**🧠 SmartSDLC: AI-Powered Software Development Assistant**

**Project Report**

**1. INTRODUCTION**

**1.1 Project Overview**

**SmartSDLC** is an AI-integrated assistant platform aimed at enhancing software development productivity across various phases of the Software Development Life Cycle (SDLC). It uses **FastAPI** as the backend framework, **Streamlit** for the frontend, and is powered by **IBM Watsonx** (Granite Model) to deliver intelligent features such as requirement analysis, test case generation, bug detection, and code improvement suggestions.

**1.2 Purpose**

The main goal is to assist developers in streamlining their development tasks with the help of Generative AI, reducing manual effort and improving software quality by automating repetitive and logic-driven tasks.

**2. IDEATION PHASE**

**2.1 Problem Statement**

Modern software development is time-consuming, error-prone, and requires repeated refinement. Developers face:

* **Inefficient requirement analysis**
* **Lack of intelligent code generation or debugging support**
* **Manual test case creation**

SmartSDLC aims to solve these using AI-powered automation, specifically trained and optimized for SDLC tasks.

**2.2 Empathy Map Canvas**

**User Persona (Akhil, 26, Junior Developer):**

* **Thinks & Feels:** Overwhelmed with documentation, struggles with test writing.
* **Sees:** GitHub Copilot, ChatGPT, StackOverflow.
* **Says & Does:** “Wish something could generate boilerplate/test cases for me.”
* **Hears:** “Stick to deadlines,” “Write more clean code.”
* **Pains:** Manual documentation, debugging, test coverage issues.
* **Gains:** Code suggestions, documentation helper, test writer, AI assistant.

**2.3 Brainstorming**

Key features brainstormed:

* AI-based **requirement analysis**
* Code & bug explanation
* **Test case generation**
* Real-time **code improvement**
* Integration with frontend & APIs

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**

**Stages:**  
*Onboarding → Feature Use → Feedback → Enhancement → Continuous Use*

SmartSDLC supports users from idea to code optimization, with AI-driven interactions ensuring smooth transitions across SDLC stages.

**3.2 Solution Requirements**

**Functional Requirements:**

* Prompt-to-code generation
* Code explanation
* Bug fixing
* Test case creation
* SDLC phase classification

**Non-Functional Requirements:**

* Security (API Key storage)
* Performance (< 2s response time)
* Scalability
* Availability (via FastAPI)
* Usability (Streamlit frontend)

**3.3 Data Flow Diagram**

User Prompt → FastAPI Backend → IBM Watsonx → AI Response → Returned JSON → Streamlit UI Display

**3.4 Technology Stack**

* **Frontend:** Streamlit
* **Backend:** FastAPI, Python
* **AI Model:** IBM Granite 13B Instruct
* **Middleware:** Watsonx SDK / REST API
* **Data Handling:** JSON
* **Security:** python-dotenv
* **Deployment:** Localhost

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit**

SmartSDLC addresses real issues faced by developers in early software phases.  
AI automation improves developer speed, confidence, and software quality.

**4.2 Proposed Solution**

An AI-powered SDLC assistant that:

* Accepts prompts for code/test generation
* Suggests improvements
* Explains logic
* Assists in bug resolution
* Fits into existing dev environments

**4.3 Solution Architecture**

* **Frontend:** Streamlit
* **Backend API:** FastAPI
* **Model Layer:** IBM Watsonx (Granite-13B)
* **Routing Layer:** /ai/generate-code, etc.
* **Security:** API Keys via .env

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

* **Approach:** Agile Scrum
* **Sprints:** 4 (each 5 days)
* **Tools:** GitHub, VS Code, Trello, Streamlit, Watsonx

| **Sprint** | **Tasks** |
| --- | --- |
| 1 | Folder setup, API keys, IBM auth |
| 2 | Backend AI route, Watsonx service |
| 3 | Streamlit frontend |
| 4 | Testing, bug fixing, documentation |

**6. FUNCTIONAL AND PERFORMANCE TESTING**

**6.1 Performance Testing**

**Functional Tests:**

* API response to prompt
* Model ID validation
* Output JSON formatting
* .env loading check

**Performance Goals:**

* Backend response < 3s
* Streamlit page load < 2s

**Result:**  
All features passed; IBM Watsonx integration validated with sample prompts.

**7. RESULTS**

**7.1 Output Screenshots**

(Screenshots should include):

* Prompt input field
* Generated code
* Explanation area
* Test case suggestion section
* Debug output (if applicable)

**8. ADVANTAGES & DISADVANTAGES**

**8.1 Advantages**

* AI-automated software assistance
* Easy to use for students and developers
* Based on real-world SDLC tasks
* IBM Watsonx ensures enterprise-grade quality
* Backend/frontend separation for scalability

**8.2 Disadvantages**

* IBM API requires active internet and account
* Limited to prompt-based input
* Output quality depends on model parameters
* Free tier has limits on usage

**9. CONCLUSION**

SmartSDLC bridges the gap between manual software engineering tasks and intelligent automation. By integrating IBM Watsonx and building a responsive Streamlit frontend, it allows developers to experience smart development assistance. Future enhancements can make this a complete AI SDLC assistant.

**10. FUTURE SCOPE**

**Planned Enhancements:**

* Multi-route support (e.g., explain, test-case, bug-fix)
* Integration with GitHub Repos
* Voice-to-prompt interface
* User authentication (Firebase)
* Deployment on IBM Cloud
* Granular SDLC phase classification
* Dark mode frontend toggle
* Export results to PDF