SmartSDLC Final Project Report

1. INTRODUCTION

1.1 Project Overview

SmartSDLC is an AI-driven software platform designed to automate core components of the Software Development Lifecycle (SDLC). It integrates generative AI models such as IBM Watsonx's Granite series and technologies like FastAPI, LangChain, and Streamlit to transform unstructured requirements into structured artifacts such as code, test cases, bug fixes, and documentation.

1.2 Purpose

The purpose of SmartSDLC is to enhance developer productivity and accuracy by automating repetitive and complex SDLC tasks using AI, making the software development process faster, more efficient, and accessible to non-technical stakeholders.

---

2. IDEATION PHASE

2.1 Problem Statement

Traditional SDLC processes are time-consuming and prone to manual errors. Teams often struggle with unclear requirements, slow development cycles, and poor documentation. There is a need for an intelligent platform that can automate and streamline the entire development lifecycle.

2.2 Empathy Map Canvas

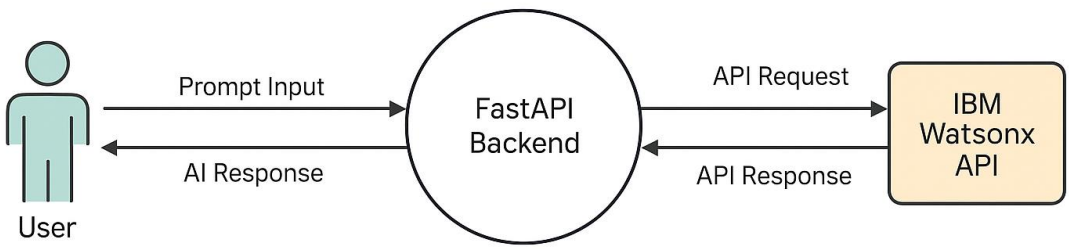
Users: Developers, Project Managers, Non-technical stakeholders

Needs: Faster delivery, clearer documentation, easier debugging, and requirement analysis.

Pains: Manual effort, redundant tasks, slow communication.

Gains: Automation, clarity, AI assistance, speed.

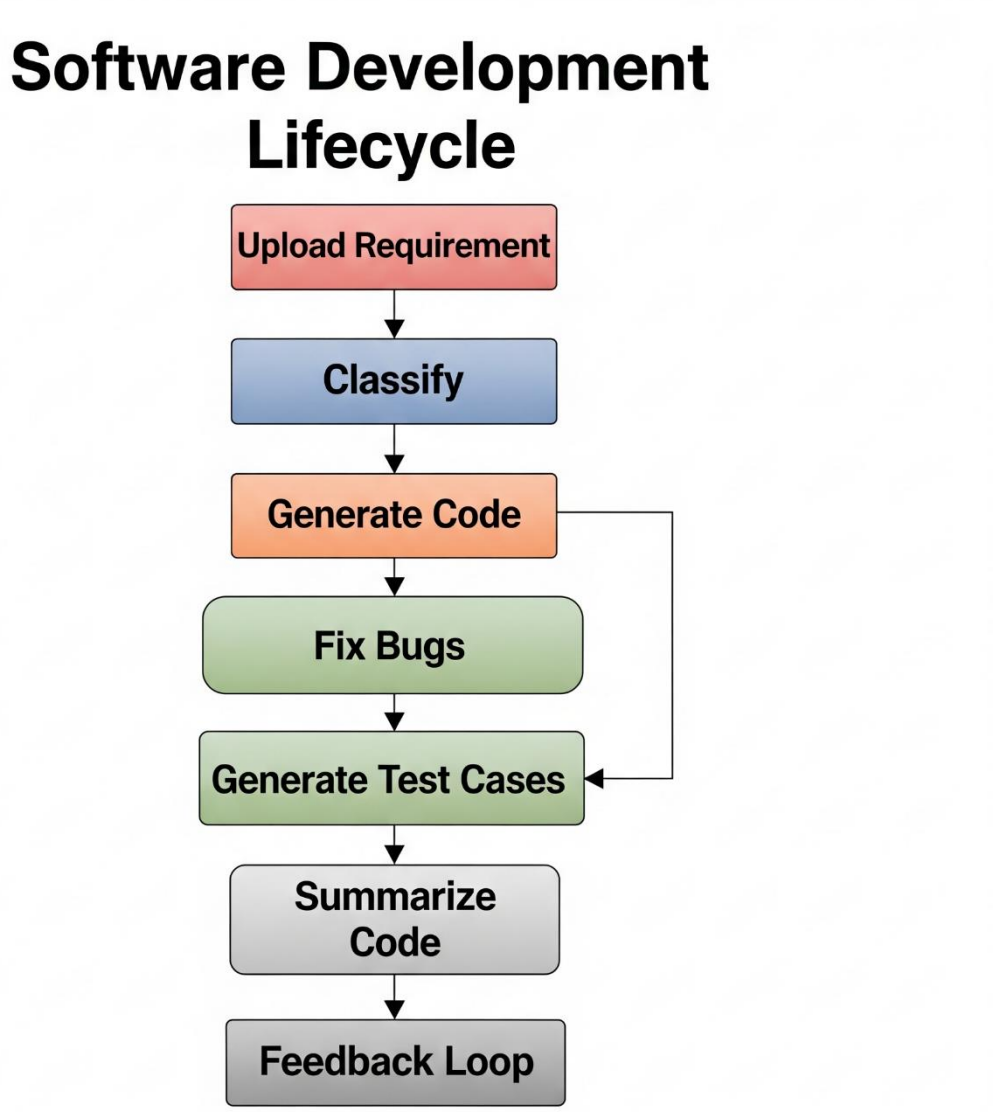
2.3 Brainstorming



- AI for requirement classification
- Auto code generation
- Test case automation
- Bug fixing using NLP
- AI chatbot for SDLC guidance
- Code summarization for documentation

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

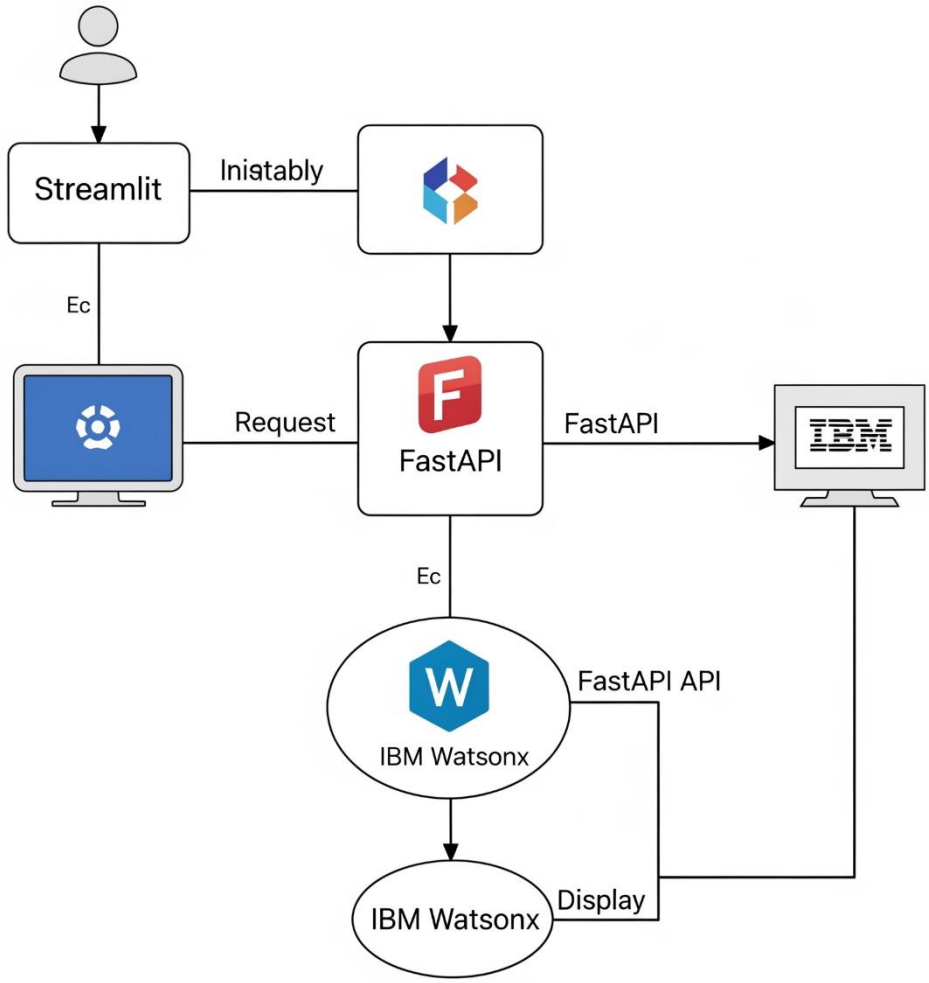


Stages: Upload requirement → Classify → Generate code → Fix bugs → Generate test cases → Summarize code → Feedback loop

3.2 Solution Requirements

- Functional: Upload PDFs, classify requirements, generate code/tests, fix bugs, summarize code, chatbot
- Non-Functional: Fast response, AI reliability, security, scalability

3.3 Data Flow Diagram



User → Streamlit Frontend → FastAPI Backend → IBM Watsonx API → Response → Frontend Display

3.4 Technology Stack

- **Frontend:** Streamlit
- **Backend:** FastAPI
- **AI Models:** IBM Watsonx Granite-20B
- **NLP Orchestration:** LangChain
- **PDF Parsing:** PyMuPDF
- **Deployment Tools:** Uvicorn, Python-dotenv

---

4. PROJECT DESIGN

4.1 Problem Solution Fit

The platform offers AI-based assistance for each SDLC phase, solving real-world developer bottlenecks with automation.

4.2 Proposed Solution

SmartSDLC accepts raw input (e.g., PDFs or text prompts), classifies requirements, generates code, identifies/fixes bugs, creates test cases, and summarizes code with AI.

4.3 Solution Architecture

Streamlit UI ↔ FastAPI ↔ Watsonx API

└→ PDF Processor

└→ Code Generator

└→ Bug Resolver

└→ Test Case Generator

└→ Chatbot Handler

---

5. PROJECT PLANNING & SCHEDULING: This is the main section heading, indicating that the following points will detail how the project will be planned and its timeline.

5.1 Project Planning: A sub-section focusing specifically on the planning aspects.

Sprint-1: Data Collection, Loading, Preprocessing (8 Story Points)

- Sprint-1: In Agile Scrum, a "sprint" is a short, fixed-length period (usually 1-4 weeks) during which a specific set of work is completed and made ready for review. This indicates the first iteration of work.
- Data Collection, Loading, Preprocessing: These are the specific tasks or features planned to be completed within Sprint 1. This suggests the project involves working with data.
- (8 Story Points): "Story points" are a unit of measure in Agile Scrum used to estimate the effort required to implement a user story or complete a task. They often represent a combination of complexity, effort, and risk. So, the work in Sprint 1 is estimated to require 8 units of effort.

Sprint-2: Model Building, Testing, Deployment (16 Story Points)

- Sprint-2: The second iteration of work.
- Model Building, Testing, Deployment: The tasks planned for Sprint 2. This suggests that after preparing the data in Sprint 1, the team will then focus on building, testing, and making a model available.

- (16 Story Points): The work in Sprint 2 is estimated to require 16 units of effort.

Team Velocity: 12 Story Points per Sprint

- Team Velocity: This is a key metric in Scrum. It represents the average amount of work (in story points) that a development team can complete in a single sprint. It's usually calculated by averaging the story points completed in previous sprints.
- 12 Story Points per Sprint: This means that, based on past performance, this particular team can typically complete 12 story points of work within one sprint.

---

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

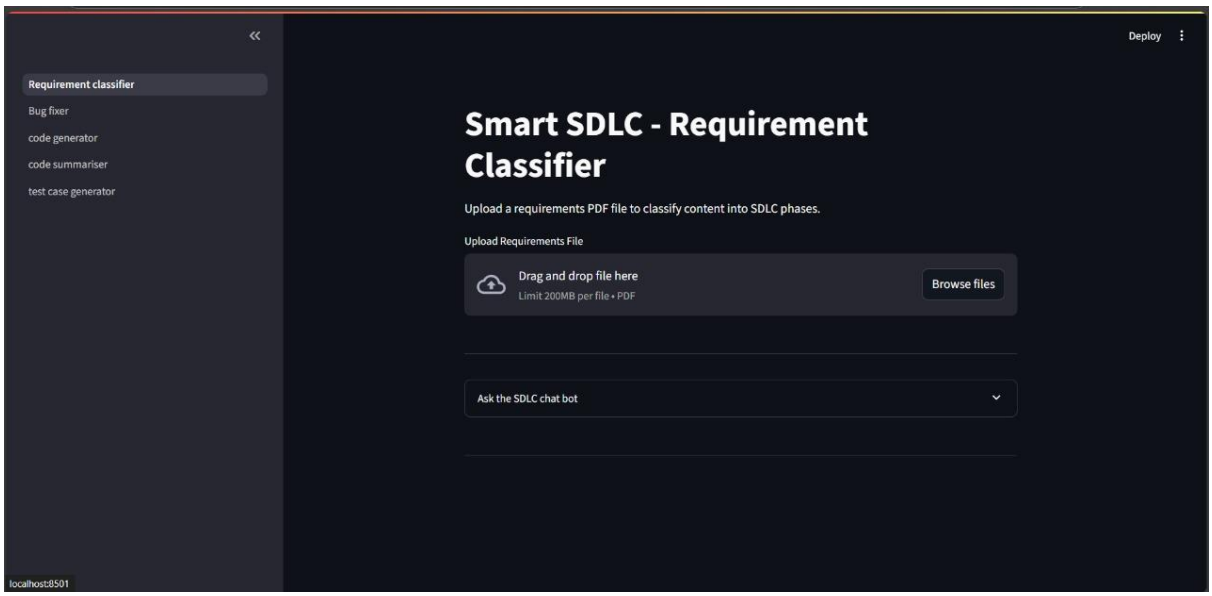
- **FT-01:** Text input validation ✔
- **FT-02:** Number input validation ✔
- **FT-03:** Content generation ✔
- **FT-04:** API connection check ✔
- **PT-01:** Response under 3s ✔
- **PT-02:** Parallel API calls ✔
- **PT-03:** Multiple file uploads ✔

---

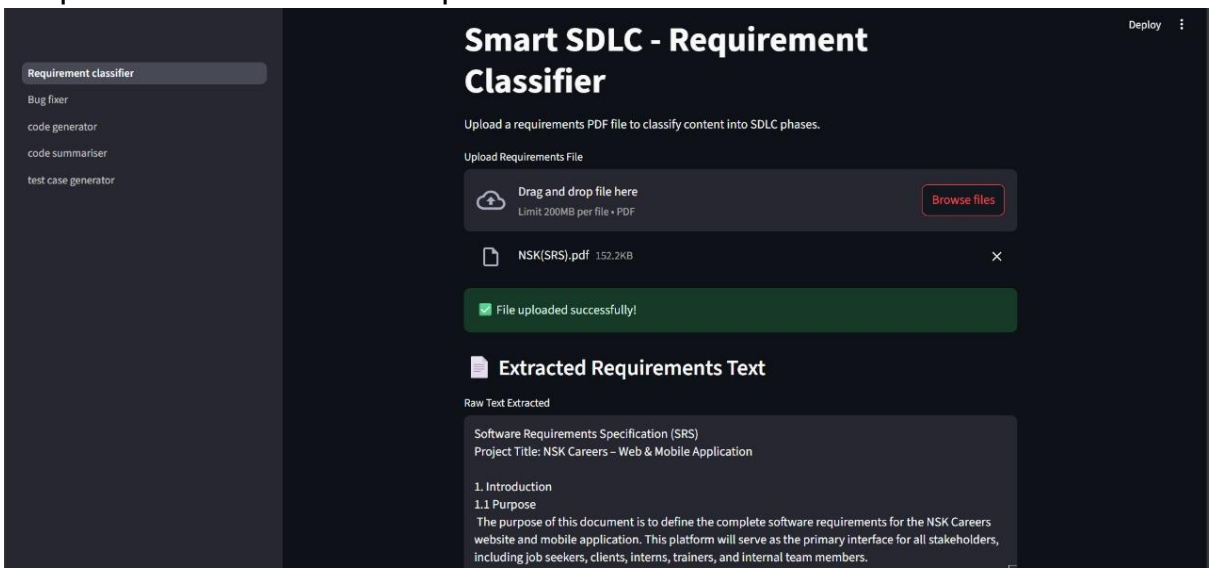
7. RESULTS

7.1 Output Screenshots

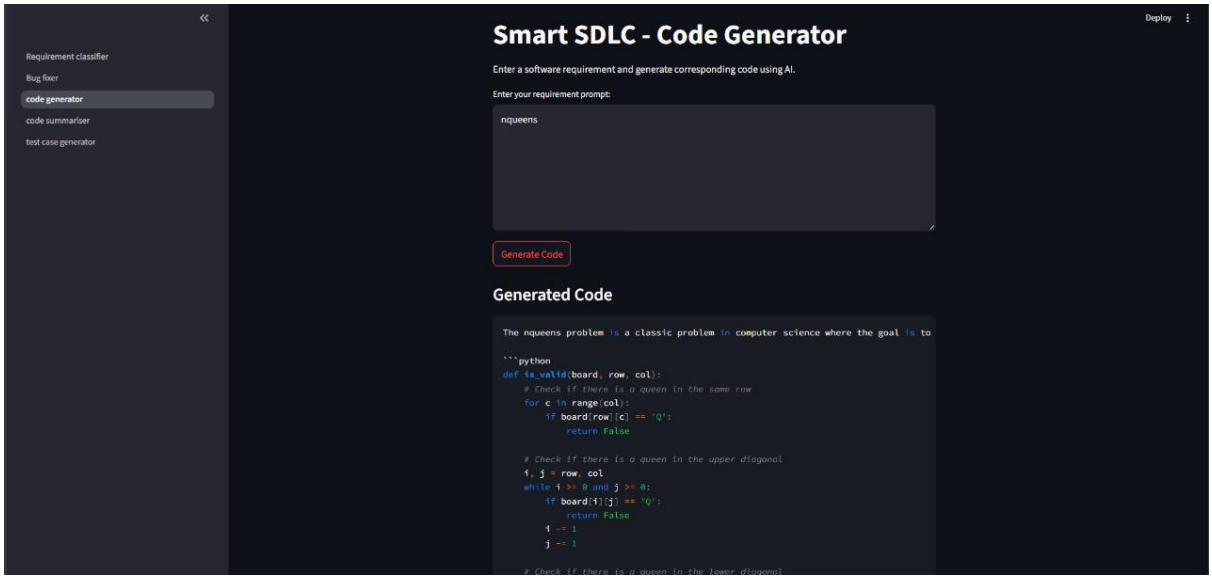
- Smart Dashboard Interface



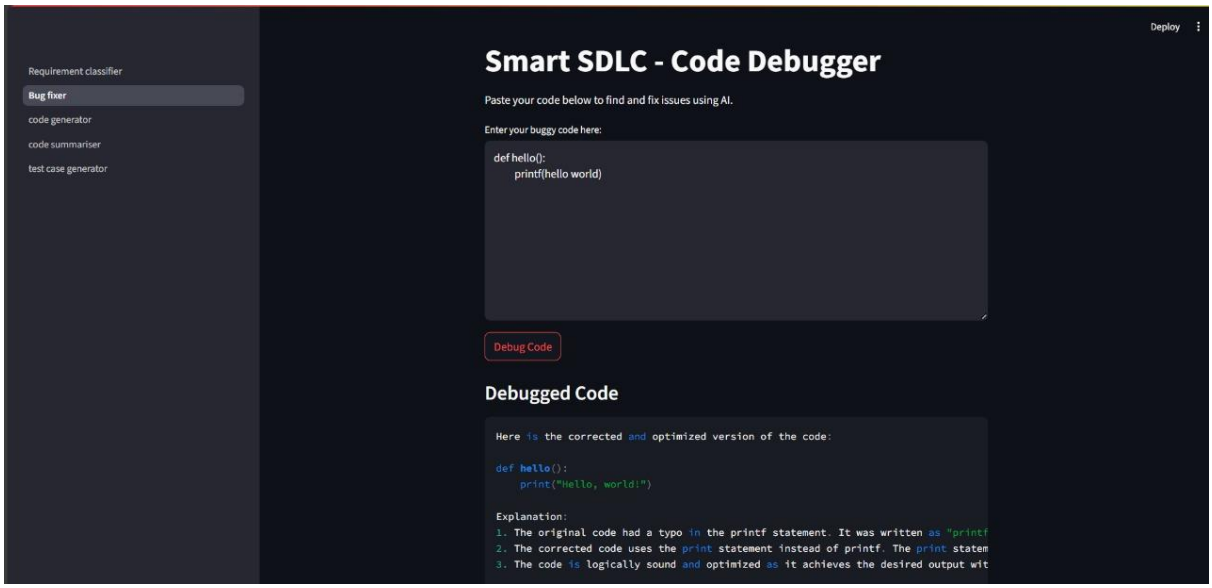
- Requirement Classifier Output



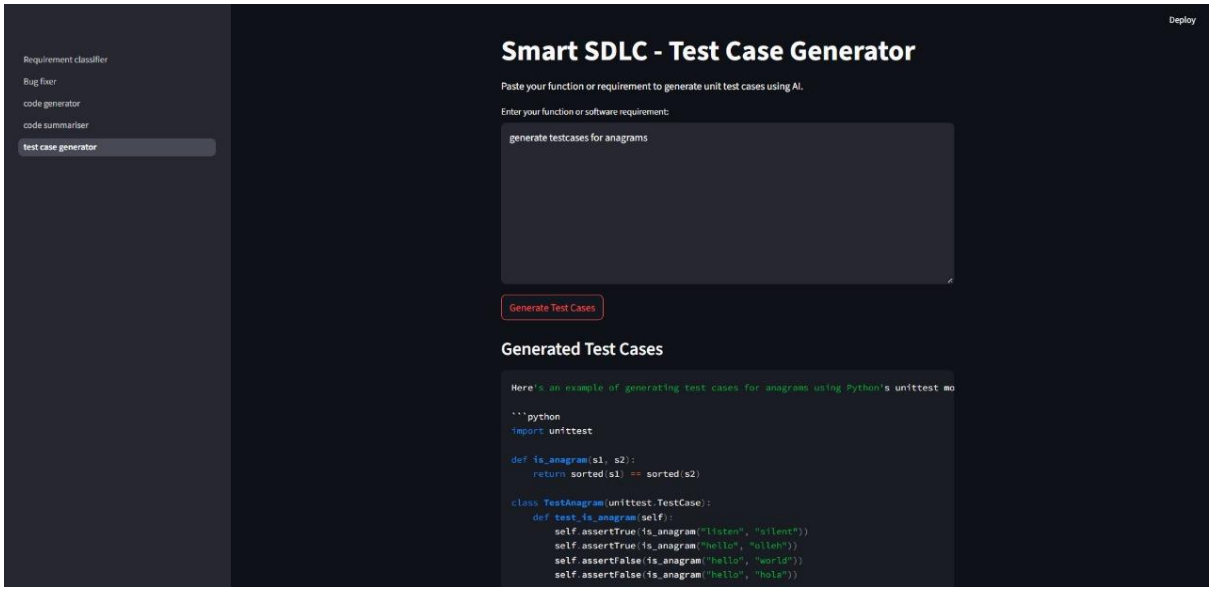
- AI Code Generator Output



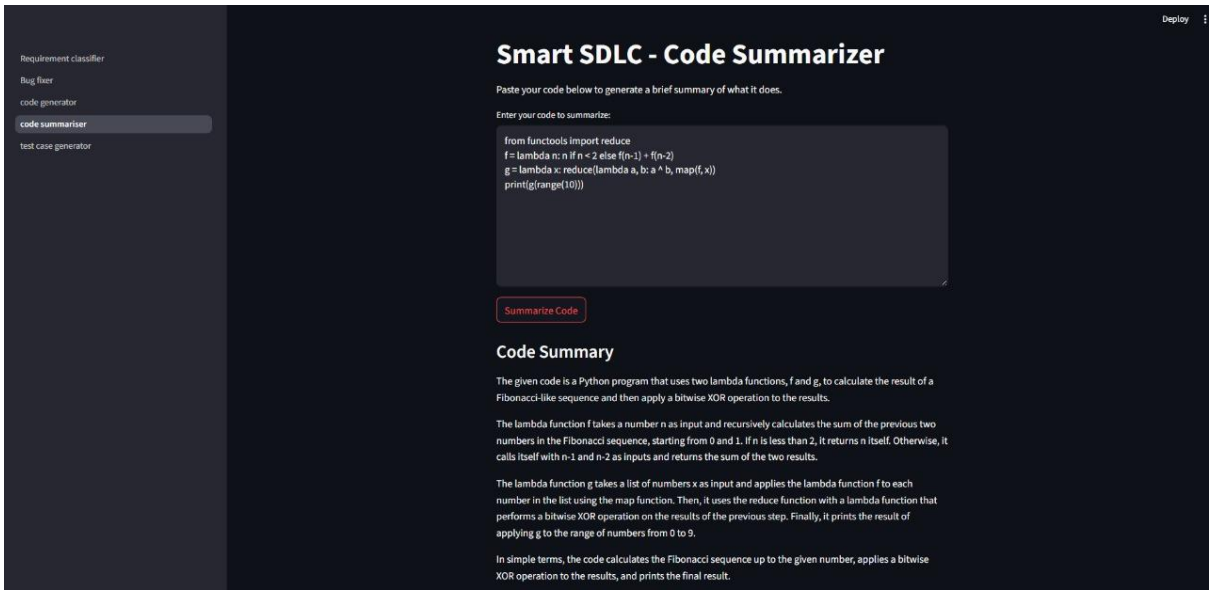
- Bug Fix Comparison



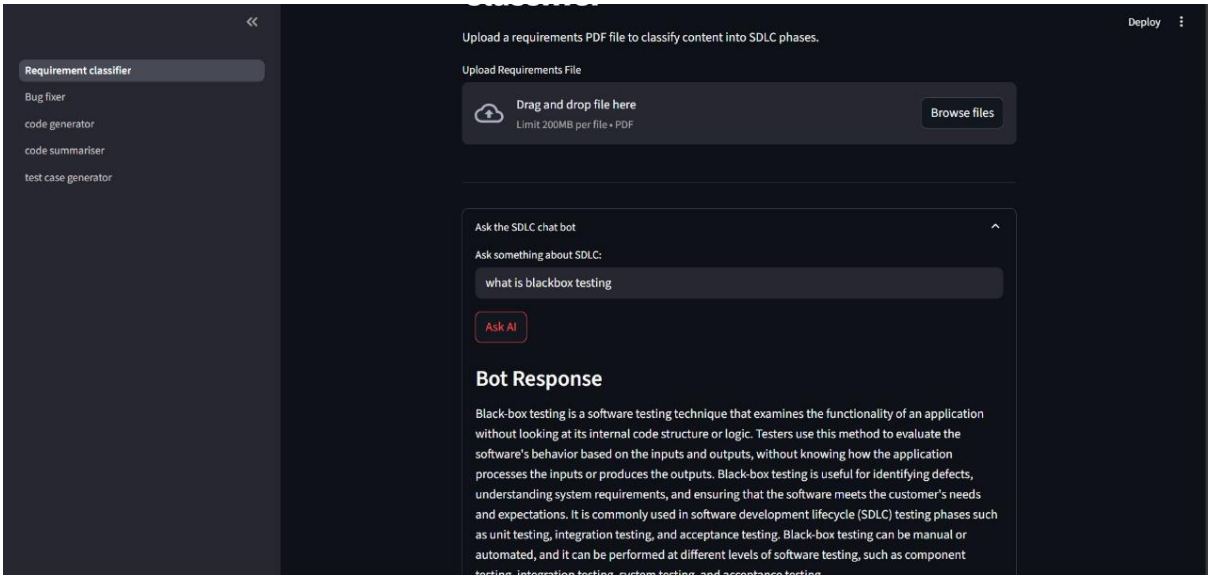
- Test Case Generator Display



- Code Summarizer Output



- Chatbot Interaction Logs



---

8. ADVANTAGES & DISADVANTAGES

Advantages:

- Reduces manual effort
- Faster SDLC cycle
- AI-powered clarity in requirements
- Modular architecture

Disadvantages:

- Dependency on AI model quality
- Limited customization per user
- Requires internet for Watsonx API access

---

9. CONCLUSION

SmartSDLC redefines software development by intelligently automating core SDLC tasks. It increases productivity, reduces development time, and empowers users with minimal technical expertise to navigate complex software workflows with the help of generative AI.

---

10. FUTURE SCOPE

- Cloud Deployment (IBM Cloud, Render)
- CI/CD pipeline integration
- Real-time team collaboration
- Multi-language support
- GitHub Actions and versioning

---

11. APPENDIX

GitHub Repository: *[https://github.com/gowtham-rajolu/ibm]*

End of Report