# Project Documentation

## 1. Introduction

• Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle  
• Team Members:  
- Deion Daniel Bernard  
- Antony Staney M  
- Naveen M  
- Surya Kumar S

## 2. Project Overview

SmartSDLC leverages IBM Granite models from Hugging Face to accelerate the software development lifecycle. The project simplifies requirement gathering, code generation, bug fixing, testing, and documentation through AI automation. Users interact via Google Colab and Gradio interfaces, making the system easy to deploy and accessible for rapid prototyping and learning. It improves productivity, ensures accuracy, and enhances collaboration in software engineering workflows.

## 3. Architecture

The architecture of SmartSDLC is designed to provide seamless integration of AI-powered features into the software development lifecycle. It includes the following components:  
• Frontend (Gradio): Provides a user-friendly interface for uploading PDFs, entering prompts, and interacting with the AI assistant.  
• Backend (Google Colab Environment): Executes Python scripts, manages model interactions, and handles data processing.  
• LLM Integration (IBM Granite Models): Powers natural language understanding, requirement extraction, code generation, and bug detection.  
• Document Handling: Uses PyPDF2 for PDF processing and requirement parsing.  
• Testing Framework: Generates automated test cases and validates outputs.

## 4. Setup Instructions

1. Install Python 3.9 or later.  
2. Open Google Colab and create a new notebook.  
3. Change Runtime Type to T4 GPU.  
4. Install dependencies: !pip install transformers torch gradio PyPDF2.  
5. Import IBM Granite model from Hugging Face (granite-3.2-2b-instruct).  
6. Run provided code cells to initialize the project.  
7. Launch the Gradio interface for interaction.

## 5. Folder Structure

project\_root/ Main folder containing project files  
notebooks/ Google Colab notebooks for development and execution  
src/ Source code for requirement extraction, code generation, and testing  
models/ Pre-trained IBM Granite models or integration scripts  
docs/ Documentation files and guides  
tests/ Generated and custom test cases  
outputs/ Generated code, test reports, and documentation

## 6. Running the Application

To start the SmartSDLC project:  
1. Launch the Google Colab notebook.  
2. Set runtime to GPU and install required libraries.  
3. Run code cells to initialize the Granite model.  
4. Use the Gradio link to access the application.  
5. Upload project PDFs or prompts.  
6. Interact with the AI assistant to generate requirements, code, tests, and documentation.

## 7. API Documentation

POST /generate-requirements Extracts and summarizes requirements from uploaded PDFs.  
POST /generate-code Transforms requirements into Python code.  
POST /fix-bugs Analyzes and corrects code errors.  
POST /generate-tests Creates unit test cases for generated code.  
POST /chat Enables interaction with AI assistant for clarification and guidance.

## 8. Authentication

SmartSDLC runs in an open environment for demonstration purposes. Secure deployment may include:  
• Token-based authentication (JWT or API keys)  
• OAuth2 with IBM Cloud credentials  
• Role-based access control (admin, developer, tester)  
• User session management for tracking interactions

## 9. User Interface

The Gradio-based interface provides:  
• Upload functionality for PDFs  
• Input fields for prompts  
• Real-time response display for code, requirements, and tests  
• Tabs for bug fixing, documentation, and code generation  
• Option to download generated files (code/test docs)

## 10. Testing

Testing was performed in multiple phases:  
• Unit Testing: Verifying requirement extraction and code generation modules.  
• Integration Testing: Ensuring interaction between requirements, code, and tests.  
• Manual Testing: Uploading files, checking outputs, and validating corrections.  
• Edge Case Handling: Large PDFs, incomplete prompts, and malformed code inputs.

## 11. Screenshots

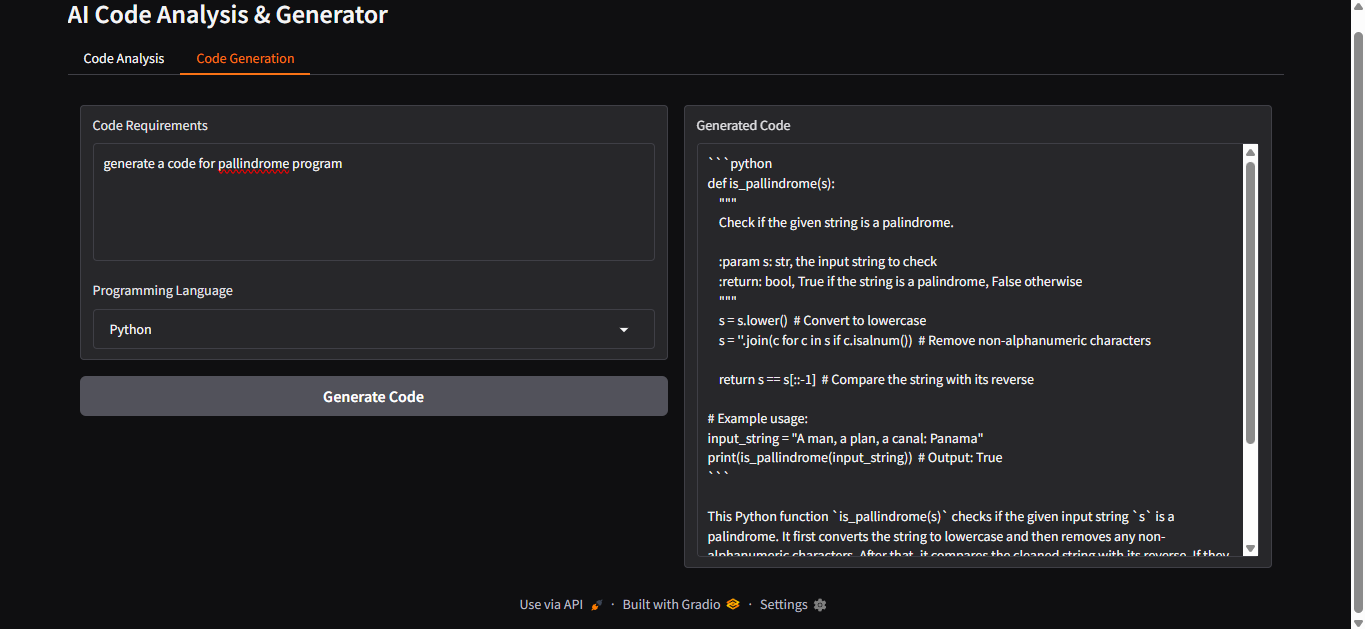


Figure 1: Code Generation Interface

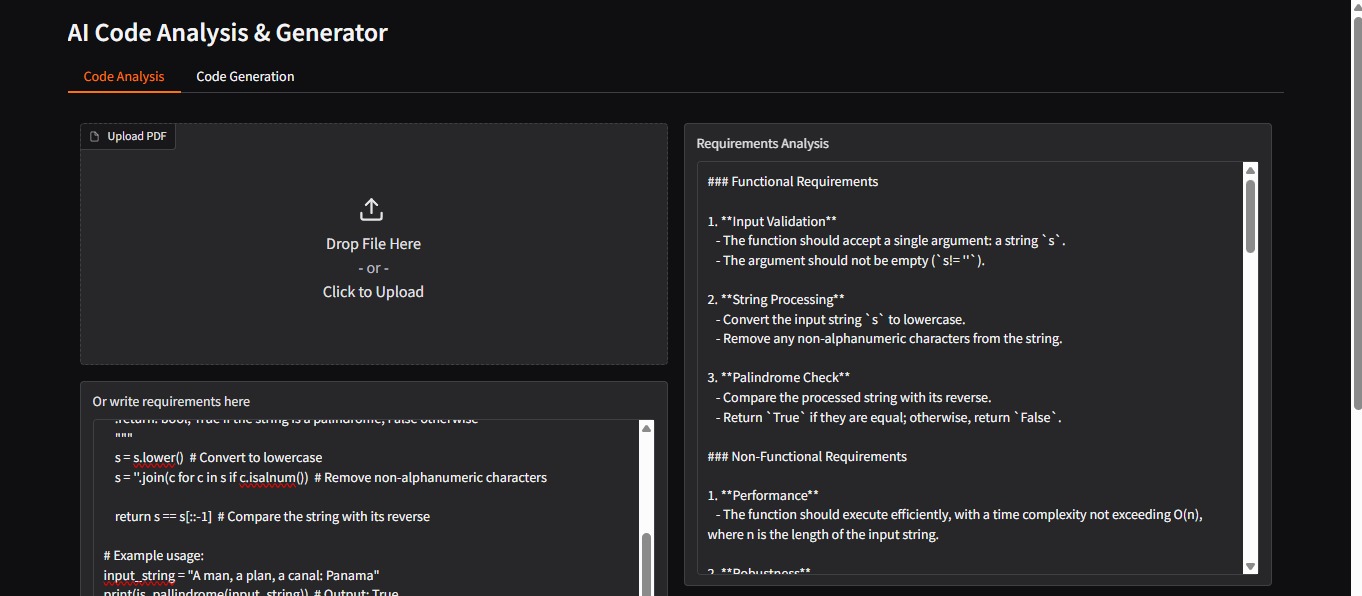


Figure 2: Requirements Analysis Interface

## 12. Known Issues

• Limited to Python code generation currently.  
• Dependency on Google Colab runtime availability.  
• Large files may slow down processing.  
• Requires stable internet for Hugging Face model integration.

## 13. Future Enhancement

• Expand support for multiple programming languages.  
• Add cloud-based deployment options (AWS, IBM Cloud, Azure).  
• Improve error handling and performance on large datasets.  
• Develop advanced visualization dashboards for tracking progress.  
• Integration with CI/CD pipelines for automated deployments.