**Project Report**

**On**

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

**SUBMITTED BY**

**Team ID :** LTVIP2025TMID31232

**Team Size :** 4

**Team Leader :** Raga Loknath

**Team member :** Mohammad Ismail Shaik

**Team member :** M Vardhan

**Team member :** Kuruba Roopavathi

**SUBMITTED TO**



**Project Report on**

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

**1. Abstract**

SmartSDLC is an AI-powered platform designed to automate critical phases of the Software Development Lifecycle (SDLC). By integrating **Natural Language Processing (NLP)** and **Generative AI** technologies, SmartSDLC can transform **unstructured requirements** into **structured user stories, production-ready code, test cases, bug fixes, and documentation**. This automation reduces manual efforts, accelerates project timelines, and enhances accuracy, making it an innovative solution for modern software development teams.

**2. Introduction**

**2.1 Problem Statement**

Traditional SDLC requires significant manual effort for requirement gathering, coding, testing, and documentation. This often leads to:

* **Time delays** due to repetitive tasks.
* **Misinterpretation of requirements** due to unstructured data.
* **Errors in code and testing** caused by manual processes.
* **Documentation challenges**, especially for large codebases.

SmartSDLC addresses these challenges by providing **AI-driven automation** across multiple SDLC phases.

**2.2 Objective**

The primary objective of SmartSDLC is to:

* Automate requirement classification.
* Generate production-ready code from user prompts.
* Debug code using AI-based analysis.
* Automatically generate test cases.
* Summarize and document code effectively.
* Provide an **AI chatbot assistant** for real-time development support.

**3. System Overview**

SmartSDLC is built as a **full-stack platform** with both frontend and backend components, integrated with **IBM Watsonx AI** and **LangChain**.

**3.1 Functional Modules**

**3.1.1 Requirement Upload and Classification**

* Upload unstructured requirement documents (PDF).
* Extract text using **PyMuPDF**.
* Classify content into SDLC phases (Requirement, Design, Development, Testing, Deployment).
* Output: **Structured user stories** for planning and traceability.

**3.1.2 AI Code Generator**

* Converts natural language prompts or user stories into **ready-to-use code**.
* Uses **Watsonx Granite-20B AI model** for context-aware code generation.
* Outputs code with **syntax highlighting** for easy integration.

**3.1.3 Bug Fixer**

* Accepts buggy code snippets.
* AI detects **syntax and logic errors**.
* Returns optimized and corrected code.

**3.1.4 Test Case Generator**

* Generates **unit test cases** from functional code or requirements.
* Uses **unittest/pytest** for automation.
* Enhances **test coverage** and consistency.

**3.1.5 Code Summarizer**

* Creates **human-readable summaries** of code modules.
* Useful for **documentation and knowledge transfer**.

**3.1.6 Floating AI Chatbot**

* Provides **real-time conversational support** for SDLC queries.
* Integrated via **LangChain** and Watsonx AI.

**4. System Architecture**

The platform follows a **client-server architecture**:

**4.1 Frontend**

* Developed using **React.js** and **Tailwind CSS**.
* Displays classified requirements, AI-generated code, test cases, and summaries.
* Provides a **floating AI chatbot** interface.

**4.2 Backend**

* Developed using **Python (FastAPI or Flask)**.
* Connects with **IBM Watsonx AI** for NLP and code generation.
* Handles **document parsing** (PyMuPDF), **prompt processing**, and **AI responses**.

**4.3 Database**

* **PostgreSQL** or **MongoDB** for structured storage of user stories, generated code, and test cases.

**4.4 Deployment**

* Docker containers for isolated microservices.
* Kubernetes for scalability and orchestration.
* CI/CD pipelines using **GitHub Actions/Jenkins**.

**5. Workflow**

The overall workflow of SmartSDLC is as follows:

1. **Upload Requirements → AI Classification → Structured Stories.**
2. **Stories → AI Code Generator → Production-Ready Code.**
3. **Buggy Code → AI Bug Fixer → Optimized Code.**
4. **Code/Requirements → AI Test Case Generator → Automated Testing.**
5. **Codebase → AI Code Summarizer → Documentation.**
6. **Interactive AI Chatbot → Developer Assistance.**

**6. Technology Stack**

* **Languages:** Python, JavaScript (React.js)
* **Frontend:** React.js, Tailwind CSS, Prism.js (for syntax highlighting)
* **Backend:** FastAPI / Flask, LangChain
* **AI Model:** IBM Watsonx Granite-20B AI
* **Database:** PostgreSQL / MongoDB
* **Tools:** PyMuPDF, Docker, Kubernetes, GitHub Actions

**7. Advantages of SmartSDLC**

* **Time Efficiency:** Rapid code and test generation.
* **Reduced Human Error:** AI ensures logical consistency.
* **Better Documentation:** Automatic code summarization.
* **Improved Collaboration:** Structured stories improve traceability.
* **Scalability:** Easily integrates into enterprise workflows.

**8. Future Enhancements**

* **Voice-based requirement analysis** using speech-to-text AI.
* **Multi-language code generation** (Java, C++, Go).
* **AI-driven DevOps automation** (CI/CD pipeline creation).
* **Security vulnerability detection** in code.

**9. Conclusion**

SmartSDLC is a next-generation **AI-powered SDLC automation platform** that redefines how software is built, tested, and documented. By leveraging **NLP, Generative AI, and intelligent automation**, it significantly reduces development time, enhances code quality, and streamlines the entire software lifecycle.

**10. References**

* IBM Watsonx AI Documentation
* LangChain Framework Documentation
* PyMuPDF Library Docs
* SDLC Best Practices (IEEE & Agile methodologies)

https://drive.google.com/file/d/1jPvMNtFTyMdKsGoBO-NV6fpNhVY14WXI/view?usp=drivesdk