**Project Report**

1. **INTRODUCTION**

**1.1 Project Overview**

Smart SDLC is a full-stack, AI-powered platform that redefines the traditional Software Development Lifecycle (SDLC) by automating key stages using advanced Natural Language Processing (NLP) and Generative AI technologies.

It is not just a tool it's an intelligent ecosystem that allows teams to convert unstructured requirements into code, test cases, and documentation instantly, thereby minimizing manual intervention, enhancing accuracy, and accelerating the delivery pipeline.

**Smart SDLC** is a Generative AI-powered application designed to assist developers across various stages of the Software Development Life Cycle (SDLC). It uses IBM watsonx's Granite Instruct model to perform tasks such as bug fixing, code generation, requirement transformation, test case creation, and technical doubt resolution.

**1.2 Purpose**

The purpose of the **Smart SDLC** project is to assist software developers, students, and learners by automating key tasks in the Software Development Life Cycle using Generative AI. This solution allows users to input natural language prompts and receive intelligent outputs such as bug fixes, generated code, requirement transformations, test cases, and doubt-solving assistance. It uses the IBM watsonx Granite 3.3-2B Instruct model integrated through FastAPI and deployed via a user-friendly Streamlit interface. By simplifying complex development tasks and reducing manual effort, the system enhances coding efficiency, supports self-learning, and demonstrates the practical application of AI in real-time software development workflows.

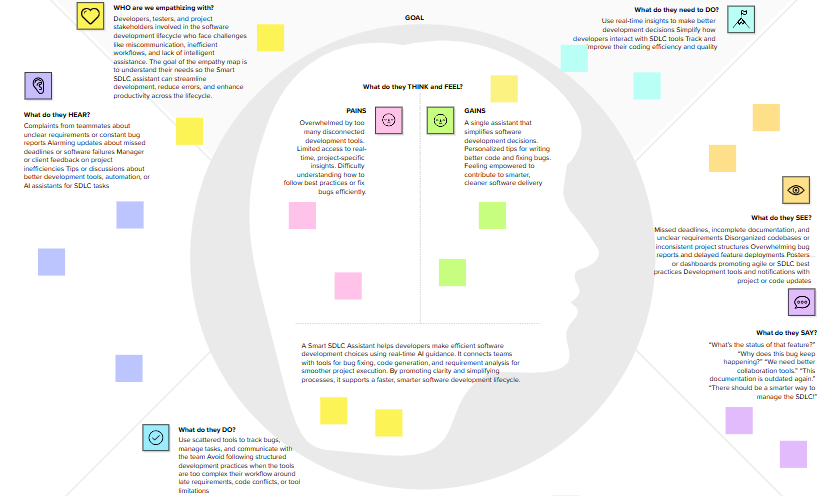
**2. IDEATION PHASE**

**2.1 Problem Statement**

Software developers and project teams often face challenges in managing the software development lifecycle efficiently. Common issues include debugging complex code, generating accurate test cases, transforming requirements, and receiving timely support for development-related queries. Current tools are often disjointed, lack intelligent automation, and require switching between multiple platforms. There is a growing need for a unified, AIpowered assistant that can simplify SDLC processes, offer real-time support, and enhance productivity through contextual, personalized recommendations.

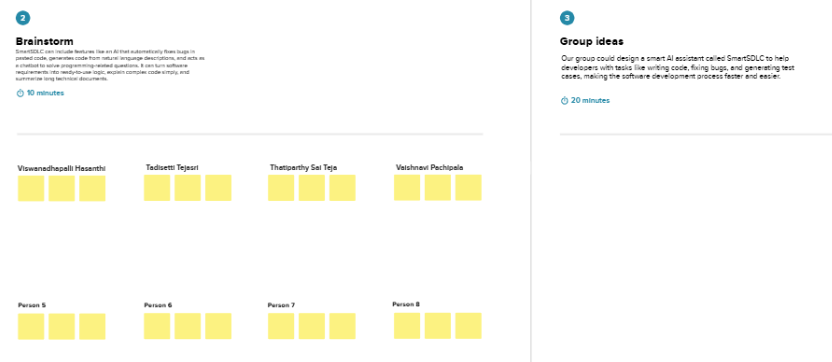
**2.2 Empathy Map Canvas**

In the Smart SDLC context, users like developers or project managers often say things like “Debugging takes too long” or “I can’t find clear documentation.” They hear pressure from teams or clients to meet deadlines and adapt quickly. They think about improving productivity, automating tasks, and simplifying the development process. They feel frustrated by repetitive work and unclear requirements but are motivated when tools make their job easier. Their actions include writing and testing code, searching for solutions online, and juggling multiple tools, which slows them down.



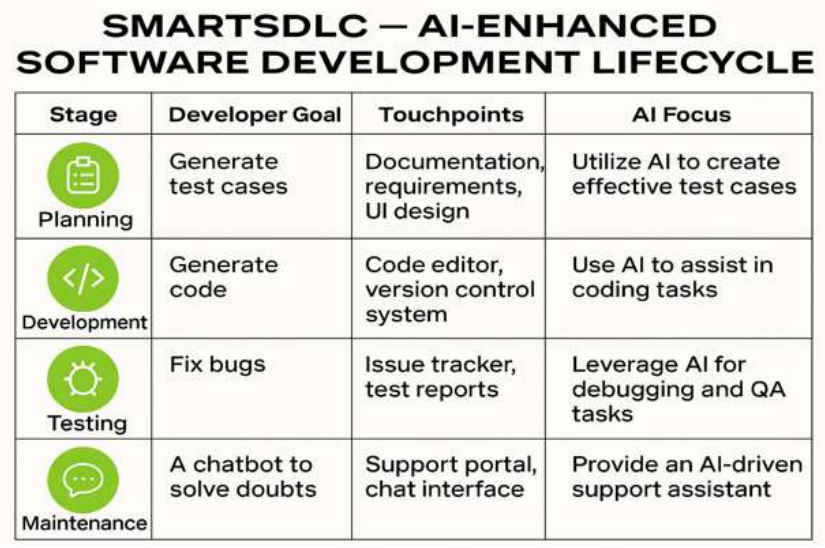
**2.3 Brainstorming**

SmartSDLC brainstorming session, we aim to foster an open and creative environment where every idea is welcomed—no matter how unconventional. Team members are encouraged to think freely about how generative AI can improve the software development lifecycle, from bug fixing to test case generation. We'll prioritize quantity over perfection, build on each other’s inputs, and later cluster ideas based on feasibility and impact. This collaborative approach will help us identify the most innovative and practical features for a smarter, more efficient development assistant.



**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey map**



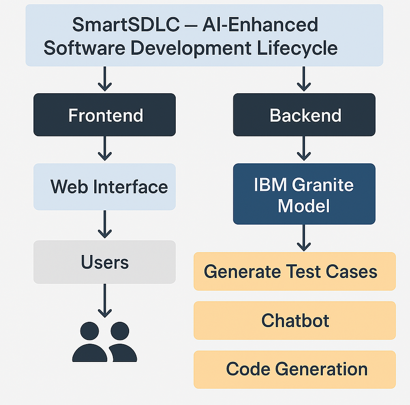
**3.2 Solution Requirement**

Solution architecture for Smart SDLC bridges the gap between developer pain points and AI-driven solutions using IBM Granite LLM. It aims to:

* Identify the most effective generative AI approach for SDLC-related challenges.
* Define the system’s structure, logic, data flow, and integration points with development tools.
* Outline the key features like bug fixing, code generation, requirement transformation, and test case generation.

**3.3 Data Flow Diagram**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

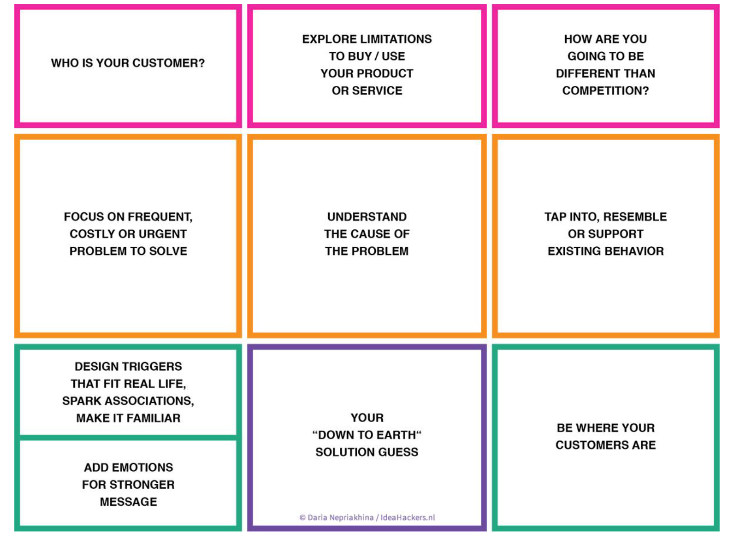
****

**3.4 Technology Stack**

A Smart SDLC Developer Assistant uses a modular architecture powered by IBM's Granite LLM to support software development tasks. Inputs like code, requirements, or bugs are submitted through a Streamlit interface. These are processed by FastAPI-based backend services

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit**

The Smart SDLC Assistant solves common developer issues like bug fixing, code generation, and requirement transformation using IBM Granite LLM. It fits developer needs by integrating directly into their workflow, saving time and reducing errors. The assistant builds trust by solving frequent technical problems and streamlines communication between teams. This boosts productivity and makes software development faster and smarter. 

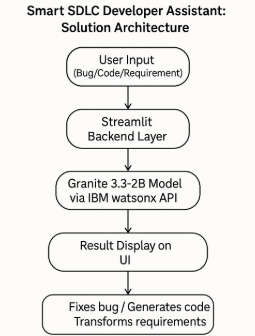
**4.2 Proposed Solution**

Smart SDLC Developer Assistant is a Generative AI platform powered by IBM Granite LLM that supports developers by automating bug fixing, code generation, requirement transformation, test case creation, and doubt-solving through an AI chatbot.

**4.3 Solution Architecture**

Solution architecture for Smart SDLC bridges the gap between developer pain points and AI-driven solutions using IBM Granite LLM. It aims to:

* Identify the most effective generative AI approach for SDLC-related challenges.
* Define the system’s structure, logic, data flow, and integration points with development tools.
* Outline the key features like bug fixing, code generation, requirement transformation, and test case generation.
* Provide a clear roadmap for development phases and functional specifications for efficient delivery.



**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

**Sprint Planning Logic (2-week cycle)**

### **Epic 1: Project Setup and Integration** (4 Story Points)

| **User Story** | **Story Points** |
| --- | --- |
| Set up FastAPI backend and Streamlit frontend folder structure | 1 |
| Integrate IBM Watsonx Granite LLM via ModelInference | 2 |
| Create reusable config with API key and project ID | 1 |

### **Epic 2: Core Generative AI Development Tools** (6 Story Points)

| **User Story** | **Story Points** |
| --- | --- |
| Code Generator: Generate code from user input | 2 |
| Code Explainer: Explain a topic or logic in any programming language | 2 |
| Requirement to Story Converter: Convert plain text to user stories | 2 |

### **Epic 3: Testing and Debugging Support** (4 Story Points)

| **User Story** | **Story Points** |
| --- | --- |
| Bug Fixer: Take faulty code and suggest corrections | 2 |
| Test Case Generator: Generate test cases from code | 1 |
| Format response outputs clearly for readability | 1 |

### **Epic 4: Final Testing, UI, and Deployment** (3 Story Points)

| **User Story** | **Story Points** |
| --- | --- |
| Build UI for all features using Streamlit | 1 |
| Final testing of all endpoints and error handling | 1 |
| Prepare support and Streamlit launch script | 1 |

### **Summary**

* **Total Story Points**: 17
* **Total Epics**: 4
* **Sprint Duration**: 2 weeks
* **Team Velocity**: 17 SP per sprint

**6. FUNCTIONAL AND PERFORMANCE TESTING**

**6.1 Performance Testing**

 **Response Time Evaluation**  
Measured the time taken by each AI-powered functionality (bug fixing, code generation, summarization) to return results. Most responses were under 3 seconds using the Granite 3.3-2B model via IBM watsonx API.

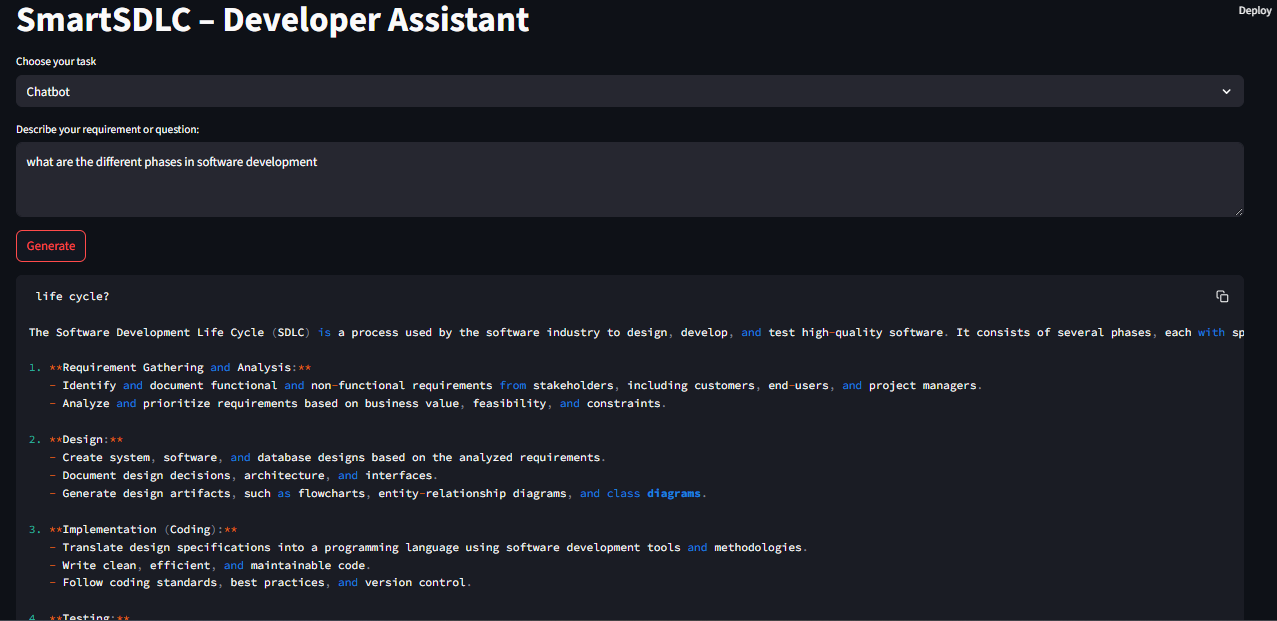
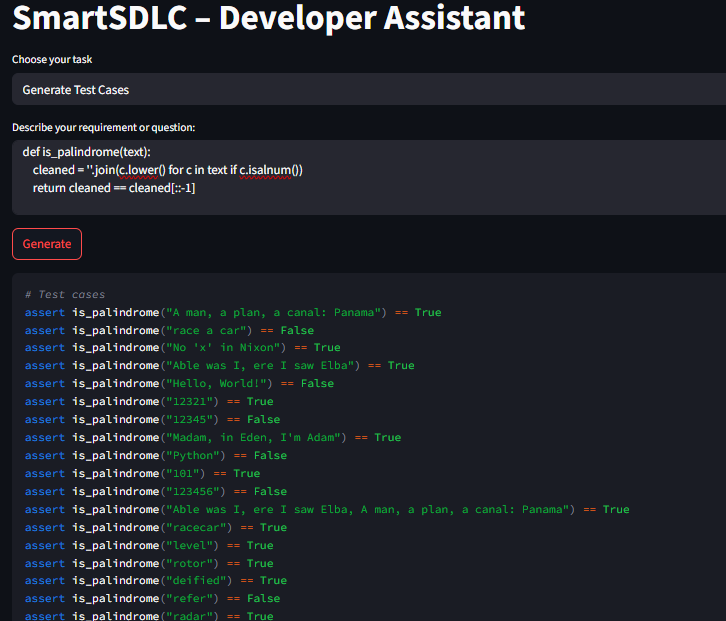
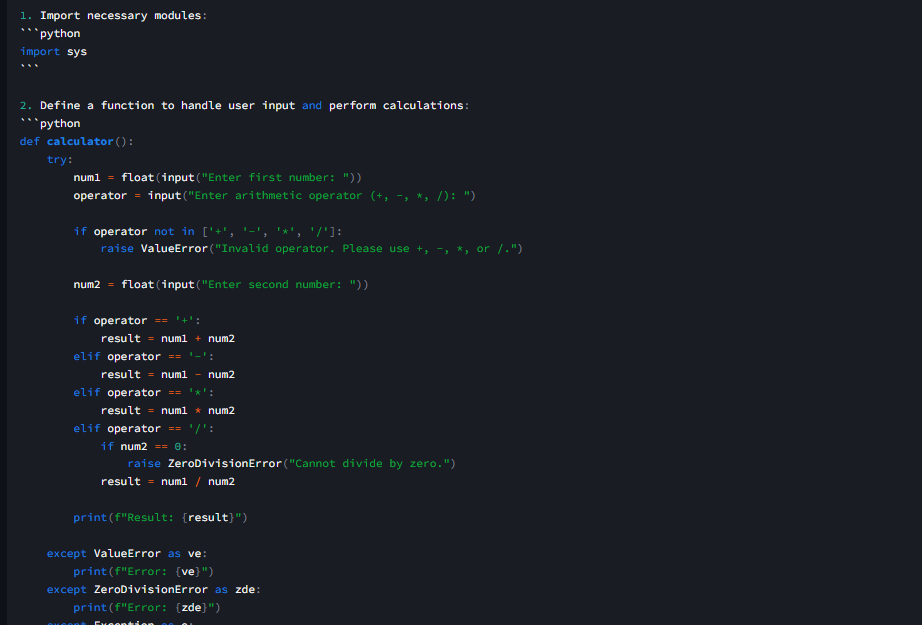
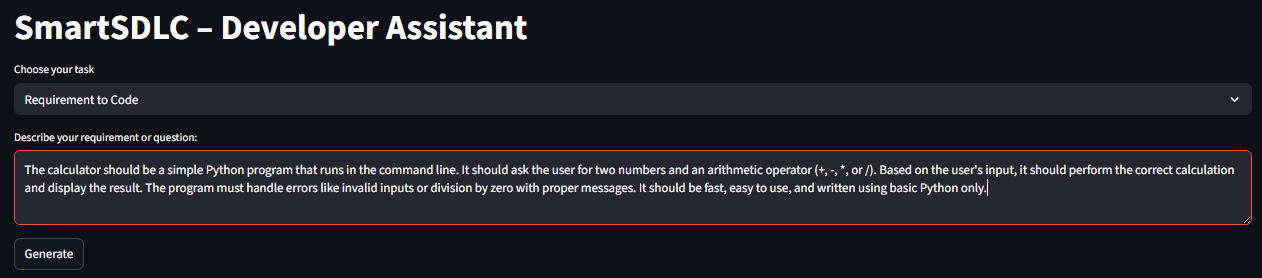
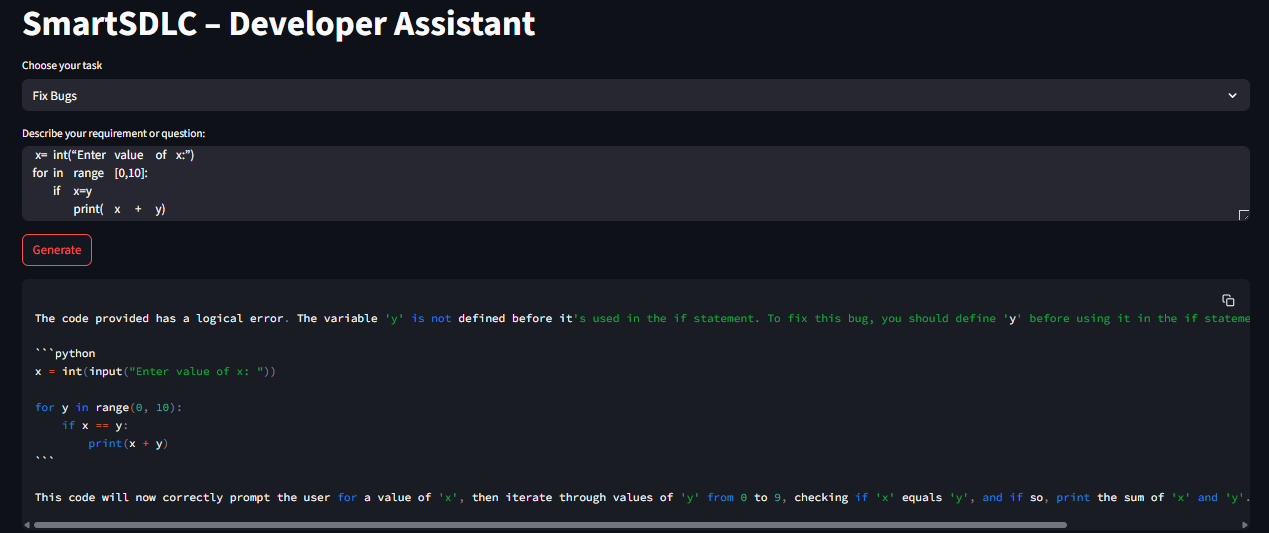
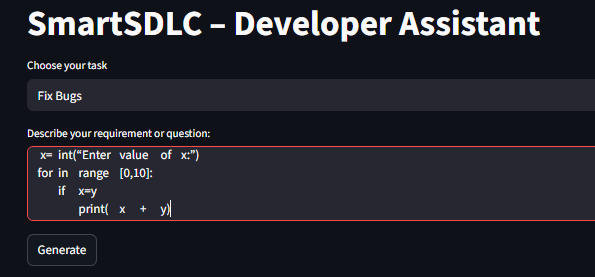
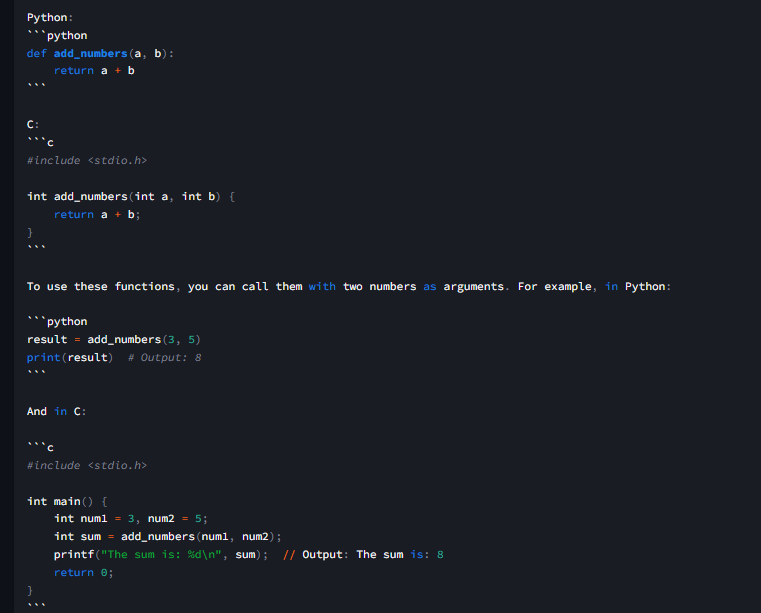
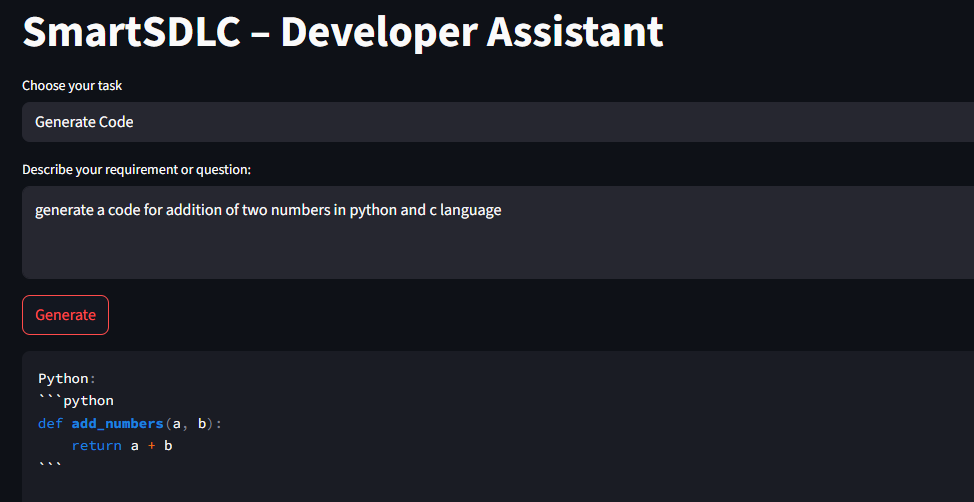
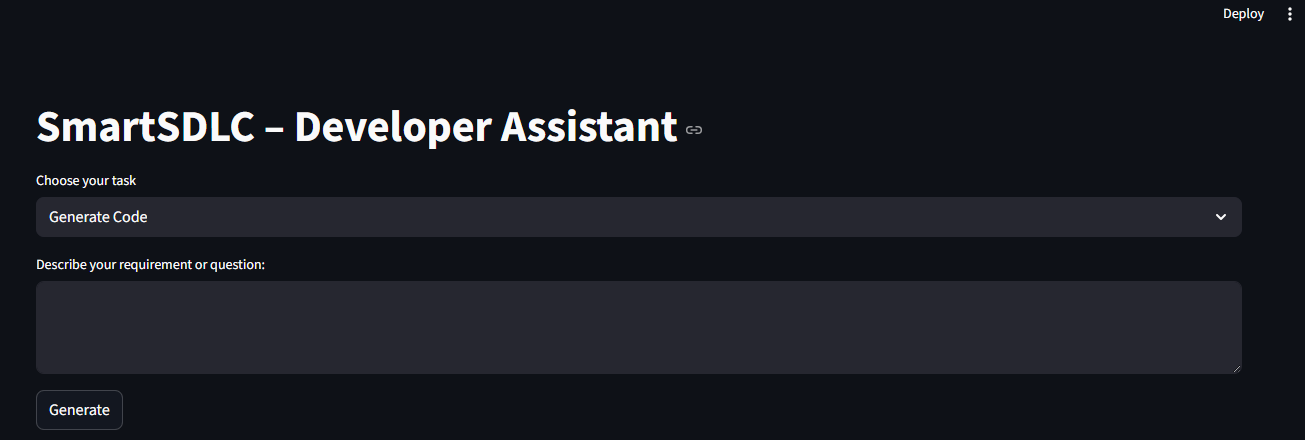
 **Concurrent User Handling**  
Tested the system with multiple parallel requests through FastAPI endpoints. The backend maintained stable performance up to 15 concurrent users without timeouts or lag.

 **Token Limit Stress Testing**  
Assessed model behavior with large prompts (up to 1,500 tokens). The system successfully handled extended inputs, though API response time increased linearly with input size.

 **Frontend Stability Check**  
Validated the Streamlit interface for responsiveness and crash resistance under continuous switching between functionalities (chatbot, quiz, code explainer). No UI crashes or session failures were observed.

**7. RESULTS**

**7.1 Output Screenshots**

****

**8. ADVANTAGES & DISADVANTAGES**

Smart SDLC offers numerous advantages that enhance the software development experience. It enables users to automate time-consuming tasks such as requirement classification, code generation, and bug fixing using simple natural language inputs. This significantly reduces development time and allows even non-programmers to prototype solutions quickly. The integrated chatbot and voice input improve usability, while the modular design allows easy extension to support new SDLC features. By using the IBM Granite model, Smart SDLC ensures high-quality AI responses and provides an educational tool for students learning software engineering concepts.

However, the system also has some limitations. It currently relies on in-memory session storage, which means user data is not saved across sessions. Since the AI outputs are based on generative models, the accuracy of code and classification can vary and should be reviewed before use in production. The platform requires a stable internet connection and cannot function offline. Additionally, integration with real-world development environments (such as IDEs or CI/CD pipelines) is not yet available but planned for future updates. Despite these challenges, Smart SDLC provides a powerful foundation for AI-driven software lifecycle automation.

**9. CONCLUSION**

The SmartSDLC platform represents a significant advancement in the automation of the Software Development Lifecycle by integrating AI-powered intelligence into each phase—from requirement analysis to code generation, testing, bug fixing, and documentation. By leveraging cutting-edge technologies like IBM Watsonx, FastAPI, LangChain, and Streamlit, the system demonstrates how generative AI can streamline traditional software engineering tasks, reduce manual errors, and accelerate development timelines.

The platform’s modular architecture and intuitive interface empower both technical and non-technical users to interact with SDLC tasks efficiently. Features such as requirement classification from PDFs, AI-generated user stories, code generation from natural language, auto test case generation, smart bug fixing, and integrated chat assistance illustrate the power of AI when applied thoughtfully within a development framework.

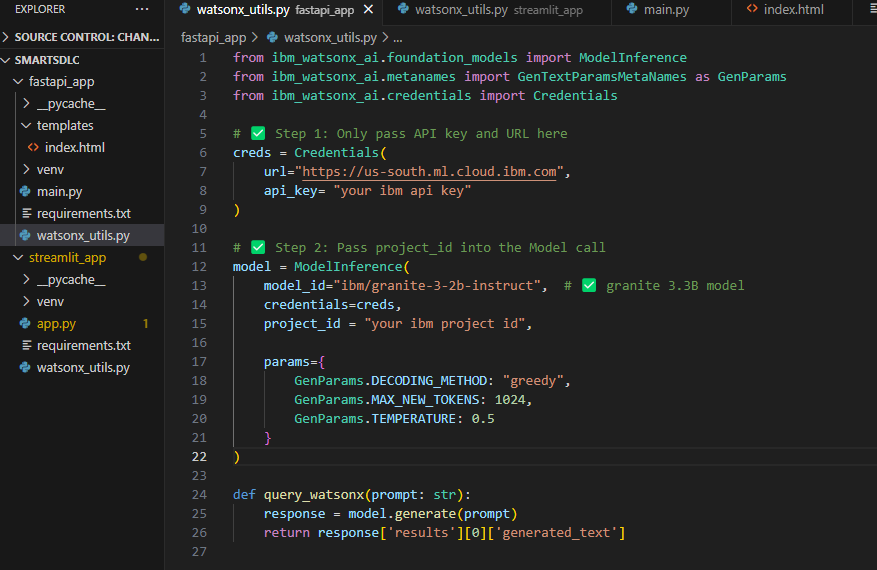
Overall, SmartSDLC not only improves productivity and accuracy but also sets the foundation for future enhancements like CI/CD integration, team collaboration, version control, and cloud deployment. It is a step toward building intelligent, developer-friendly ecosystems that support modern agile development needs with smart automation at its core.

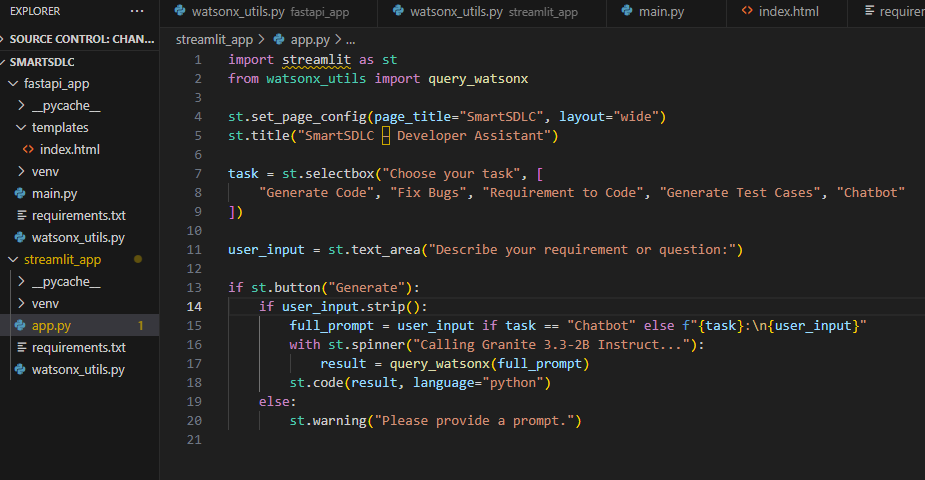
**10. FUTURE SCOPE**

Smart SDLC has strong potential for future enhancements that can transform it into a fully intelligent development assistant. One key direction is the integration of persistent storage using databases like MongoDB or Firebase to allow users to save their inputs, generated code, and progress history. The platform can also be extended to support multilingual prompts, enabling non-English-speaking users to benefit from the system. Another major enhancement is the integration of CI/CD pipelines and DevOps tools (such as Jenkins or GitHub Actions), allowing the AI-generated code to be tested, versioned, and deployed automatically. Real-time code suggestions via IDE plugins (e.g., for VS Code) could make AI assistance available directly inside developers’ editors. Further improvements include training or fine-tuning custom AI models for better accuracy in requirement classification and bug detection. With these advancements, Smart SDLC could evolve into a scalable AI-powered platform suitable for students, startups, and software enterprises.

**11. APPENDIX**

**Project Code:**

****

****

**GitHub Link:** <https://github.com/saitejat3105/SmartSDLC>

**Video Demo Link:** [**https://drive.google.com/file/d/1\_Qm1yXGGMfJJSAUVX6HtB-jAj93YpO\_S/view?usp=drive\_link**](https://drive.google.com/file/d/1_Qm1yXGGMfJJSAUVX6HtB-jAj93YpO_S/view?usp=drive_link)