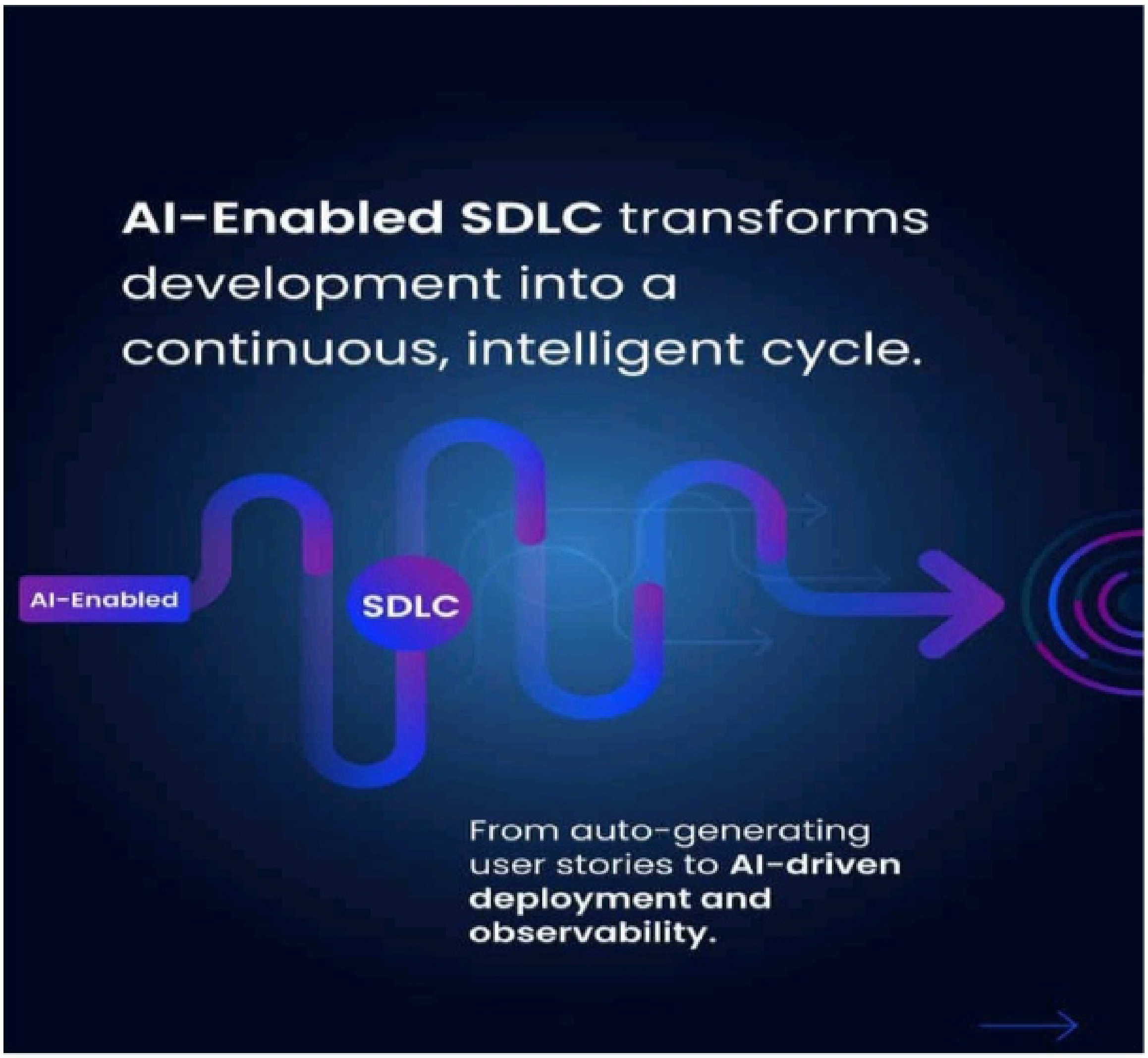
SmartSDLC—AIEnhancedSoftware DevelopmentLifecycle

GenerativeAIwith1BM


AI-Enabled SDLC transforms development into a continuous, intelligent cycle.
AI-Enabled
SDLC
From auto-generating user stories to AI-driven deployment and observability.

1 . INTRODUCTION:

The Smart SDLC (Software Development Life Cycle) is a new-age methodology that integrates Artificial Intelligence (AI) into every phase of the development process. Unlike traditional models. it leverages AI driven automation, forecasting, and user interaction to enhance productivity and accuracy.

TEAM MEMBERS

SRIJA.L

SUBATHRA. A

SUSHMITHA. N-R

SWATHI.K

2. PROJECTOVERVIEW:

The Smart SDLC is designed to overcome the challenges of conventional models such as waterfall,agile and DevOps by' integrating AI into the core processes. This ensures efficient planning, minimal errors, faster delivery, and improved quality,

. Conversation interface, developers, testers and clients interact with the AI as if chatting with a colleague. TheAI helps generate documentation, code snippets testing scripts, and even deployment instructions.

· Poley Summarization: AI reviews organizational policies and ensures compliance In development, This reduces human effort in lengthy policy checks while guaranteeing ethical and legal practices.

· ResourceForecasting:AIpredictshuman,financial,andtechnicalresources required for each phase. This ensures projects do not run into shortages or budget overruns.

. Eco Tip generation, suggest energy, efficient coding practices, optimal server usage and cloud strategies that reduce carbon footprint during development and deployment.

· CitizenFeedbackLoop:Collectsfeedbackdirectlyfromend-usersandincorporatesitinto the development cycle, ensuring the product evolves with real world needs

. KPI forecasting evaluates key performance indicates like cost timeline and quality in advance, allowing early risk detection and proactive adjustments.

· Anamoly detection: constantly monitors system logs condequaity,andinfrastructureto detect unusual patterns that could signal risks or failures.

· Streamlit do Gradio UI: Integrates user-friendly dashboards (Streamlit) with conversation AI tools gradio in both developer and stakeholder accessi ble inter action points.

3. ARCHITECTURE:

The Smart SDCC architecture is divided into frontend and backend. The frontend provides user-friendly dashboards. feedback forms, and chatbats, while the backend integrates A' engines, databases, and ML modules.

· L.LM'ntegration:Large Language Model spowerrequirementanalysis, code generation, and feedback automation.

· Vector sector store: past project knowledge given in vector format enabling AI to recallprevioug work.

· MLModulesSpecializedAIcomponentsforcading,testing,forecasting,anda nomaly detection

4.SETUP INSTRUCTION:

To run Smart SDLC, users must install Python 3.8+, set up Ai/ML libraries like TensorFlöW or PyTOfCh, and configure Streamlit/ Gradio for the UI. A Vector database such as FAISS or Pinecone is also required.

Step-by-step install ation ensures reproducibility and environment stability fordevelopment teams.

5.FOLDER STRUCTURE:

The folder structure ig designed for modularity and clarity

- Frontend → UI
- Backend → APIs & services
- Ml_modules → AI/ML smart modules
- Llm_integration → LLM setup
- Vector_store → Vector database
- Tests → Quality checks
- Configs → Settings
- Docs → Documentation

6.RUNNINGTHEAPPLICATION,

To start the project; the repository, set up the environment, configure databases and APIs, and launch the backend (FastAPl) and frontend (Streamlit,/Gradio), This ensures smooth collaboration and quiCk testing cycles,

7.API DOCUMENTATION:

The APIs connect the frontend with backend AI functionalities. They enable tasks like requirement gathering, anomaly detection, KPV resource forecasting, eco tips, and integrating citizen feedback, Each API is secured and optimized for high performance.

8.AUTHENTICATION:

Role-based authentication ensures data and AI modules are secure, Access tokens, encrypted logins, and privilege levels protect sensitive data while seamless collaboration_

9.USERINTERFACE:

Smart SDLC UI combines dashboards, real-time charts, and AI-powered chatbots_ Nontechnical take holders can view progress visually, while developers interact directly with AI modules for automation.

10. TESTING:

AI automates test case generation, regression testing. and defect detection, Real-time monitoring ensures quick resolution of issues, making software more reliable.

11.KNOWN ISSUES:

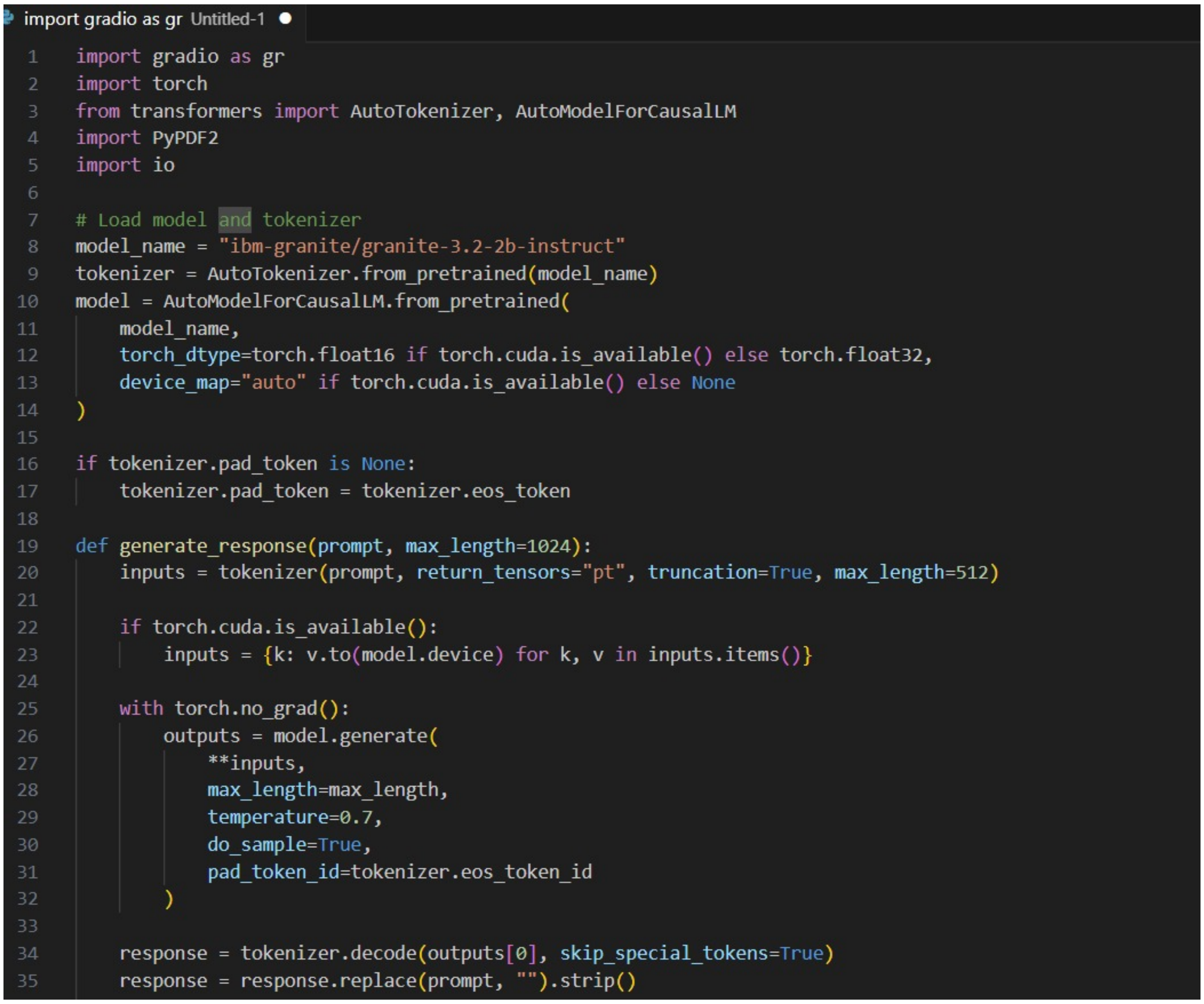
Smart SDLC may face challenges like LIM latency, multimodal input misinterpretation, or false positives in anomaly detection, These limitations are acknowledged' and continuous improvements are ongoing

12.FUTURE ENHANCEMENTS:

Future updates include integrating domain-specific LLMs, expanding multimodal input (voce. sketch-to-code, video), enhancing predictive analytics, and incorporating DevOps pipelines for full automation.

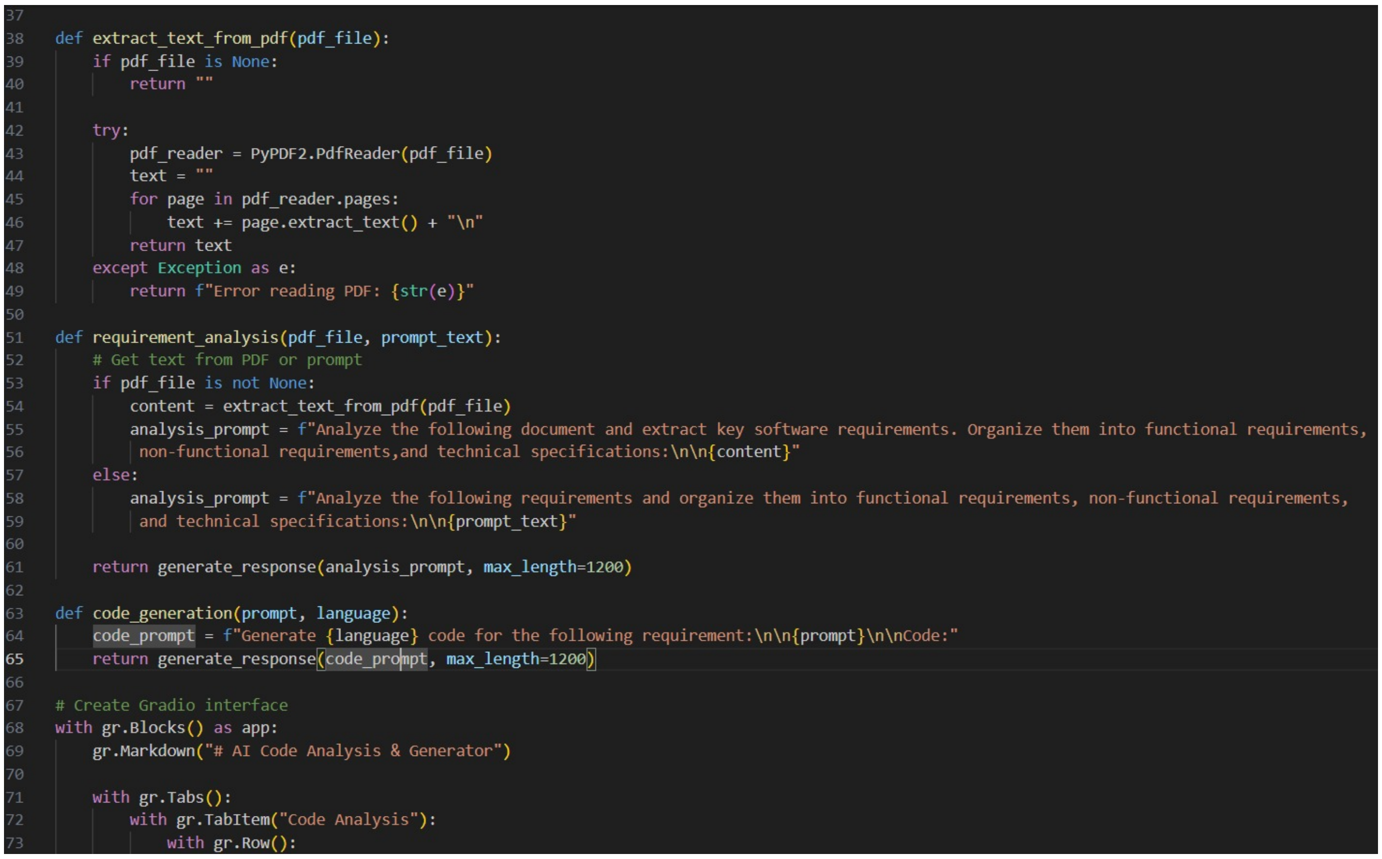
13.PROJECT SCREENSHOTS:

Screenshots of smartsdlc dashboards,chatbot Interaction and anamoly Detection.

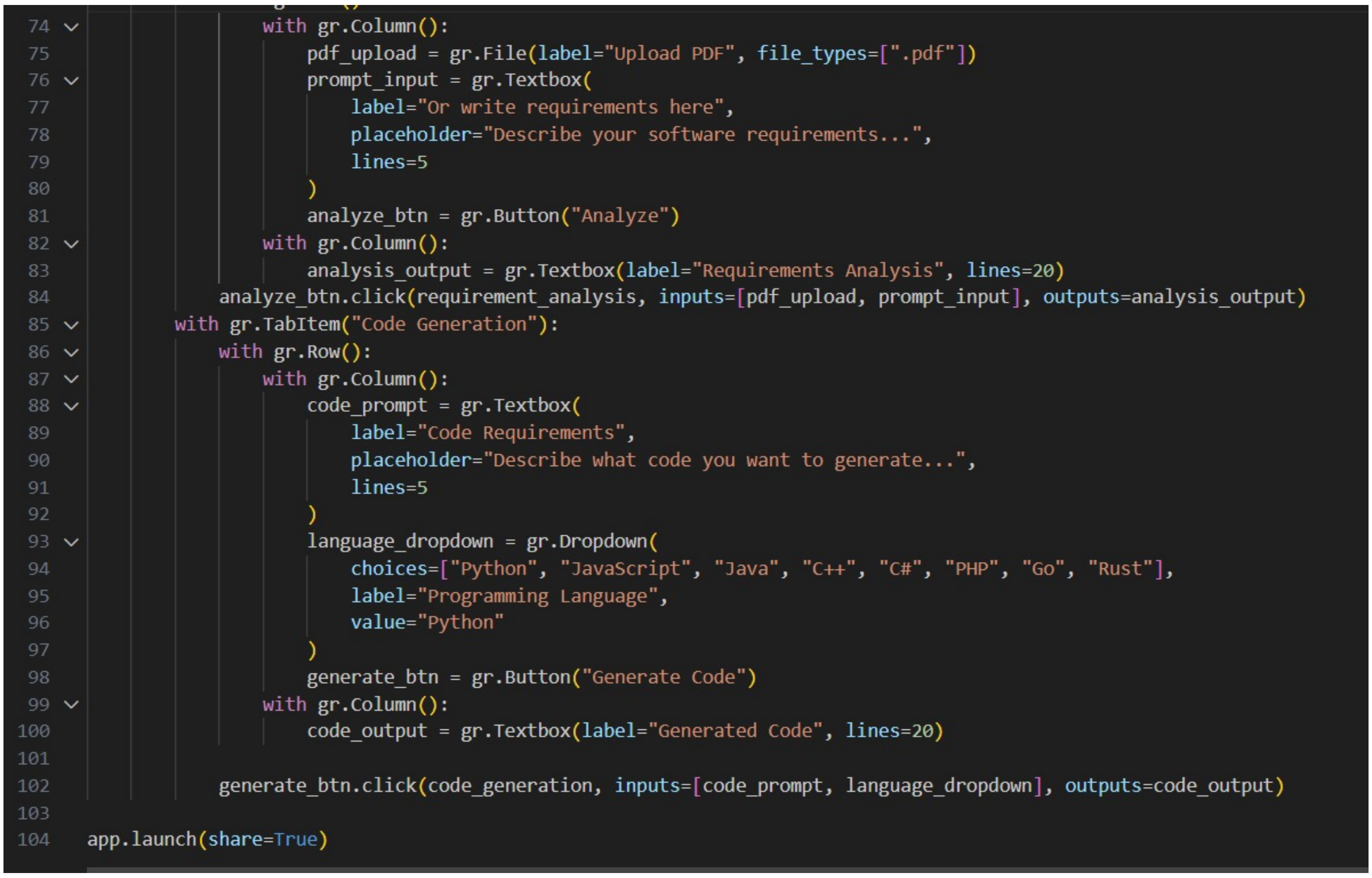


import gradio as gr Untitled-1

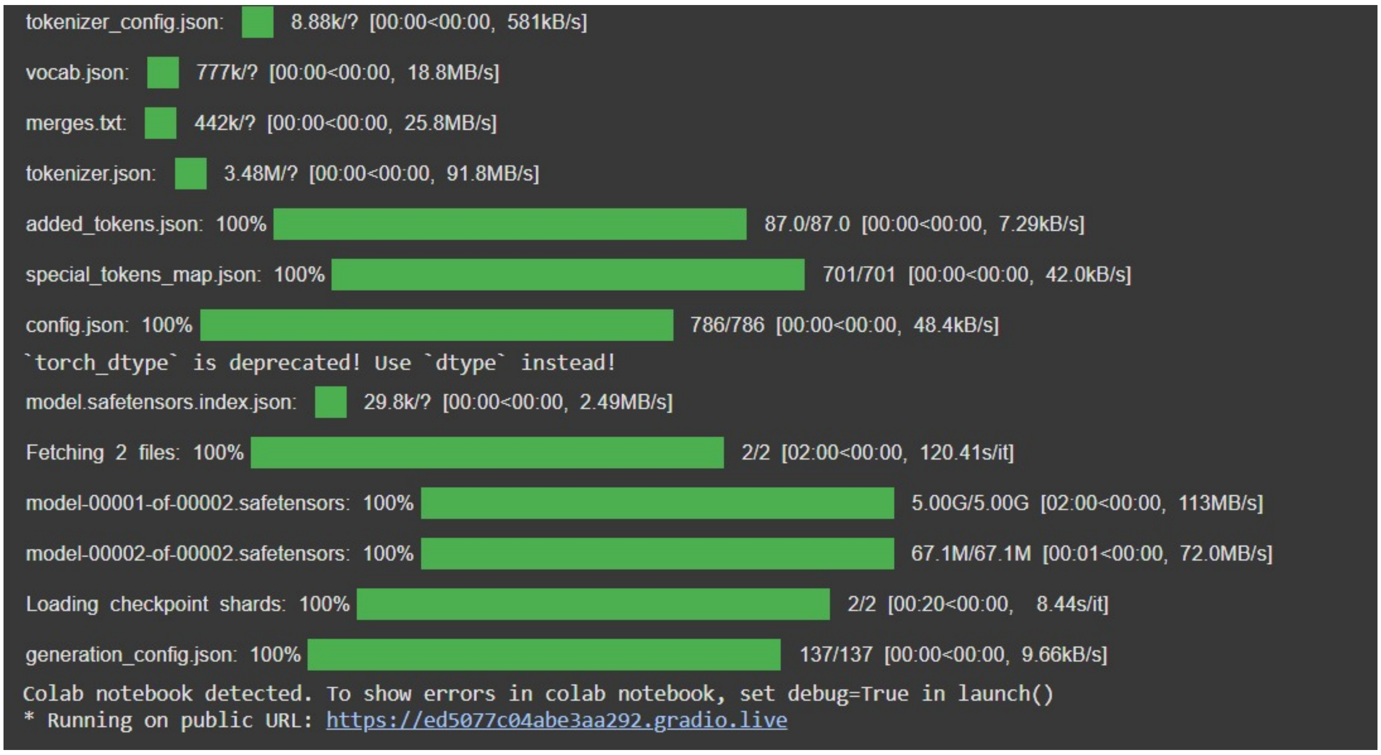
```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2
import io

# 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=1024):
    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()
```



```python

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,and technical specifications:\n\n{content}"
    else:
        analysis_prompt = f"Analyze the following requirements and organize them into functional requirements, non-functional requirements,
         and technical specifications:\n\n{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("# AI Code Analysis & Generator")

    with gr.Tabs():
        with gr.TabItem("Code Analysis"):
            with gr.Row():
```



```python
                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="Generated Code", lines=20)

            generate_btn.click(code_generation, inputs=[code_prompt, language_dropdown], outputs=code_output)

app.launch(share=True)
```



```
tokenizer_config.json:  8.88k/? [00:00<00:00, 581kB/s]
vocab.json:  777k/? [00:00<00:00, 18.8MB/s]
merges.txt:  442k/? [00:00<00:00, 25.8MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 91.8MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 7.29kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 42.0kB/s]
config.json: 100%  786/786 [00:00<00:00, 48.4kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 2.49MB/s]
Fetching 2 files: 100%  2/2 [02:00<00:00, 120.41s/it]
model-00001-of-00002.safetensors: 100%  5.00G/5.00G [02:00<00:00, 113MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:01<00:00, 72.0MB/s]
Loading checkpoint shards: 100%  2/2 [00:20<00:00, 8.44s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 9.66kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://ed5077c04abe3aa292.gradio.live
```