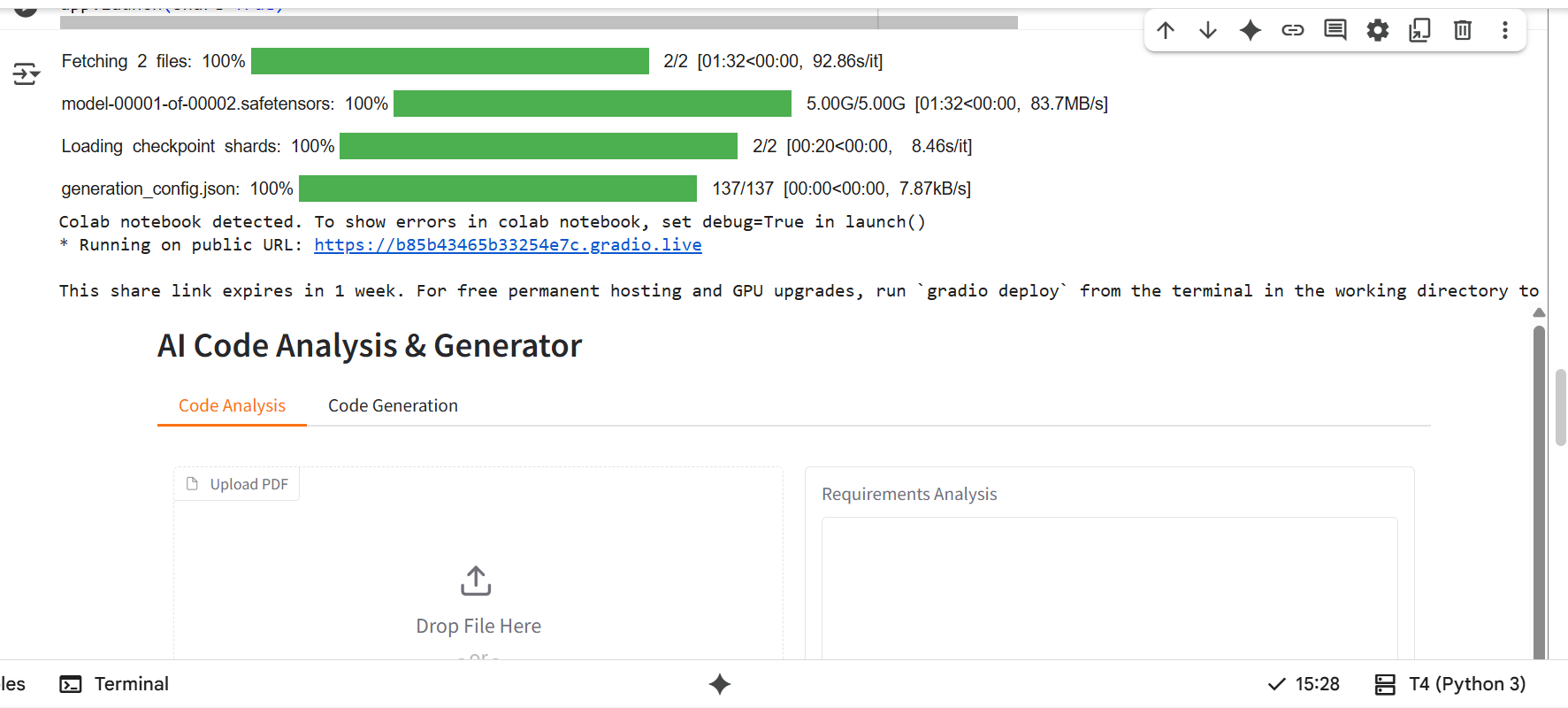
**Step 4:**

****

**Step 5: click the play button**

****

**Step 6: Click the public URL to view the output**

****

**Step 7:**

* **The gradio page is visible here**
* **Upload code as PDF or copy and paste the code A**
* **Click the Analyze button and it will analyze the code in the box**