PROJECT REPORT

TITLE: SMART SDLC – AI-POWERED SOFTWARE-DEVELOPMENT-LIFE-CYCLE ASSISTANT USING IBM GRANITE

Team ID : LTVIP2025TMID32004

Team Size: 4
Team Leader: Kuraganti Bharath
Team Member: Kasturi Sandeep
Team Member: Kantamaneni Prudhvi
Team Member: Kode Devi Sri Vallika

CONTENTS:

1. INTRODUCTION

1.1 PROJECT OVERVIEW

SMART SDLC is an intelligent assistant that automates and augments every major SDLC phase—requirements analysis, code generation, bug fixing, testcase creation, code summarization and conversational help—using IBM Watsonx Granite 13B and a Streamlit-based user interface.

1.2 PURPOSE

The system accelerates software delivery while improving code quality. It enables developers to prototype faster, eliminate repetitive tasks and gain instant insight into legacy code from a single dashboard.

2. IDEATION PHASE

2.1 PROBLEM STATEMENT

Developers spend significant time reading old code, writing boilerplate and tracking down bugs. Existing tools address only fragments of this workflow. A unified large-language-model copilot can remove those bottlenecks.

2.2 EMPATHY MAP CANVAS

SAYS "How do I fix this bug quickly?" "Can the AI write my unit tests?"

THINKS "Will the generated code be safe and maintainable?"

DOES Searches Stack Overflow, copies snippets, writes ad-hoc scripts

FEELS Stressed by deadlines, frustrated by repetitive tasks

PAINS Manual debugging, boilerplate coding, unclear legacy logic

GAINS Faster turnaround, fewer errors, clearer understanding

2.3 BRAINSTORMING

Standalone ideas such as bug fixer, test generator and story extractor were combined into one end-to-end assistant covering the complete SDLC.

3. REQUIREMENT ANALYSIS

3.1 CUSTOMER JOURNEY MAP

4. Start the Streamlit web application.

5. Select a module: Requirement, Code Generation, Bug Fix, Tests, Summary or Chat.

6. Provide input by uploading a PDF or code snippet.

7. The Granite model produces stories, code, fixes or answers.

8. Review the output, copy or download as needed.

9. Switch modules or end the session.

3.2 SESSION REQUIREMENTS

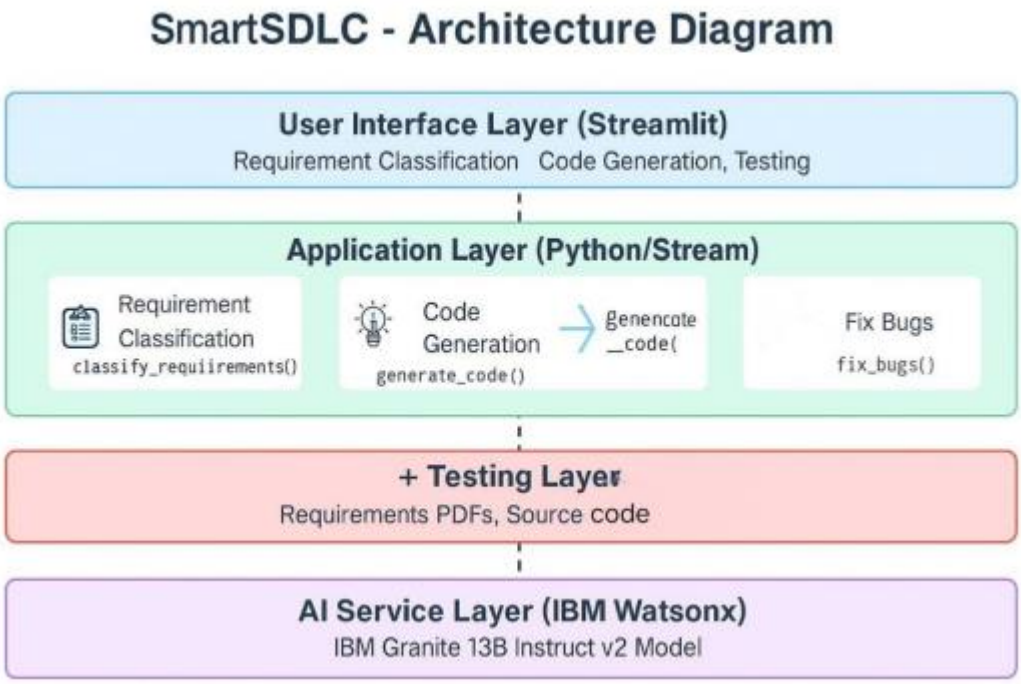
a) Upload PDF and code files.

b) Receive real-time AI responses.

c) Download generated assets.

d) Preserve chat history within the session.

3.3 DATA FLOW DIAGRAM



3.4 TECHNOLOGY STACK

Frontend	Streamlit
Backend	Python 3.11
AI Service	IBM Watsonx Granite 13B Instruct v2
PDF Parsing	PyMuPDF
Environment Management	virtualenv and .env secrets

4. PROJECT DESIGN

4.1 PROBLEM–SOLUTION FIT

Teams need faster, higher-quality delivery. Embedding Granite LLMs inside daily tools provides intelligent automation that meets this need.

4.2 PROPOSED SOLUTION

Layer 1 User interface: individual Streamlit pages per module

Layer 2 Core logic: Python helpers for PDF handling, code cleanup and API calls

Layer 3 AI layer: cached Granite model accessed with secure credentials

4.3 SOLUTION ARCHITECTURE

UI Layer Sidebar navigation, chat window, file widgets

Application Logic app.py and pages route requests

Helper Layer watson.py, pdf_utils.py, cleaning.py

AI Layer Granite service with retry and rate-limit control

5. PROJECT PLANNING AND SCHEDULING

Week 1 (12 Jun – 19 Jun) Idea finalisation, Streamlit skeleton, PDF ingestion

Week 2 (20 Jun – 26 Jun) Granite API integration, module logic, unit tests

Week 3 (27 Jun – 03 Jul) Bug-fix loop, UI polish, report creation

Week 4 (04 Jul – 10 Jul) Final demonstrations, documentation, deployment script

6. FUNCTIONAL AND PERFORMANCE TESTING

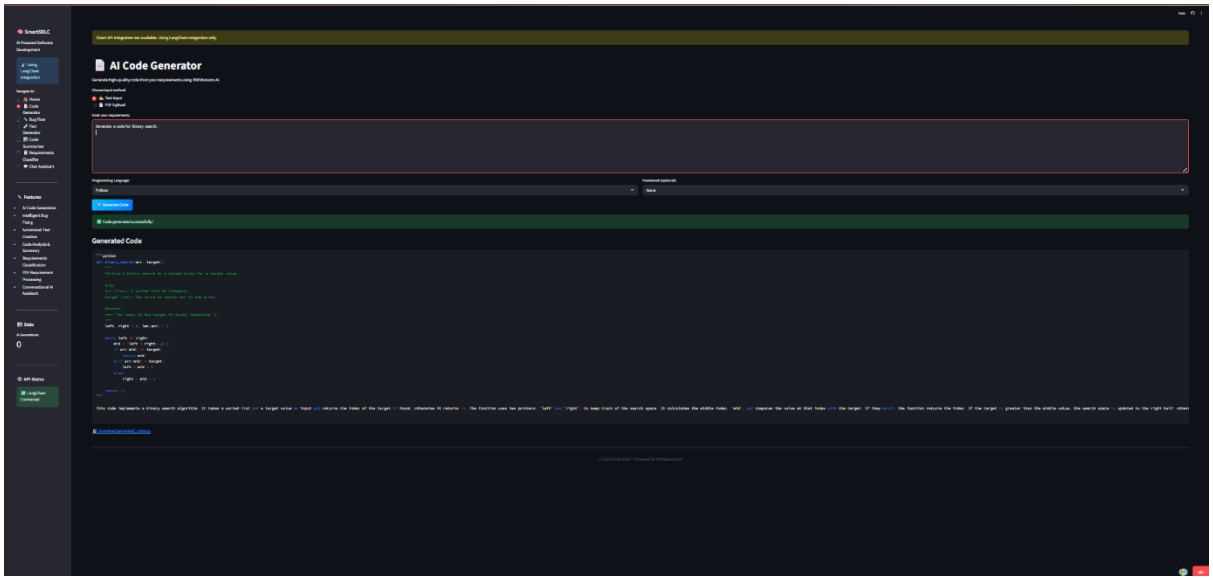
Unit testing PDF parser and code-cleanup utilities

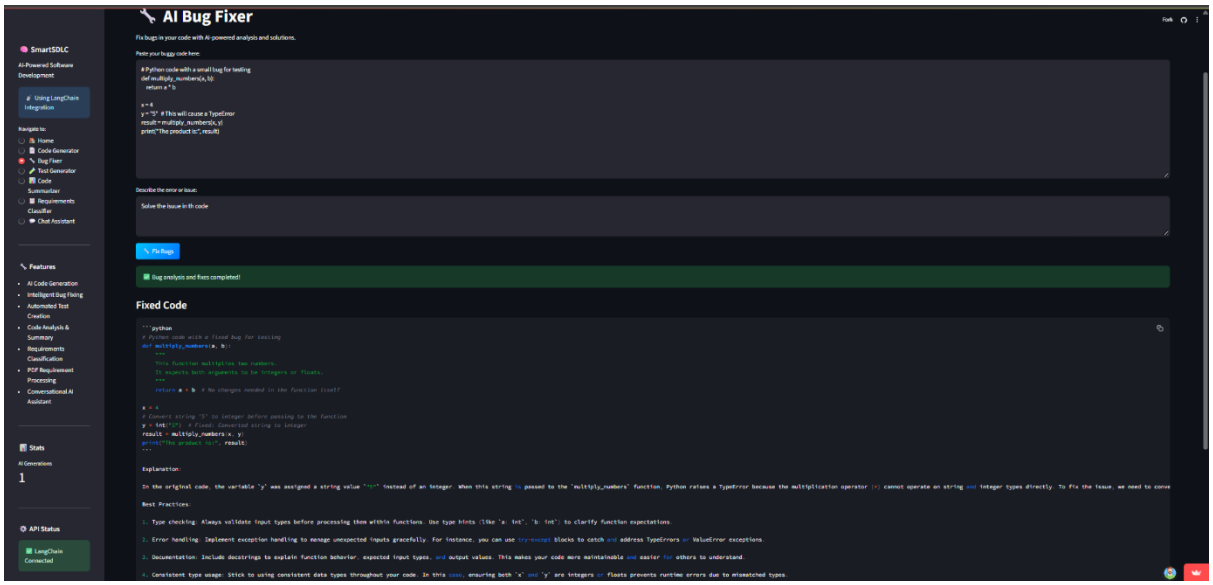
Integration testing End-to-end Streamlit to Granite response

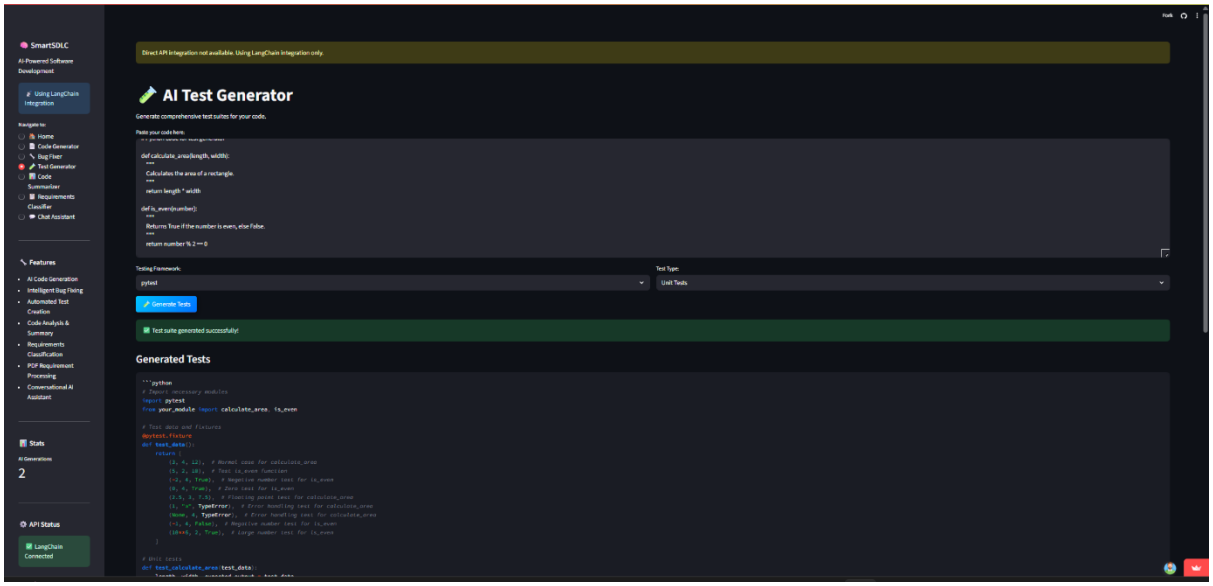
Manual testing Real project PDFs and GitHub codebases

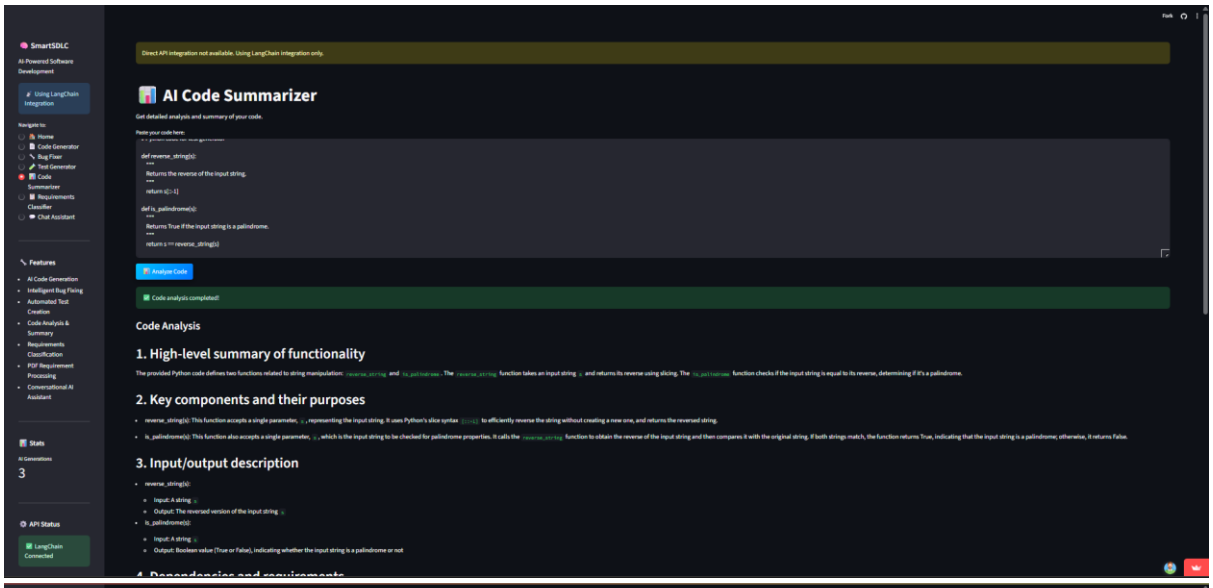
Error handling Network drops, oversized files, API-quota exhaustion

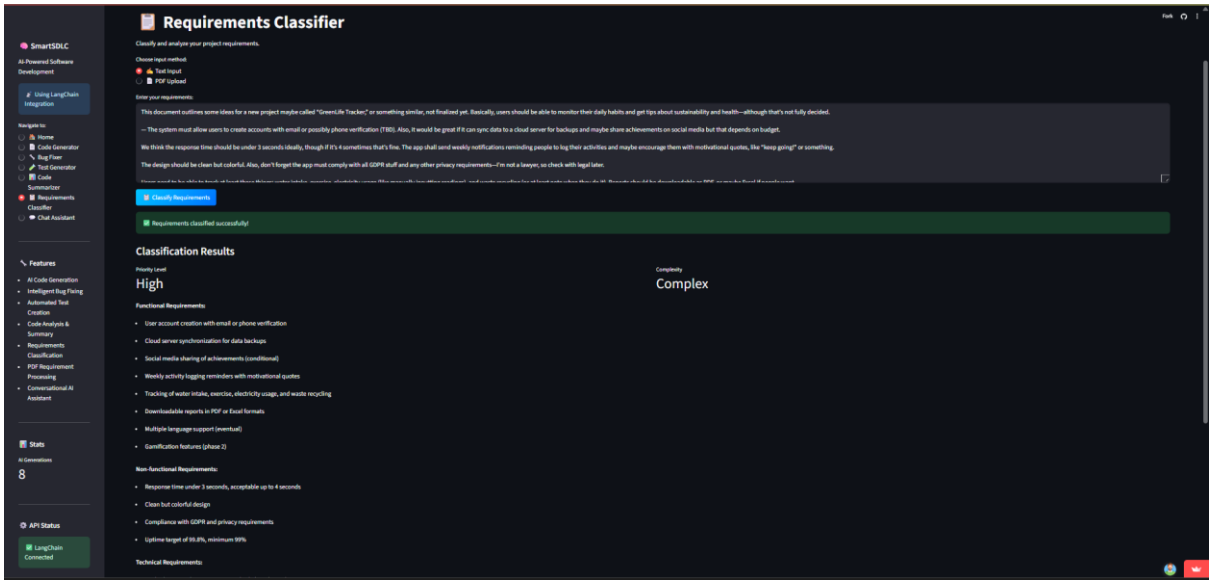
7. RESULTS

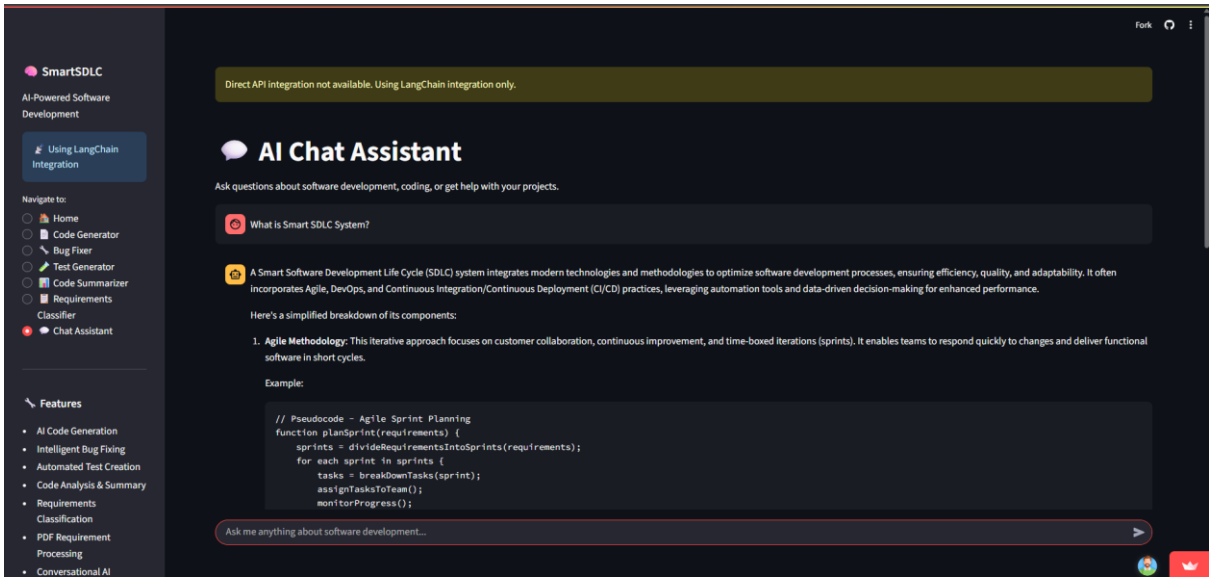


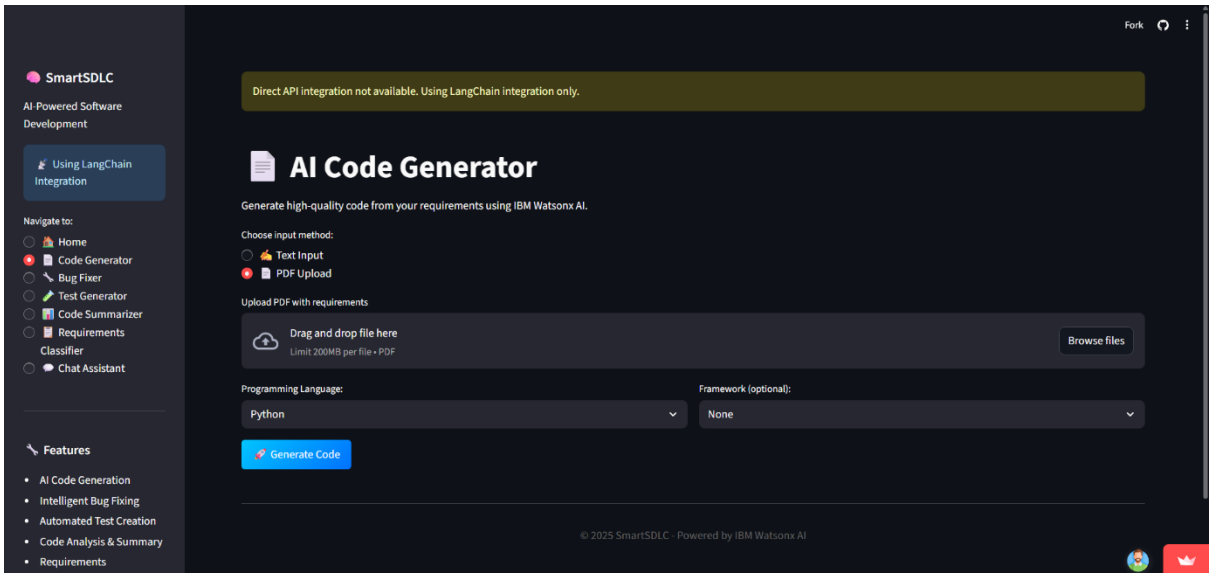


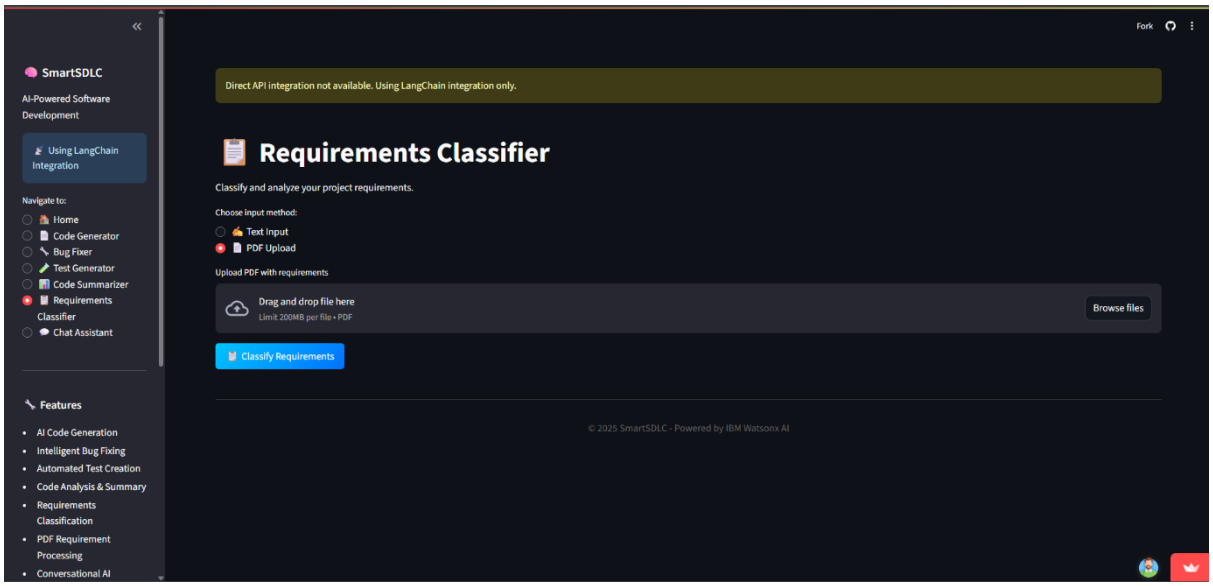

SmartSDLC
AI-Powered Software Development
Using LangChain Integration
Navigate to:
Home
Code Generator
Bug Fixer
Test Generator
Code Summarizer
Requirements Classifier
Chat Assistant
Features
AI Code Generation
Intelligent Bug Fixing
Automated Test Creation
Code Analysis & Summary
Requirements Classification
PDF Requirement Processing
Conversational AI Assistant
Stats
AI Generations
2
API Status
LangChain Connected
Direct API integration not available. Using LangChain integration only.
AI Test Generator
Generate comprehensive test suites for your code.
Paste your code here:
def calculate_area(length, width):
"""
Calculates the area of a rectangle.
"""
return length * width
def is_even(number):
"""
Returns True if the number is even, else False.
"""
return number % 2 == 0
Testing Framework:
pytest
Test Type:
Unit Tests
Generate Tests
Test suite generated successfully!
Generated Tests
```python
# Import necessary modules
import pytest
from your_module import calculate_area, is_even
# Test data and fixtures
@pytest.fixture
def test_data():
return [
(3, 4, 12), # Normal case for calculate_area
(5, 2, 10), # Test is_even function
(-3, 4, True), # Negative number test for is_even
(0, 4, True), # Zero test for is_even
(2.5, 2, 5.0), # Floating point test for calculate_area
(1, "a", TypeError), # Error handling test for calculate_area
(None, 4, TypeError), # Error handling test for calculate_area
(-1, 4, False), # Negative number test for is_even
(10**8, 2, True), # Large number test for is_even
]
# Unit tests
def test_calculate_area(test_data):


SmartSDLC
AI-Powered Software Development
Using LangChain Integration
Navigate to:
Home
Code Generator
Bug Fixer
Test Generator
Code Summarizer
Requirements Classifier
Chat Assistant
Features
AI Code Generation
Intelligent Bug Fixing
Automated Test Creation
Code Analysis & Summary
Requirements Classification
PDF Requirement Processing
Conversational AI Assistant
Stats
AI Generations
3
API Status
LangChain Connected
Direct API integration not available. Using LangChain integration only.
AI Code Summarizer
Get detailed analysis and summary of your code.
Paste your code here:
def reverse_string(s):
"""
Returns the reverse of the input string.
"""
return s[::-1]
def is_palindrome(s):
"""
Returns True if the input string is a palindrome.
"""
return s == reverse_string(s)
Analyze Code
Code analysis completed!
Code Analysis
1. High-level summary of functionality
The provided Python code defines two functions related to string manipulation: reverse_string and is_palindrome. The reverse_string function takes an input string s and returns its reverse using slicing. The is_palindrome function checks if the input string is equal to its reverse, determining if it's a palindrome.
2. Key components and their purposes
reverse_string(s): This function accepts a single parameter, s, representing the input string. It uses Python's slice syntax [::-1] to efficiently reverse the string without creating a new one, and returns the reversed string.
is_palindrome(s): This function also accepts a single parameter, s, which is the input string to be checked for palindrome properties. It calls the reverse_string function to obtain the reverse of the input string and then compares it with the original string. If both strings match, the function returns True, indicating that the input string is a palindrome; otherwise, it returns False.
3. Input/output description
reverse_string(s):
Input: A string s
Output: The reversed version of the input string s
is_palindrome(s):
Input: A string s
Output: Boolean value (True or False), indicating whether the input string is a palindrome or not


SmartSDLC
AI-Powered Software Development
Using LangChain Integration
Navigate to:
Home
Code Generator
Bug Fixer
Test Generator
Code Summarizer
Requirements Classifier
Chat Assistant
Features
AI Code Generation
Intelligent Bug Fixing
Automated Test Creation
Code Analysis & Summary
Requirements Classification
PDF Requirement Processing
Conversational AI Assistant
Stats
AI Generations
8
API Status
LangChain Connected
Requirements Classifier
Classify and analyze your project requirements.
Choose input method:
Text Input
PDF Upload
Enter your requirements:
This document outlines some ideas for a new project maybe called "GreenLife Tracker," or something similar, not finalized yet. Basically, users should be able to monitor their daily habits and get tips about sustainability and health—although that's not fully decided.
— The system must allow users to create accounts with email or possibly phone verification (TBD). Also, it would be great if it can sync data to a cloud server for backups and maybe share achievements on social media but that depends on budget.
We think the response time should be under 3 seconds ideally, though if it's 4 sometimes that's fine. The app shall send weekly notifications reminding people to log their activities and maybe encourage them with motivational quotes, like "keep going!" or something.
The design should be clean but colorful. Also, don't forget the app must comply with all GDPR stuff and any other privacy requirements—I'm not a lawyer, so check with legal later.
Classify Requirements
Requirements classified successfully!
Classification Results
Priority Level
High
Complexity
Complex
Functional Requirements:
User account creation with email or phone verification
Cloud server synchronization for data backups
Social media sharing of achievements (conditional)
Weekly activity logging reminders with motivational quotes
Tracking of water intake, exercise, electricity usage, and waste recycling
Downloadable reports in PDF or Excel formats
Multiple language support (eventual)
Gamification features (phase 2)
Non-functional Requirements:
Response time under 3 seconds, acceptable up to 4 seconds
Clean but colorful design
Compliance with GDPR and privacy requirements
Uptime target of 99.8%, minimum 99%
Technical Requirements:
```


Fork
SmartSDLC
AI-Powered Software Development
Using LangChain Integration
Navigate to:
Home
Code Generator
Bug Fixer
Test Generator
Code Summarizer
Requirements Classifier
Chat Assistant
Features
AI Code Generation
Intelligent Bug Fixing
Automated Test Creation
Code Analysis & Summary
Requirements Classification
PDF Requirement Processing
Conversational 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...


Fork
SmartSDLC
AI-Powered Software Development
Using LangChain Integration
Navigate to:
Home
Code Generator
Bug Fixer
Test Generator
Code Summarizer
Requirements Classifier
Chat Assistant
Features
AI Code Generation
Intelligent Bug Fixing
Automated Test Creation
Code Analysis & Summary
Requirements
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
Browse files
Programming Language:
Python
Framework (optional):
None
Generate Code
© 2025 SmartSDLC - Powered by IBM Watsonx AI


Fork
SmartSDLC
AI-Powered Software Development
Using LangChain Integration
Navigate to:
Home
Code Generator
Bug Fixer
Test Generator
Code Summarizer
Requirements Classifier
Chat Assistant
Features
AI Code Generation
Intelligent Bug Fixing
Automated Test Creation
Code Analysis & Summary
Requirements Classification
PDF Requirement Processing
Conversational 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

8. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

• Complete SDLC coverage in one tool

• Faster prototype-to-production pipeline

• High-quality code and summaries from Granite 13B

• Open and extensible foundation

DISADVANTAGES

• No user authentication yet

• Limited language support beyond Python (road-map)

• Internet connectivity required for AI service

9. CONCLUSION

SMART SDLC demonstrates that generative AI can streamline software engineering. By pairing Streamlit's simplicity with Watsonx power, it reduces development time and improves reliability, forming a basis for enterprise adoption.

10. APPENDIX

GitHub Repository	https://github.com/Sandeepkasturi/SmartSDLC.git
Key Files	SMART_SDLC.py, pages directory
Watsonx Model	ibm/granite-13b-instruct-v2
License	MIT