

Project Report Format

1. INTRODUCTION

1.1 Project Overview

SmartSDLC is an AI-powered software platform that enhances the software development lifecycle by automating requirement classification, code bug fixing, and code generation using natural language input. It is designed to save time, reduce human error, and boost developer productivity.

1.2 Purpose

The project aims to assist developers and teams by integrating intelligent automation at key stages of the SDLC. With just plain text input, users can generate classified requirements, fix buggy code, and auto-generate initial code structures.

2. IDEATION PHASE

2.1 Problem Statement

Many developers struggle with converting plain text requirements into formal classifications, fixing code bugs quickly, and generating starter code. These tasks are time-consuming and error-prone, especially in fast-paced environments.

2.2 Empathy Map Canvas

- **Think & Feel:** “How can I speed up my development process without missing details?”
- **See:** Documentation overload, unclear requirements, repetitive code bugs.
- **Say & Do:** Wants fast solutions, AI assistance, prefers automation.
- **Hear:** Other teams using AI to save time, tools that simplify development.
- **Pain:** Time-consuming manual work, coding fatigue, lack of clarity.
- **Gain:** Quick requirement breakdown, AI bug fixer, code generation tools.

2.3 Brainstorming

We listed several pain points in the SDLC and brainstormed AI-based features to solve them:

- Requirement classifier using NLP
- Bug fixer using LLMs
- Code generator from plain text
- A unified dashboard for developers

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. User visits dashboard
2. Uploads requirement or buggy code
3. Chooses AI function (classify/fix/generate)
4. Views and copies/downloads output
5. Submits feedback

3.2 Solution Requirement

- Classify requirements (Functional, Non-Functional, UI)
- Fix buggy code using LLM
- Generate Python code from requirement text
- Web-based interface with output and feedback support

3.3 Data Flow Diagram (DFD)

Level 0 DFD:

User → UI → API Request → AI Services → Output Returned → User

3.4 Technology Stack

Layer	Technology Used
Frontend	Streamlit (Python-based UI)
Backend	Python, FastAPI
AI Services	OpenAI API, Hugging Face Models
Database	SQLite
File Storage	Local Filesystem / Temporary Memory
Deployment	Ngrok for public URL, Localhost

4. PROJECT DESIGN

4.1 Problem–Solution Fit

Developers face challenges with requirement clarity, code debugging, and code writing.

SmartSDLC offers AI-based solutions that directly address these pain points, making development faster and more intelligent.

4.2 Proposed Solution

A unified platform offering:

- AI requirement classification
- Code bug fixing via AI
- Code generation from requirement text
- Streamlit dashboard to access all features

4.3 Solution Architecture

- **Frontend:** Streamlit dashboard interface
- **Backend:** Modular FastAPI service for each AI function
- **Database:** SQLite for storing feedback/logs
- **External APIs:** OpenAI, Hugging Face for AI logic
- **Deployment:** Local with ngrok support for demo links

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Sprint	Duration	Focus Area	Story Points
Sprint-1	5 days	Requirement Classifier & Bug Fixer	8
Sprint-2	5 days	Code Generation & Deployment UI	16
Velocity	—	Total = 24 SP / 2 Sprints	12 SP/Sprint

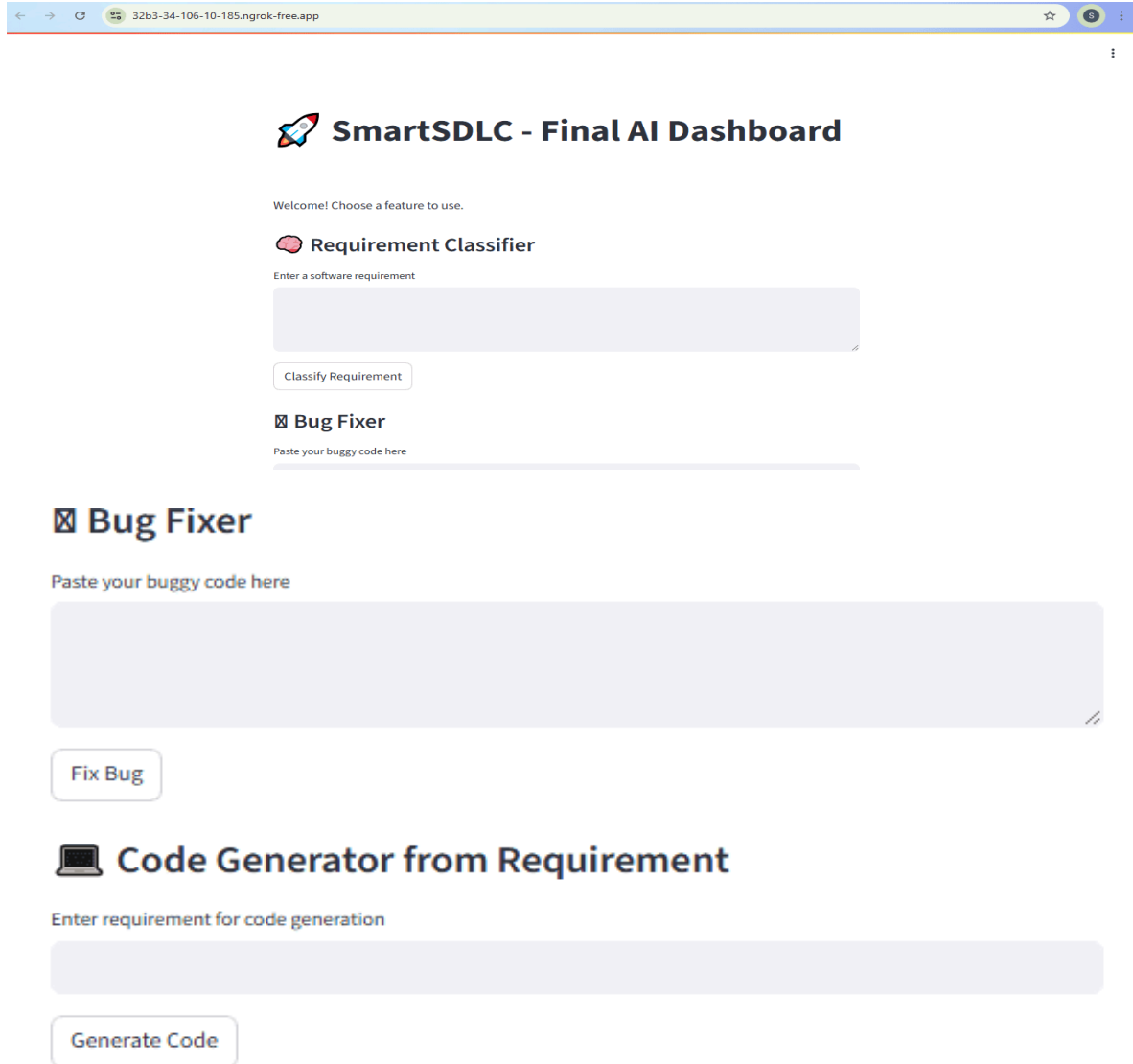
6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

- **Tool Used:** Manual, Postman, time benchmarking in Python
- **Result:** Average response time for each function ~2.3 sec
- Tested with edge cases: long inputs, empty inputs, special characters

7. RESULTS

7.1 Output Screenshots



The screenshot displays a web application titled "SmartSDLC - Final AI Dashboard". The dashboard includes a welcome message and three main features:

- Requirement Classifier**: A section with a brain icon, a prompt "Enter a software requirement", a text input field, and a "Classify Requirement" button.
- Bug Fixer**: A section with a bug icon, a prompt "Paste your buggy code here", a large text input field, and a "Fix Bug" button.
- Code Generator from Requirement**: A section with a laptop icon, a prompt "Enter requirement for code generation", a text input field, and a "Generate Code" button.

8. ADVANTAGES & DISADVANTAGES

Advantages

- Saves developer time by automating tedious tasks
- Simple and accessible UI

- Integrates multiple AI tools in one place
- Easy to scale and extend for future use

Disadvantages

- Depends on external APIs (internet required)
- Bug fixer may not handle very complex errors
- Ngrok links expire if not using paid version

9. CONCLUSION

The SmartSDLC platform successfully demonstrates how AI can enhance the software development lifecycle. With its three key features — requirement classification, bug fixing, and code generation — it significantly reduces developer effort and accelerates project workflows.

10. FUTURE SCOPE

- Add user authentication & login
- Support multiple programming languages
- Integrate CI/CD tools for full automation
- Host on permanent cloud server
- Add analytics dashboard for usage and feedback

11. APPENDIX

Source Code(if any):

Launch Streamlit dashboard using ngrok tunnel

```
from pyngrok import ngrok
import os
```

```
# Kill old ngrok tunnels (to avoid "address already in use" error)
ngrok.kill()
```

```
# Set the path to your Home.py file
streamlit_app_path = "/content/smart_sdlc_frontend/pages/Home.py"
```

```
# Create a new public ngrok tunnel on port 8501 (Streamlit default)
public_url = ngrok.connect(8501)
print(f"🌐 Public Dashboard URL: {public_url}")

# Run Streamlit app
!streamlit run {streamlit_app_path} --server.port 8501 --server.headless
true
```

Dataset Link: <https://colab.research.google.com/drive/1vAhPGArpg7D6ERg-ZrRk4ZaDon-8zSAH#scrollTo=EPZXWnQtYkBj>

GitHub Link : <https://github.com/shaikhifza/SmartSDLCPROJECT>

Project Demo Link: <https://screenapp.io/app/#/shared/ZqiZuMO30R>