SmartSDLC – AI-Enhanced Software Development Lifecycle

1. Introduction

Team ID: NM2025TMID07289
Team Size: 4
Team Leader: Sharifa Jameela Mohamed Mohidee N
Team Member: Shehanas N
Team Member: Hairunnish A
Team Member: Asiya Mariyam H

2. Project Overview

Purpose: SmartSDLC aims to accelerate and streamline the software development lifecycle by integrating IBM Granite AI models. It allows teams to upload PDFs, automatically generate clear requirements, convert prompts into code, create and run tests, fix bugs, write documentation, and chat with an AI assistant — all from a single interface. The project will be deployed on Google Colab using the Granite model for easy setup and performance.

Features:

- Requirements Extraction: Upload PDFs and generate project requirements automatically.
- Code Generation: Convert user prompts into Python code snippets.
- Bug Fixing: AI-assisted debugging and patch generation.
- Test Case Generation: Automatic creation of test cases for uploaded code.
- Documentation Writer: Generate concise technical documentation.
- Interactive Chat Assistant: Provide on-demand help and tips during development.
- Cloud Deployment: Run and test the app in Google Colab using T4 GPU.

3. Architecture

Frontend (Gradio): A Gradio-based interface to upload files, interact with the AI assistant, view generated code, test results, and documentation.

Backend (Python + IBM Granite): Python modules integrate IBM Granite models (granite-3.2-2b-instruct) from Hugging Face to handle requirements extraction, code generation, testing, and documentation.

Deployment (Google Colab): The entire project runs in Google Colab with GPU acceleration.

Version Control (GitHub): All project code is stored and versioned on GitHub for easy collaboration.

4. Setup Instructions

Prerequisites: Python 3.x, Gradio, Transformers, Torch, PyPDF2, IBM Granite model access on Hugging Face, Google Colab T4 GPU knowledge, GitHub account.

Installation Process:
1. Open Google Colab.
2. Create a new notebook and change runtime to T4 GPU.
3. Run:
!pip install transformers torch gradio PyPDF2 -q
4. Copy and paste the project code into the notebook.
5. Run cells to launch the Gradio app.
6. Upload PDFs and interact with the AI modules.

5. Folder Structure (GitHub Repo)

```
smartSDLC/
■■■ app/ # Core Python modules
■■■ requirements.txt # Dependencies
■■■ README.md # Project Overview
■■■ notebook.ipynb # Google Colab notebook
```

6. Running the Application

Open the Google Colab notebook. Change runtime to T4 GPU. Run installation commands and code cells. Click on the Gradio URL to launch the app in a new tab. Upload PDFs, generate code, tests, or documentation interactively.

7. API/Module Documentation

Requirements Extractor Module: Upload PDF → Extract clear requirements.
Code Generator Module: Input prompt → Generate Python code.
Test Case Generator Module: Input code → Generate tests.
Bug Fixer Module: Input code → Fix errors automatically.
Doc Writer Module: Generate Markdown/HTML documentation.

8. Authentication

Public demo runs without authentication.
Planned future enhancement: integrate OAuth2 or API keys for secure deployments.

9. User Interface

Clean Gradio-based dashboard. Tabs for Requirements, Code, Tests, Docs, and Chat. Real-time preview of outputs.

10. Testing

Unit Testing: Verify each module individually.
Integration Testing: Ensure smooth workflow between modules.
Manual Testing: Upload PDFs and verify outputs.
Edge Case Handling: Large files, malformed prompts, and failed API calls.

12. Known Issues

Limited to Python code generation initially.
Requires a stable internet connection for Hugging Face API.

13. Future Enhancements

Add support for more programming languages.
Implement token-based authentication.
Enhance UI with progress tracking and analytics.