# 

# CHIDAMBARAM PILLAI COLLEGE FOR WOMEN

# SMARTSDLC-AI-ENHANCED SOFTWARE DEVELOPMENT LIFECYCLE

# 

# Team Leader : Sundharambal N

# Team member : Yogeshwari A

# INTERNAL EXTERNAL

# ABSTRACT

# 

The SmartSDLC project is an AI-enhanced platform designed to automate and streamline the Software Development Lifecycle (SDLC**)** using advanced technologies like IBM Watsonx, FastAPI, LangChain, and Streamlit. It integrates generative AI to handle key SDLC phases, including requirement analysis, code generation, test case creation, bug fixing, and documentation. The platform features a user-friendlyinterface that allows users to upload PDFs, generate structured requirements, and transform natural language prompts into functional code. It also includes an AI-powered chatbot for real-time assistance and support. The backend, built with FastAPI, efficiently processes API requests, while the frontend, developed using Streamlit, offers a visually appealing and interactive dashboard. The system is modular, scalable, and secure, featuring a robust authentication mechanism and seamless integration between AI models and user inputs. Deployment is streamlined with local hosting via Uvicorn and Streamlit, with comprehensive API documentation available through Swagger UI. By leveraging AI in a practical and efficient manner, SmartSDLC significantly reduces manual workload, accelerates development processes, and enhances software quality, making it an invaluable tool for developers and project teams.

# 

# 

# PROJECT DESCRIPTION



## 

## INTRODUCTION TO PYTHON

Python is a high-level, interpreted, and general-purpose programming language created by Guido van Rossum and first released in 1991. It is known for its simple syntax that closely resembles natural language, making it easy to learn and use especially for beginners.

### INTRODUCTION TO AI

Artificial Intelligence (AI) is a branch of computer science that focuses on creating machines or systems capable of performing tasks that normally require human intelligence. These tasks include problem-solving, learning, decision-making, understanding language, recognizing patterns, and adapting to new situations.AI works by using algorithms and large amounts of data to simulate human-like thinking. It enables machines to analyze information, draw conclusions, and improve their performance over time without being explicitly programmed for every task.

**IBM GRANITE**

IBM Granite is a family of **open-source, enterprise-ready AI models** designed for businesses. They include large language models (LLMs), safety-focused models (Guardian), vision-language models, speech models, and embedding.

**FRONTEND**

**Streamlit**

Streamlit is an open-source Python framework that makes it easy to create interactive web applications for data science, machine learning, and analytics — without needing HTML, CSS, or JavaScript.With just Python code, you can quickly build dashboards, visualize data, and deploy apps.

**BACKEND**

**FastAPI**

FastAPI is a modern, high-performance web framework for building APIs with Python.It is built on top of Starlette (for web handling) and Pydantic (for data validation).FastAPI is widely used for RESTful APIs, backend services, and ML model deployment because it’s fast, easy to use, and developer-friendly.

**IBM Watsonx**

IBM Watsonx is an enterprise AI and data platform that includes Watsonx.ai for building AI models, Watsonx.data for managing data, and Watsonx.governance for ensuring responsible AI, helping businesses train, deploy, and scale trustworthy generative AI and machine learning across hybrid cloud environments**.**

**MODULES:**

**Requirement Upload and Classification**

The platform simplifies the complex task of requirement gathering by allowing users to upload PDF documents containing raw, unstructured text. The backend extracts content using PyMuPDF and leverages IBM Watsonx’s Granite-20B AI model to classify each sentence into specific SDLC phases such as Requirements, Design, Development, Testing, or Deployment. These classified inputs are then transformed into structured user stories, enabling clear planning and traceability. The frontend displays this output in an organized, readable format grouped by phase, significantly improving clarity and saving manual effort.

**AI Code Generator**

AI Code Generator, addresses the development phase, where developers can input natural language prompts or structured user stories. These prompts are sent to the Watsonx model, which generates contextually relevant, production-ready code. This reduces the time needed for boilerplate or prototype creation and enhances coding efficiency. The code is presented in a clean, syntax-highlighted format on the frontend, ready for use or further enhancement.

**Bug Fixer**

Bug Fixer, the platform supports debugging by accepting code snippets in languages such as Python or JavaScript. Upon receiving the buggy code, the Watsonx AI analyzes it for both syntactical and logical errors and returns an optimized version. This not only assists developers in identifying mistakes without extensive manual reviews but also provides immediate, corrected code directly in the frontend for comparison.

Coding

import random

class SmartSDLC\_AI:

def \_\_init\_\_(self, project\_name, requirements):

self.project\_name = project\_name

self.requirements = requirements

def requirement\_analysis(self):

return f"Requirements for {self.project\_name}: {self.requirements}"

def design\_phase(self):

designs = ["Use MVC architecture",

"Adopt Microservices design",

"Apply Layered Architecture",

"Use Client-Server model" ]

return f"Suggested Design: {random.choice(designs)}"

def development\_phase(self):

return f"Suggested Tech Stack: {random.choice(tech\_stacks)}"

def testing\_phase(self):

tests = ["Unit Testing, Integration Testing, System Testing",

"Automated Testing using Selenium or PyTest",

"Performance & Security Testing"]

return f"Testing Plan: {random.choice(tests)}"

def deployment\_phase(self):

return f"Deployment Plan: {random.choice(deployments)}"

def maintenance\_phase(self):

maintenance = [

"Regular Bug Fixes & Updates",

"System Monitoring with Prometheus",

"User Feedback Collection",

"Security Patching" ]

return f"Maintenance Plan: {random.choice(maintenance)}"

def run\_sdlc(self):

print("------ Smart SDLC AI Assistant ------")

print(self.requirement\_analysis())

print(self.design\_phase())

print(self.development\_phase())

print(self.testing\_phase())

print(self.deployment\_phase())

print(self.maintenance\_phase())

if \_\_name\_\_ == "\_\_main\_\_":

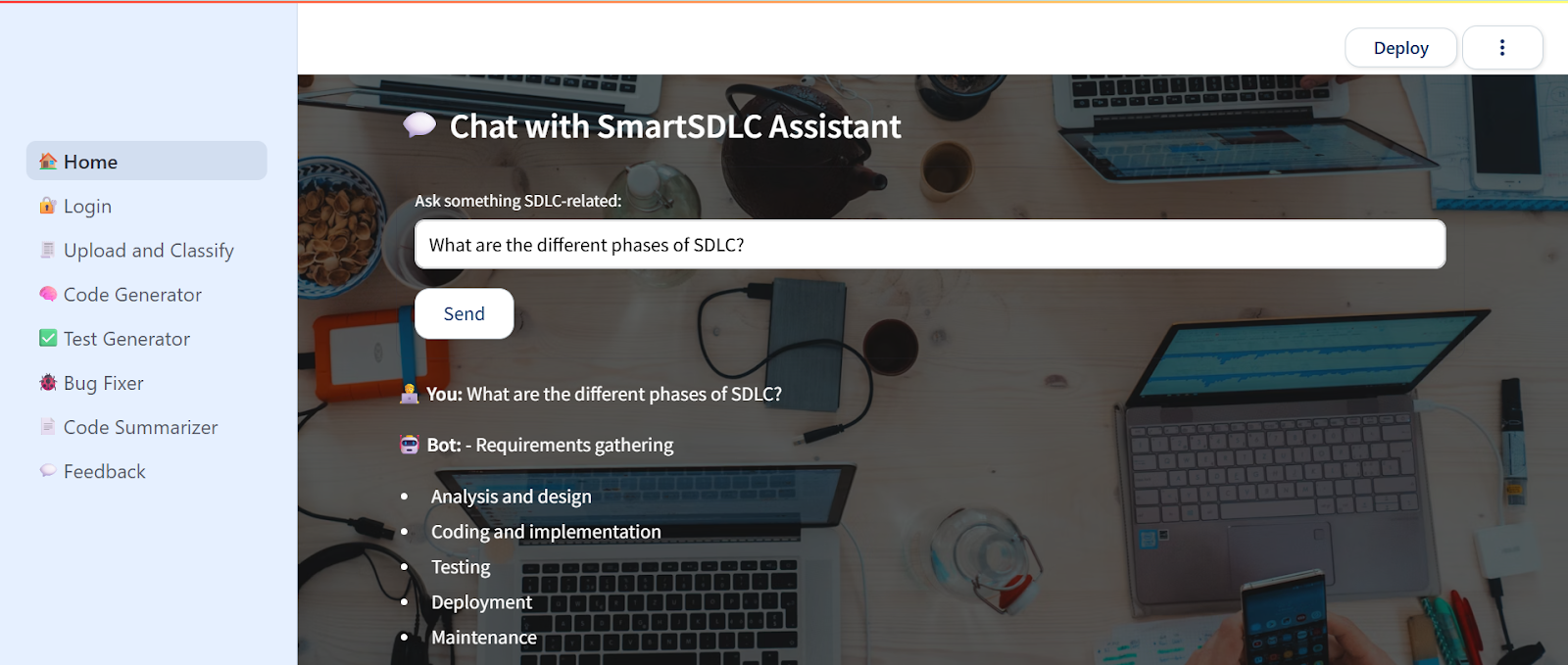
project = SmartSDLC\_AI(

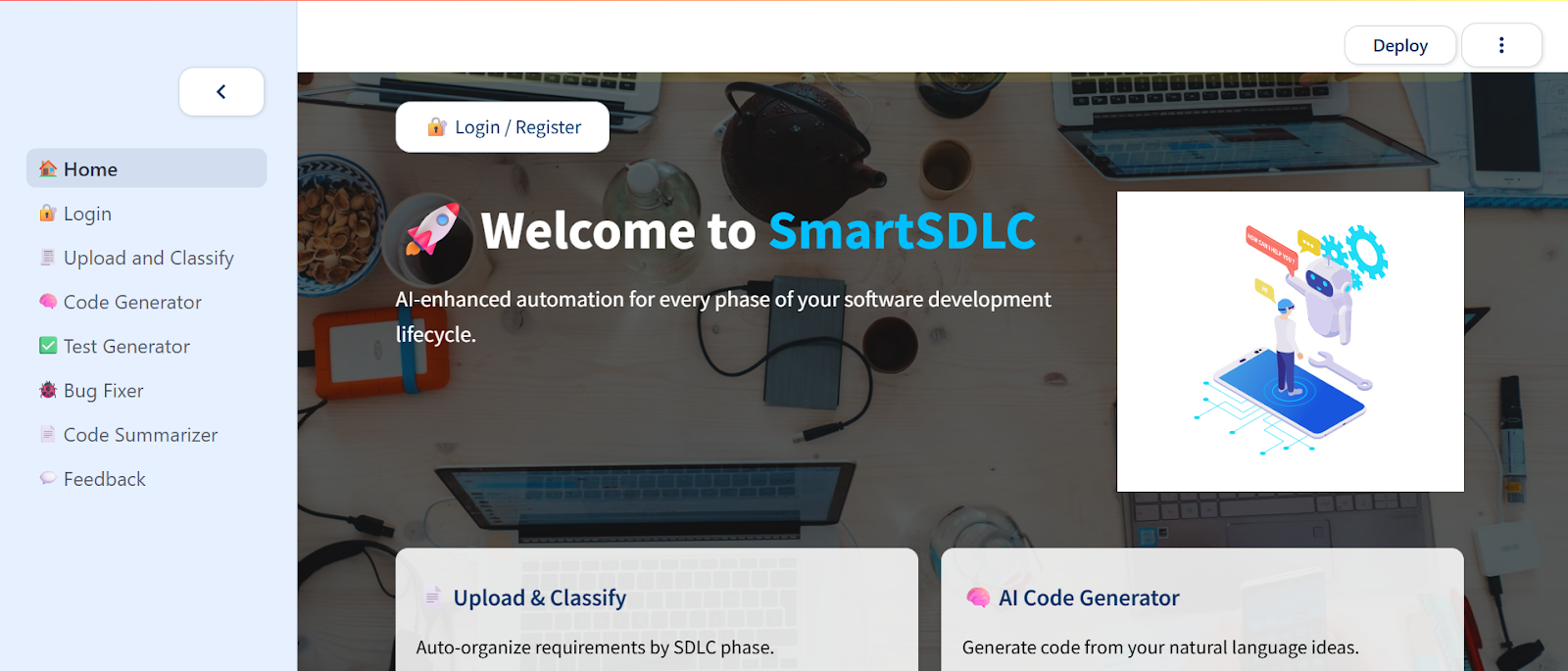
project\_name="EduTutor AI",

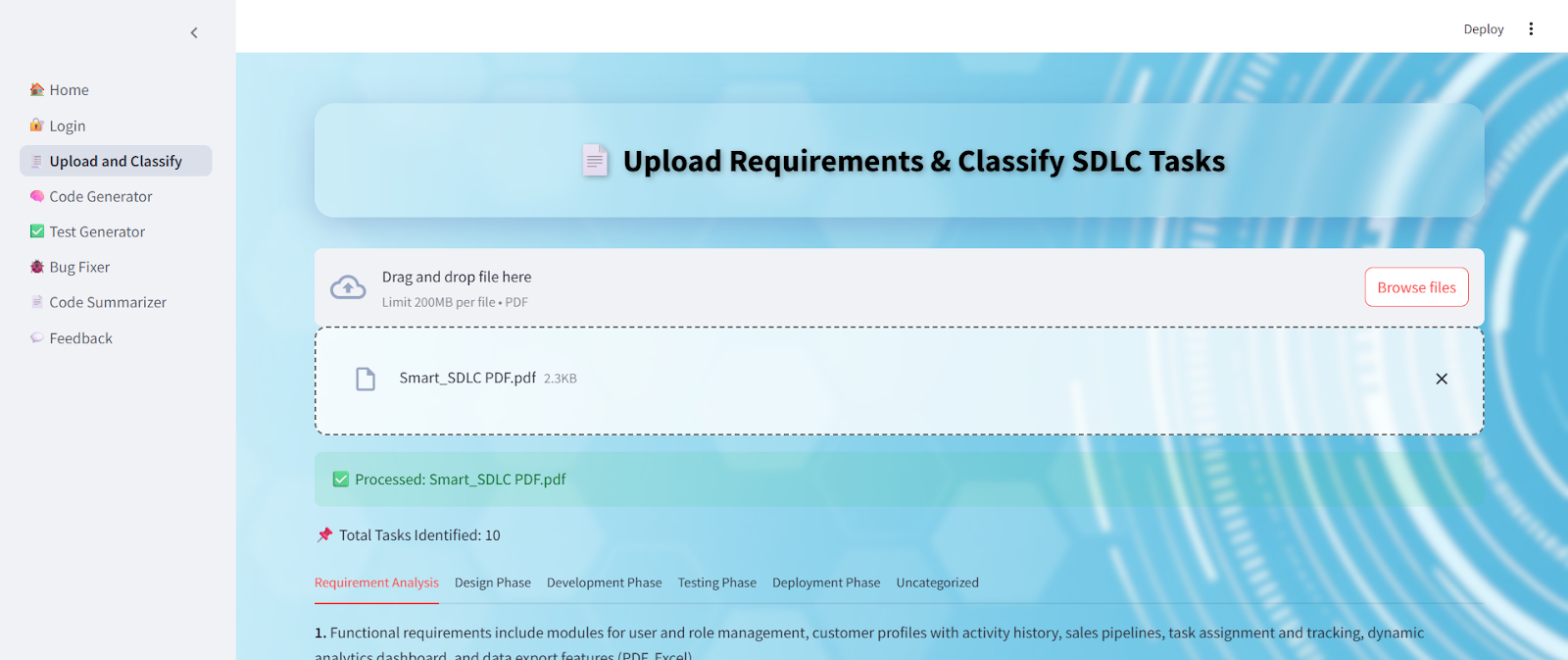
requirements="An AI-based e-learning tutor that answers student questions and provides study materials." )

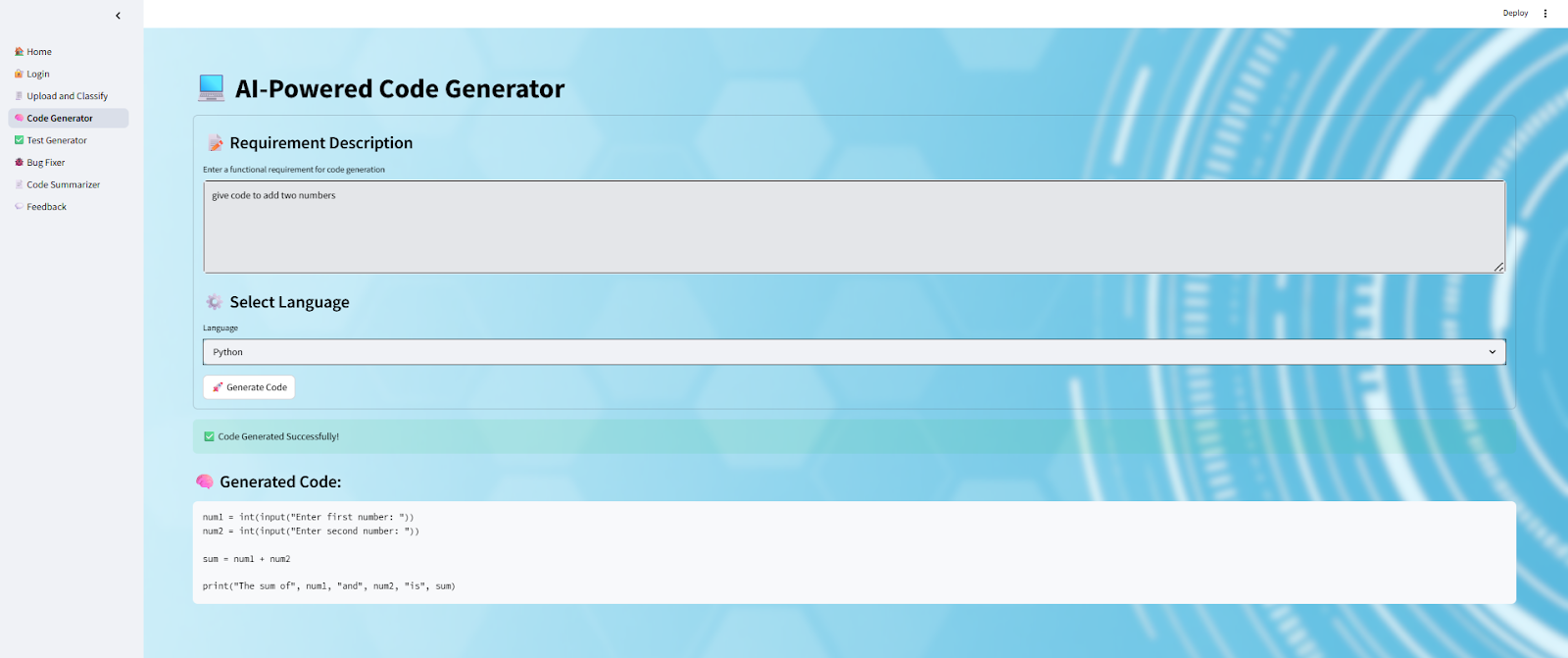
project.run\_sdlc()

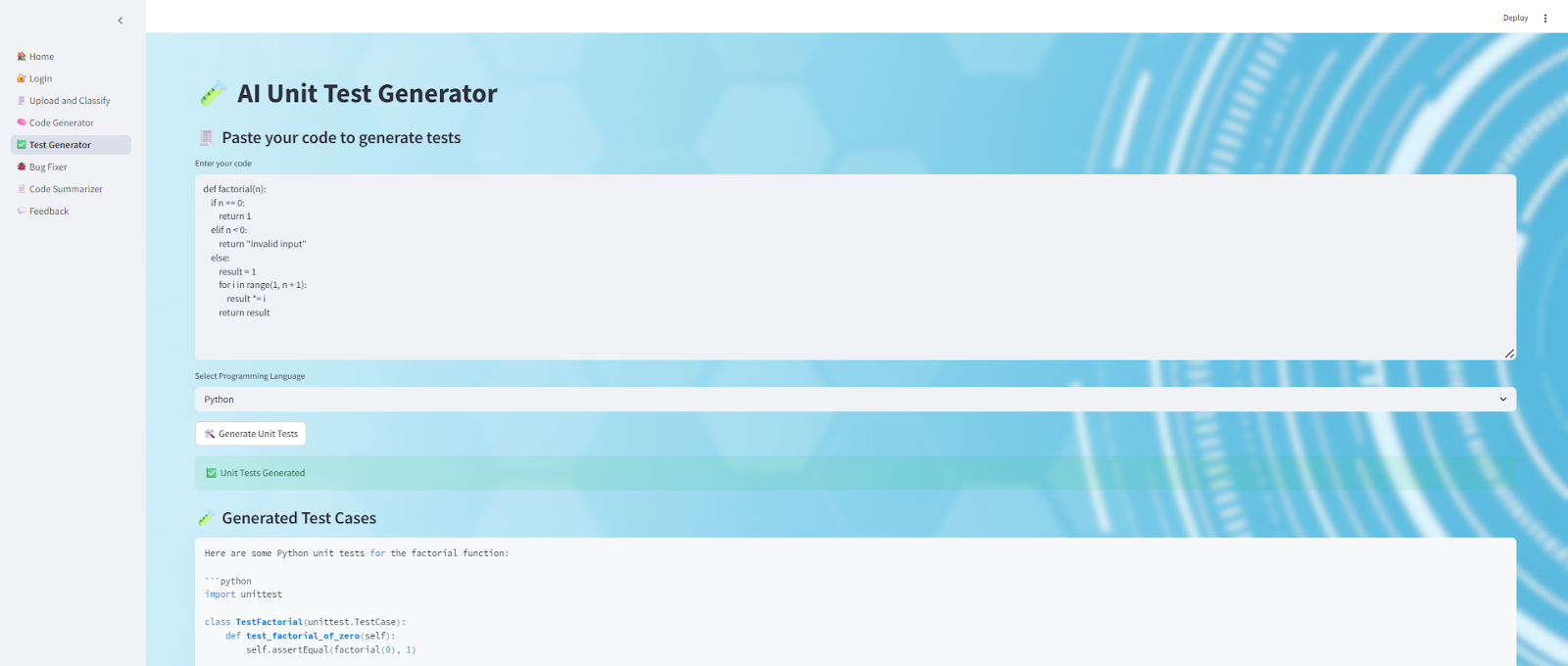
SCREENSHOT:

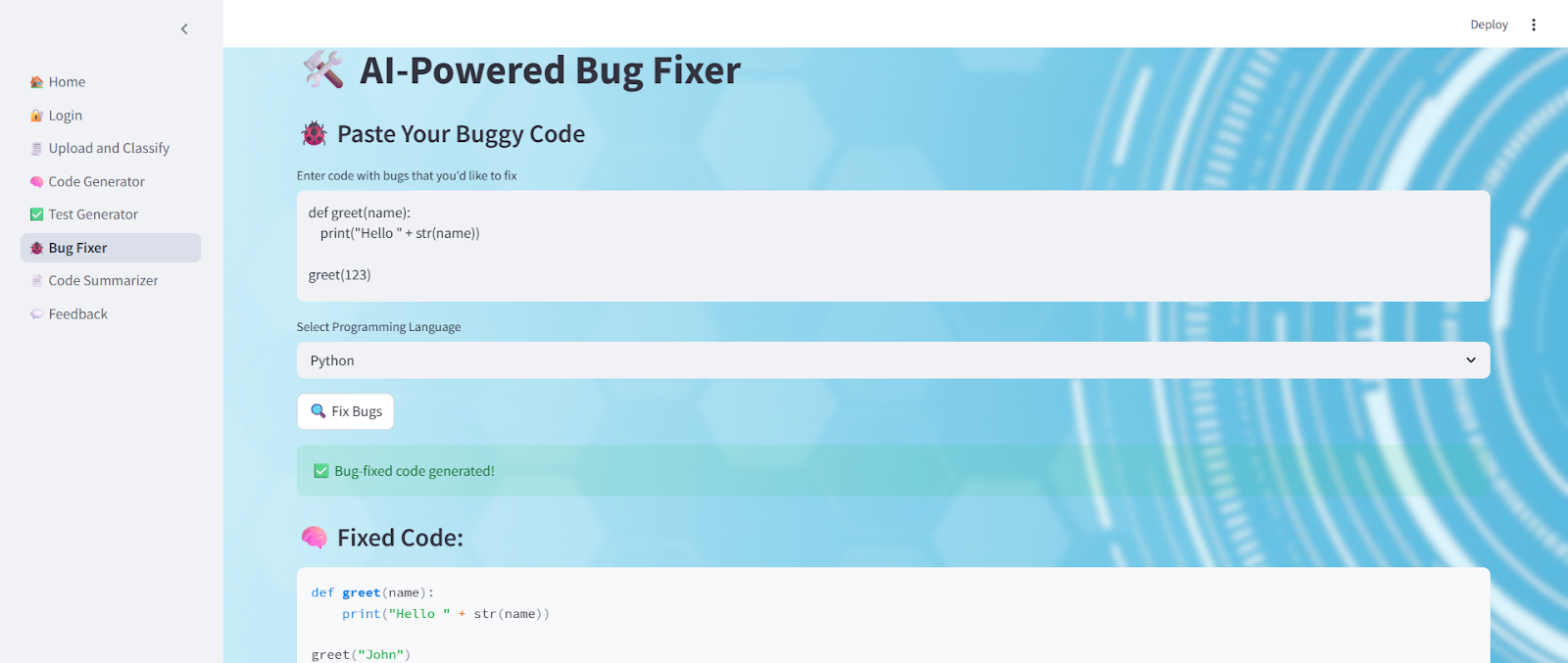


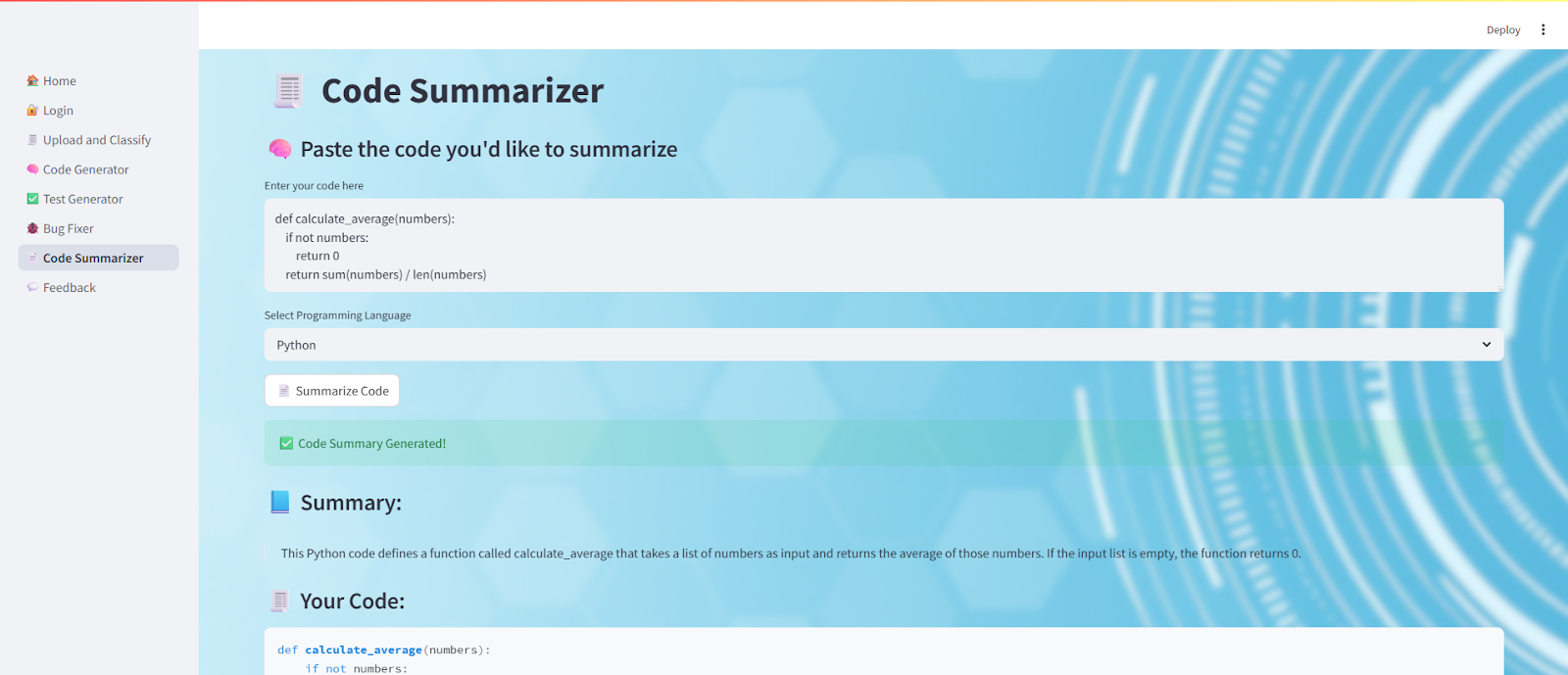


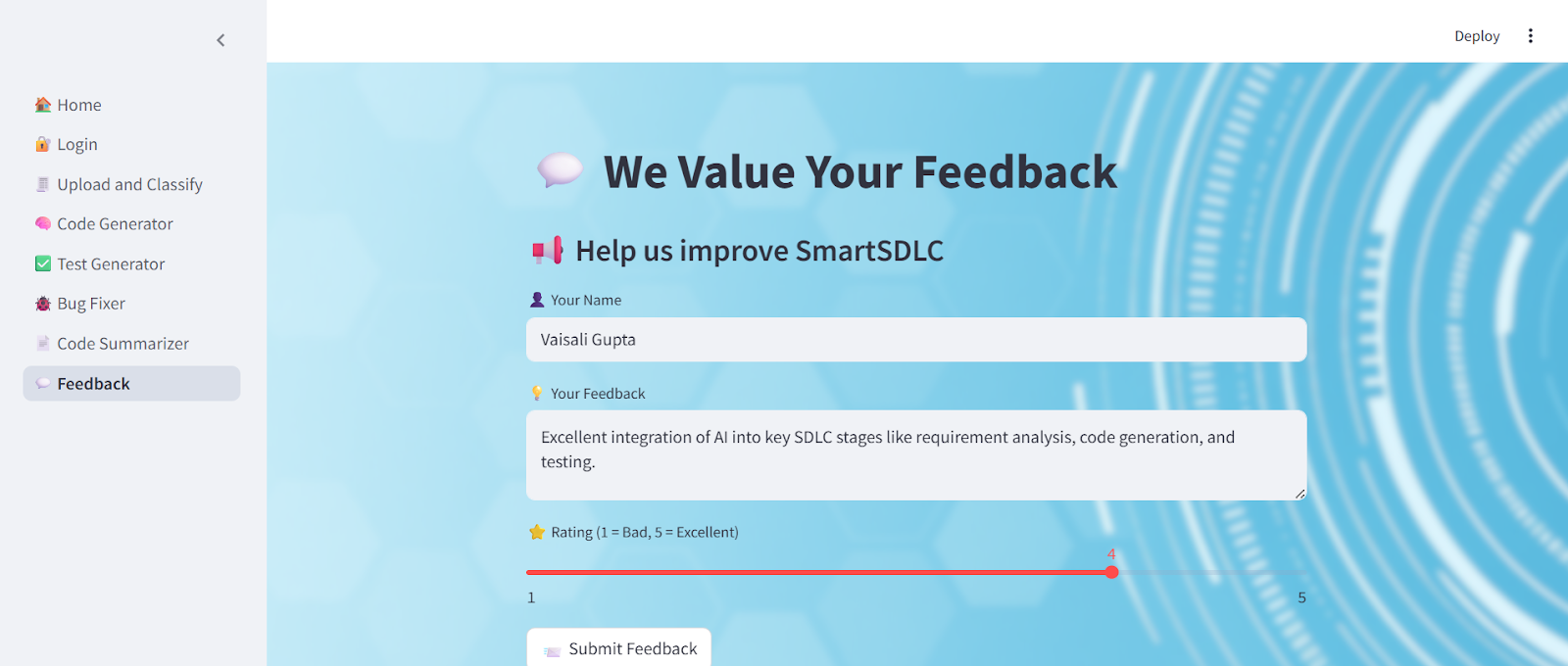












**CONCLUSION:**

SmartSDLC integrates artificial intelligence into every stage of the software development lifecycle, enabling smarter planning, automated code generation, intelligent testing, proactive risk management, and continuous optimization. By combining traditional SDLC practices with AI-driven insights and automation, SmartSDLC reduces development time, improves software quality, enhances collaboration, and ensures adaptability to evolving business needs. Ultimately, it represents a future-ready approach that empowers organizations to build reliable, efficient, and innovative software solutions at scale.