SMARTSDLC: AI Enhanced Software Development Lifecycle

Project Documentation format

1. Introduction

• **Project Title:** SmartSDLC – AI-Enhanced Software Development Lifecycle

• **Team Members:**

Team Members	**Role**
Shaik Hifza	Team Leader and Project Manager
Anand Kumar	Frontend Developer
Shaik Irfan Basha	Backend Developer
Ramisetty Ajay Kumar	AI Integrator

2. Project Overview

• **Purpose:** The purpose of SmartSDLC is to automate and enhance key stages of the software development lifecycle using AI. It reduces manual effort in classifying requirements, fixing bugs, and generating code, thereby increasing developer productivity and project quality.

• **Features:**

AI-powered Requirement Classifier

Automated Bug Fixing System

Natural Language to Code Generator

Streamlit Dashboard with easy-to-use interface

Public accessibility via ngrok for demo/testing

Modular backend and scalable architecture

3. Architecture

• **Frontend:** The frontend is built using Streamlit (a Python-based framework for UI), structured to display three main features: Requirement Classifier, Bug Fixer, and Code Generator. Each feature has its own input/output panel for user interaction.

● **Backend:** The backend consists of Python FastAPI services organized into AI modules. Each module (classifier, fixer, generator) is independently callable through RESTful endpoints. Communication between Streamlit frontend and backend is handled via API calls.

● **Database:** A simple SQLite database is used to store feedback, logs, and example inputs/outputs. The database is connected to the backend using SQLAlchemy for easy object-relational mapping.

4. Setup Instructions

• Prerequisites:

Python 3.10+

pip (Python package manager)

Ngrok (for public URL)

Streamlit

Git

• Installation:

```
git clone https://github.com/your-repo/smartsdlc.git

cd  smartsdlc

pip install -r requirements.txt
```

Set up environment variables (e.g., OpenAI/Hugging Face API keys) in a .env file:

OPENAI_API_KEY= os.getenv("WATSONX_API_KEY")

HUGGINGFACE_TOKEN=2yzdkGccsge5XPxYeUGQwQFfdCh_3t8syj3kx1TxDyqRcY3v

5. Folder Structure

• Client:

```
smart_sdlc_frontend/
├── app.py              # Main dashboard
├── pages/              # Subpages for each feature
├── assets/             # Images, logos
└── style.css           # Custom styling
```

• Server:

```
smart_sdlc_backend/
├── main.py             # FastAPI app
├── app/
│   ├── models/          # Data models
│   ├── routes/          # API routes for classifier, fixer, generator
│   └── services/        # Core AI logic
└── db/                  # SQLite database + init scripts
```

6. Running the Application

commands to start the frontend and backend servers locally

o Frontend:

```
cd smart_sdlc_backend
uvicorn main:app --reload
```

o Backend:

```
cd smart_sdlc_frontend
streamlit run app.py
```

7. API Documentation

Endpoint	**Method**	**Description**
/classify	POST	Classify input requirements into functional, non-functional, UI
/fix-code	POST	Accept buggy code and return AI-fixed version
/generate-code	POST	Convert natural language requirement into code
/feedback	POST	Submit feedback data
/feedback	GET	View all submitted feedback

8. Authentication

Token-based authentication (JWT)

Admin access for viewing user feedback and logs

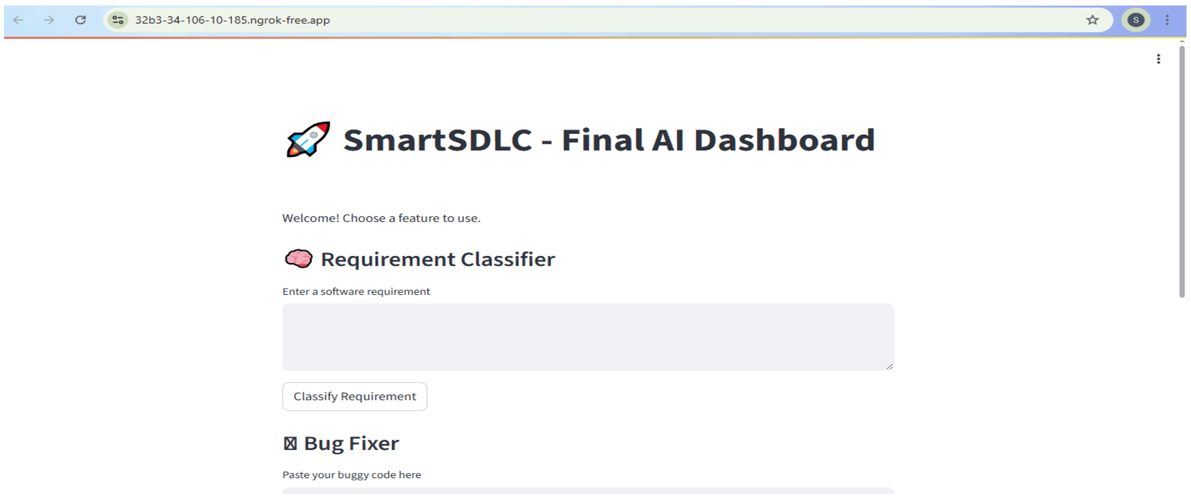
9. User Interface

- Simple and clean Streamlit-based UI
- Three sections: Requirement Classifier, Bug Fixer, and Code Generator
- Users can:
  - Paste input
  - Click "Submit"
  - View and copy/download results
- Feedback submission form included for user input

10. Testing

- **Tools Used:**
  - Manual Testing via UI
  - Postman for API validation
  - Python unittest for backend services
  - Mock input testing for AI components
- **Strategy:**
  - Functional testing of each AI module independently
  - End-to-end flow testing from frontend to backend
  - Edge cases: missing input, invalid code, unsupported languages

11. Screenshots or Demo



🚀 SmartSDLC - Final AI Dashboard

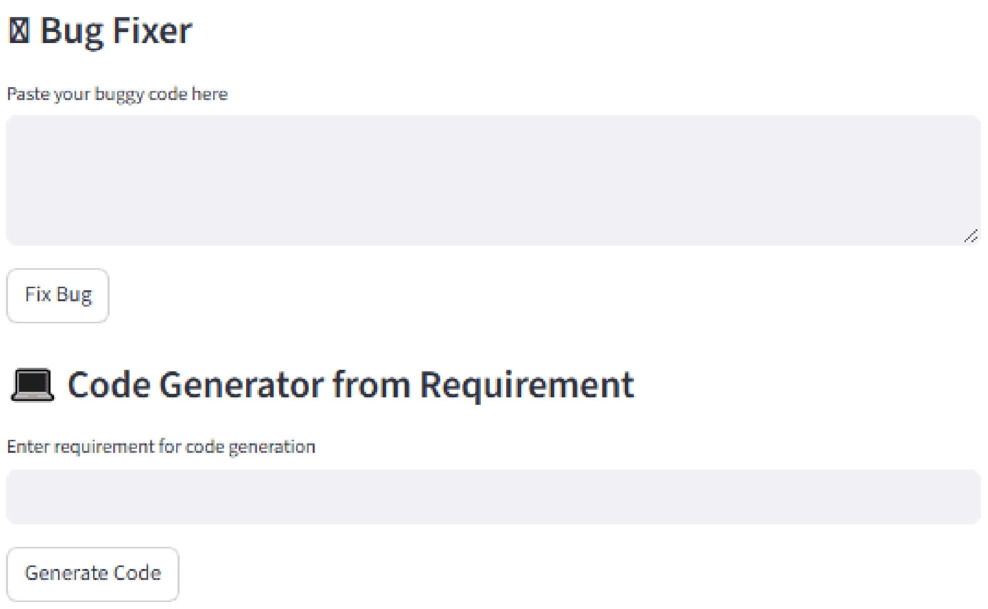
Welcome! Choose a feature to use.

🧠 Requirement Classifier

Enter a software requirement

⊠ Bug Fixer

Paste your buggy code here



⊠ Bug Fixer

Paste your buggy code here

Fix Bug

💻 Code Generator from Requirement

Enter requirement for code generation

Generate Code

Demo link : https://screenapp.io/app/#/shared/ZqiZuMO30R

12. Known Issues

- May fail on highly complex or domain-specific requirements
- Bug fixer sometimes misses syntax nuances in rare languages
- Code generation is basic and ideal for simple use cases
- Ngrok link needs to be refreshed every session (unless premium)

13. Future Enhancements

- Add login/authentication with session management
- Support more programming languages in code generation
- Enable drag-and-drop file input
- Connect to cloud database (Firebase/MongoDB Atlas)
- Add analytics dashboard for admin insights
- Offline export feature for SDLC documents