

Project Documentation

ON

SmartSDLC – AI-Enhanced Software Development Lifecycle

By

Team ID: LTVIP2025TMID31232

Team Size:4

Team Leader: Ranga Loknath

Team member: Mohammad Ismail Shaik

Team member: M vardhan

Team member: Kuruba Roopavathi

**SmartSDLC – AI-Enhanced Software Development Lifecycle**

**1. Introduction**

SmartSDLC is a full-stack, AI-powered platform designed to automate and enhance traditional Software Development Lifecycle (SDLC) processes using cutting-edge Natural Language Processing (NLP) and Generative AI technologies.  
The platform seamlessly transforms unstructured requirements into code, test cases, bug fixes, and documentation, thus minimizing manual intervention, improving accuracy, and accelerating the software delivery pipeline.

**2. Key Features and Modules**

SmartSDLC provides a unified ecosystem with the following intelligent modules:

2.1 Requirement Upload and Classification

* **Functionality: Upload PDF documents with unstructured text.**
* **Backend:**
  + **Granite-20B AI.** Extracts content using PyMuPDF.
  + Classifies sentences into SDLC phases (Requirements, Design, Development, Testing, Deployment) using IBM Watsonx
* **Output:** Structured user stories grouped by SDLC phases.
* **Frontend**: Displays results in a phase-wise organized view.

**2.2 AI Code Generator**

* Functionality: Converts natural language prompts or user stories into production-ready code.
* Backend: Uses Watsonx AI for context-aware code generation.
* Frontend: Displays code with syntax highlighting for immediate use.

**2.3 Bug Fixer**

* Functionality: Debugs code snippets (Python, JavaScript, etc.).
* Backend: Detects syntactical and logical errors and returns optimized code.
* Frontend: Presents side-by-side comparison of original vs. fixed code.

**2.4 Test Case Generator**

* Functionality: Generates test cases based on functional code or requirements.
* Output: Structured test cases using unittest/pytest frameworks.
* Advantage: Eliminates manual test writing and improves test coverage.

**2.5 Code Summarizer**

* Functionality: Converts code snippets into human-readable explanations.
* Use Case: Helpful for onboarding new developers and maintaining documentation.

**2.6 Floating AI Chatbot Assistant**

* Functionality: Real-time conversational support across the application.
* Backend: Powered by LangChain, intelligently answering SDLC-related queries.
* Frontend: Interactive chat interface for instant help.

**3. Technology Stack**

**Frontend:**

**React.js, Tailwind CSS**

**Syntax Highlighter Libraries (e.g., Prism.js)**

**Backend:**

* Python (FastAPI / Flask)
* PyMuPDF for document parsing
* LangChain for chatbot integration
* IBM Watsonx Granite-20B AI for NLP & code generation

**Database**:

* PostgreSQL / MongoDB

**Deployment:**

* Docker & Kubernetes
* CI/CD pipelines (GitHub Actions / Jenkins)

**4. Workflow Overview**

1. Requirement Upload → AI Classification → Structured User Stories
2. User Stories / Prompts → AI Code Generation
3. Buggy Code → AI Debugging → Corrected Code
4. Code / Requirement → AI Test Case Generation
5. Codebase → AI Summarization → Documentation
6. Real-Time Support via Floating AI Chatbot

**5. Benefits**

* Time-Saving: Instant code generation and test case creation.
* Accuracy: AI-based classification and bug fixing reduce errors.
* Collaboration: Structured user stories enable better traceability.
* Automation: Eliminates repetitive manual tasks in SDLC.
* Scalability: Easily adapts to large projects with diverse teams**.**

**6. Future Enhancements**

* Integration of multi-language code support (C++, Java, Go).
* Advanced DevOps automation (CI/CD pipeline generation).
* AI-driven performance optimization recommendations.
* Voice-based requirement gathering using speech-to-text NLP.

**7. Conclusion**

SmartSDLC revolutionizes software development by creating an AI-powered ecosystem that accelerates the entire SDLC, making it faster, smarter, and more reliable for both technical and non-technical users.