# SmartSDLC – Project Documentation

1. Introduction

Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle

Team Member 1: Thenmozhi.T (Leader) 23RCN57

Team Member 2: Varsha.N 23RCN58

Team Member 3: Vinoliya.S 23RCN59

Team Member 4: Yogeshwari.T 23RCN60

2. Project Overview

## Purpose:

SmartSDLC is an AI-driven assistant that enhances every stage of the software development lifecycle (SDLC). It helps developers automatically extract requirements from documents, generate code from prompts, create test cases, debug code, write documentation, and interact with an AI helper for guidance.

The project was entirely implemented using Google Colab for execution, with source code and documentation hosted on GitHub.

## Features:

* - Requirement Extraction – Upload a PDF to generate clear software requirements.
* - Prompt-to-Code Generation – Converts natural language prompts into functional code.
* - Test Case Creation – Automatically generates unit test cases.
* - Bug Fixing – Identifies and fixes coding errors.
* - Documentation Generator – Creates structured documentation from project inputs.
* - AI Chat Assistant – Provides guidance and explanations.
* - Colab Integration – Runs fully in Google Colab (no local setup needed).
* - Gradio Frontend – Interactive UI for easy use.
* - GitHub Hosting – All project files are version-controlled.

3. Architecture

## Google Colab Environment:

* - Used as the main execution platform.
* - Handles library installation, model loading, and launching the Gradio app.

## Frontend (Gradio):

* - Simple interface with tabs for requirements, code, tests, bug fixing, and docs.
* - Runs inside Colab with a shareable link.

## Backend (Python logic inside Colab):

* - All logic runs inside Python notebooks.
* - Integrates IBM Granite models for requirement and code generation.

## LLM Integration (IBM Granite):

* - Model used: granite-3.2-2b-instruct from Hugging Face.

## Data Storage:

* - Source code & outputs stored in GitHub.
* - Users can download reports directly from Colab.

4. Setup Instructions

## Prerequisites:

* - Google Account with Colab access
* - Hugging Face account for IBM Granite models
* - GitHub repository for hosting project files
* - Internet connection for model loading

## Installation in Colab:

!pip install transformers torch gradio PyPDF2 -q

5. Folder Structure (GitHub Repository)

smart-sdlc/

│── main.ipynb # Google Colab notebook

│── requirements.txt # Dependencies

│── model\_integration.py # Handles Granite model

│── gradio\_ui.py # UI definition

│── utils/ # Helper functions (PDF parsing, docs)

│── README.md # Documentation

6. Running the Application

1. Open Google Colab.

## 2. Clone your GitHub repo:

!git clone https://github.com/your-username/smart-sdlc.git

%cd smart-sdlc

3. Install dependencies.

4. Run main.ipynb cells.

5. A Gradio link will be generated → open it in your browser.

6. Interact with SmartSDLC (upload PDF, generate code, tests, etc.).

7. API Documentation

## Internal functions (executed inside Colab):

* - extract\_requirements(pdf\_file) → Extracts software requirements.
* - generate\_code(prompt) → Produces Python code from text prompt.
* - generate\_tests(code) → Creates unit tests for generated code.
* - fix\_bugs(code) → Suggests fixes for buggy code.
* - generate\_docs(inputs) → Creates documentation from outputs.

(No external FastAPI endpoints – all logic is in Colab.)

8. Authentication

* - Hugging Face API key required to access IBM Granite models.
* - GitHub account used for uploading final project files.

9. User Interface

* - Tab 1 – Requirement Extraction (Upload PDFs).
* - Tab 2 – Prompt-to-Code Generator.
* - Tab 3 – Test Case Generator.
* - Tab 4 – Bug Fixing Assistant.
* - Tab 5 – Documentation Generator.

Runs fully inside Colab with a Gradio shareable link.

10. Testing

* - Unit Testing: Verified PDF parsing, prompt-to-code conversion.
* - Manual Testing: Tested end-to-end workflow in Colab.
* - Edge Cases: Checked invalid PDFs, empty prompts, long inputs.

11. Screenshots

* - Colab notebook running installation commands.
* - Gradio link generated inside Colab.
* - Requirement Extraction output.
* - Code generation result.
* - Test case output.
* - Bug fixing demonstration.

12. Known Issues

* - Colab session expires after ~12 hours.
* - Requires internet connection for Hugging Face models.
* - No persistent database (outputs need to be saved manually or to GitHub).

13. Future Enhancements

* - Store project history in Google Drive or Pinecone DB.
* - Add GitHub Actions integration for CI/CD.
* - Export generated reports as PDF/Word.
* - Support for multiple programming languages (Java, C++, etc.).
* - Role-based UI (developer, tester, manager).