Project Documentation

# 1. Project Title

Smart SDLC – AI Enhanced Software Development Lifecycle

# 2. Project Overview

## Purpose:

To create an AI-driven framework for improving the software development lifecycle (SDLC). By using AI at every stage — planning, requirements, design, coding, testing, deployment, and maintenance — the system reduces errors, improves productivity, predicts risks, and accelerates delivery. It acts as a decision-support and automation engine for development teams.

## Features:

* AI-driven Requirement Analysis – Extracts and refines requirements using NLP and knowledge bases.
* Automated Design Suggestions – Uses AI to recommend optimal architectures and patterns.
* Smart Code Generation – Generates code snippets and unit tests based on requirements.
* Predictive Risk Management – Forecasts project delays, budget overruns, and defect hotspots.
* Continuous Quality Monitoring – Applies anomaly detection and automated testing throughout SDLC.
* KPI Dashboard – Shows productivity, defect density, and deployment readiness.
* Feedback Loop – Gathers developer and stakeholder feedback for iterative improvements.

# 3. Architecture

Frontend (Streamlit or Gradio): Interactive dashboard for project managers and developers to view AI-generated insights, upload documents, and see KPIs.

Backend (FastAPI): Processes uploaded project artifacts, requirements, test cases, and code files. Serves as API hub for AI modules.

AI/LLM Integration: Uses a large language model for natural language understanding, summarization, and code generation.

Vector Search Database: Stores project documents, requirements, and test cases for quick semantic search.

ML Modules: Forecasting project timelines, anomaly detection in code metrics, automatic test case generation.

# 4. Setup Instructions

* Python 3.9+
* API keys for AI/LLM and vector database
* Install dependencies
* Launch FastAPI backend and Streamlit dashboard

# 5. Folder Structure

app/ – Backend logic (FastAPI routes, AI modules)  
ui/ – Streamlit dashboard components  
ai\_modules/ – Requirement analyzer, code generator, test predictor  
data/ – Uploaded requirements, test cases, code snippets  
smart\_sdlc\_dashboard.py – Main entry point

# 6. Running the Application

* Start FastAPI server
* Run Streamlit dashboard
* Upload requirements or code files
* Generate analysis, test cases, or reports

# 7. API Documentation

* POST /requirements/analyze – Extracts functional and non-functional requirements
* POST /generate/code – Generates code snippets/test cases
* GET /kpi/dashboard – Retrieves project health metrics
* POST /submit/feedback – Collects user input

# 8. Authentication

JWT or OAuth2 with project credentials, role-based access for developers, QA, and managers.

# 9. User Interface

Minimalist dashboard with tabs for:

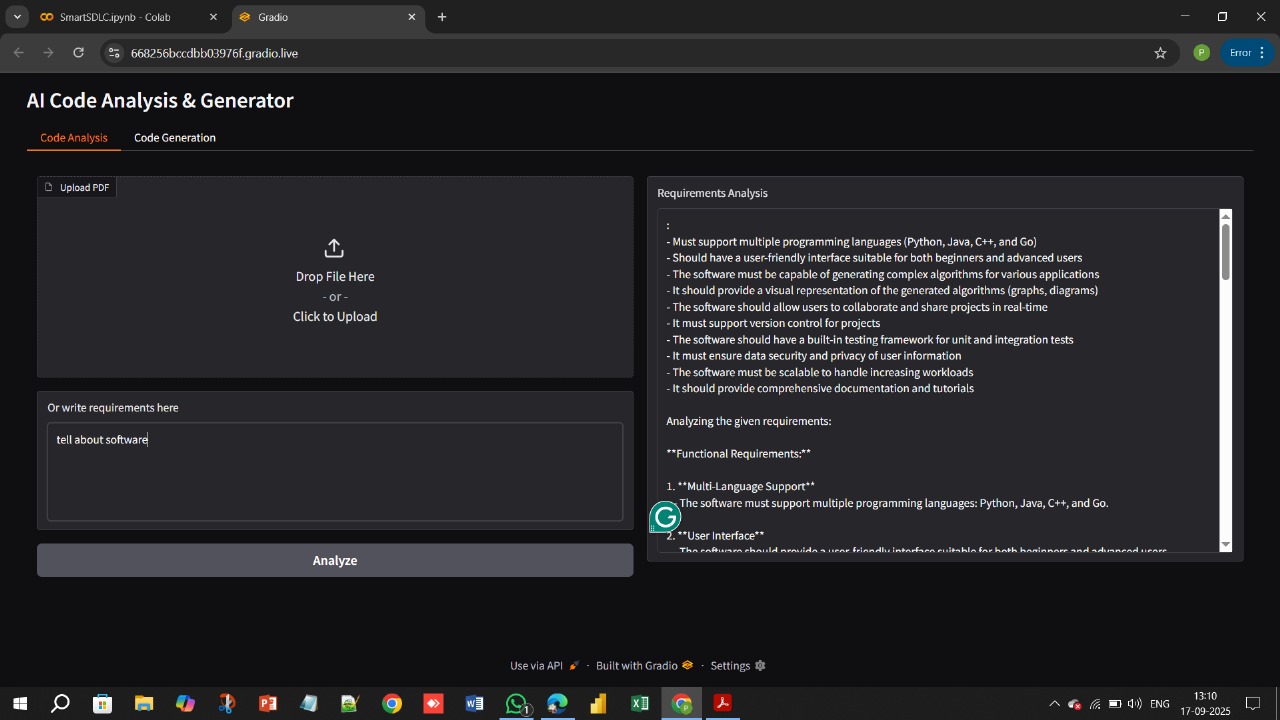
* Requirements Analysis
* Code Generation
* KPI Visualization
* Risk Forecasting

# 10. Testing

* Unit testing for AI modules
* API testing via Swagger/Postman
* Manual testing of dashboards and uploads
* Edge case handling (malformed requirements, missing code snippets)

11.Screenshot

# 



# 12. Known Issues

Limited accuracy on poorly written requirements; AI may generate incorrect code that needs human review.

# 13. Future Enhancements

* Deeper integration with CI/CD pipelines
* Automated security testing
* Multi-language code generation
* Real-time project progress alerts