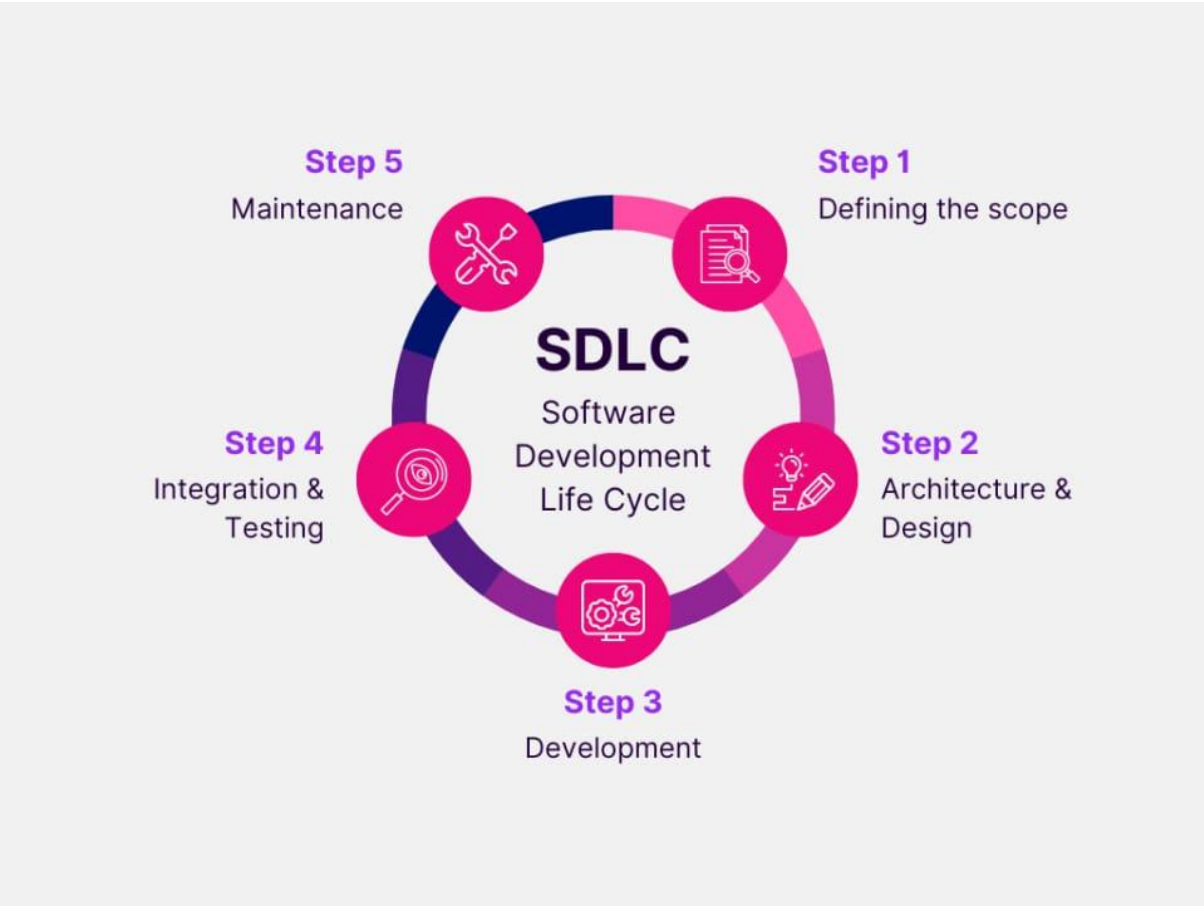


Project Documentation

1. Introduction

- **Project Title**: SmartSDLC Assistant
- **Team ID :** NM2025TMID02722
- **Team Leader :** THENNARASU R
- **Team member :** Lokeshwaran C
- **Team member :** DHANASINGH.M
- **Team member :** THAYUMANAVAN.E
- **Team member :** VENKATESAN M

---

2. Project Overview

- **Purpose**:
  The purpose of the **SmartSDLC Assistant** is to guide students, developers, and professionals in understanding and applying Software Development Life Cycle (SDLC) phases effectively. By leveraging AI (IBM Granite LLM) and an interactive Gradio-based interface, the assistant simplifies SDLC concepts, generates intelligent checklists, and provides smart suggestions tailored to different project scenarios.
- **Features**:
  Explain SDLC Phases
    - *Key Point*: Step-by-step guidance
    - *Functionality*: Explains any SDLC phase (Planning, Analysis, Design, Implementation, Testing, Deployment, Maintenance) in plain language.

AI-Enhanced Checklist

- *Key Point*: Practical task breakdown
- *Functionality*: Generates actionable checklists for each SDLC stage to help in real project execution.

Smart Suggestions

- *Key Point*: Intelligent project insights
- *Functionality*: Provides recommendations, best practices, and risk warnings based on the chosen SDLC phase.

Conversational Interface

- *Key Point*: Natural language queries
- *Functionality*: Allows users to ask SDLC-related questions and receive AI-powered responses.

Gradio UI

- *Key Point*: Simple and interactive
- *Functionality*: Provides a clean, tabbed interface for phase explanation, checklists, and suggestions.

---

3. Architecture

- **Frontend (Gradio in Google Colab)**
  - Built using **Gradio**, providing an easy-to-use tabbed UI directly inside Colab or as a shareable web link.
  - Tabs: *Explain Phase*, *Checklist*, *Smart Suggestions*.

- **Backend (Python + IBM Granite Model)**
  - Uses Hugging Face Transformers to load and interact with **IBM Granite LLM**.
  - Handles text generation and response logic.
- **LLM Integration (IBM Granite)**
  - Model: ibm-granite/granite-3.2-2b-instruct.
  - Used for natural language understanding and content generation.

---

4. Setup Instructions

- **Prerequisites**
  - Google Colab account
  - Python 3.9+ (pre-installed in Colab)
  - Hugging Face Transformers library
  - Gradio library
- **Installation Process (Colab cells)**

1. Install dependencies (!pip install transformers gradio torch).
2. Import model and tokenizer from Hugging Face.
3. Run the Gradio app cell to launch UI.
4. Use the provided link to interact with the app.

---

5. Folder Structure

(Since this is a Colab project, structure is simple)

SmartSDLC/
|— SmartSDLC_Project.ipynb # Main Colab Notebook
|— model/ # IBM Granite model (auto-downloaded from HF)
|— outputs/ # (optional) save generated outputs

---

6. Running the Application

1. Open the Colab Notebook.
2. Run setup and model loading cells.
3. Launch Gradio app using demo.launch().
4. Use tabs to:
   - **Explain Phase** → learn any SDLC stage.
   - **Checklist** → get tasks for that phase.
   - **Smart Suggestions** → receive recommendations.

---

7. API Documentation

- **Internal Functions (inside Colab)**
    - generate_response(prompt) → gets AI-generated text from Granite.
    - explain_phase(phase) → explains a chosen SDLC phase.
    - generate_checklist(phase) → creates checklist tasks.
    - generate_suggestions(phase) → provides smart recommendations.

(No external REST API is used, only Colab functions + Gradio UI)

---

8. Authentication

- Not required.
- Runs openly in Colab environment for learning and demonstration.

---

9. User Interface

- **Gradio Tabs**:
    - **Explain Phase** → Dropdown for SDLC phases + text output.
    - **AI Checklist** → Auto-generated checklist in bullet points.
    - **Smart Suggestions** → Actionable insights and warnings.
- **UI Style**:
    - Simple, minimalistic, student-friendly.

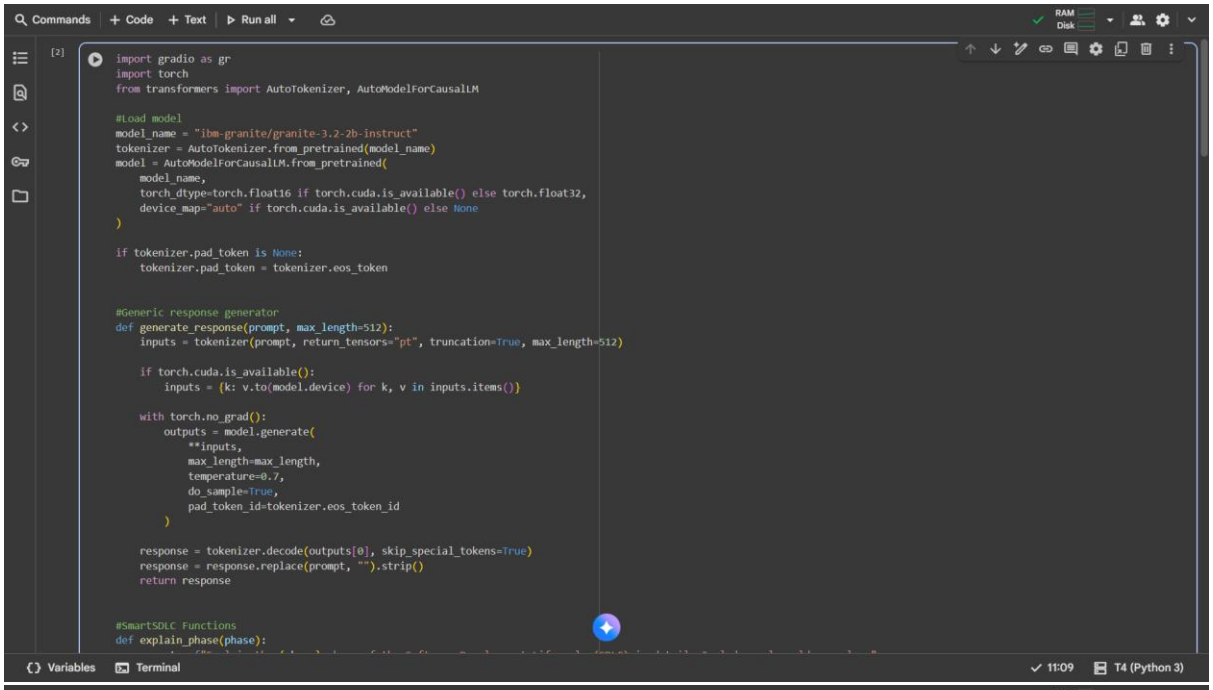
---

10. Testing

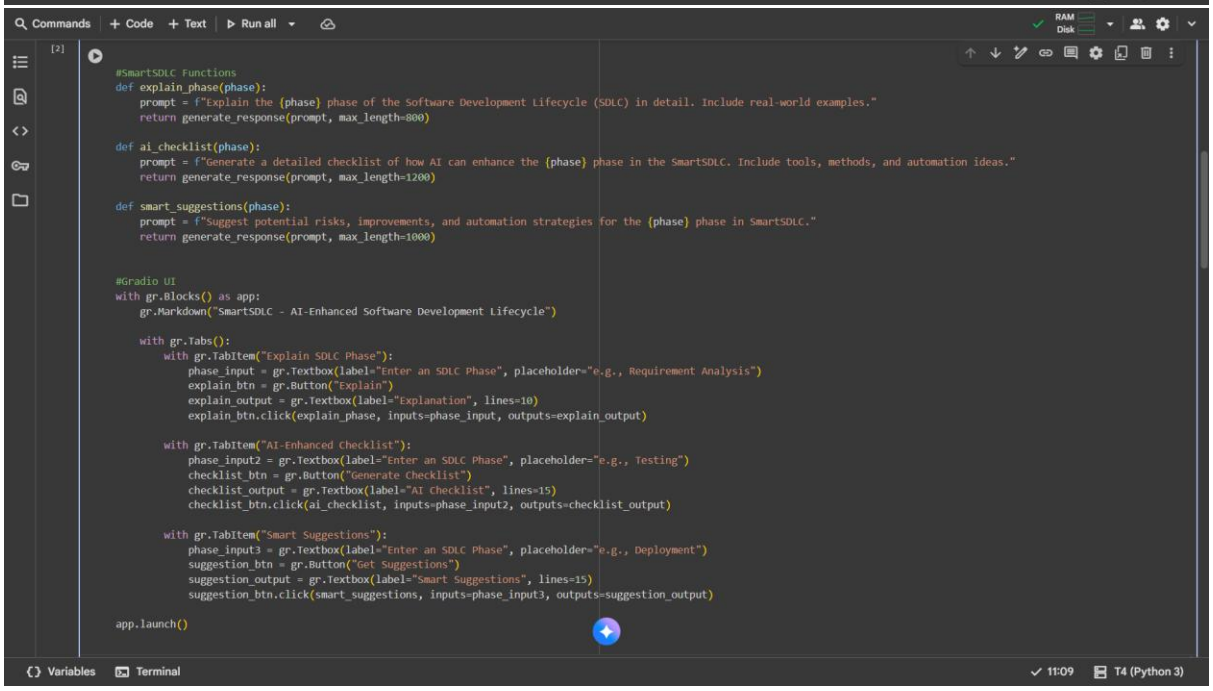
- **Unit Testing**: Tested response generation for each SDLC phase.
- **Manual Testing**: Verified dropdown inputs and tab switching.
- **Edge Cases**: Handled invalid inputs (e.g., unknown phase names).

---

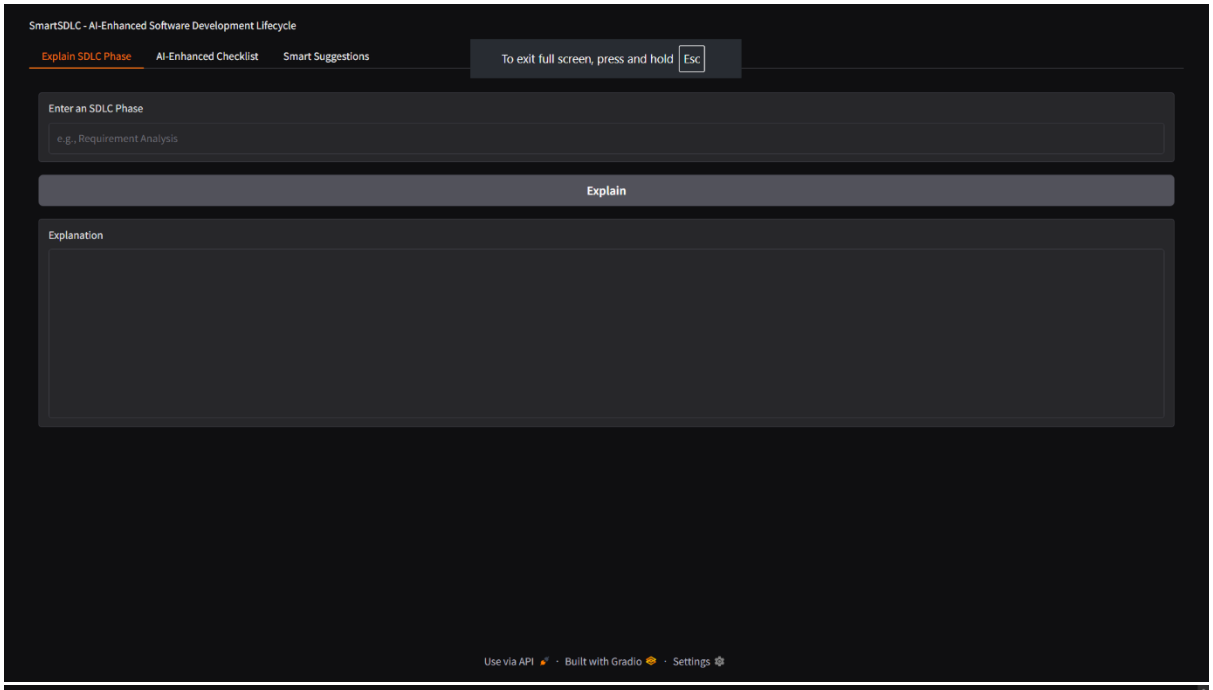
11. Screenshots

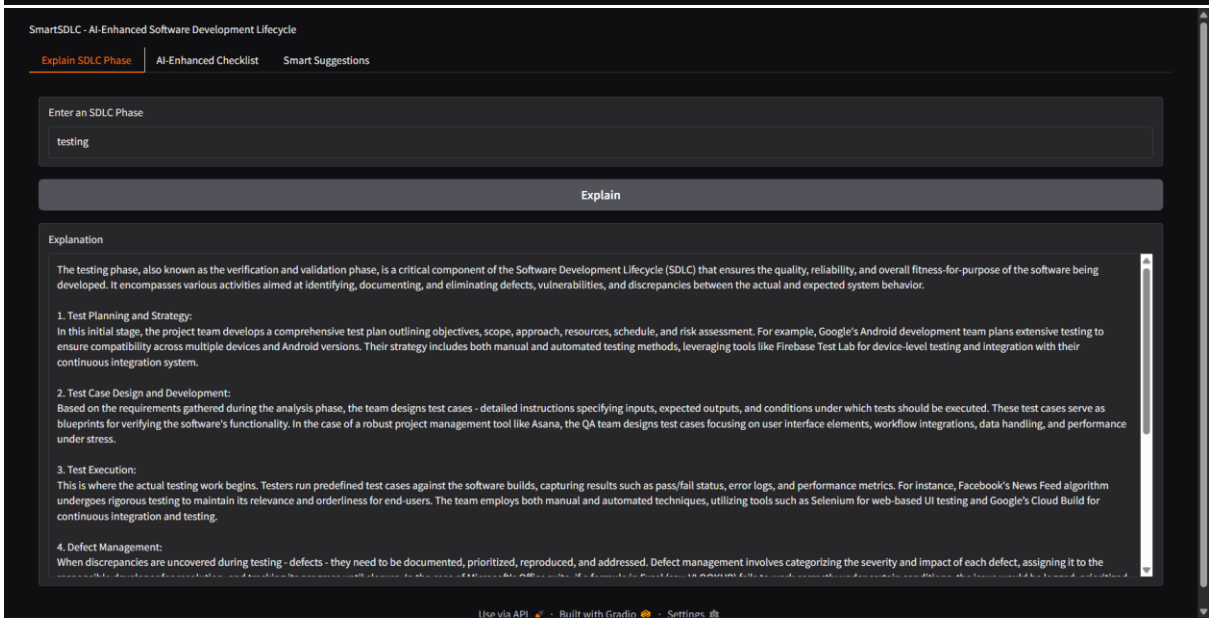


```
[2]
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

#Load model
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


#Generic response generator
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


#SmartSDLC Functions
def explain_phase(phase):
```



```
[2]
#SmartSDLC Functions
def explain_phase(phase):
    prompt = f"Explain the {phase} phase of the Software Development Lifecycle (SDLC) in detail. Include real-world examples."
    return generate_response(prompt, max_length=800)

def ai_checklist(phase):
    prompt = f"Generate a detailed checklist of how AI can enhance the {phase} phase in the SmartSDLC. Include tools, methods, and automation ideas."
    return generate_response(prompt, max_length=1200)

def smart_suggestions(phase):
    prompt = f"Suggest potential risks, improvements, and automation strategies for the {phase} phase in SmartSDLC."
    return generate_response(prompt, max_length=1000)


#Gradio UI
with gr.Blocks() as app:
    gr.Markdown("SmartSDLC - AI-Enhanced Software Development Lifecycle")

    with gr.Tabs():
        with gr.TabItem("Explain SDLC Phase"):
            phase_input = gr.Textbox(label="Enter an SDLC Phase", placeholder="e.g., Requirement Analysis")
            explain_btn = gr.Button("Explain")
            explain_output = gr.Textbox(label="Explanation", lines=10)
            explain_btn.click(explain_phase, inputs=phase_input, outputs=explain_output)

        with gr.TabItem("AI-Enhanced Checklist"):
            phase_input2 = gr.Textbox(label="Enter an SDLC Phase", placeholder="e.g., Testing")
            checklist_btn = gr.Button("Generate Checklist")
            checklist_output = gr.Textbox(label="AI Checklist", lines=15)
            checklist_btn.click(ai_checklist, inputs=phase_input2, outputs=checklist_output)

        with gr.TabItem("Smart Suggestions"):
            phase_input3 = gr.Textbox(label="Enter an SDLC Phase", placeholder="e.g., Deployment")
            suggestion_btn = gr.Button("Get Suggestions")
            suggestion_output = gr.Textbox(label="Smart Suggestions", lines=15)
            suggestion_btn.click(smart_suggestions, inputs=phase_input3, outputs=suggestion_output)

app.launch()
```





---

12. Known Issues

- Model responses depend on IBM Granite training data (sometimes generic).
- Internet required to download the model.
- Colab runtime resets clear the session.

---

13. Future Enhancements

- Add **project-specific templates** for documentation.
- Expand checklist with real-time project management tasks.
- Export results as **PDF reports** directly.

- Add **voice input support** for queries.