**REQUIREMENT ANALYSIS**

**1.Objective**

To thoroughly understand and document the client’s needs, ensuring that both functional and non-functional requirements are clearly defined before development begins.

**2.Key activities**

**Requirement Elicitation**: Interacting with stakeholders through interviews, workshops, and document analysis to capture needs.

**Classification of Requirements**: Dividing them into functional, non-functional, and technical specifications.

**Feasibility Study**: Analyzing whether the requirements are technically and economically feasible.

**Validation**: Reviewing requirements with clients to avoid ambiguity and contradictions. Documentation: Preparing a Software Requirement Specification (SRS) document.

**3.Deliverables**

Detailed SRS document with diagrams and use cases. Requirement categorization table (functional, non-functional, technical). Use case diagrams or flowcharts for better visualization. Requirement traceability matrix to track fulfillment.

**4. Tools and Techniques**

Requirement gathering tools (interviews, questionnaires). Natural Language Processing (AI model for automated extraction). Documentation tools (MS Word, Confluence, Notion). Gradio interface for PDF/text analysis**.**

**5. Expected Outcomes**

A validated and well-structured set of requirements. Reduced ambiguity and misinterpretation during development. Clear alignment between client needs and development goals.

**PROJECT PLANNING**

**1.Objective**

The purpose of project planning is to outline the roadmap for execution. It ensures resources are allocated optimally, risks are identified, and timelines are realistic. This stage translates requirements into actionable steps and organizes the workflow

**2.Key Activities**

**Scope Definition**: Finalizing the boundaries of the project.

**Resource Estimation**: Identifying manpower, budget, and technology requirements.

**Scheduling**: Creating timelines, Gantt charts, and milestones.

**Risk Analysis**: Identifying potential risks and mitigation strategies.

**Communication Planning**: Defining how progress and updates will be shared among team members**.**

**3.Deliverables**

Project Plan document. Timeline with milestones (Gantt chart). Resource allocation plan. Risk management plan.

**4.Tools and Technology**

Project management tools (Jira, Trello, MS Project). Collaboration platforms (Slack, Teams). Gantt chart/roadmap tools.

**5.Expected Outcomes**

Clear roadmap for execution. Effective resource utilization. Transparent progress tracking. Risk reduction.

**PROJECT DESIGN PHASE**

**1.Objective**

To transform requirements into a clear design blueprint that guides developers. It defines how the system will be structured and ensures compatibility between components.

**2.Key Activities**

**High-Level Design (HLD):**Outlining the architecture, system modules, and data flow.

**Low-Level Design (LLD):** Detailed description of algorithms, database schema, and class diagrams.

**UI/UX Wireframes:** Designing user-friendly interfaces.

**Technology Stack Selection:** Choosing programming languages, frameworks, and databases. Design Review: Getting client approval before moving to development.

**3.Deliverables**

High-level and low-level design documents. Database design documents. UI wireframes or prototypes. Approved architecture blueprint.

**4.Tools and Technology**

UML modeling tools (Lucidchart, Draw.io). Prototyping tools (Figma, Adobe XD). Database design tools (MySQL Workbench).

**5.Expected Outcomes**

Clear blueprint for developers to follow. Reduced rework during coding. Alignment of design with requirements and constraints.

**PERFORMANCE TESTING**

**1.Objective**

To ensure the software performs optimally under varying conditions and meets the desired benchmarks in terms of speed, scalability, and reliability.

**2.Key Activities**

**Defining Metrics**: Setting KPIs like response time, throughput, and error rates. Test Case

**Design**: Creating realistic scenarios with varying load.

**Execution of Load/Stress Tests**: Simulating peak traffic conditions.

**Monitoring & Analysis**: Identifying bottlenecks in CPU, memory, or network.

**Optimization**: Fixing performance issues through tuning and refactoring.

**3.Deliverables**

Performance test plan and test cases. Test reports with metrics. Optimization recommendations.

**4.Tools and Technology**

Performance testing tools (JMeter, LoadRunner, Gatling). Monitoring tools (Grafana, Prometheus).

**5. Expected Outcomes**

Validated system performance under load. Identification of scalability and bottlenecks. Improved user experience and system reliability.

**IDEATION PHASE**

**1.Objective**

To encourage innovation by brainstorming and refining creative solutions that align with project goals and user expectations.

**2.Key Activities**

**Brainstorming Sessions**: Team discussions to generate diverse ideas.

**Problem Identification:** Studying market gaps and user pain points.

**Conceptualization**: Drafting concept notes and solution sketches.

**Feasibility Analysis**: Evaluating each idea for technical and financial viability.

**Prioritization:** Selecting the most impactful and realistic ideas.

**3.Deliverables**

List of potential ideas and solutions. Concept notes and prototypes. Feasibility and impact assessment.

**4.Tools and Technology**

Mind mapping tools (Miro, XMind). Collaboration platforms (Google Workspace, Notion). AI-assisted ideation tools.

**5.Expected Outcomes**

A pool of innovative, feasible solutions. Clear direction for requirement definition. Stronger alignment with user expectations and market trends.

**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=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()

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. "

f"Organize them into functional requirements, non-functional requirements, and "

f"technical specifications:\n\n{content}"

)

else:

analysis\_prompt = (

f"Analyze the following requirements and organize them into functional requirements, "

f"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():

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 Requirement",

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(

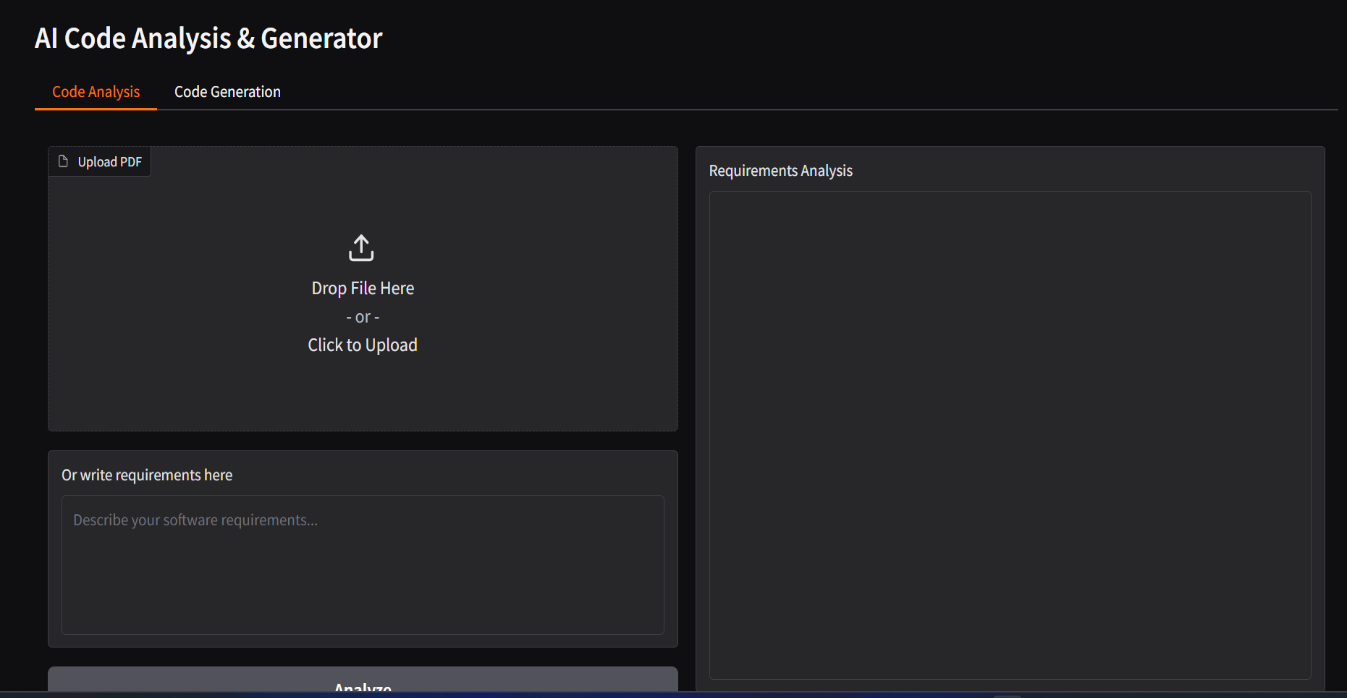
code\_generation,

inputs=[code\_prompt, language\_dropdown],

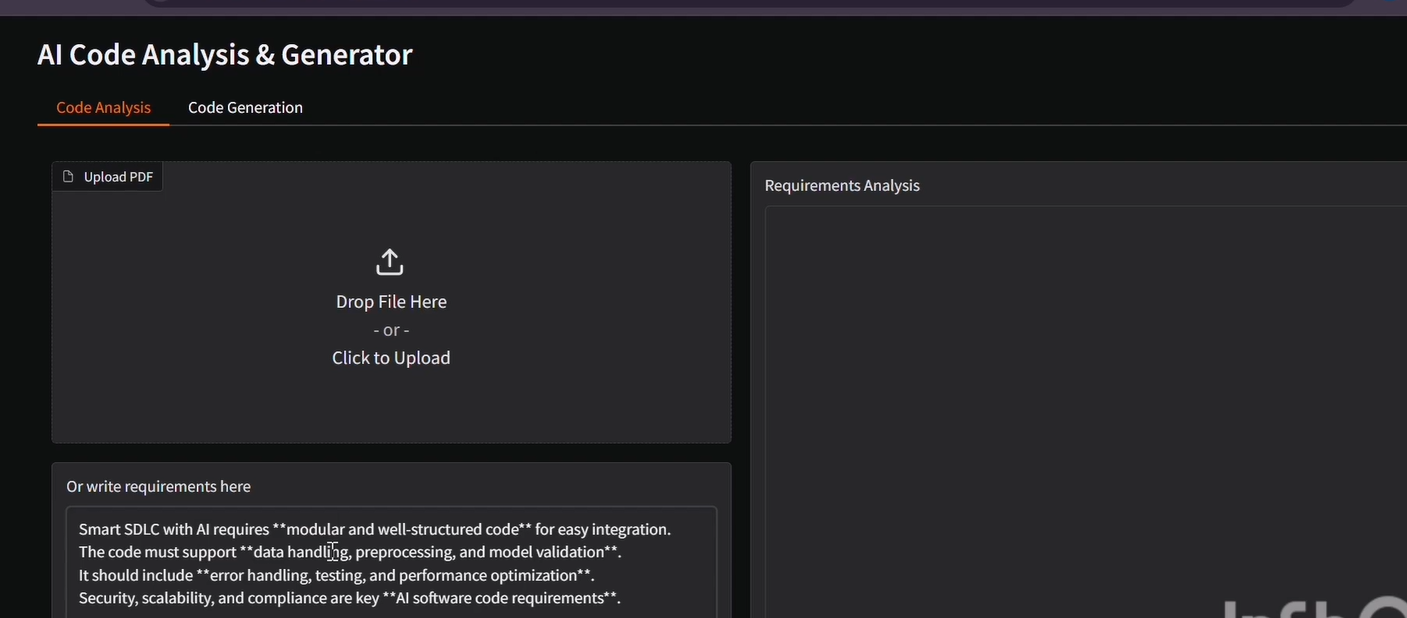
outputs=code\_output

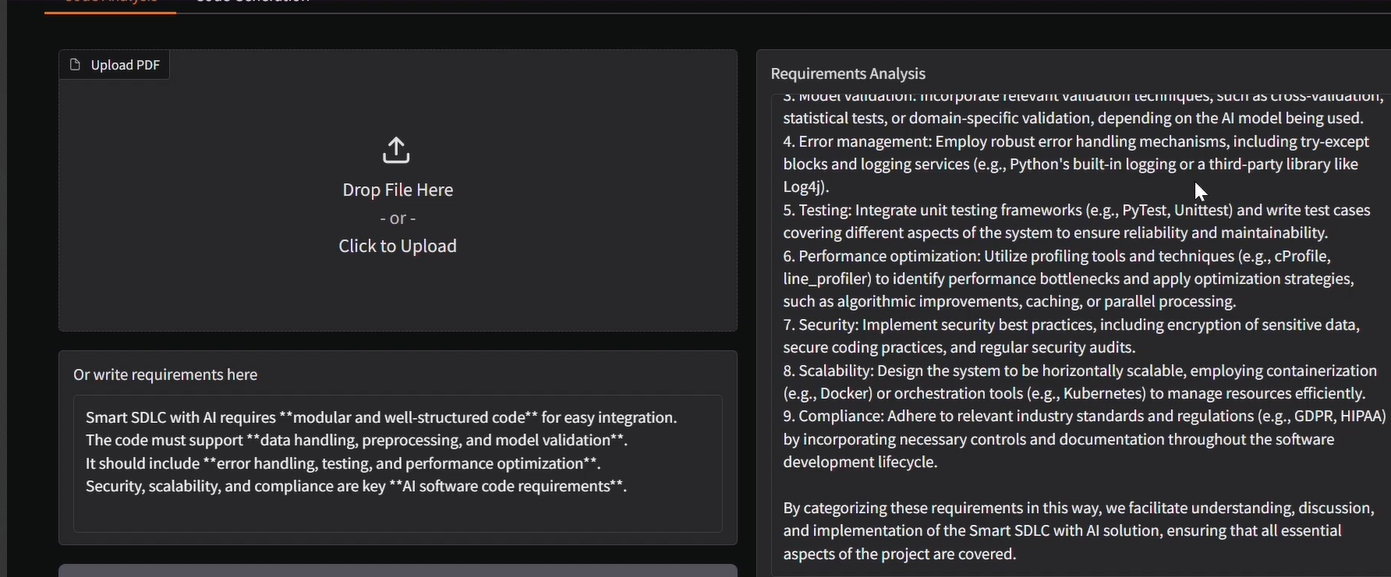
)

app.launch(share=True)

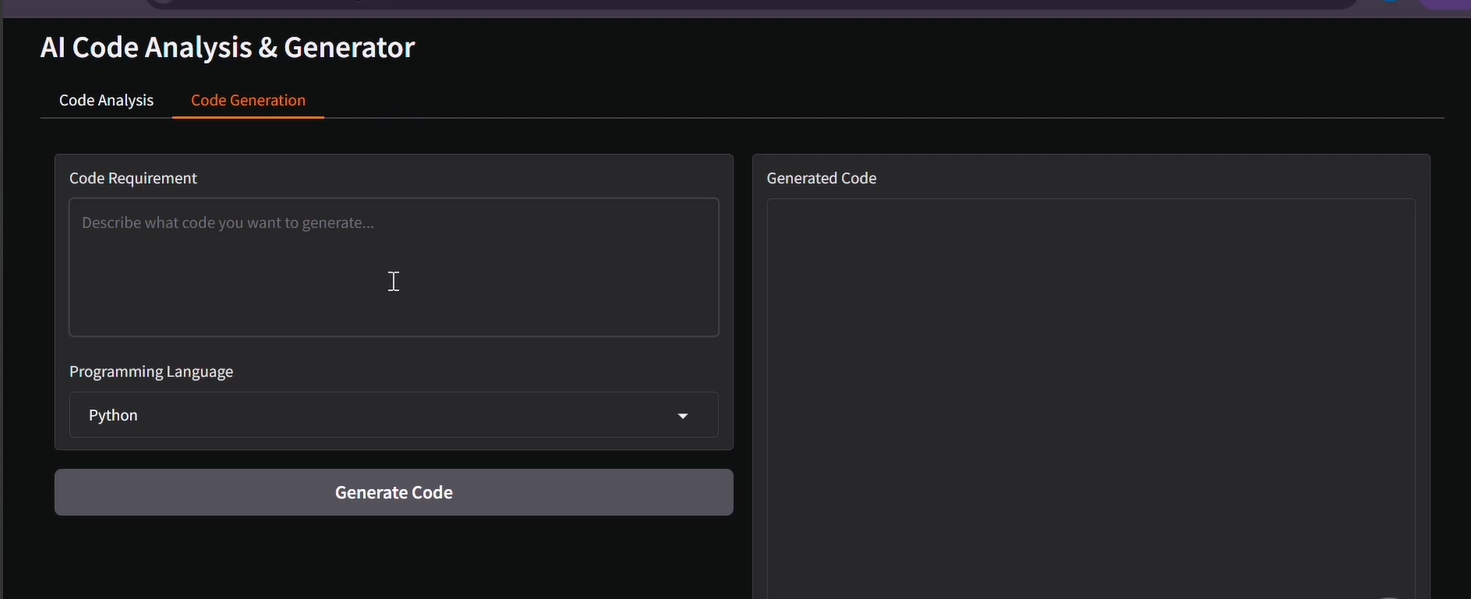
**OUTPUT**

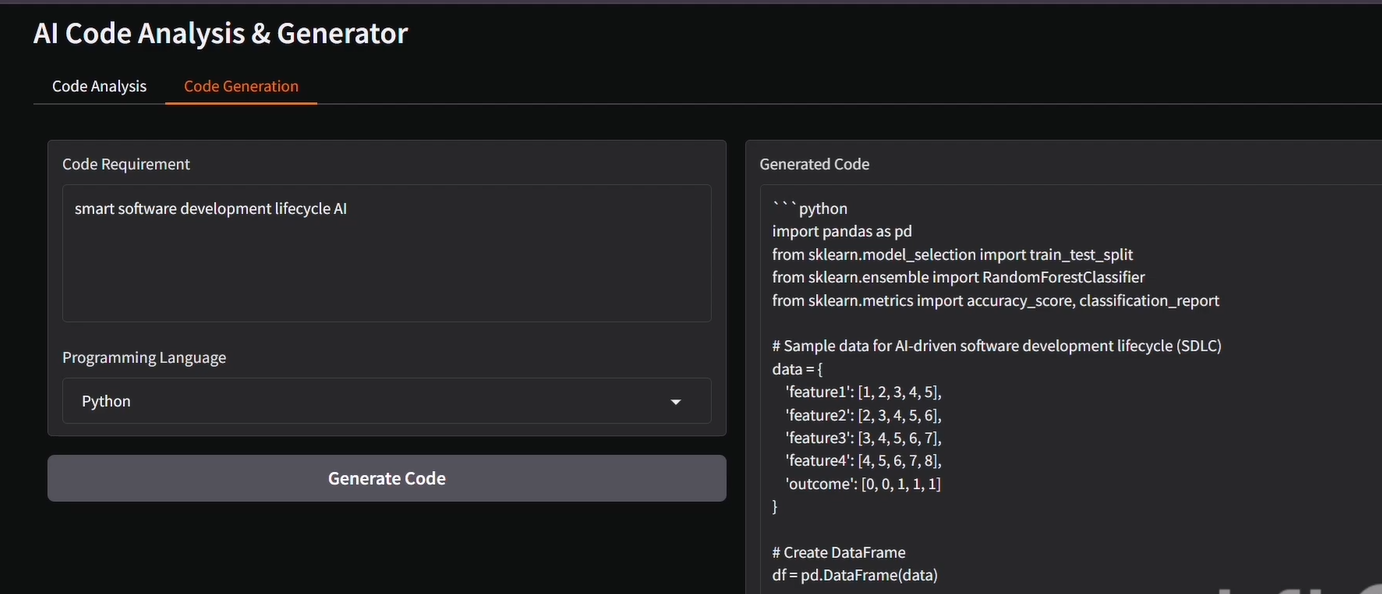
**CODE ANALYSIS**

****

****

**CODE GENERATION**

****

****