

Project Documentation

1. Introduction

Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle

Team Member: RITVIK S SUKKAL

Team Member: PRAVEEN C

Team Member: RAGHUL H

Team Member: RAJESHKANNA G

2. Project Overview

Purpose

The purpose of SmartSDLC is to streamline the Software Development Life Cycle (SDLC) by automating two critical tasks:

1. Requirements Analysis – Extract and classify requirements from PDF documents or user inputs.
2. Code Generation – Translate natural language requirements into functional code snippets in multiple programming languages.

This tool empowers developers, project managers, and analysts by reducing manual effort, improving accuracy, and accelerating the software design process.

Features

Requirements Analysis

Extracts text from PDF files.

Organizes requirements into Functional, Non-Functional, and technical categories.

Code Generation

Generates AI-powered code snippets.

Supports languages such as Python, Java, JavaScript, C++, C#, PHP, Go, and Rust.

Interactive Gradio Interface

Tabbed design: “Code Analysis” and “Code Generation.”

Supports both file uploads and manual text input.

3. Architecture

Frontend (Gradio):

Provides an intuitive and responsive user interface with tabs for analysis and code generation.

Backend (PyTorch & Transformers):

Handles text processing, model inference, and output generation.

LLM Integration (IBM Granite 3.2 Instruct Model):

Utilizes ibm-granite/granite-3.2-2b-instruct for natural language understanding and generation.

PDF Processing (PyPDF2):

Extracts content from uploaded documents for requirement analysis.

4. Setup Instructions

Prerequisites Python 3.9+

pip (package manager)

Internet connection (to download models)

Run the application: `python smartsdlc.py`

5. Folder Structure

smartsdlc.py

README.md

SmartSDLC- Project Documentation

Demo video

6. Running the Application

1. Launch the App: `python smartsdlc.py`

2. Access the Gradio Interface:

- Code Analysis Tab → Upload a PDF or enter text, click Analyze, and view categorized requirements.
- Code Generation Tab → Enter a requirement, choose a programming language, and click Generate Code.

3. If share=True, a public link will be generated for remote access.

7. User Interface

Code Analysis Tab:

- Upload PDF or enter text.
- Displays structured requirement analysis.

Code Generation Tab:

- Enter requirement text.
- Choose programming language.
- Displays generated code snippet.

8. Testing

○ Unit Testing:

Verified core functions such as PDF extraction, requirement classification, and code generation.

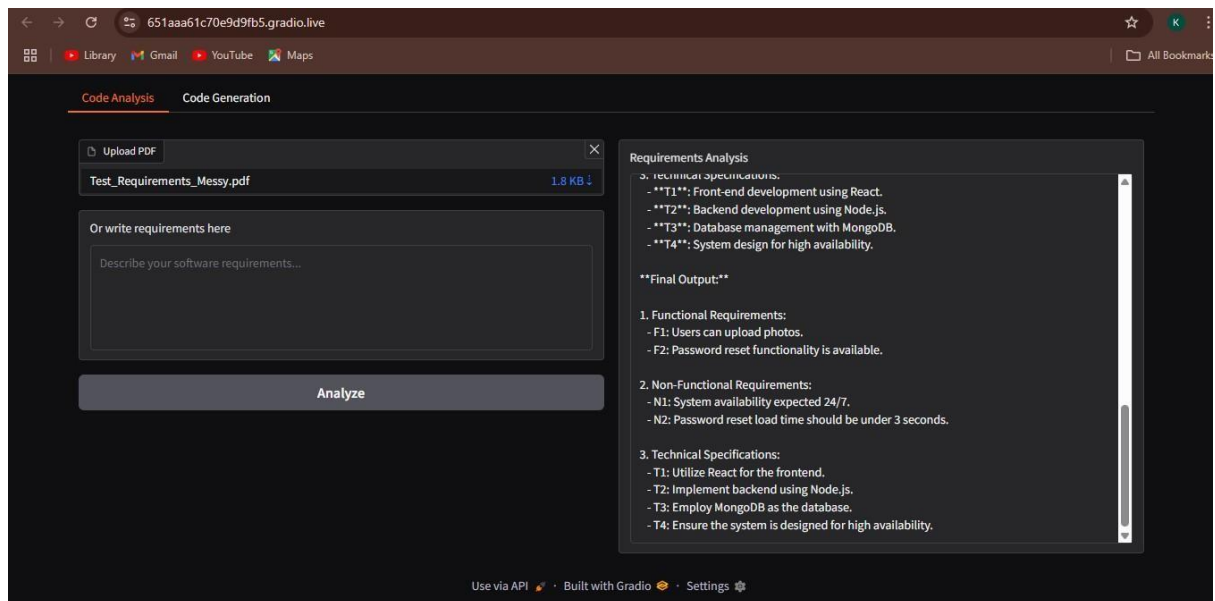
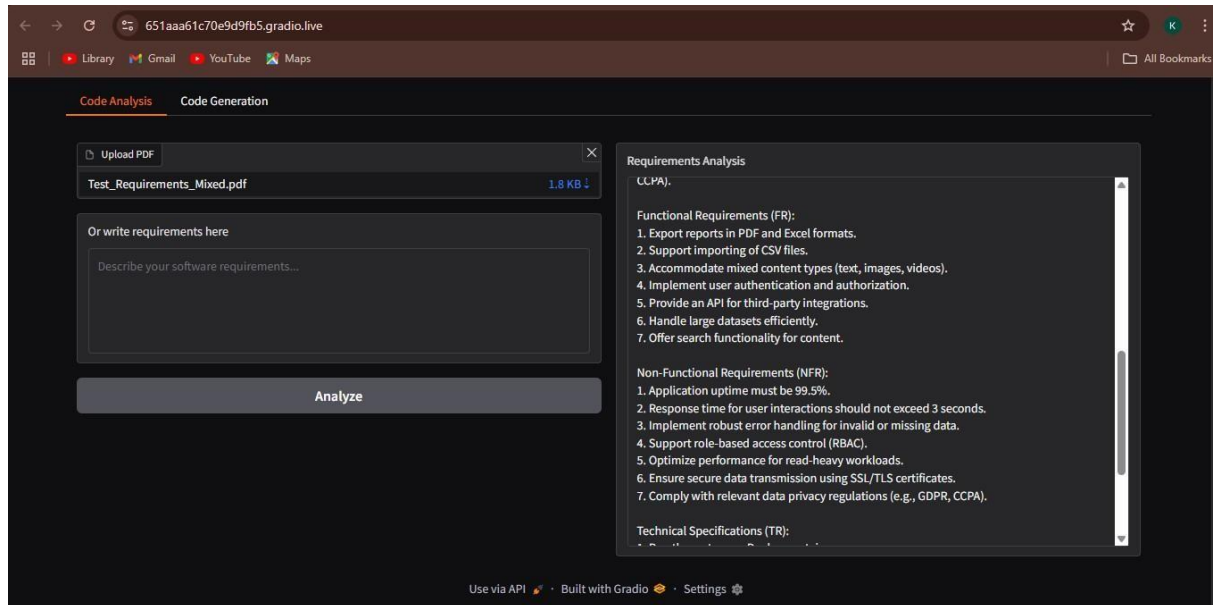
○ Functional Testing:

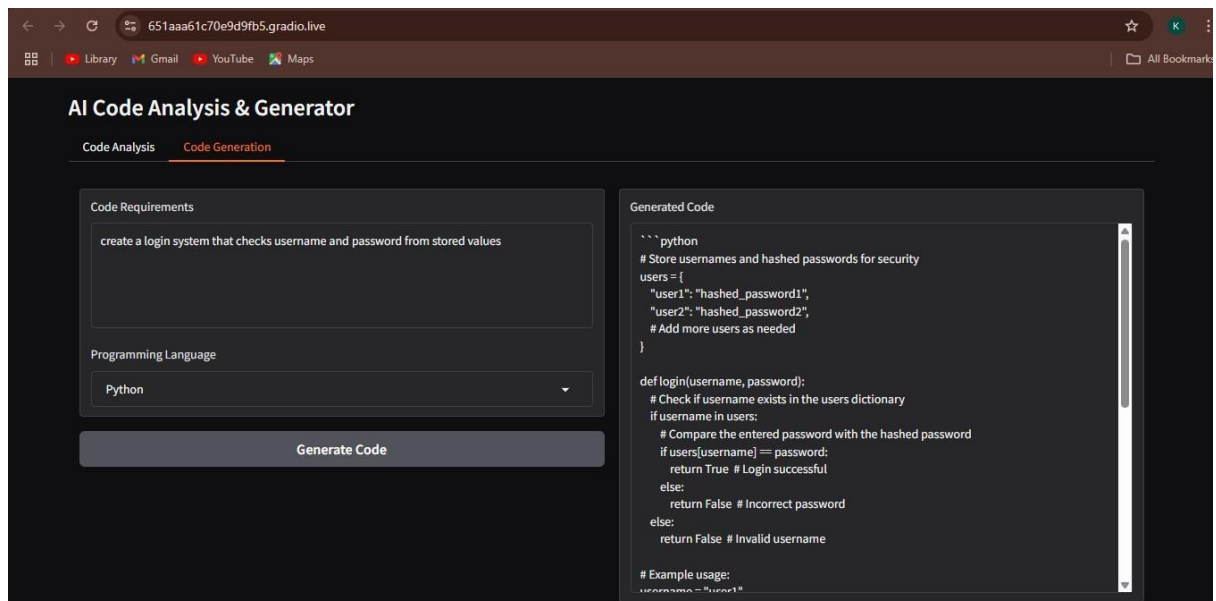
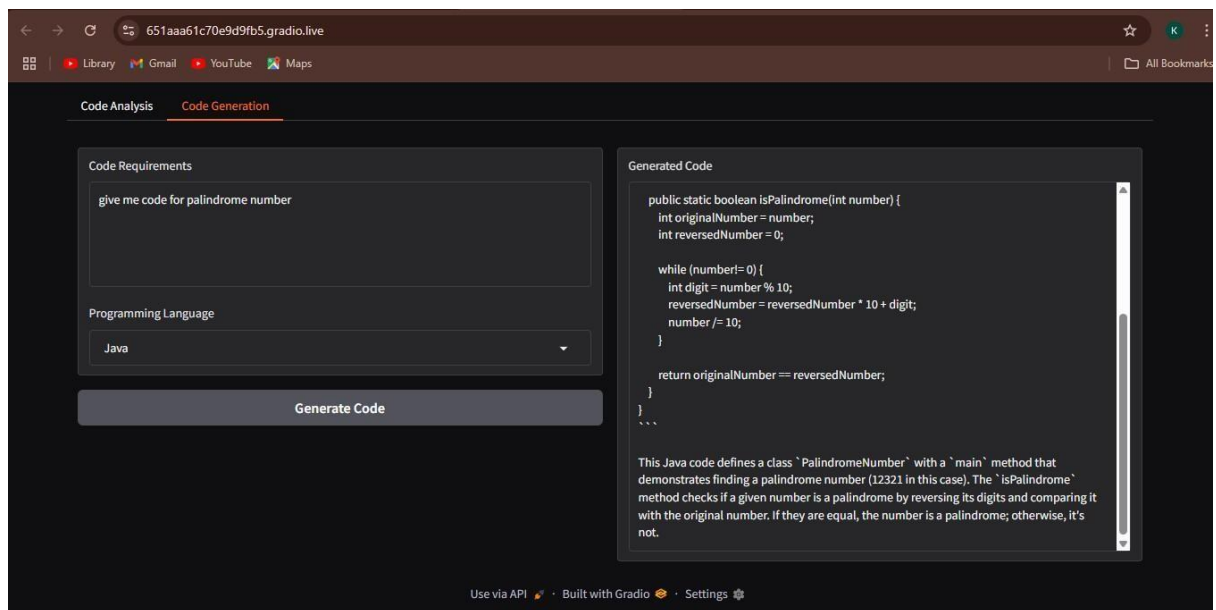
Tested end-to-end flows using sample PDFs and requirement prompts.

○ Manual Testing:

Validated Gradio interface usability and accuracy of generated outputs.

9.Screenshots





10. Known Issues

- Generated code may require manual refinement.
- Large PDFs may slow down processing.
- Model accuracy depends on prompt clarity.

11. Future Enhancements

- Add export functionality (Word, PDF, JSON).
- Integrate versioning for generated requirements.
- Expand supported programming languages.
- Introduce authentication for secure deployments.