**Project Report Format**

**1. INTRODUCTION**

1.1 Project Overview:

| Phase | AI-Enhanced Functionalities |
| --- | --- |
| 1. Requirements Analysis | - AI chat assistant to gather user stories- Convert natural language to user requirements- Detect missing requirements |
| 2. System Design | - Auto-generate UML diagrams- Suggest best architectural patterns |
| 3. Code Generation | - Use LLM (e.g. IBM Granite 3.3) to generate backend/frontend code- Code templates based on prompt |
| 4. Testing | - AI-generated unit and integration test cases- Predict possible bugs from requirements |
| 5. Debugging | - LLM-based bug explanation and fixing suggestions |
| 6. Documentation | - Auto-generate API docs, READMEs, design docs |
| 7. Deployment | - Suggest CI/CD pipelines- Integrate with GitHub or GitLab |
| 8. Maintenance & Monitoring | - AI chatbot for issue resolution- Summarize logs and detect anomalies |
| 9. Project Management | - Smart sprint planning- Task prioritization and time estimation |

1.2 Purpose :

The purpose of the SmartSDLC project is to modernize and optimize the software development process by embedding Artificial Intelligence (AI) capabilities into every stage of the Software Development Lifecycle (SDLC).

2. **IDEATION PHASE**

2.1 Problem Statement

2.2 Empathy Map Canvas

2.3 Brainstorming

3. **REQUIREMENT ANALYSIS**

3.1 Customer Journey map

3.2 Solution Requirement

3.3 Data Flow Diagram

3.4 Technology Stack

4. **PROJECT DESIGN**

4.1 Problem Solution Fit

4.2 Proposed Solution

4.3 Solution Architecture

5. **PROJECT PLANNING & SCHEDULING**

5.1 Project Planning

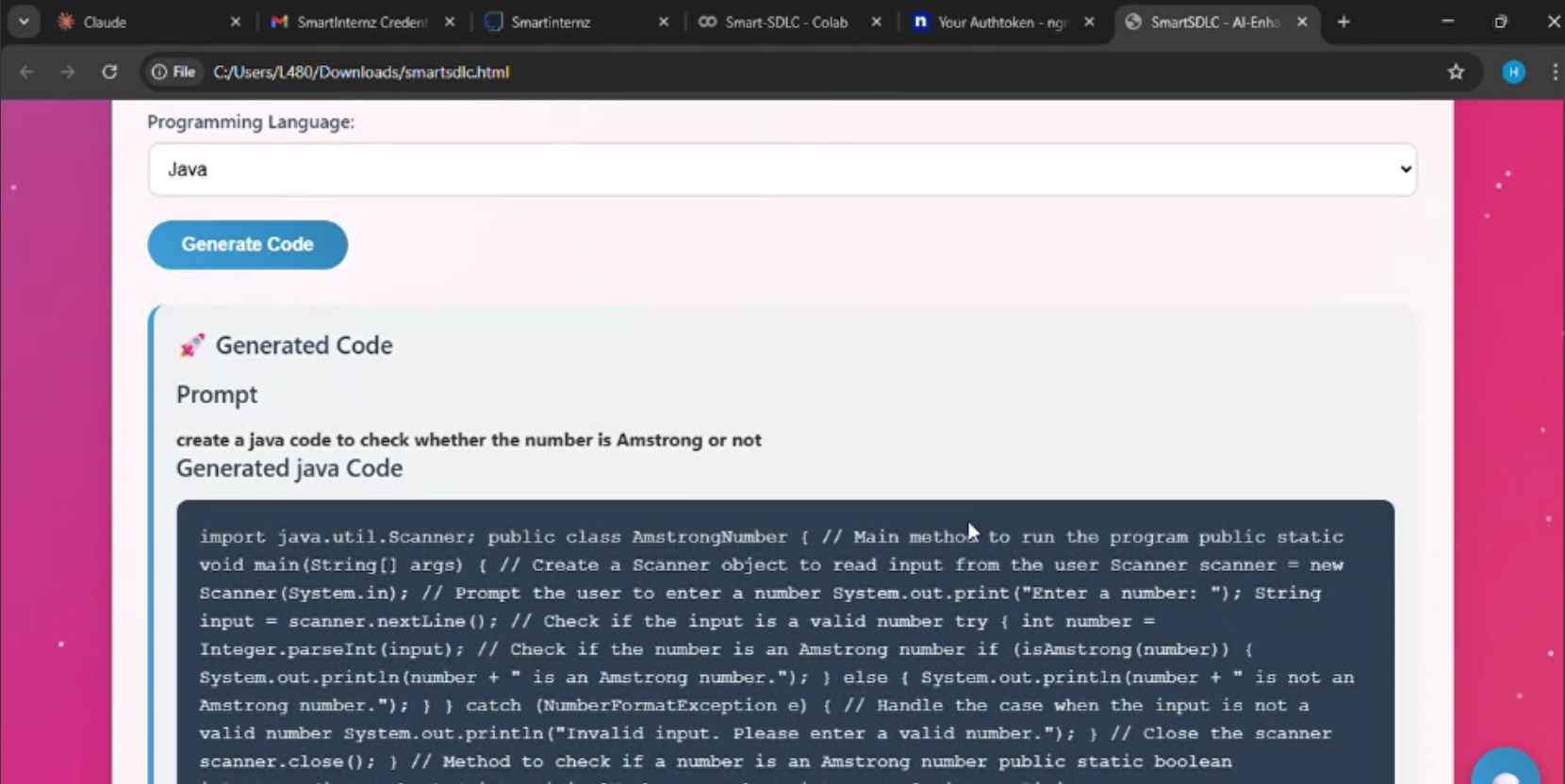
6. **FUNCTIONAL AND PERFORMANCE TESTING**

6.1 Performance Testing

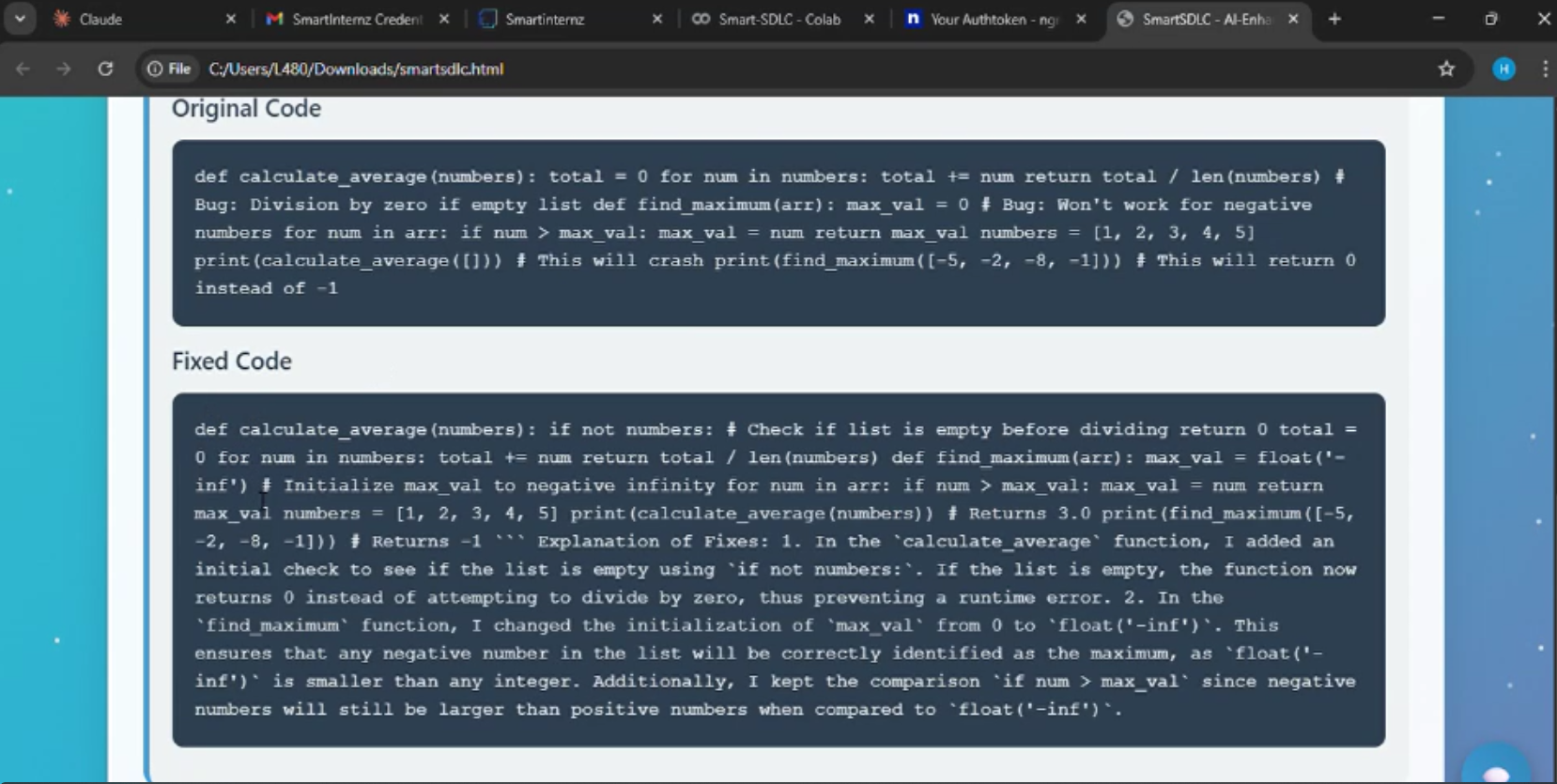
7. **RESULTS**

7.1 Output Screenshots:

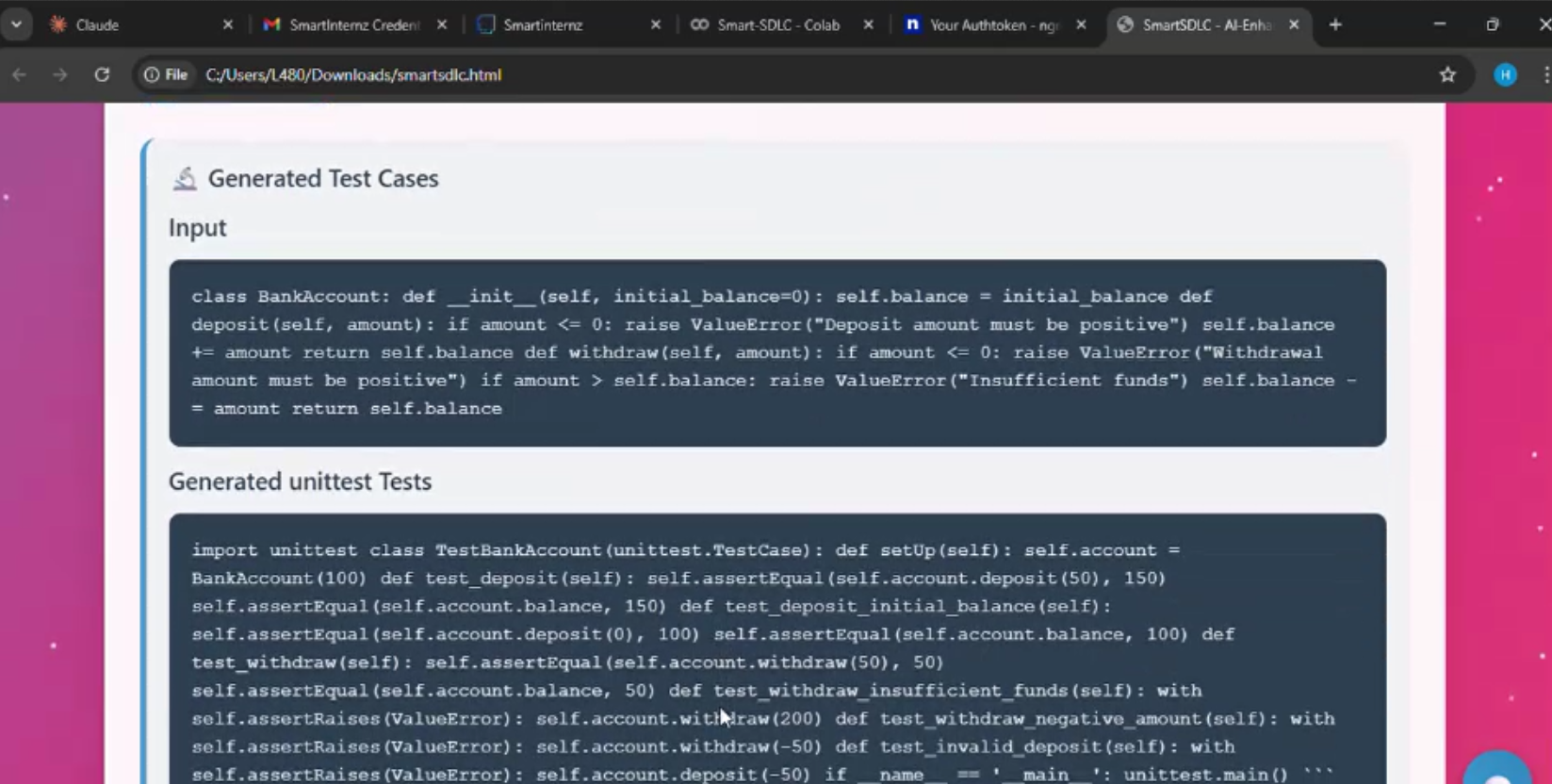
7.1.1 Code Generator:



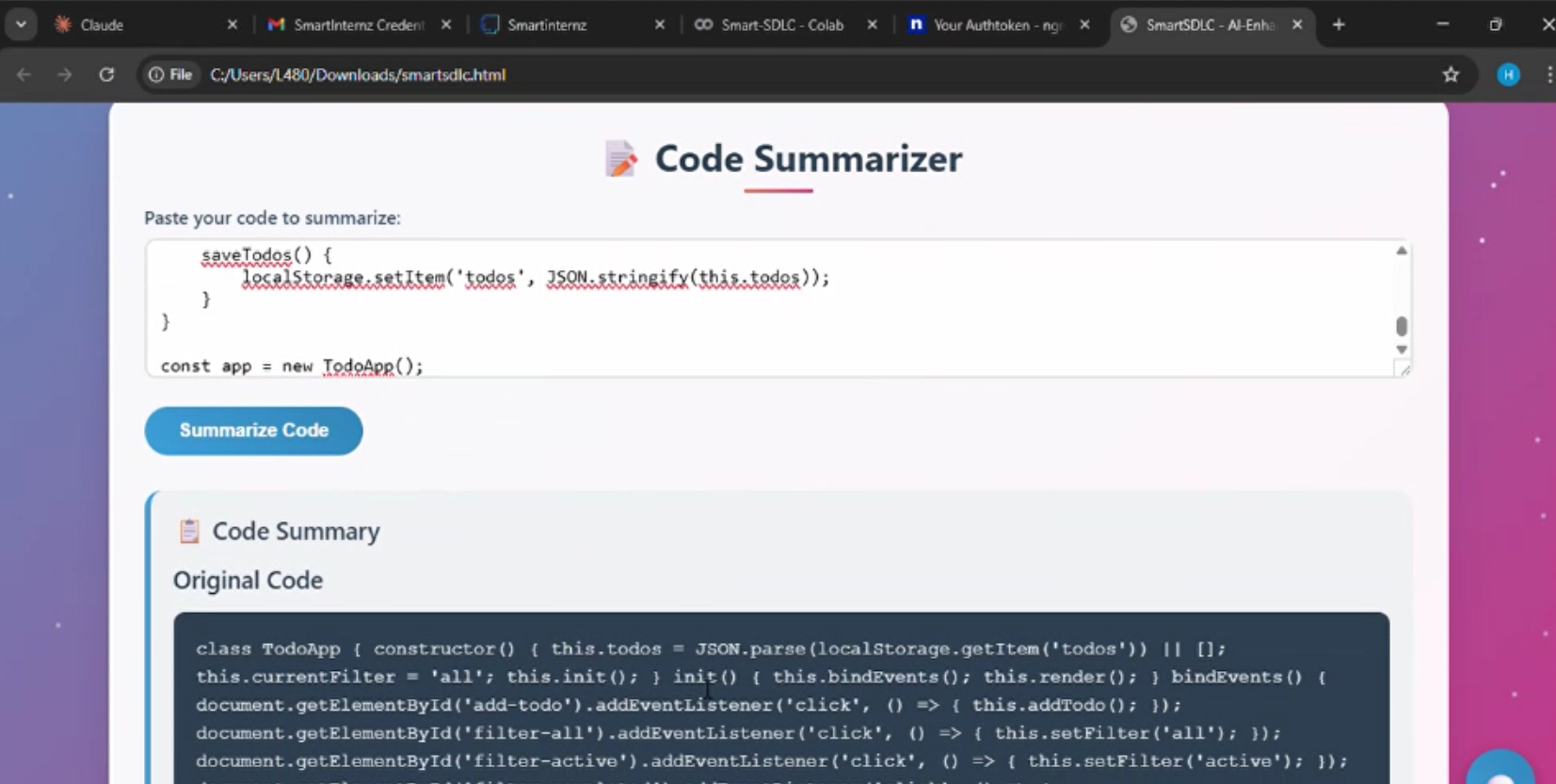
7.1.2 Bug Fixer:



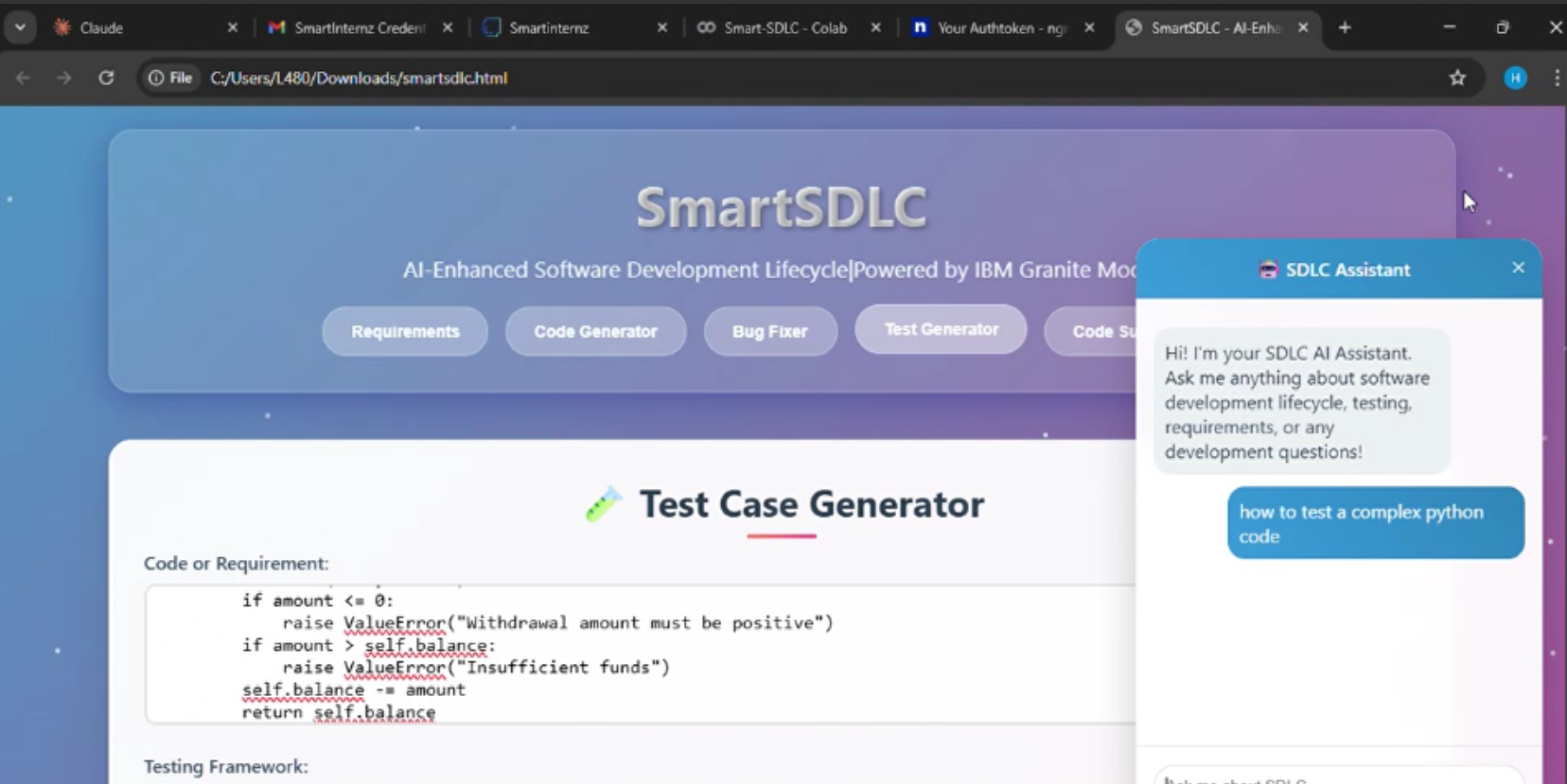
7.1.3 Test Generator:



7.1.4 Code Summerizer:



7.1.5 SDLC Assistant:



8. **ADVANTAGES & DISADVANTAGES**

| # | Advantages | Disadvantages |
| --- | --- | --- |
| 1 | Automates requirement analysis, coding, testing, and documentation | AI may misinterpret vague or unclear requirements |
| 2 | Saves time and effort for developers and teams | Initial setup of APIs and cloud services can be complex |
| 3 | Reduces manual errors and improves consistency | Heavy dependence on AI model accuracy and quality of prompts |
| 4 | Provides intelligent suggestions via LLMs (e.g., for bug fixing, code generation) | Not all domain-specific logic can be auto-generated without fine-tuning |
| 5 | User-friendly interface built using Streamlit | May require a stable internet connection to access cloud-based AI models |
| 6 | Scalable and modular via microservices and cloud deployment | Customizing backend or AI models can involve a learning curve |
| 7 | Supports GitHub integration and CI/CD workflows | Sensitive data must be handled with strict security measures |
| 8 | Enhances productivity through AI-driven insights and templates | Ongoing costs for cloud infrastructure and premium APIs |

9. **CONCLUSION**

*SmartSDLC revolutionizes the software development lifecycle by automating key stages like requirement analysis, code generation, testing, and documentation using advanced AI models. It boosts productivity, reduces errors, and simplifies development for teams.*

*Despite minor setup challenges and AI limitations, its impact on speed, scalability, and accuracy makes it a powerful tool for modern software engineering.*

10. **FUTURE SCOPE**

*1. Multimodal Input Support*

*2.AI Model Fine-Tuning*

*3.CI/CD Pipeline Integration*

*4.Real-Time Collaboration*

*5.Explainable AI (XAI) Integration*

*6.Mobile App Interface*

*7.AI-Powered Project Analytics*

*8.Plug-and-Play Architecture*

11. **APPENDIX**

Source Code(if any)

Dataset Link

GitHub & Project Demo Link:

https://github.com/LankeHarshitha/SDLC.git