**SmartSDLC: AI-Enhanced Software Development Lifecycle**

## **Project Documentation**

Team ID : LTVIP2025TMID32004

Team Size: 4

Team Leader:  Kuraganti Bharath

Team Member: Kasturi Sandeep

Team Member: Kantamaneni Prudhvi

Team Member: Kode Devi Sri Vallika

### **1.Introduction**

SmartSDLC is an AI-powered software application designed to automate critical phases of the Software Development Lifecycle (SDLC). Built using Python and Streamlit, and integrated with IBM Watsonx's Granite 3.3 Instruct model, it transforms textual requirements into code, test cases, summaries, and even fixes bugs through natural language interaction.

**2. Project Overview**

The purpose of SmartSDLC is to minimize human effort in software planning, development, and quality assurance by offering AI-driven support at every major SDLC stage.

Key Features:

- Requirement Upload & Classification

- AI Code Generator

- Bug Fixer

- Test Case Generator

- Code Summarizer

- Chat Assistant

Each feature integrates IBM's Granite model to interpret user input and generate relevant code or insights.

**3. Architecture**

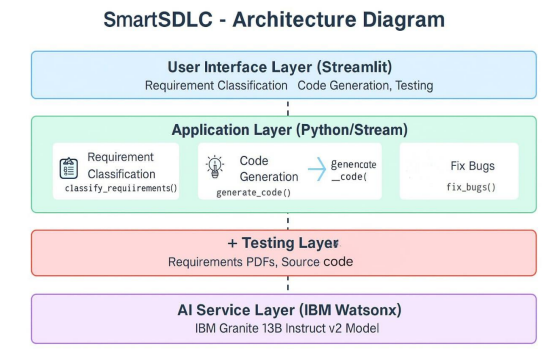
SmartSDLC follows a clean, modular architecture:

1. \*\*User Interface (Streamlit)\*\*: Collects input, displays results, and manages interactions.

2. \*\*Application Logic (Python)\*\*: Processes user commands, forms AI prompts, and handles session state.

3. \*\*AI Service Layer (IBM Watsonx)\*\*: IBM Granite 3.3 Instruct model generates context-aware outputs.

4. \*\*Temporary State Memory\*\*: User session and intermediate data stored in-memory using Streamlit session state



**4. Setup Instructions**

**Prerequisites:**

- Python 3.8 or above

- IBM Cloud account with Watsonx access

- Streamlit, pandas, python-dotenv, ibm-watsonx-ai

**Installation Steps**:

1. Clone the project and navigate to the folder

2. Create a virtual environment: `python -m venv venv`

3. Activate the virtual environment: - Windows: `.\venv\Script s\activate`

4. Install dependencies: `pip install -r requirements.txt`

5. Create `.env` file with the following: IBM\_API\_KEY="your\_key"

PROJECT\_ID="your\_project\_id" BASE\_URL="https://eu-de.ml.cloud.ibm.com"

6. Run the app: `streamlit run SMART\_SDLC.py`

**5. API Documentation**

SmartSDLC does not expose traditional REST APIs but uses IBM Watsonx's `generate\_text()` method via the Python SDK.

**Example Prompt Usage:**

- Code Generation: "Generate Python code that implements a login system."

- Bug Fixing: "Fix the following code: [BUGGY CODE]"

- Testing: "Write pytest unit tests for the given function.

**AI Parameters:**

- max\_new\_tokens = 500

- temperature = 0.7

- top\_p = 1.0

- decoding\_method = sample

**6. Authentication**

IBM Watsonx is accessed using secure API key authentication.

Credentials are stored in a `.env` file and loaded using `python-dotenv`

**Security Practice:**

- Do not hardcode API keys.

- Use `.env` and `.gitignore` to prevent accidental exposure.

- Only load variables at runtime.

**7. User Interface**

The app features a sidebar for navigation and module selection. Each module accepts different input types and shows results using `st.code`, `st.text\_area`, or `st.chat\_input`.

**Modules:**

- Requirement Upload → file uploader

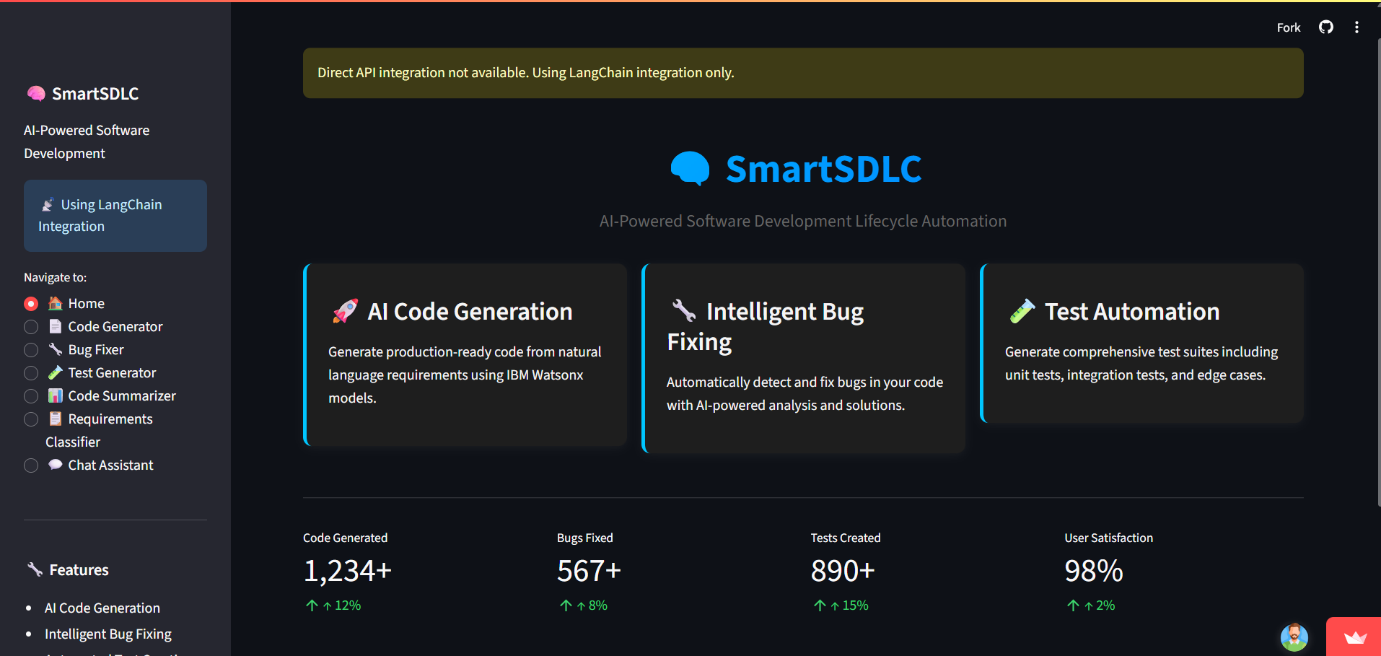
- Code Generator → text area prompt

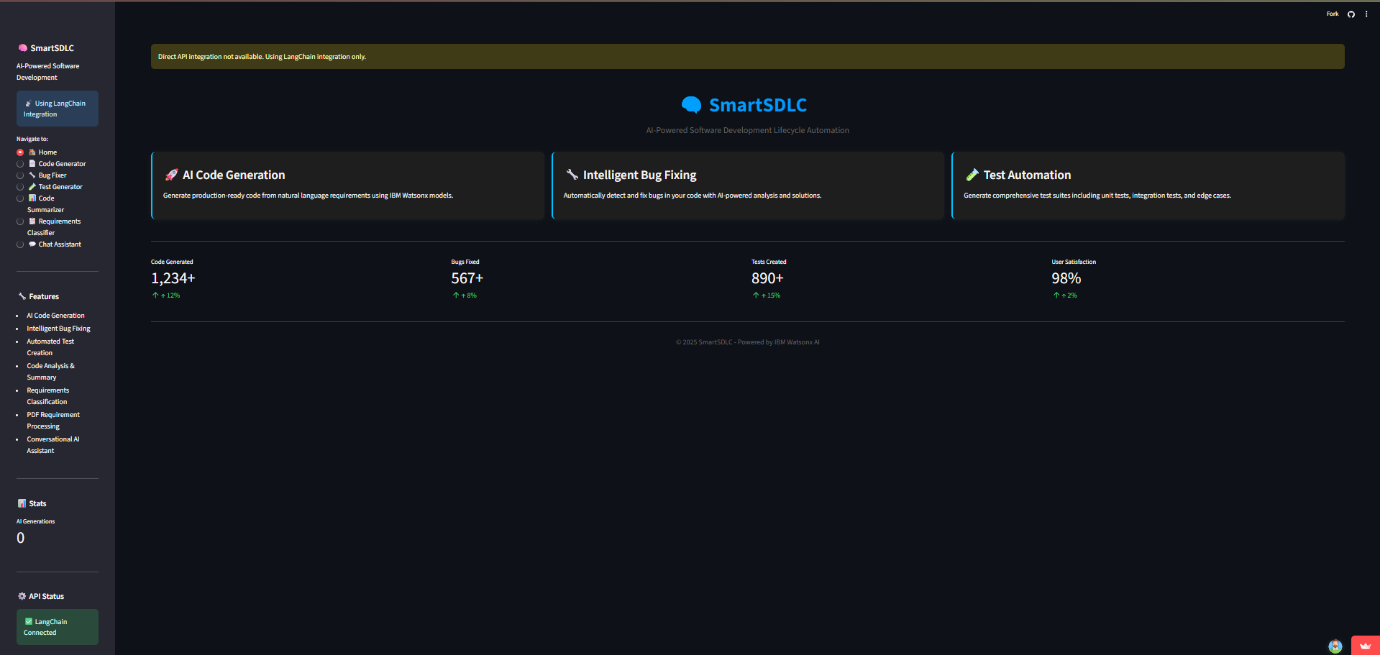
- Bug Fixer → code input

- Test Generator → requirement/code input

- Code Summarizer → code input

- Chatbot → natural language Q&A





**8. Testing**

SmartSDLC includes several test mechanisms:

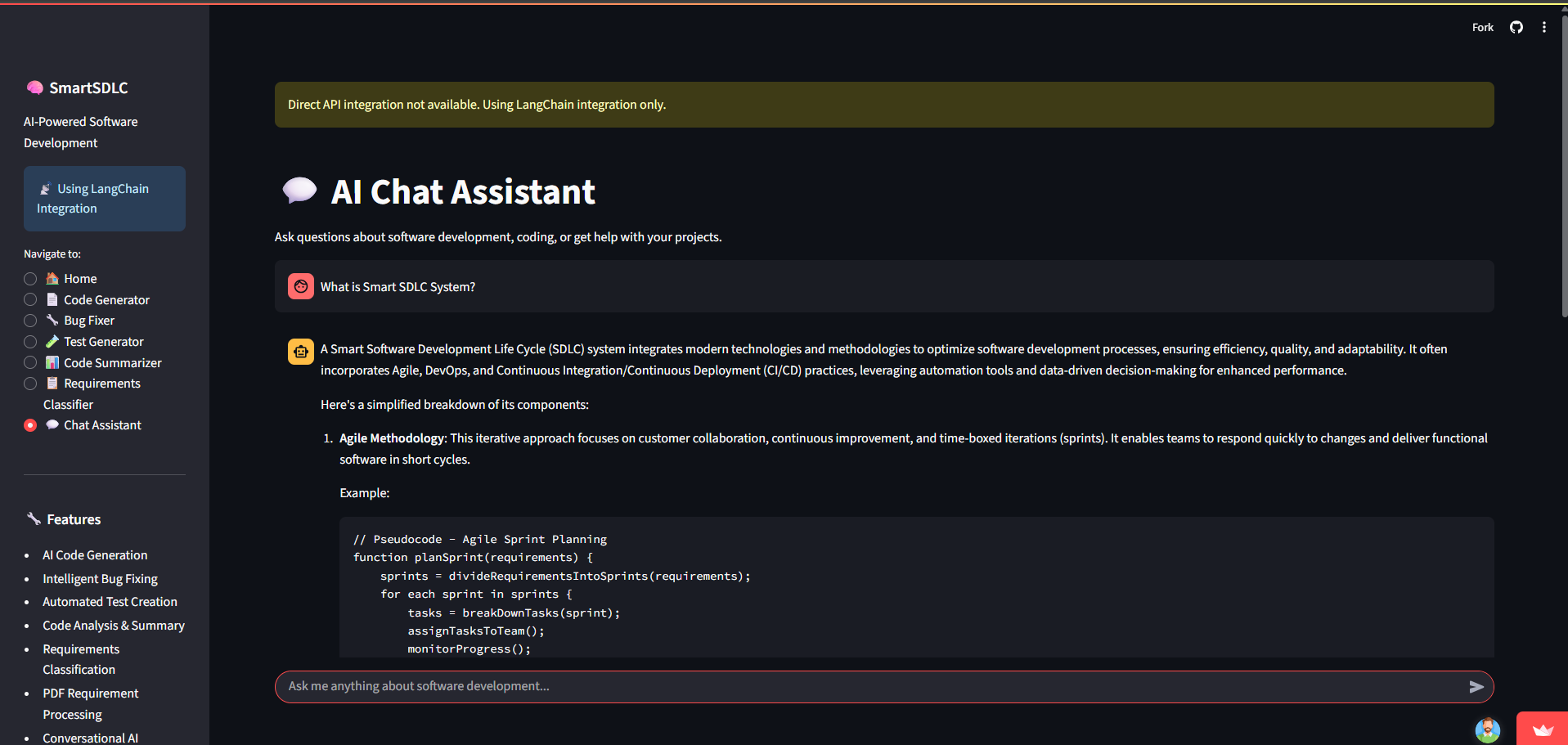
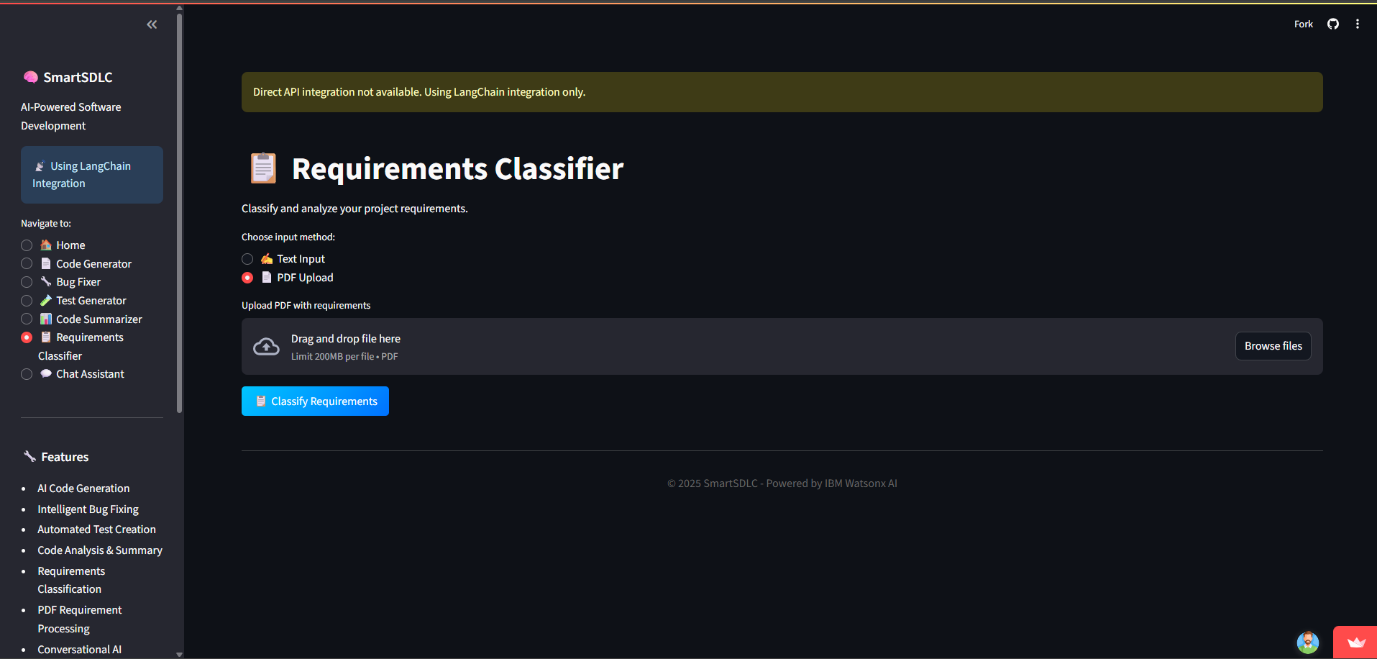
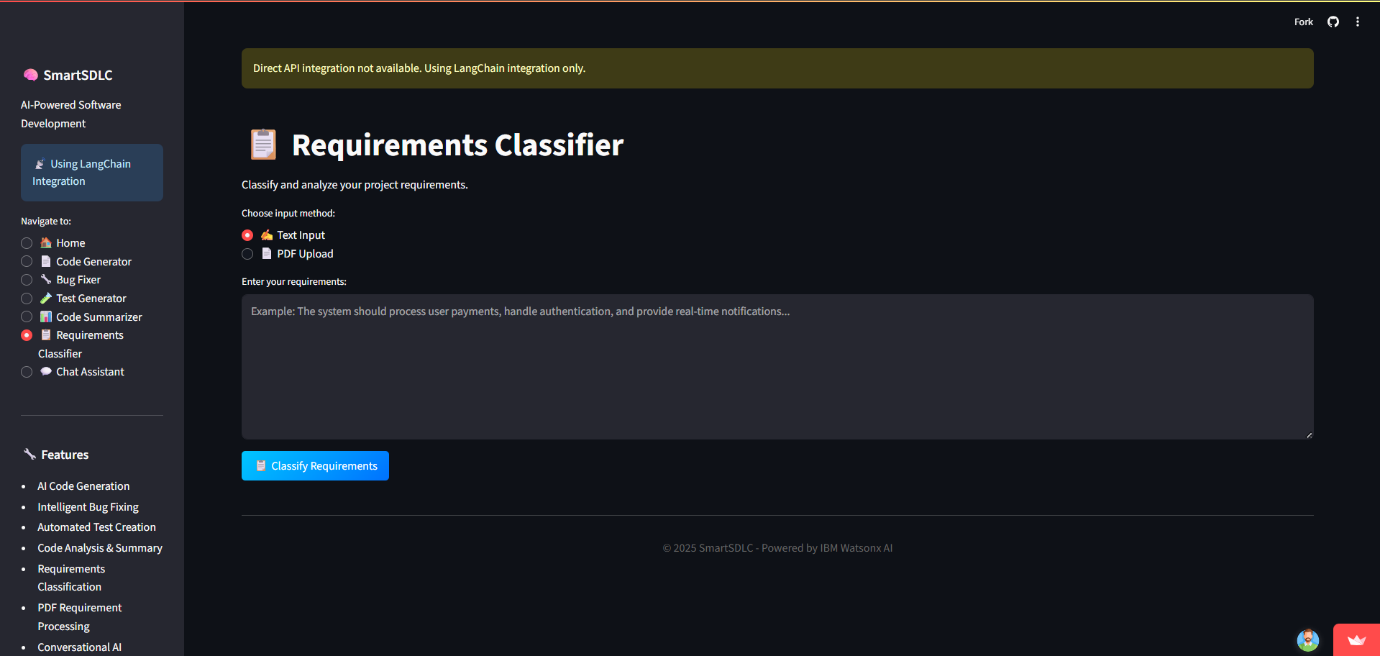
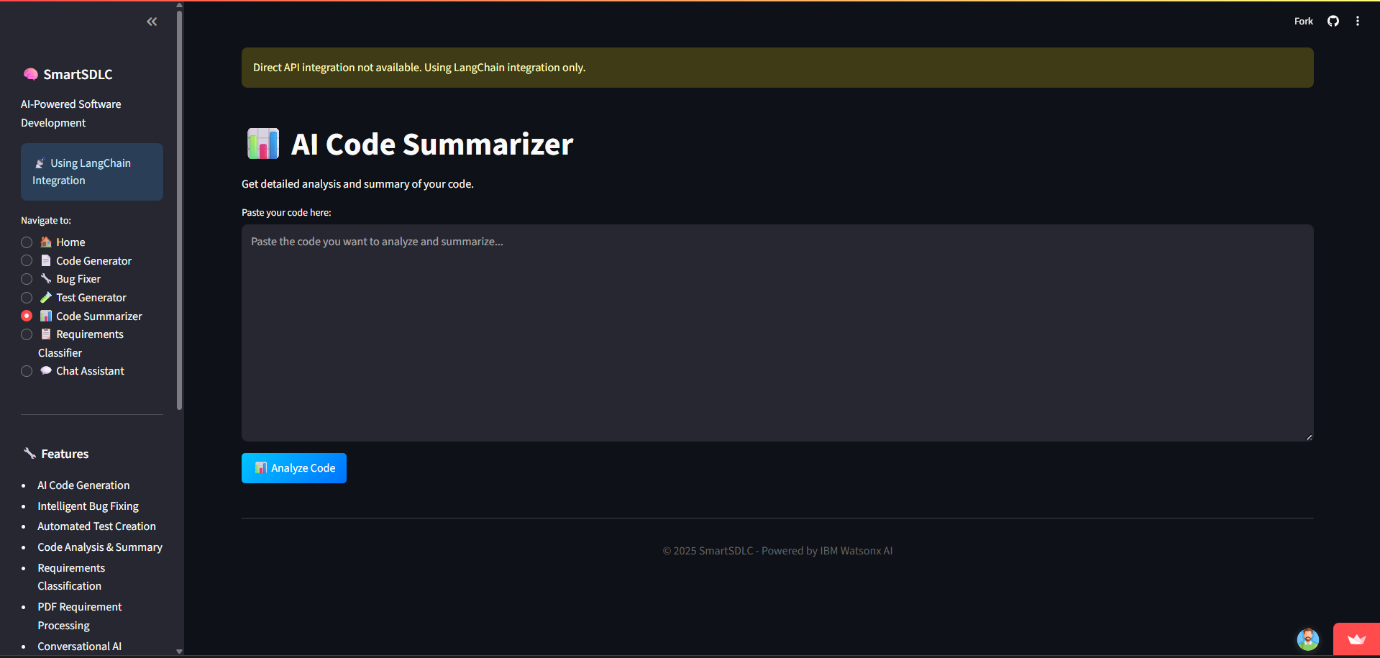
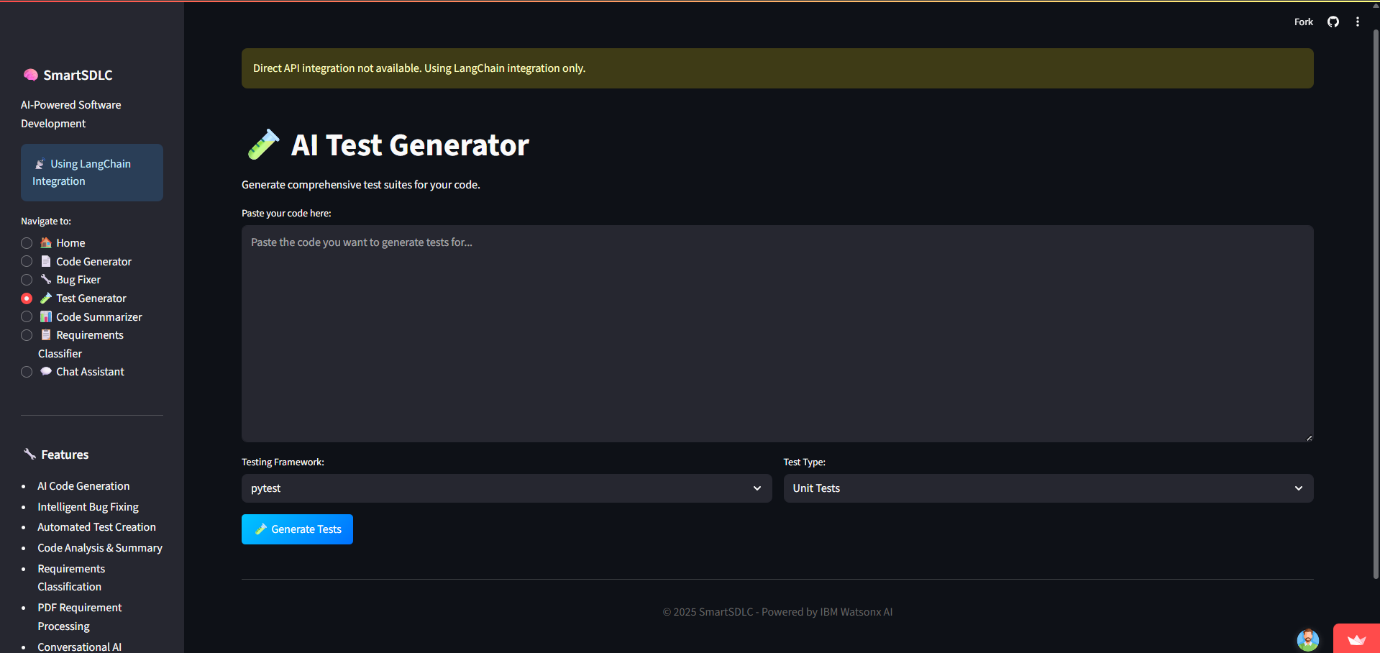
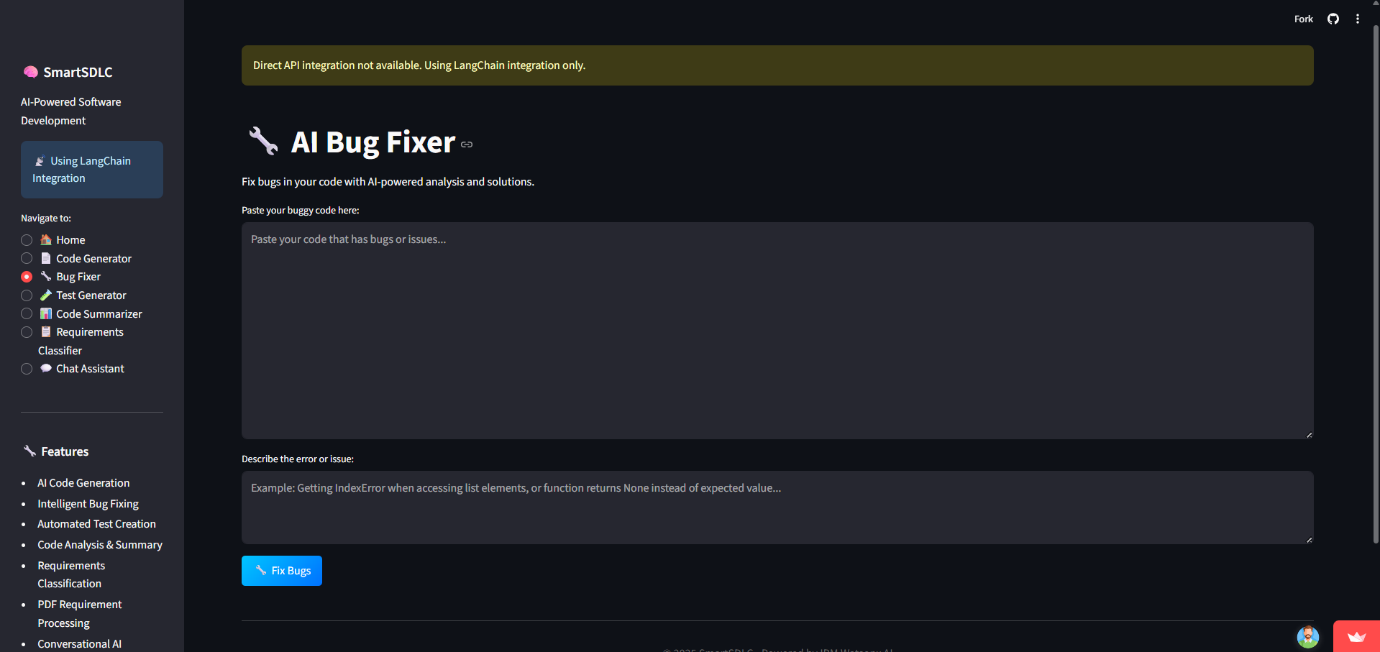
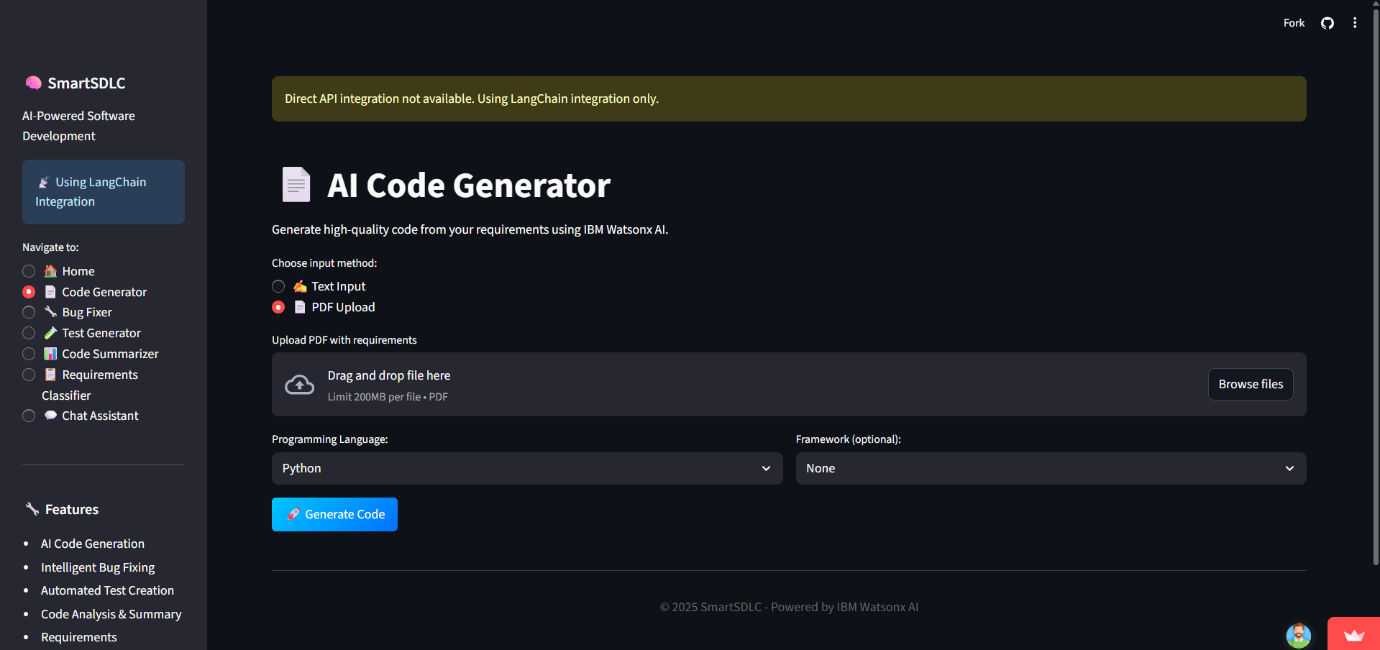
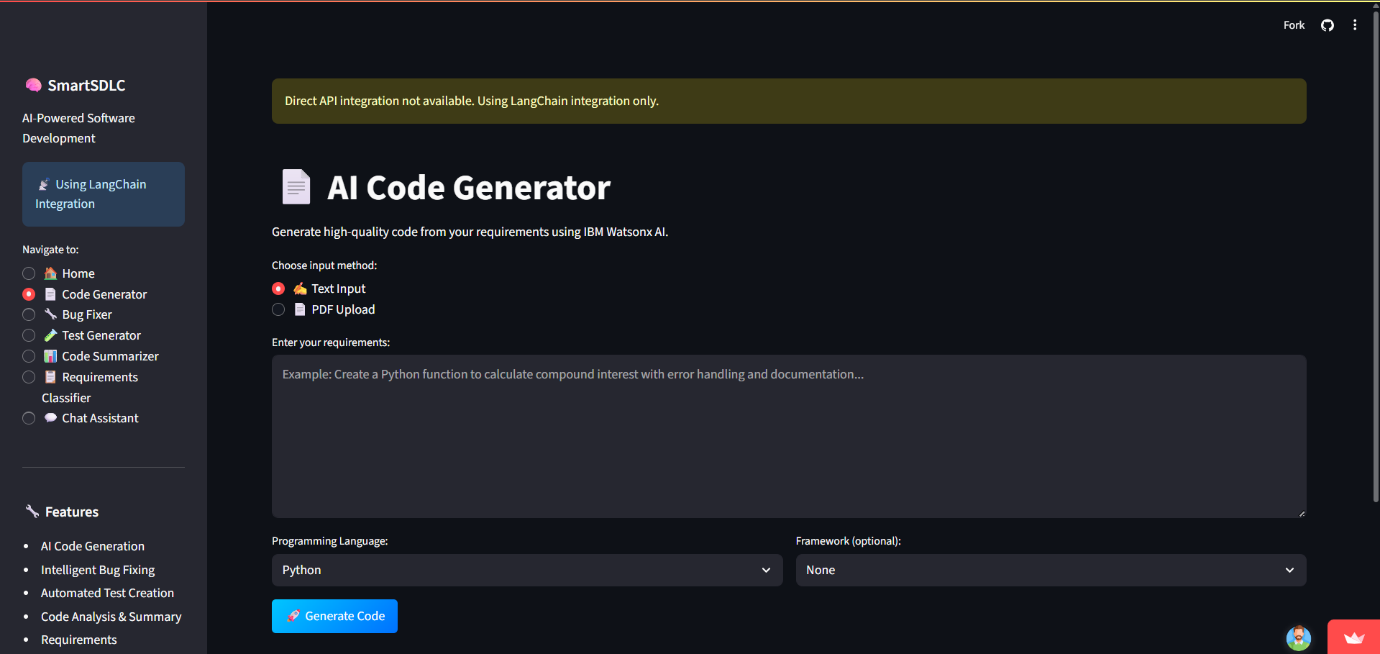
- Unit Testing: For prompt creation, output cleaning functions.

- Integration Testing: Streamlit frontend with IBM Watsonx API.

- Manual Testing: For all 6 features (input validation, output quality).

Tests are either in-code validation or handled by test cases generated using SmartSDLC's own test generator module

**9. Screenshots**



**10. Known Issues**

- No persistent user login system

- No database support (all session-based)

- No role-based access or advanced error handling

- IBM Watsonx API has rate limits depending on your cloud plan

**11. Future Enhancements**

- Add persistent database (MongoDB, PostgreSQL)

- Dockerize for CI/CD deployment

- Implement role-based login system

- Extend to support software architecture generation

- Add support for audio-based prompts or file-to-code generation

**12. Folder Structure**

SmartSDLC/

│

├── SMART\_SDLC.py → Main Streamlit app file

├── .env → Environment file for secrets

├── requirements.txt → Dependencies

├── /data → Optional sample input files

├── /venv → Python virtual environment

**13. Modules Breakdown**

Each module calls `ask\_watsonx(prompt)` to send instructions to the model.

- Requirement Classifier → PDF-to-user stories

- Code Generator → Prompt-to-code

- Bug Fixer → Debug raw code input

- Test Generator → Create unit test cases

- Summarizer → Explain what code does

- Chat Assistant → Open Q&A on SDLC topics

**14. Technology Stack**

Frontend: Streamlit

Backend: Python AI

Model: IBM Watsonx Granite 3.3 Instruct

PDF Reader: PyMuPDF (fitz)

Authentication: python-dotenv + .env

Deployment Target: IBM Cloud Foundry / Localhost

**15. Conclusion**

SmartSDLC successfully demonstrates how AI can accelerate software development by automating key stages like planning, coding, testing, and documentation. Future versions can extend its capabilities into DevOps, mobile responsiveness, and integration with GitHub workflows