

# SmartSDLC – AI-Enhanced Software Development Lifecycle

## Project Documentation

### 1. Introduction

- Project Title: SmartSDLC – AI-Enhanced Software Development LifecycleTeam Members:
  - Member 1: S. Jeya
  - Member 2: P. Gopika
  - Member 3: B. Kowsalya

#### Introduction:

Software development is a multi-phase process that begins with requirements gathering and ends with deployment. A critical challenge faced by developers is accurately interpreting requirements and transforming them into functional code. Traditional methods involve manual documentation review and coding, which is time-consuming and error-prone.

The AI Code Analysis & Generator project aims to automate this process by combining Natural Language Processing (NLP) with Large Language Models (LLMs). The system accepts requirements (from PDFs or text), organizes them into structured categories, and generates corresponding source code in the desired programming language.

### 2. Project Overview

- Purpose:  
The primary purpose of this project is to create a smart AI assistant that:
  - Simplifies requirement analysis
  - Reduces manual effort in software documentation
  - Speeds up prototype development through code generation
- Objectives:
  - Provide an easy-to-use web interface for non-technical and technical users
  - Enhance requirement analysis with AI-based categorization
  - Support multiple programming languages for code generation
  - Encourage faster software prototyping in academic and professional settings
- Features:

## 1. Requirement Analysis

- AI categorizes requirements into:
  - *Functional requirements* (features and behaviors)
  - *Non-functional requirements* (performance, usability, security, etc.)
  - *Technical specifications* (tools, technologies, standards)

## 2. Code Generation

- AI converts requirement statements into working code snippets in multiple languages
- Currently supports: Python, JavaScript, Java, C++, C#, PHP, Go, Rust

## 3. PDF Document Processing

- Accepts uploaded PDFs
- Extracts text automatically for further analysis

## 4. Interactive Gradio Web UI

- Tab-based interface
- *Code Analysis Tab* → Upload/enter requirements and get AI analysis
- *Code Generation Tab* → Enter requirement + select language → Generate code

## 3. Architecture

The architecture follows a modular layered design:

- Frontend (Gradio):
  - Provides interactive, tabbed user interface
  - Built with `gr.Blocks()`
  - Handles file uploads, text inputs, and displays outputs
- Backend (Hugging Face Transformers + PyTorch):
  - Model: `ibm-granite/granite-3.2-2b-instruct`
  - Provides natural language understanding & code generation
  - Runs inference on GPU if available (for speed)
- PDF Processing (PyPDF2):
  - Reads PDF pages

- Extracts plain text for NLP analysis
- Core Modules:
  - `generate_response()` → Model inference wrapper
  - `extract_text_from_pdf()` → PDF reader
  - `requirement_analysis()` → Requirement classification logic
  - `code_generation()` → Generates code snippets
- Data Flow:
  1. Input (PDF/Text) →
  2. Preprocessing (Tokenizer / PDF Reader) →
  3. Model (LLM for analysis or code generation) →
  4. Output (Textbox in Gradio UI)

## 4. Setup Instructions

### Prerequisites:

- Platform: Google Colab (recommended) or local Python environment
- Python Version: 3.9+
- Libraries:
  - `pip install gradio torch transformers PyPDF2`
- Internet: Required to download Hugging Face model

### Installation Steps (Google Colab):

1. Open Colab notebook
2. Paste the project code into a new cell
3. Run the cell → Model + dependencies install automatically
4. Colab generates a public Gradio link → Open in browser

## 5. Folder Structure

project-root/

```
| — app.ipynb          # Colab notebook containing code
| — requirements.txt    # Optional dependency file
| — README.md          # Documentation
```

| — /data # (Optional) Folder for storing PDFs

## 6. Running the Application

1. Start Colab runtime and run the notebook
2. Gradio launches and generates a public URL
3. Navigate to the link and use tabs:
  - Code Analysis Tab:
    - Upload a PDF or type requirements
    - Output: AI-generated categorized requirements
  - Code Generation Tab:
    - Enter requirement + select programming language
    - Output: AI-generated code snippet

## 7. API Documentation (Internal Functions)

- `generate_response(prompt, max_length)`
  - Input: Prompt string
  - Output: AI-generated text
- `extract_text_from_pdf(pdf_file)`
  - Input: PDF file
  - Output: Extracted plain text
- `requirement_analysis(pdf_file, prompt_text)`
  - Input: PDF or text
  - Output: Categorized requirements
- `code_generation(prompt, language)`
  - Input: Requirement + Language choice
  - Output: Code snippet in selected language

## 8. Authentication

- Current version is open-access for academic demo
- Future deployments may include:

- JWT tokens for secure API usage
- Role-based access (Admin, Developer, User)
- API keys for Hugging Face model usage

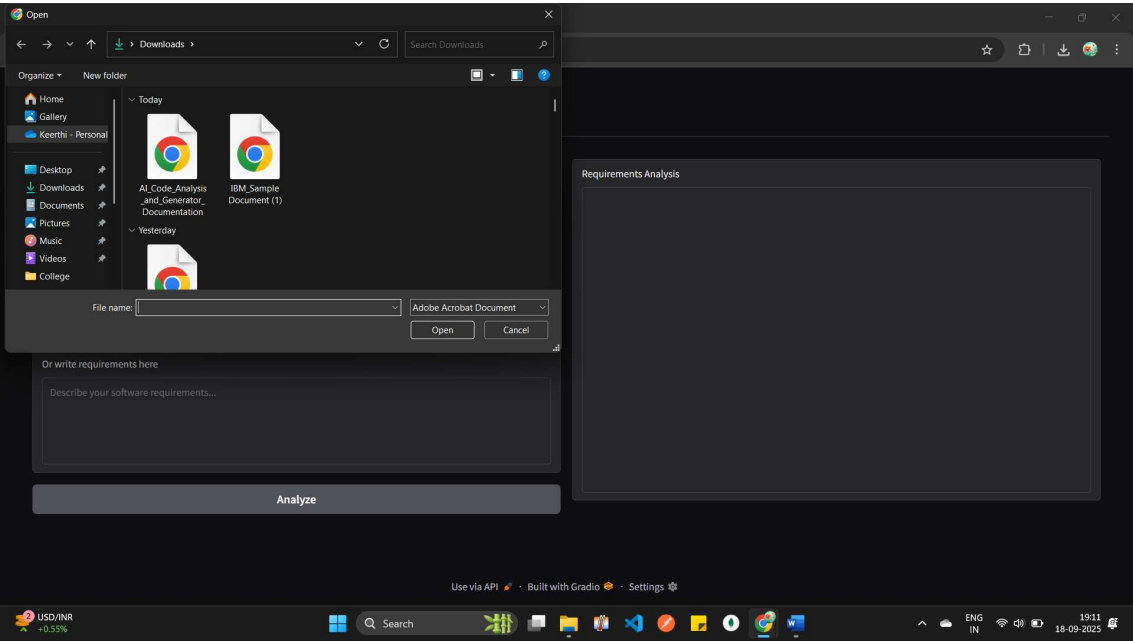
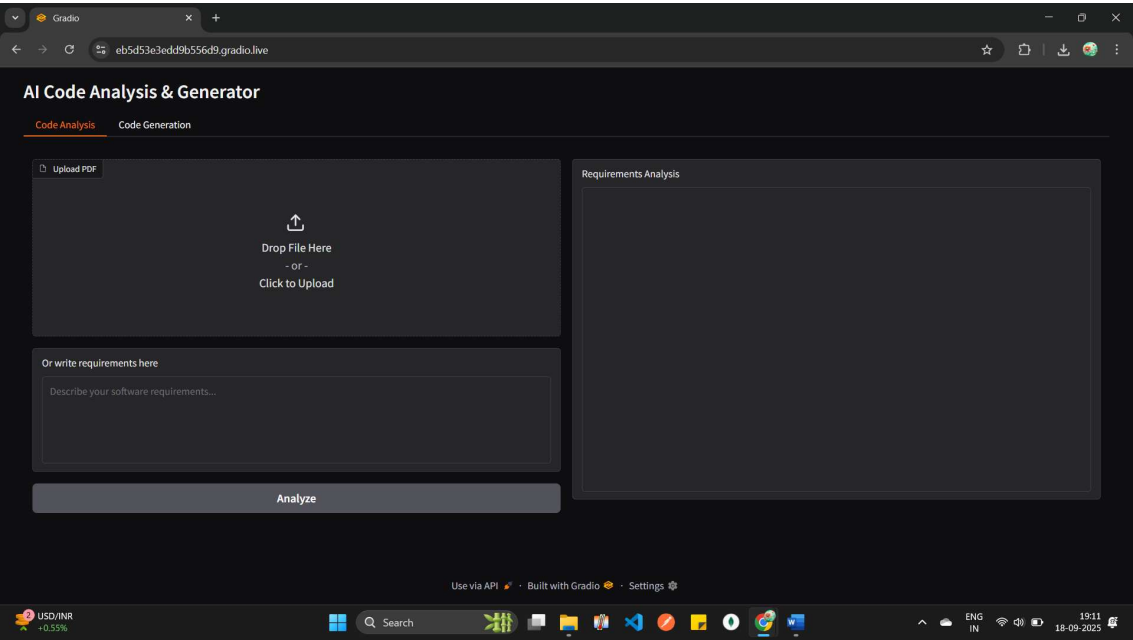
## **9. User Interface**

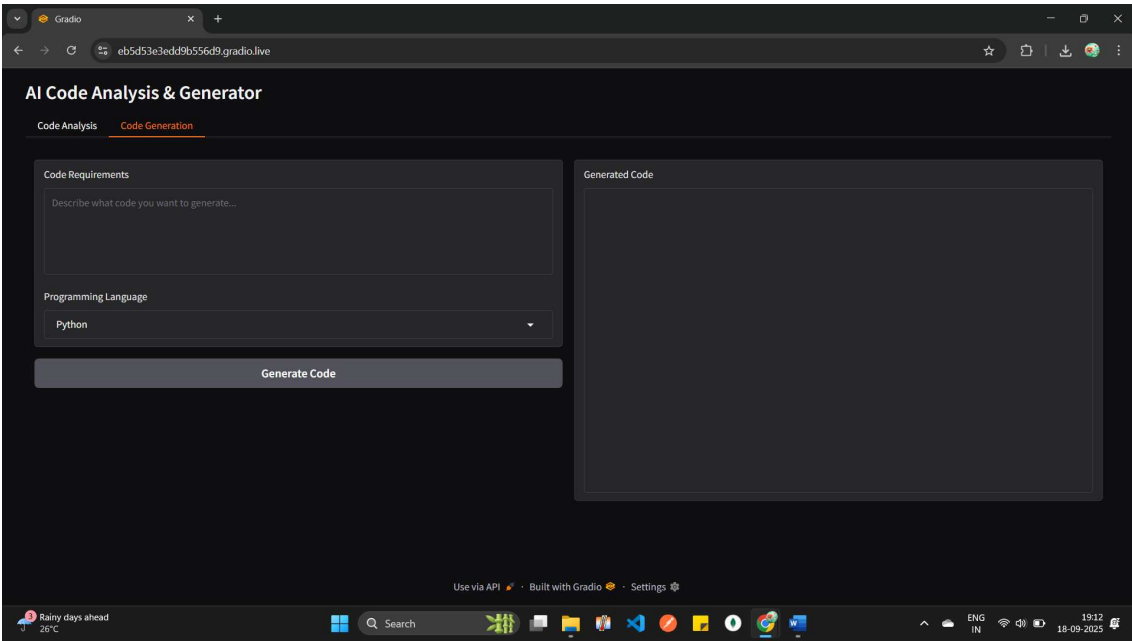
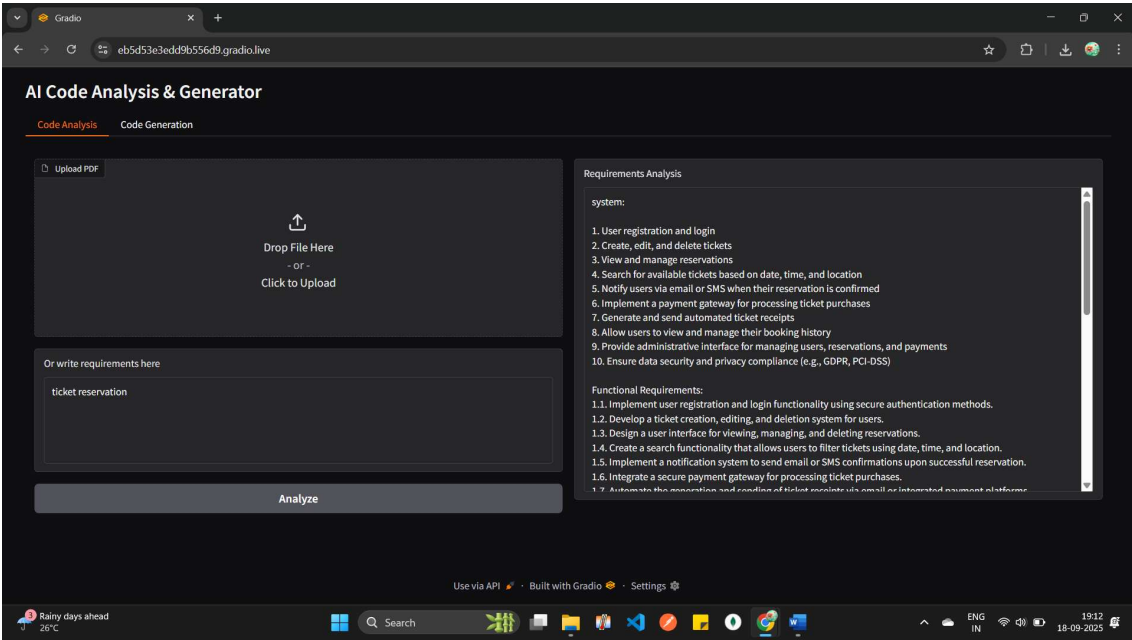
- Design Goals: Minimalist, tab-based, accessible for non-technical users
- Components:
  - Sidebar navigation (Gradio tabs)
  - File upload and text inputs
  - Drop-down menu for language selection
  - Real-time outputs for requirements and code

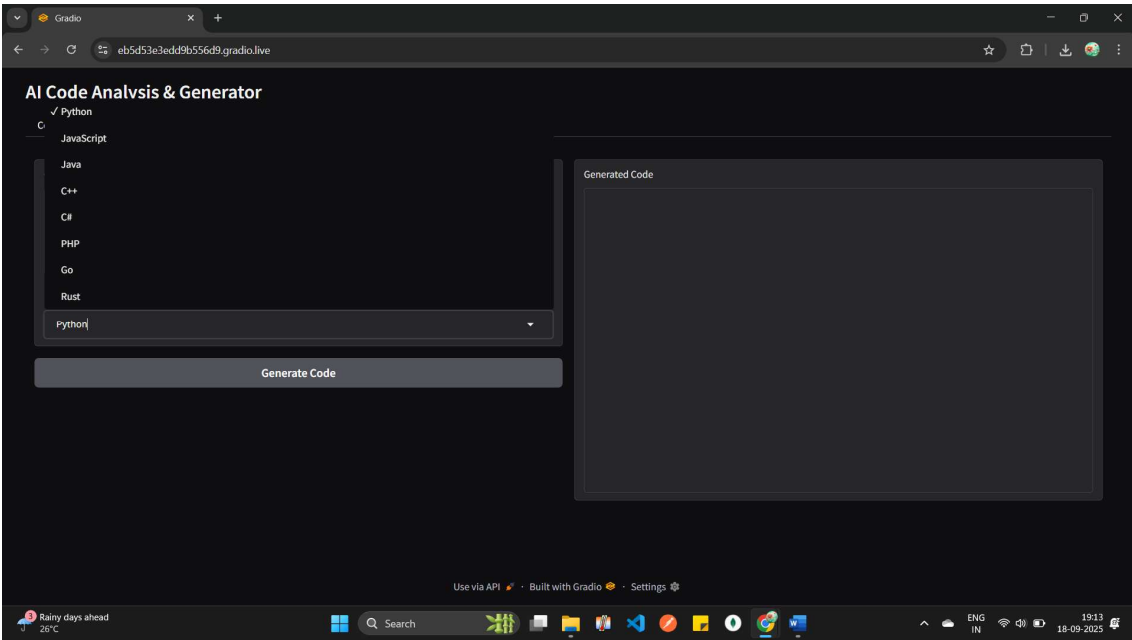
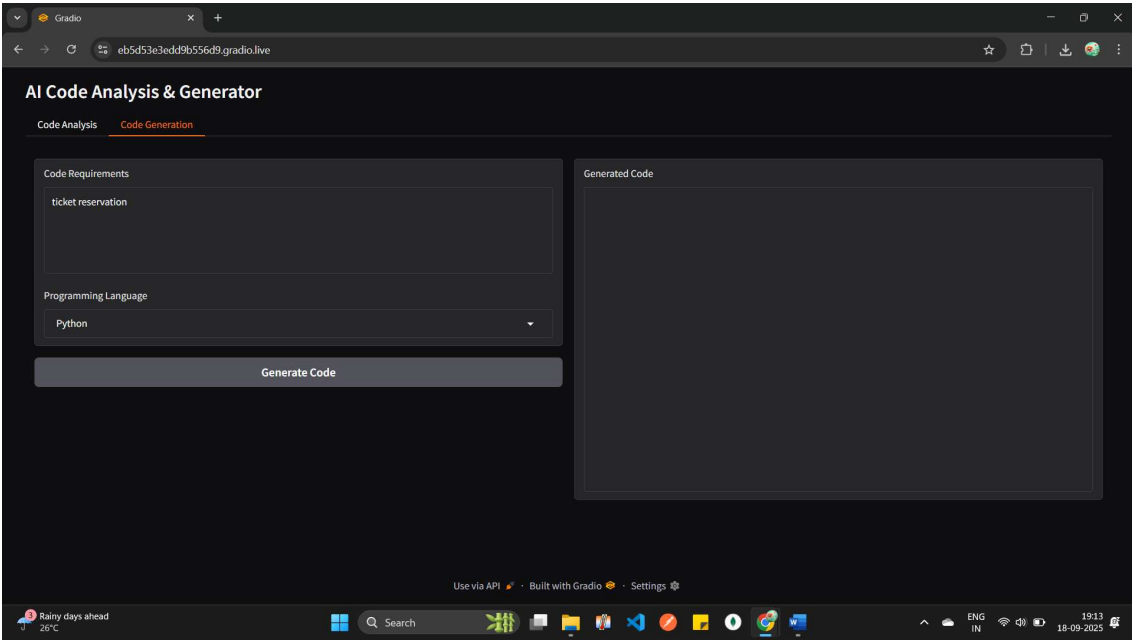
## **10. Testing**

- Unit Testing:
  - PDF extraction functions
  - AI prompt generation
- API Testing:
  - Hugging Face model responses
  - Edge cases (long/empty input)
- Manual Testing:
  - File upload + text input in Gradio
  - Code generation correctness

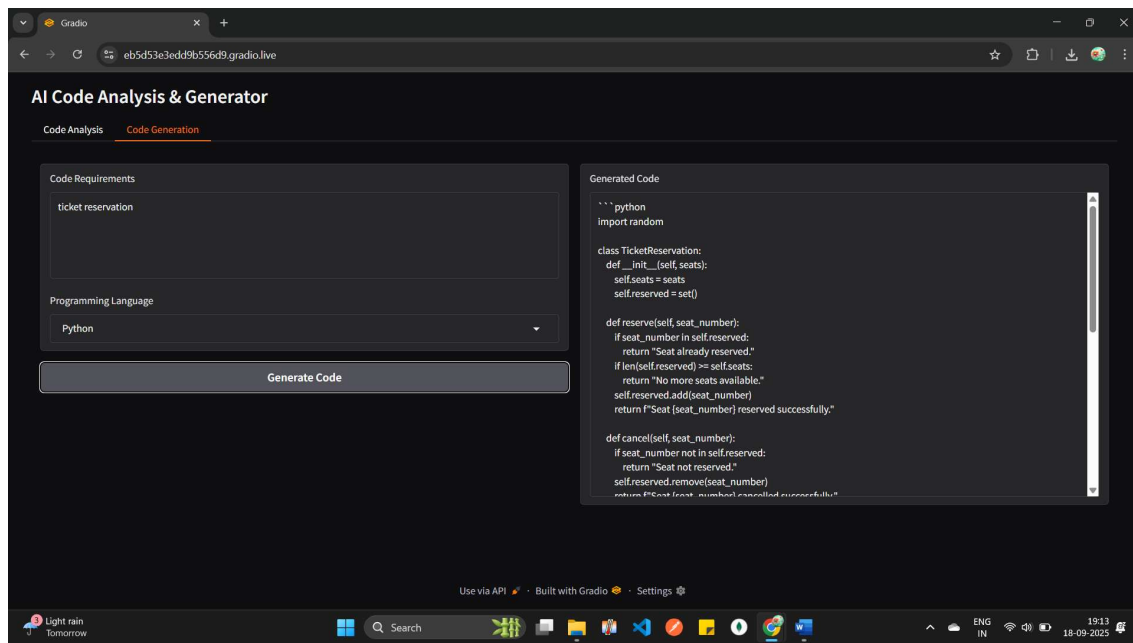
11. Screenshots











## 12. Known Issues

- Slow model loading (~500MB download on first run in Colab)
- Performance varies with document size
- Limited handling of complex/multi-page PDFs
- Generated code may require manual debugging

---

## 13. Future Enhancements

- Add database storage for past analysis results
- Extend support to Word (.docx) and Excel (.xlsx) documents
- Improve code generation with error handling and explanations
- Add multi-turn conversational capability (chatbot-style assistant)
- Integrate with project management tools like Jira or Trello