Project Documentation

1. Introduction

Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle
Team Members:
- Member 1 – Sathish
- Member 2 – Chandru
- Member 3 – Kowresh
- Member 4 – Gokul

2. Project Overview

The SmartSDLC project leverages IBM Granite LLMs to accelerate the software development lifecycle. It allows users to upload PDFs, extract requirements, generate code, create tests, fix bugs, and interact with an AI assistant. The solution improves developer productivity, reduces errors, and provides a structured workflow for building reliable software.
Features:
- Requirement Extraction – Analyze PDFs / text and categorize requirements
- Code Generation – Generate Python, Java, C++, and other language snippets
- Test Case Generation – Automatically create unit tests
- Bug Fixing & Documentation – Suggest bug fixes and generate project documentation
- AI Assistant – Provide guidance during development
- Integration with GitHub – Save and share code/projects
- Colab Deployment – Run easily in Google Colab with GPU acceleration
- Gradio UI – User-friendly interface for analysis and code generation

3. Architecture

Frontend (Gradio): Interactive UI with tabs for Requirement Analysis and Code Generation.
Backend (Python + Hugging Face Transformers): Powered by IBM Granite (granite-3.2-2b-instruct).
Deployment: Runs on Google Colab T4 GPU, integrates with GitHub.
Key Tools: transformers, torch, gradio, PyPDF2.

4. Setup Instructions

Prerequisites:
- Python 3.9+
- Google Colab with T4 GPU
- Hugging Face account & IBM Granite model access
- GitHub account for version control

Installation:
- Install dependencies: !pip install transformers torch gradio PyPDF2 -q
- Run provided Python script in Colab
- Upload requirements PDF or enter text manually
- Interact with Gradio interface
- Optional: Upload results/code to GitHub

5. Folder Structure

- app/ – Main application code
- requirements_analysis.py – Extract requirements
- code_generation.py – Generate language-specific code

- pdf_utils.py – PDF text extraction
- app_ui.py – Gradio interface
- main.py – Entry script

6. Running the Application

- Open Google Colab and enable GPU
- Upload code or clone from GitHub
- Run the script to launch the Gradio app
- Use Requirement Analysis Tab – upload PDF or enter text
- Use Code Generation Tab – enter requirement + select language
- View output in real time

7. Team Member Contributions

- Sathish – Frontend development (Gradio UI) and documentation
- Chandru – Backend integration with Hugging Face and requirement analysis module
- Kowresh – Code generation, testing and debugging
- Gokul – Deployment in Google Colab, GitHub integration and project setup

8. Future Enhancements

- Expose API endpoints (/analyze, /generate-code, /generate-tests)
- Add authentication (API keys, role-based access)
- Enhance collaboration with GitHub OAuth integration