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

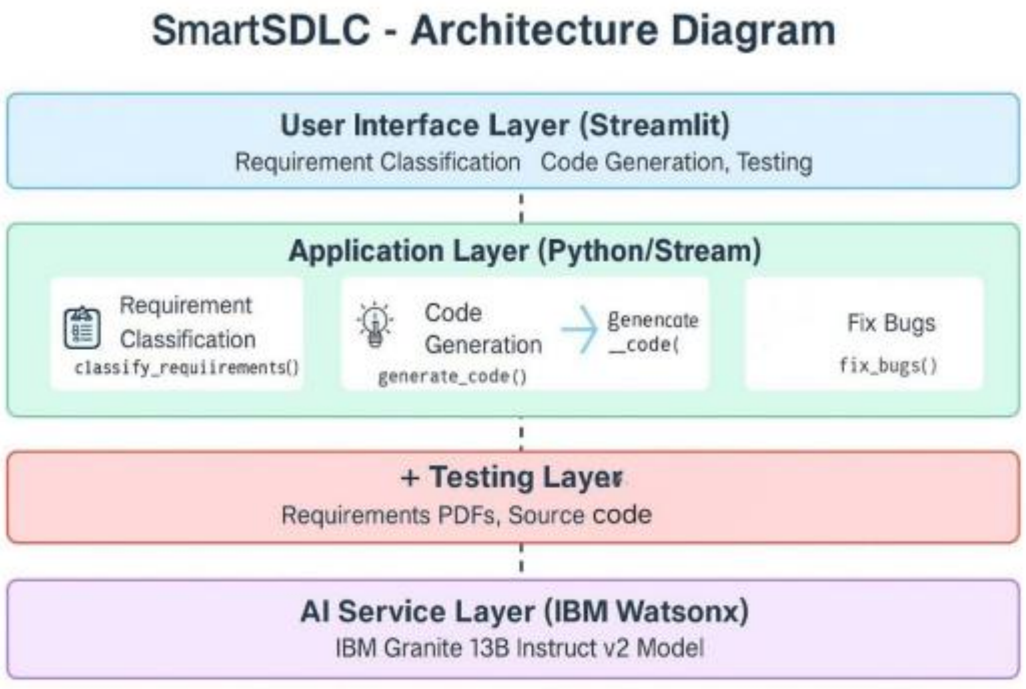
The purpose of SmartSDLC is to minimize human effort in software planning, development, and quality assurance by offering AI-driven support at every major SDLC stage.

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


SmartSDLC - Architecture Diagram
User Interface Layer (Streamlit)
Requirement Classification Code Generation, Testing
Application Layer (Python/Stream)
Requirement Classification
classify_requiirements()
Code Generation
generate_code()
genencate _code(
Fix Bugs
fix_bugs()
+ Testing Layer
Requirements PDFs, Source code
AI Service Layer (IBM Watsonx)
IBM Granite 13B Instruct v2 Model

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.

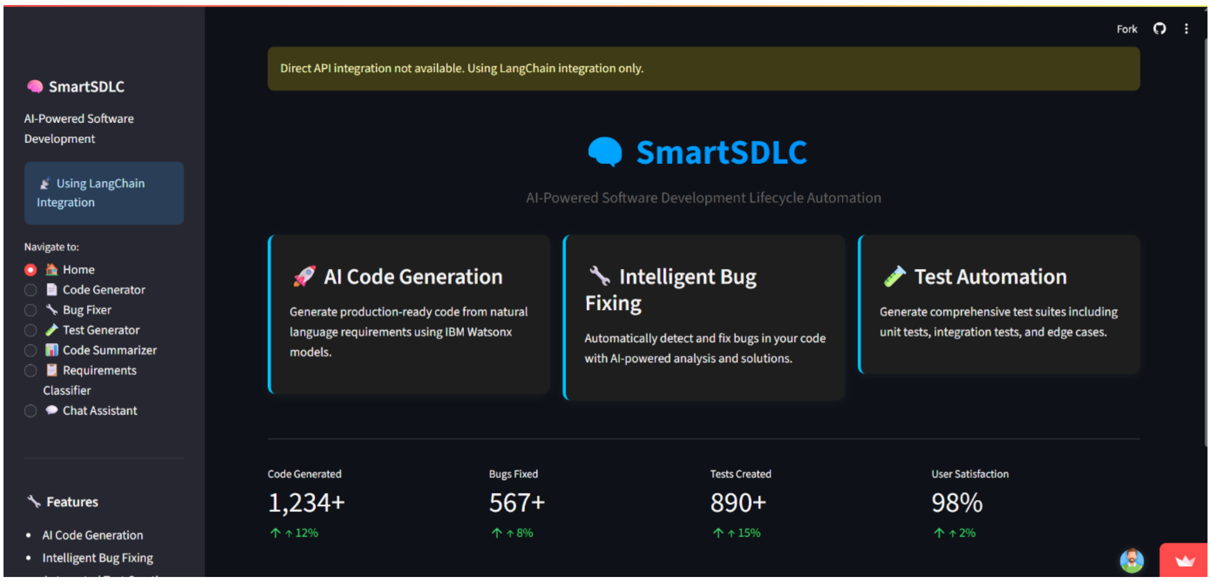
7. User Interface

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



Direct API integration not available. Using LangChain integration only.

SmartSDLC

AI-Powered Software Development Lifecycle Automation

AI Code Generation

Generate production-ready code from natural language requirements using IBM Watsonx models.

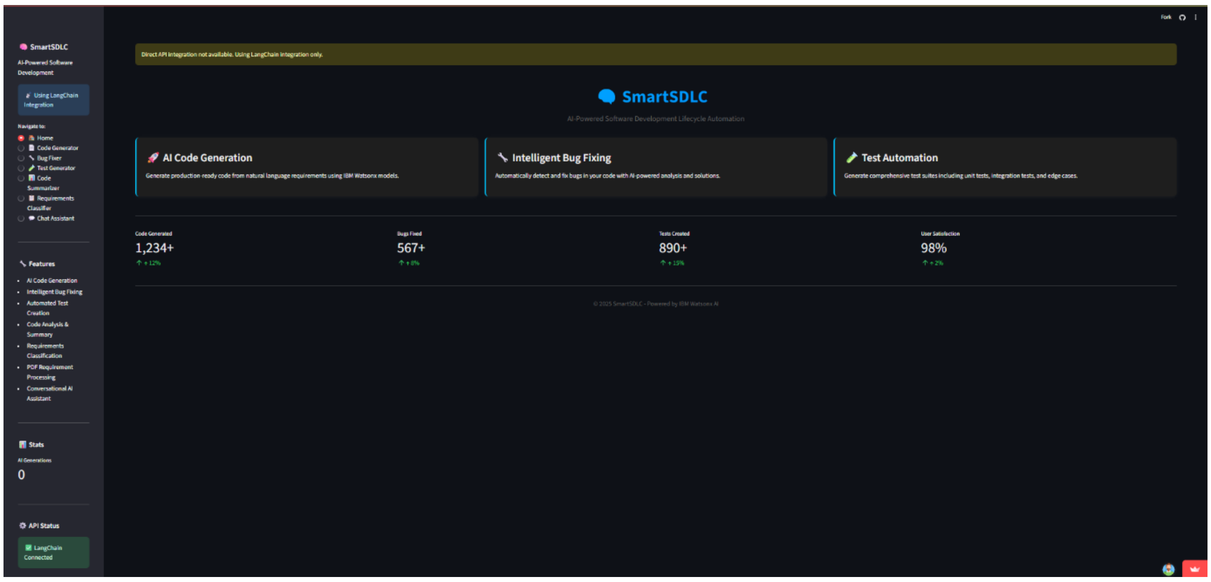
Intelligent Bug Fixing

Automatically detect and fix bugs in your code with AI-powered analysis and solutions.

Test Automation

Generate comprehensive test suites including unit tests, integration tests, and edge cases.

Code Generated	Bugs Fixed	Tests Created	User Satisfaction
1,234+	567+	890+	98%
↑ 12%	↑ 8%	↑ 15%	↑ 2%



Direct API integration not available. Using LangChain integration only.

SmartSDLC

AI-Powered Software Development Lifecycle Automation

AI Code Generation

Generate production-ready code from natural language requirements using IBM Watsonx models.

Intelligent Bug Fixing

Automatically detect and fix bugs in your code with AI-powered analysis and solutions.

Test Automation

Generate comprehensive test suites including unit tests, integration tests, and edge cases.

Code Generated	Bugs Fixed	Tests Created	User Satisfaction
1,234+	567+	890+	98%
↑ 12%	↑ 8%	↑ 15%	↑ 2%

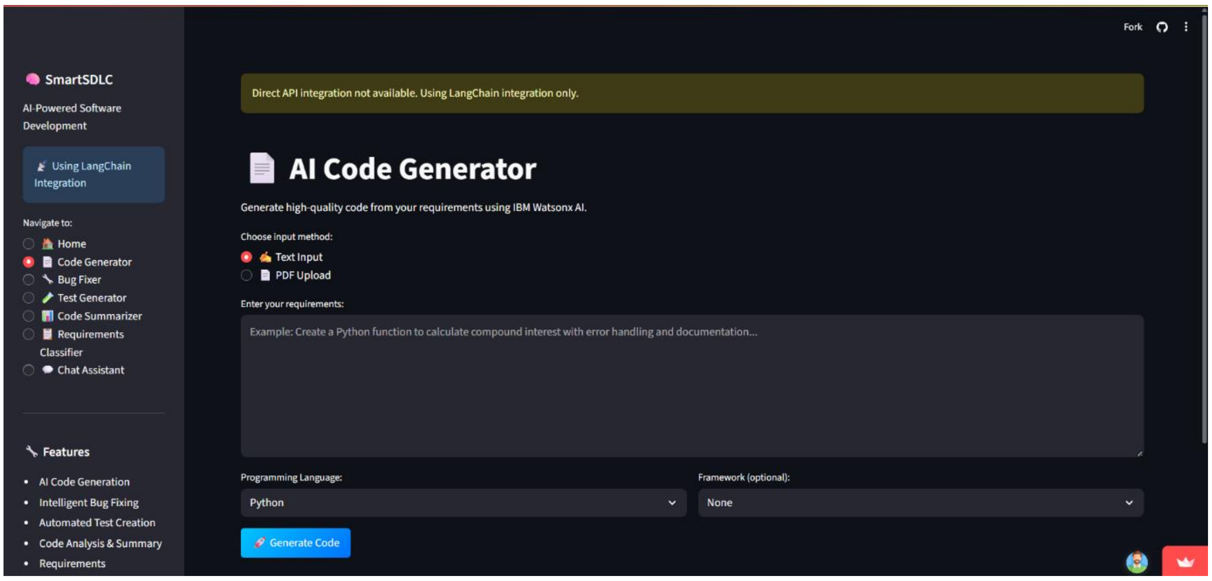
© 2025 SmartSDLC - Powered by IBM Watsonx AI

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



Direct API integration not available. Using LangChain integration only.

AI Code Generator

Generate high-quality code from your requirements using IBM Watsonx AI.

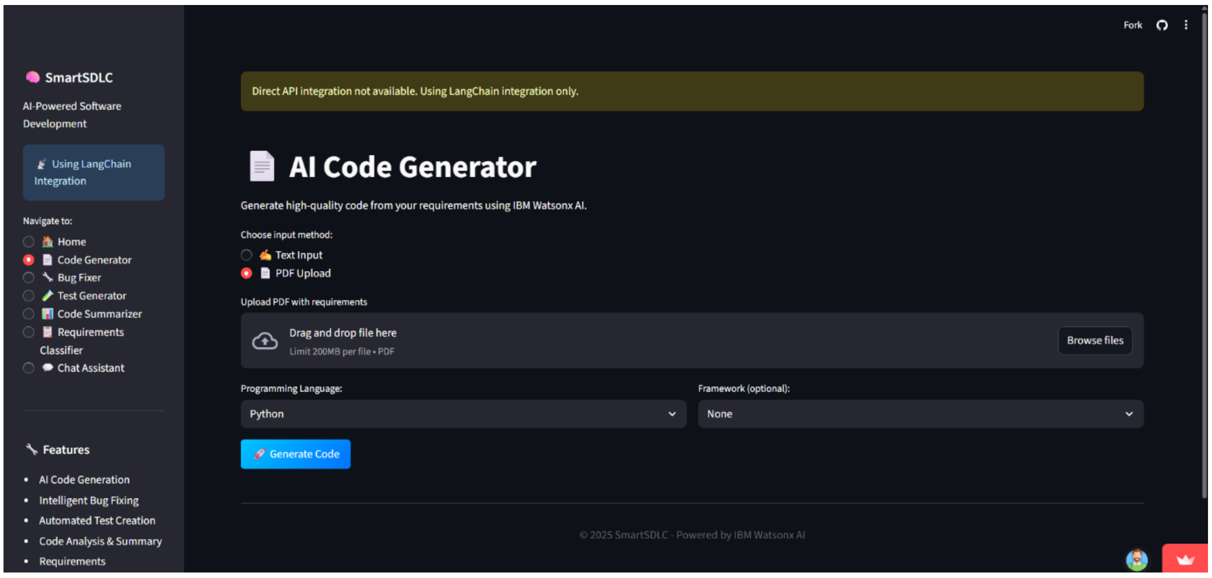
Choose input method: [✓ Text Input] PDF Upload

Enter your requirements:

Example: Create a Python function to calculate compound interest with error handling and documentation...

Programming Language: Python

Framework (optional): None



Direct API integration not available. Using LangChain integration only.

AI Code Generator

Generate high-quality code from your requirements using IBM Watsonx AI.

Choose input method: Text Input [✓ PDF Upload]

Upload PDF with requirements

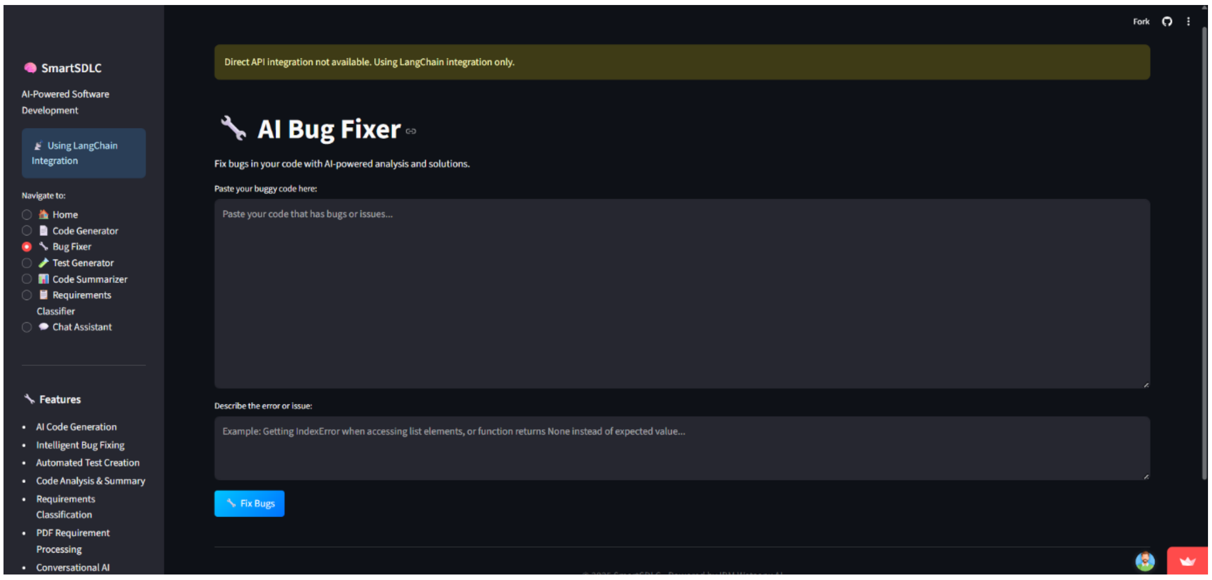
Drag and drop file here

Limit 200MB per file • PDF

Programming Language: Python

Framework (optional): None

© 2025 SmartSDLC - Powered by IBM Watsonx AI



Direct API integration not available. Using LangChain integration only.

AI Bug Fixer

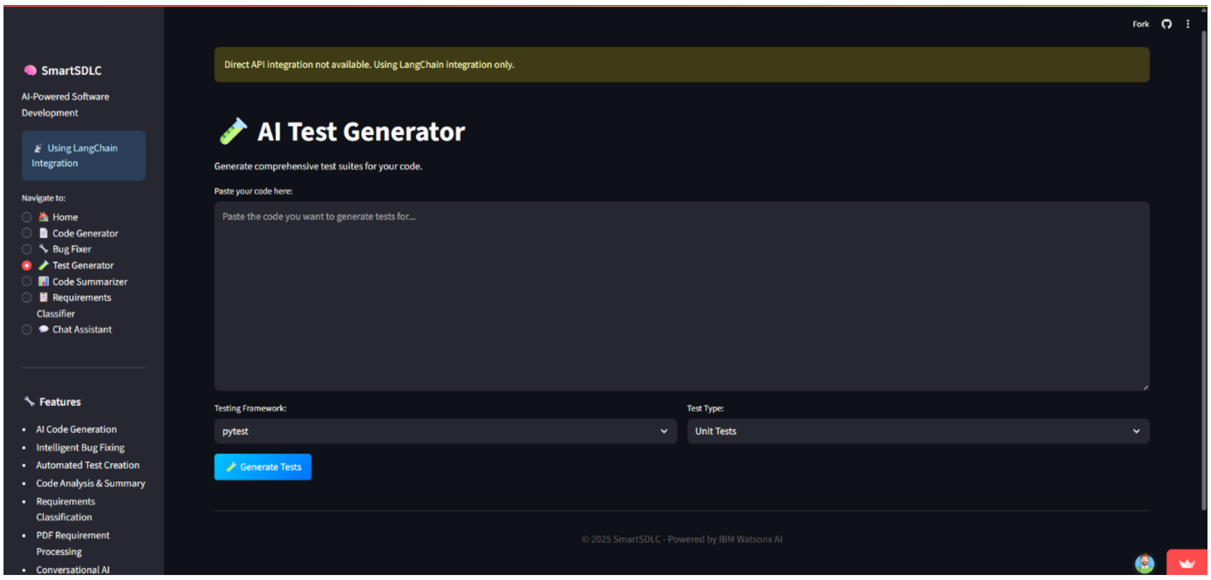
Fix bugs in your code with AI-powered analysis and solutions.

Paste your buggy code here:

Paste your code that has bugs or issues...

Describe the error or issue:

Example: Getting IndexError when accessing list elements, or function returns None instead of expected value...



Direct API integration not available. Using LangChain integration only.

AI Test Generator

Generate comprehensive test suites for your code.

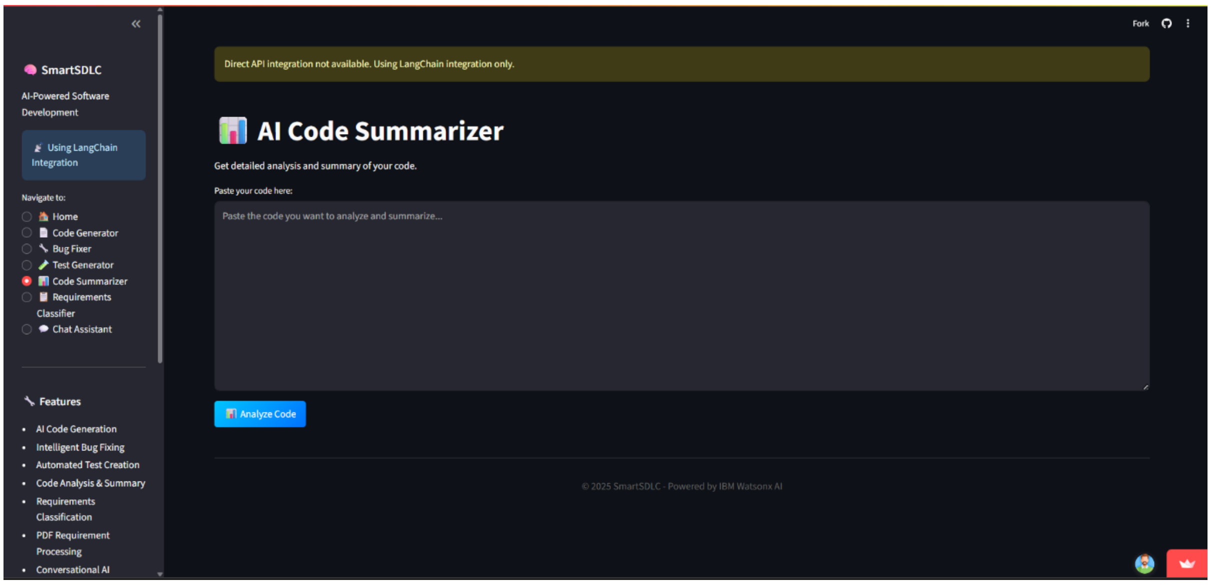
Paste your code here:

Paste the code you want to generate tests for...

Testing Framework: pytest

Test Type: Unit Tests

© 2025 SmartSDLC - Powered by IBM Watsonx AI



Direct API integration not available. Using LangChain integration only.

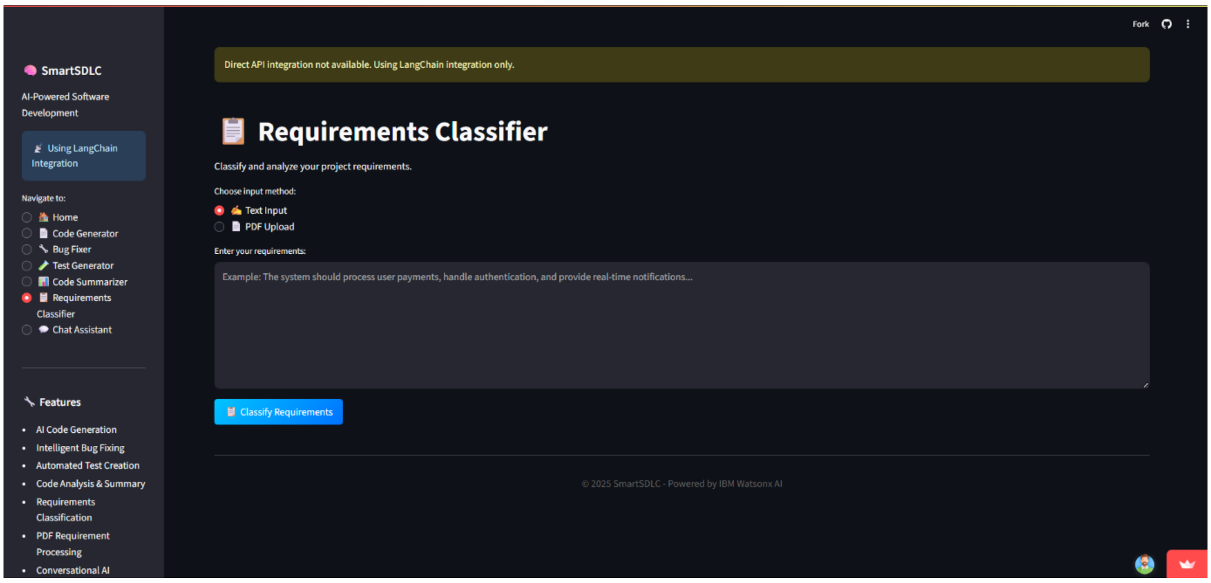
AI Code Summarizer

Get detailed analysis and summary of your code.

Paste your code here:

Paste the code you want to analyze and summarize...

© 2025 SmartSDLC - Powered by IBM Watsonx AI



Direct API integration not available. Using LangChain integration only.

Requirements Classifier

Classify and analyze your project requirements.

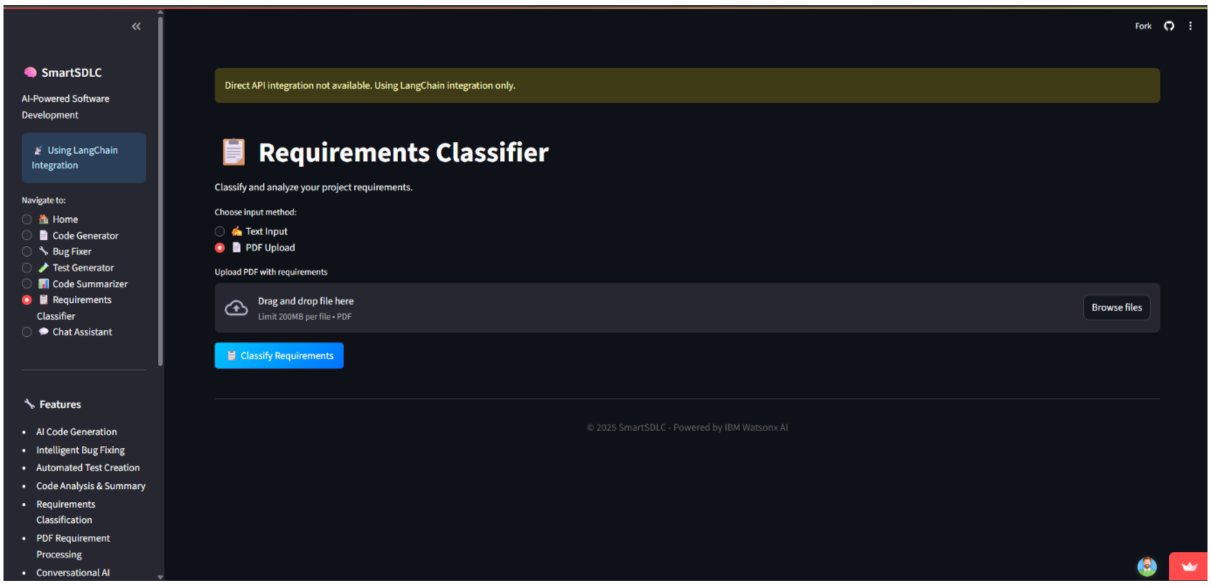
Choose input method:

[✓ Text Input] PDF Upload

Enter your requirements:

Example: The system should process user payments, handle authentication, and provide real-time notifications...

© 2025 SmartSDLC - Powered by IBM Watsonx AI



Direct API integration not available. Using LangChain integration only.

Requirements Classifier

Classify and analyze your project requirements.

Choose input method:

Text Input [✓ PDF Upload]

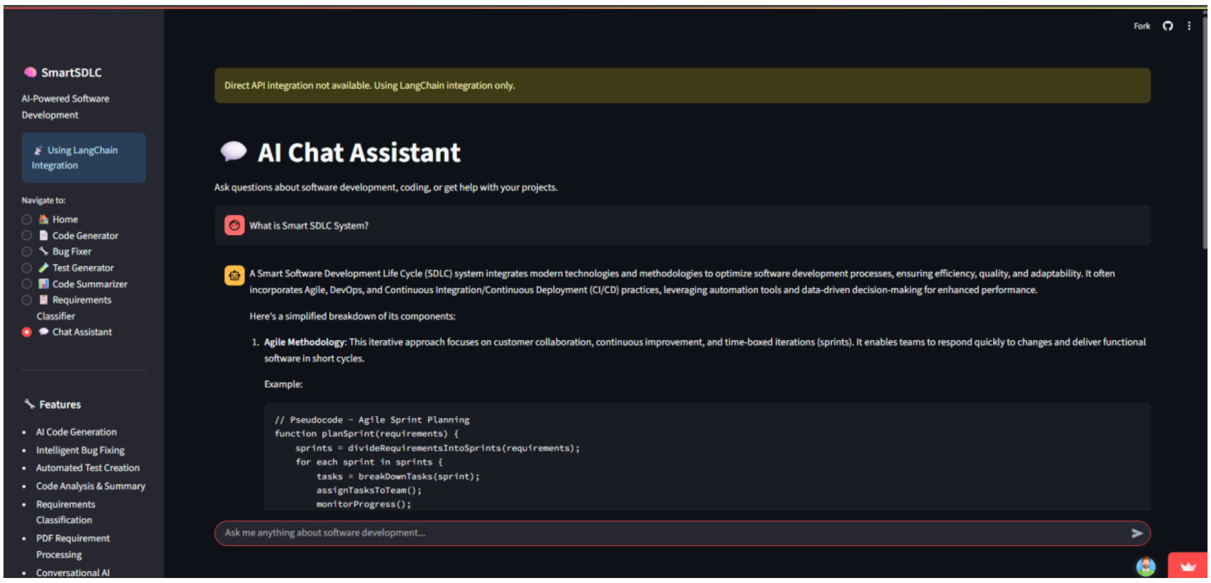
Upload PDF with requirements

Drag and drop file here
Limit 200MB per file • PDF

Browse files

Classify Requirements

© 2025 SmartSDLC · Powered by IBM Watsonx AI



Direct API integration not available. Using LangChain integration only.

AI Chat Assistant

Ask questions about software development, coding, or get help with your projects.

What is Smart SDLC System?

A Smart Software Development Life Cycle (SDLC) system integrates modern technologies and methodologies to optimize software development processes, ensuring efficiency, quality, and adaptability. It often incorporates Agile, DevOps, and Continuous Integration/Continuous Deployment (CI/CD) practices, leveraging automation tools and data-driven decision-making for enhanced performance.

Here's a simplified breakdown of its components:

1. **Agile Methodology:** This iterative approach focuses on customer collaboration, continuous improvement, and time-boxed iterations (sprints). It enables teams to respond quickly to changes and deliver functional software in short cycles.

   Example:

```
// Pseudocode - Agile Sprint Planning
function planSprint(requirements) {
    sprints = divideRequirementsIntoSprints(requirements);
    for each sprint in sprints {
        tasks = breakDownTasks(sprint);
        assignTasksToTeam();
        monitorProgress();
```

Ask me anything about software development...

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