**Project Title**

**Smart SDLC – AI Enhanced Software Development Lifecycle**

**Project Documentation**

# Introduction

* Project Title : Smart SDLC – AI Enhanced Software Development Lifecycle
* Team member :DEEPA.S
* Team member :MALATHI.R
* Team member :KAVIYA.S
* Team member :KAVIYA.S

# Project Overview

* Purpose : The purpose of Smart SDLC is to transform the traditional Software Development

Lifecycle into an AI-driven, intelligent process. By leveraging machine learning, natural language processing, and automation, Smart SDLC enhances every stage of development—from requirement analysis to deployment and maintenance. It ensures faster delivery, improved accuracy, reduced human error, and adaptive project management. The system acts as a virtual assistant for developers, testers, and project managers, helping them analyze requirements, generate code snippets, detect anomalies, predict risks, and automatically document processes.

* Features:
* **Requirement Analysis Assistant:** Automatically extracts, categorizes, and validates functional and non-functional requirements from project documents.
* **AI-driven Design Suggestions:** Generates UML diagrams, workflows, and architecture recommendations based on requirements.
* **Automated Code Generation:** Creates initial code snippets in multiple programming languages from structured requirements.
* **Policy & Document Summarization:** Summarizes lengthy requirement docs, design docs, or compliance policies into concise points.
* **Forecasting & Risk Prediction:** Predicts project delays, effort estimation, and risk factors using historical data.
* **Anomaly Detection in Development:** Identifies unusual patterns in testing, commits, or deployment logs.
* **Continuous Feedback Loop:** Collects developer and tester feedback to refine processes and ensure iterative improvement.
* **Multimodal Input:** Supports text, PDFs, CSVs, and requirement templates for analysis.
* **Interactive Dashboard (Gradio/Streamlit):** Provides a user-friendly interface for requirement analysis, code generation, and project monitoring.

# Architecture

Frontend (Streamlit/Gradio): Provides dashboards, requirement input forms, and interactive tabs for analysis, code generation, and project monitoring. Supports real-time collaboration for project stakeholders.

Backend (FastAPI): Handles requirement extraction, code generation, report creation, and anomaly detection. Optimized for asynchronous operations and easy integration with external APIs. LLM Integration (Granite/Transformers): Uses AI language models for natural language understanding, requirement analysis, and code generation.

Vector Database (Pinecone/FAISS): Stores embeddings of requirement documents and enables semantic search.

ML Modules: Implements forecasting models, risk prediction, and anomaly detection using scikit-learn and pandas.

# Setup Instructions

* Install Python 3.9 or later
* Install dependencies from requirements.txt
* Configure API keys and environment variables in .env
* Run backend server with FastAPI
* Launch frontend with Streamlit/Gradio
* Upload requirement documents and interact with AI modules

# Folder Structure

* app/ – FastAPI backend logic including routes and services
* app/api/ – API endpoints for requirement analysis, code generation, and anomaly detection
* ui/ – Streamlit/Gradio frontend components
* requirement\_analyzer.py – Extracts and classifies requirements
* code\_generator.py – Generates code snippets from structured requirements
* risk\_forecaster.py – Predicts risks and estimates project effort
* anomaly\_detector.py – Detects unusual patterns in project logs
* report\_generator.py – Creates AI-generated project reports

# Running the Application

* Start FastAPI backend server
* Run Streamlit/Gradio frontend
* Navigate through requirement analysis, code generation, and project forecasting tabs
* Upload documents and view outputs like requirement lists, generated code, and reports
* All interactions are real-time with AI backend

# API Documentation

* POST /analyze-requirements – Extracts and categorizes requirements
* POST /generate-code – Generates code in chosen language
* POST /upload-doc – Uploads and embeds project documents
* GET /search-docs – Returns semantically similar documents
* POST /predict-risk – Provides risk forecasting results
* POST /detect-anomaly – Identifies anomalies in testing or deployment logs

# Authentication

For secure deployment, Smart SDLC can integrate token-based authentication (JWT), OAuth2 for user logins, and role-based access (admin, developer, tester, manager). This ensures controlled access and project data confidentiality.

# User Interface

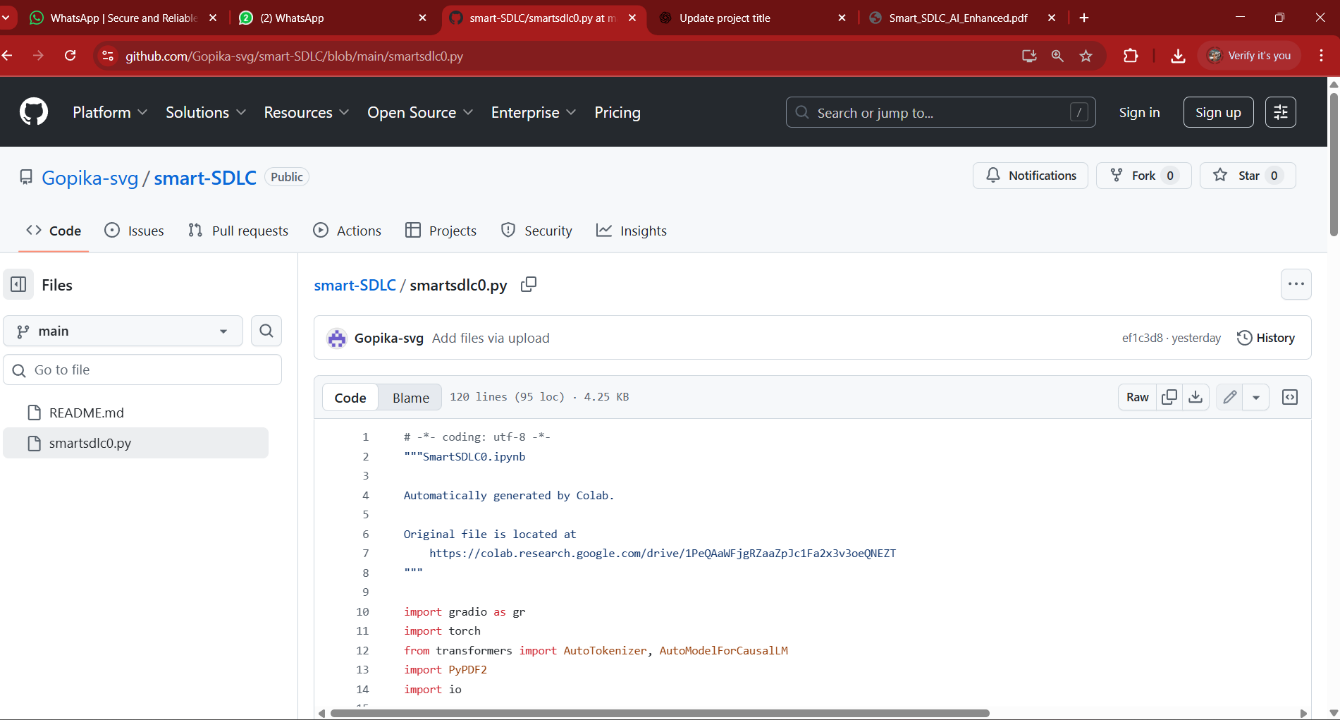
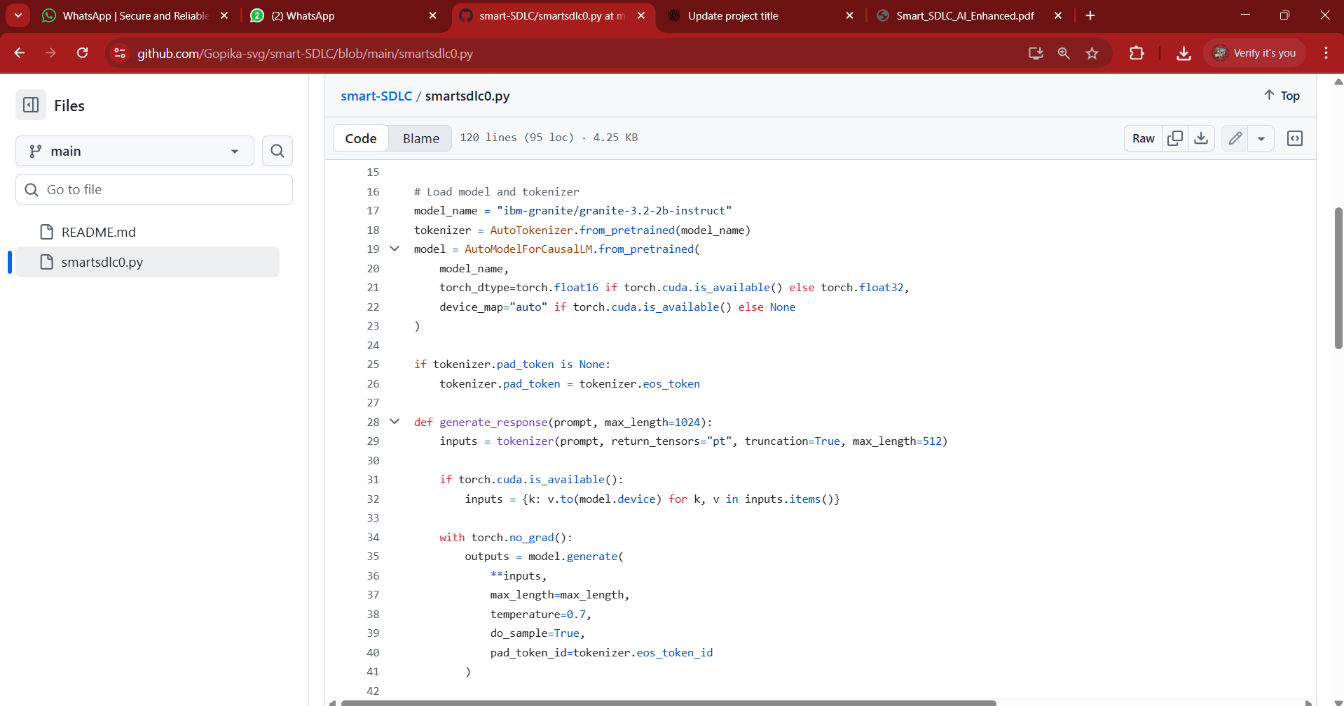
The interface is minimal and intuitive. It includes sidebar navigation, requirement visualizations, tabbed layouts for analysis, code generation, and forecasting. Supports real-time collaboration and PDF export of requirement analysis and reports.

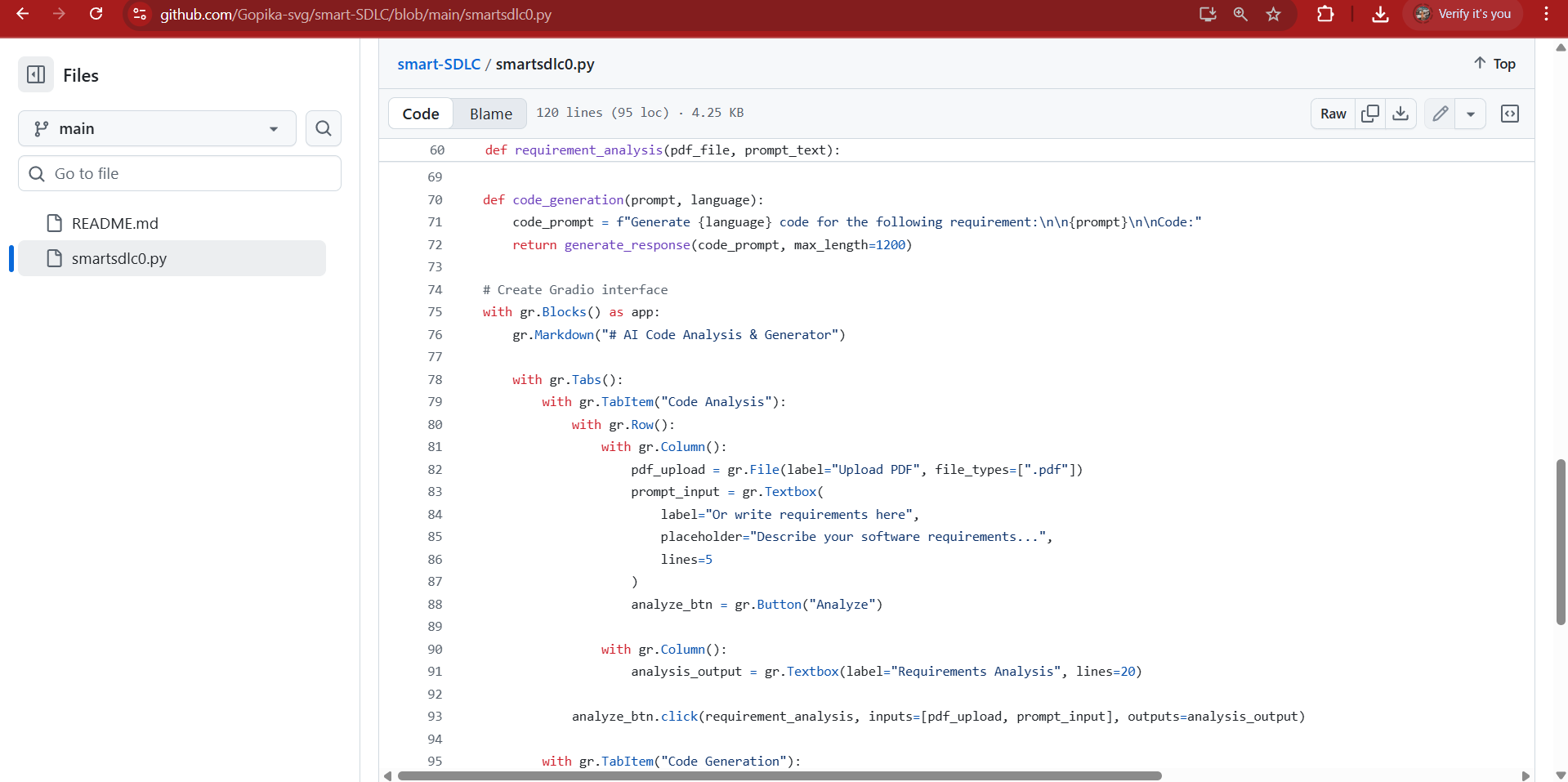
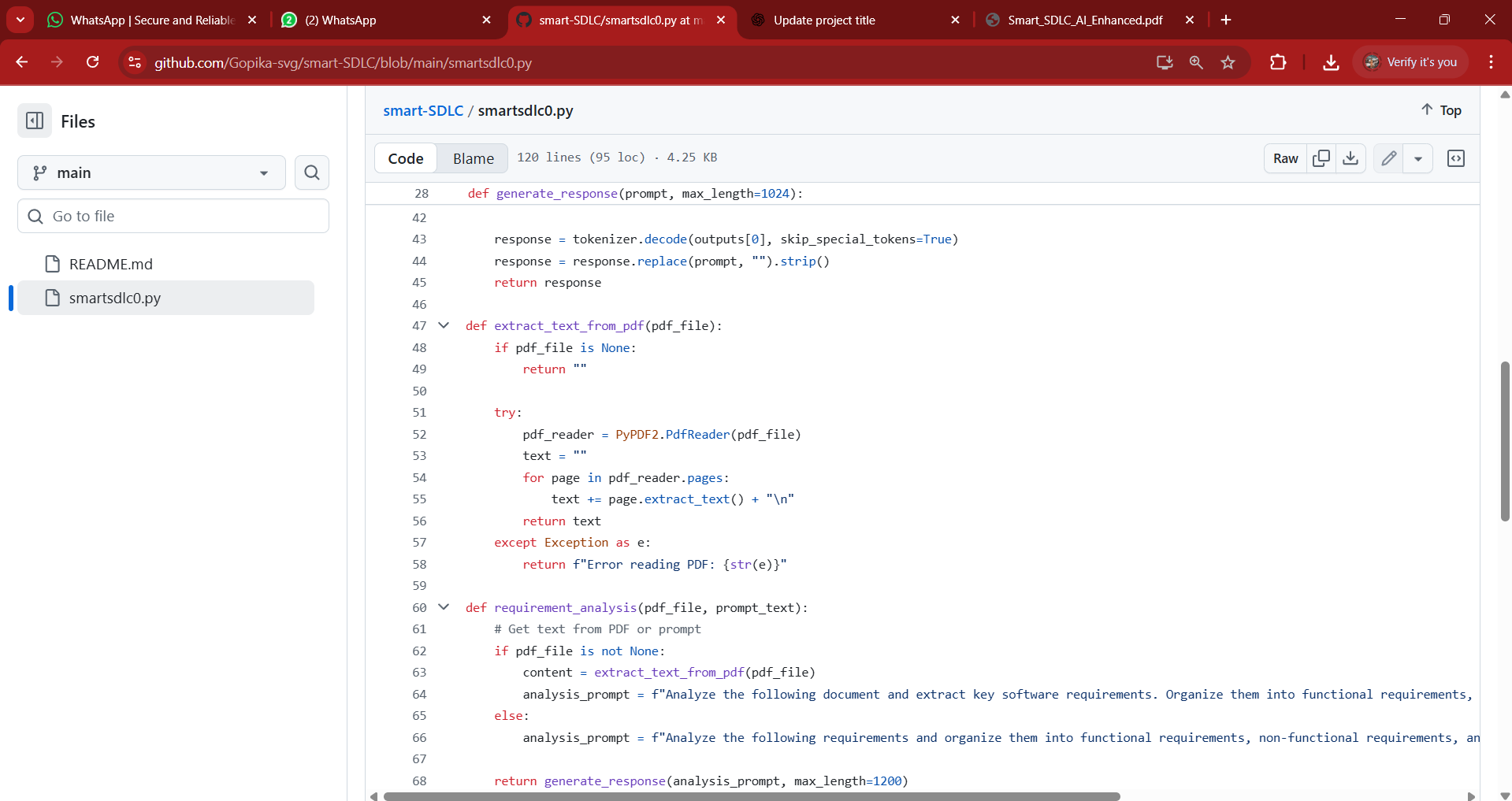
# Testing

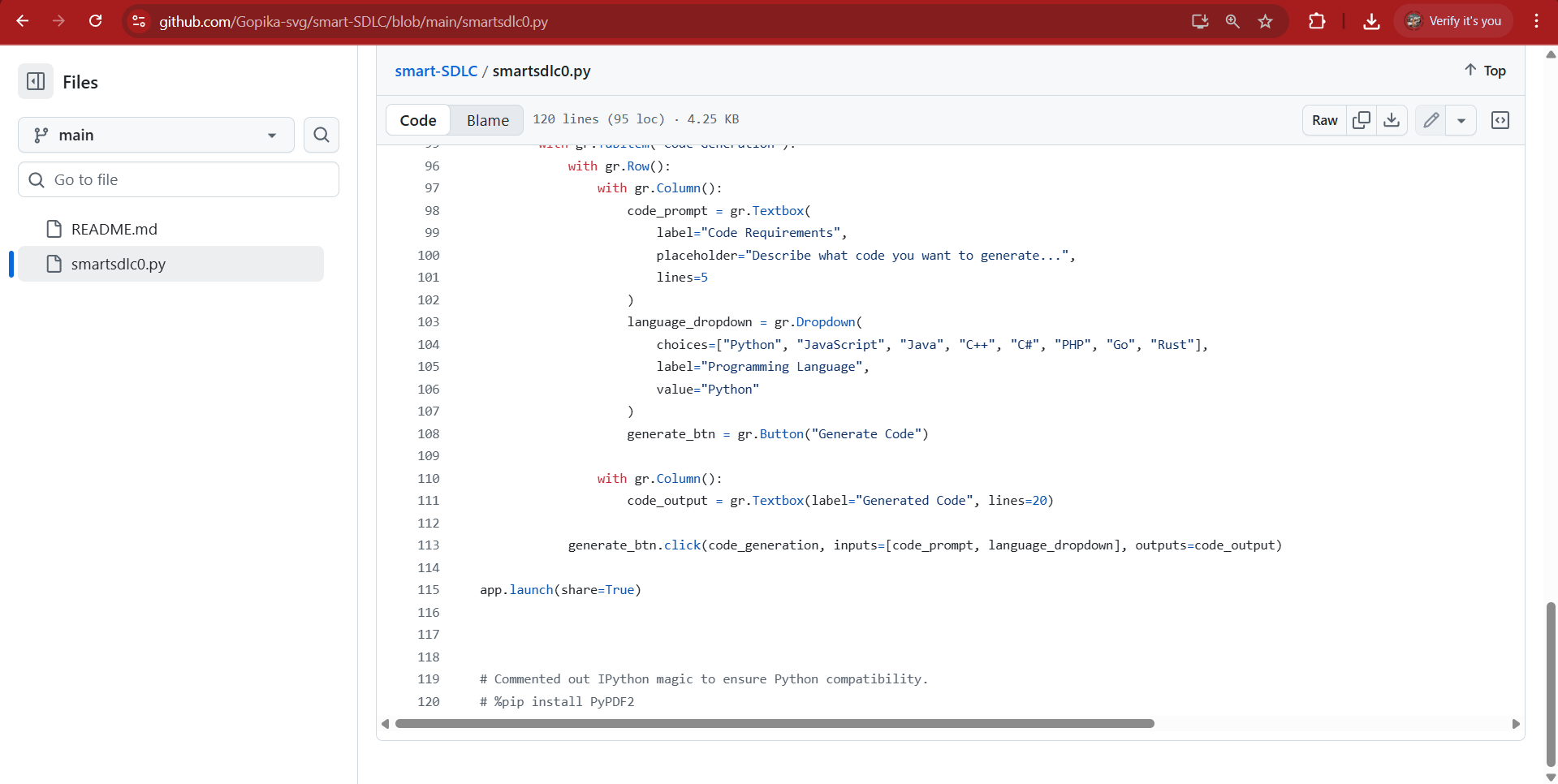
* Unit Testing – For requirement extraction and utility functions
* API Testing – Using Swagger UI, Postman, and pytest
* Manual Testing – For file uploads, code generation, and forecasting outputs
* Edge Case Handling – Large documents, malformed inputs, and missing fields

# Screenshots

Screenshots of the Smart SDLC dashboard, requirement analysis, and code generation pages can be placed here.

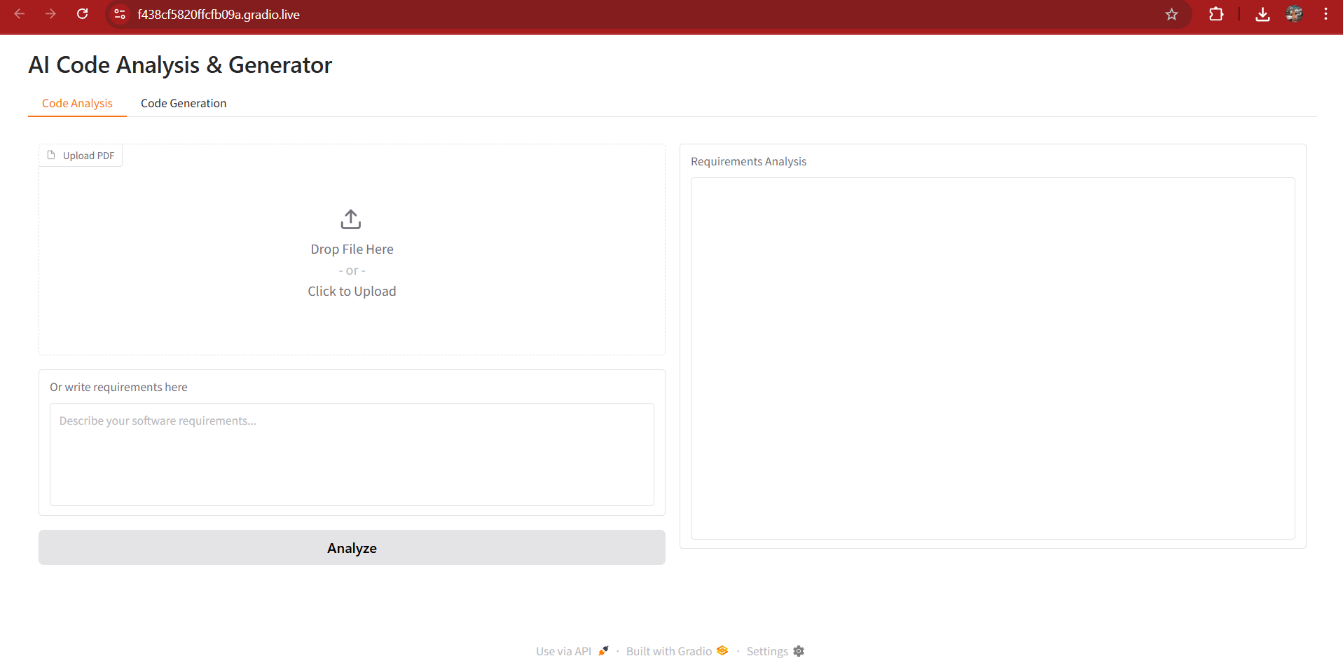






**OUTPUT:**

****

****

# Known Issues

* Limited accuracy for very ambiguous requirements
* Initial code generation may require developer refinement
* Forecasting depends on availability of historical project data
* Large file uploads may impact performance

# Future Enhancements

* Integration with popular project management tools (Jira, Trello, GitHub)
* Advanced visualization of project timelines and risks
* Support for additional programming languages
* Real-time collaboration with version control integration