FSD DOCUMENTATION

**1. Introduction**

Team ID: LTVIP2025TMID32164

Project Title: Smart SDLC – AI-powered Software Development Lifecycle Assistant

Team Leader: Chittareddy Praveen Kumar

Team Members: Bonthu Sai Chandrika

Bollina Lavanya

C Anvesh Reddy

**2. Project Overview**

The aim of Smart SDLC is to help students, developers, and engineers rapidly prototype, understand, and debug software systems using AI. It is built with FastAPI, Python, Gradio/HTML, and the IBM Granite model from Hugging Face.

**3. Architecture**

The project follows a modular backend using FastAPI and Hugging Face models. The frontend can be implemented using either Gradio or a custom HTML/CSS interface, with a chatbot embedded. The model processes natural language prompts related to coding tasks and responds in real time.

**4. Setup Instructions (Colab)**

- Clone the GitHub repo  
- Add Hugging Face API key to .env  
- Run FastAPI server using Uvicorn  
- Open Gradio/HTML UI and start testing features

**5. Notebook Code Structure**

- main.py (FastAPI entry)  
- models/ (ML model integrations)  
- templates/ (HTML files)  
- static/ (CSS, JS)  
- chatbot/ (AI response logic)  
- routes/ (feature APIs)

**6. Running the App**

Use 'uvicorn main:app --reload' to start the backend and access features via browser or Gradio link.

**7. API Documentation**

- /classify: Classifies uploaded requirements  
- /generate\_code: Generates Python code from prompts  
- /fix\_code: Debugs code using AI  
- /test\_cases: Generates test cases  
- /summarize: Summarizes code

**8. Authentication**

Basic JWT-based login can be added for saving history. Currently, session is handled in memory.

**9. User Interface**

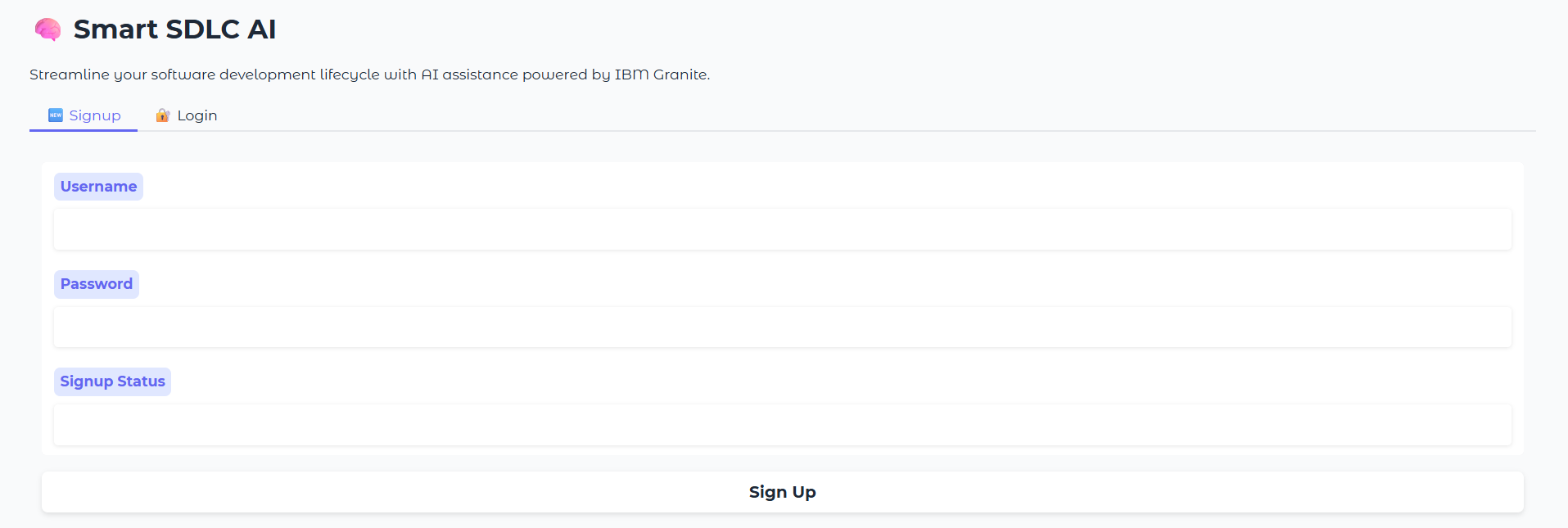
Tabbed interface with pages for each SDLC phase, floating chatbot, and support for image uploads and voice commands.

**10. Testing**

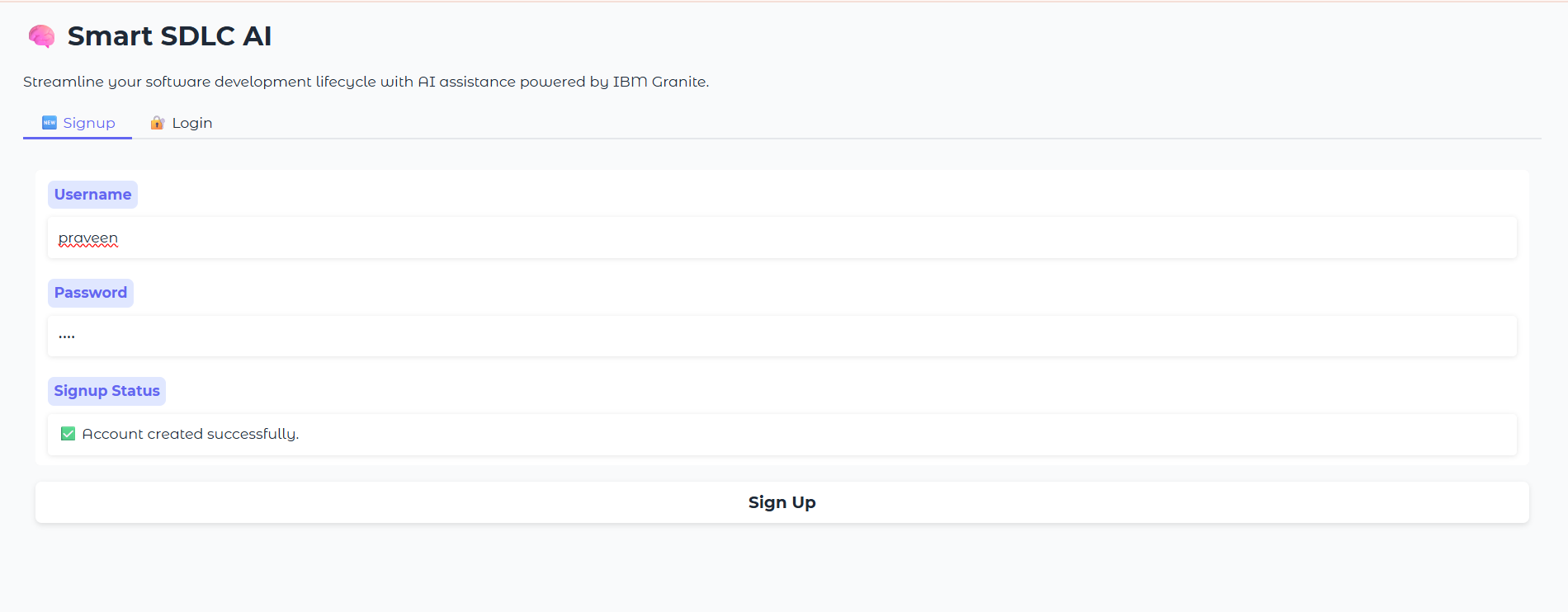
The Smart SDLC application can grow into a full AI software assistant. Features like CI/CD integration, persistent storage, multilingual prompts, and VS Code extensions will make it enterprise-ready.

**11. Screenshots or Demo**

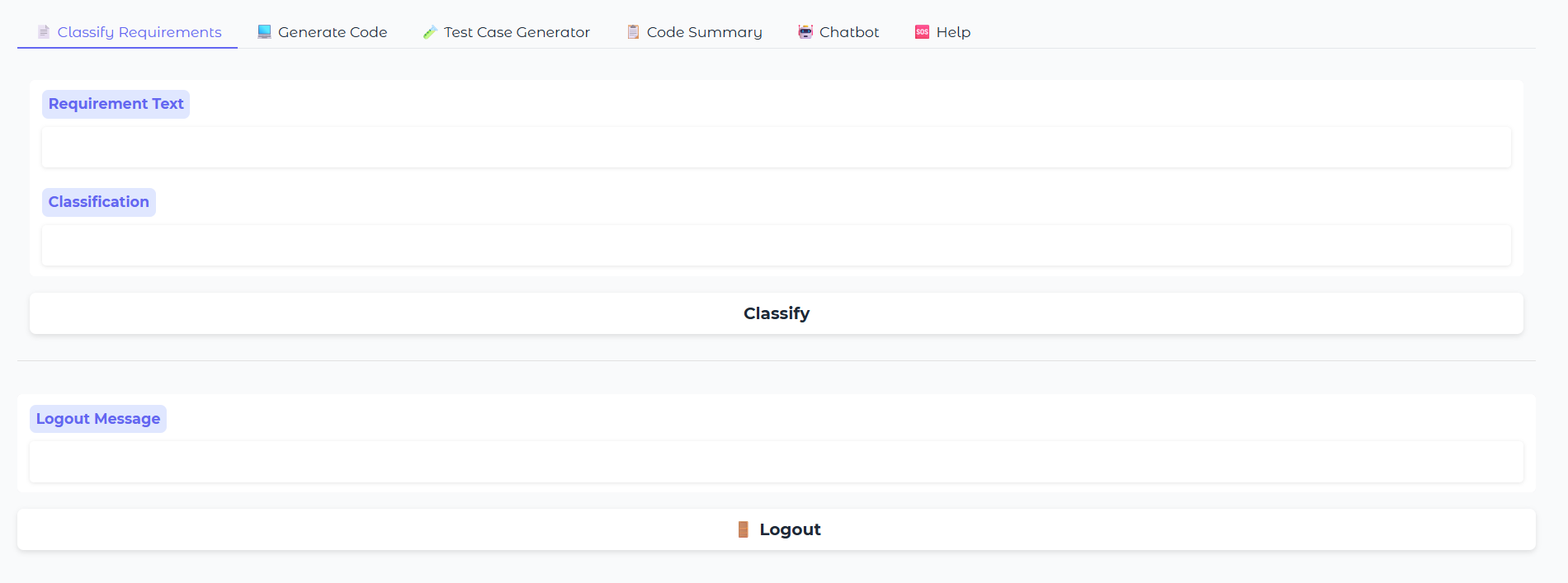
**Sign up/login page:**

****

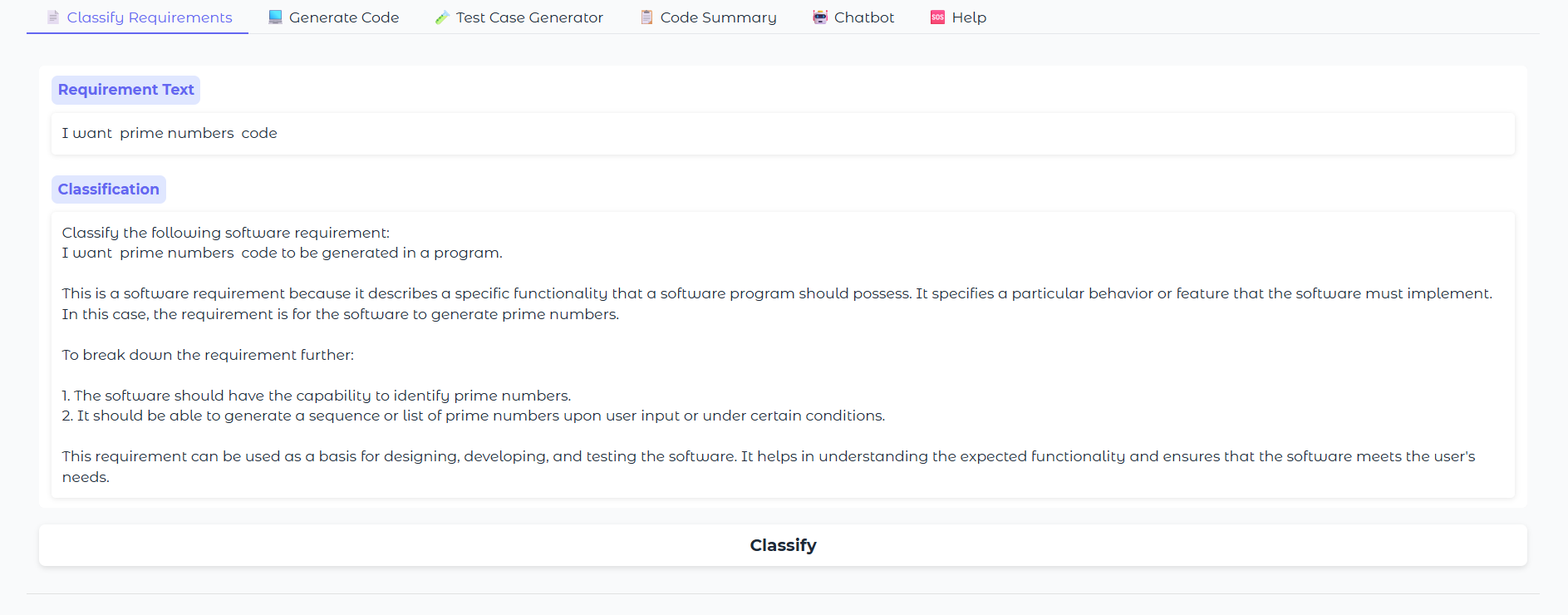
**Output:**

****

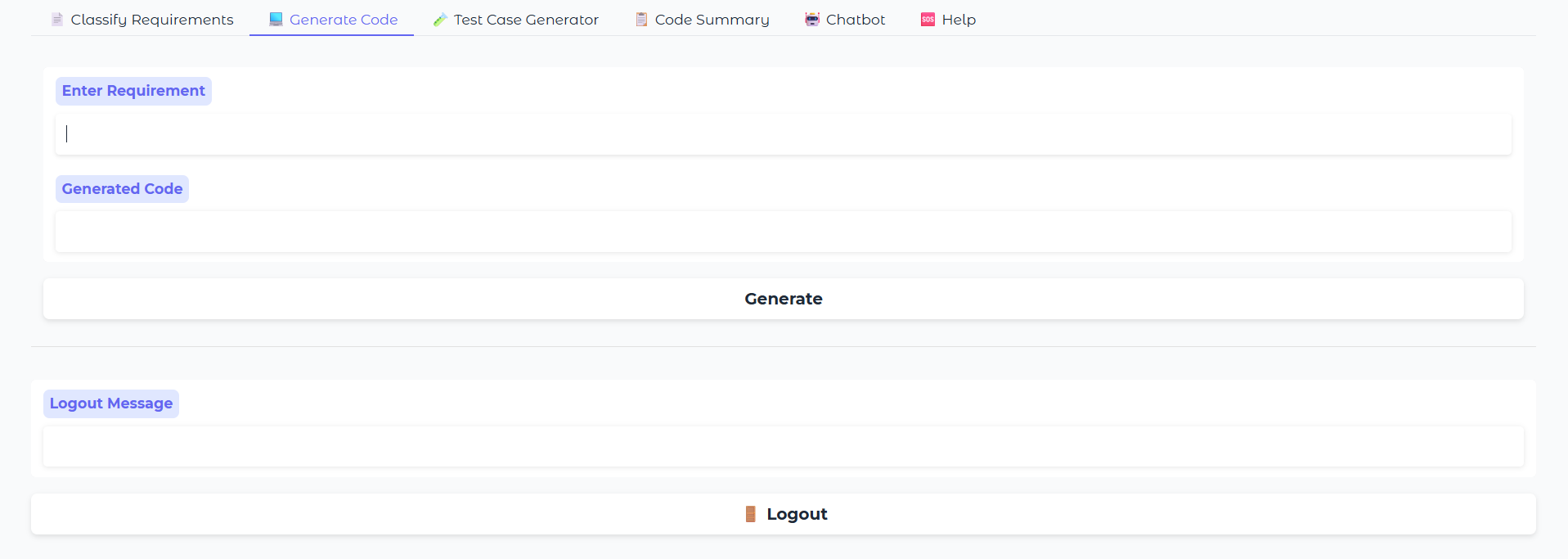
**Classify Requirements:**

****

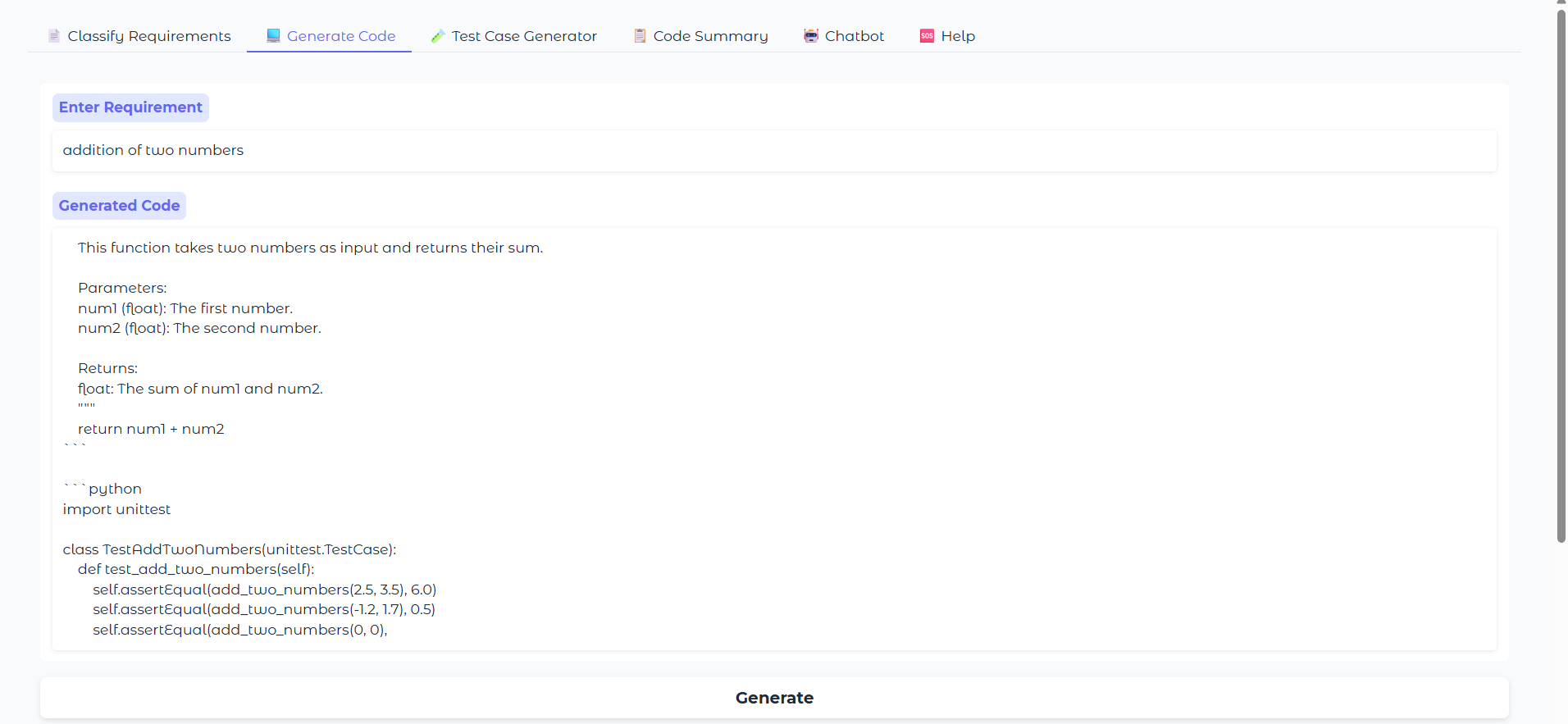
**Output:**

****

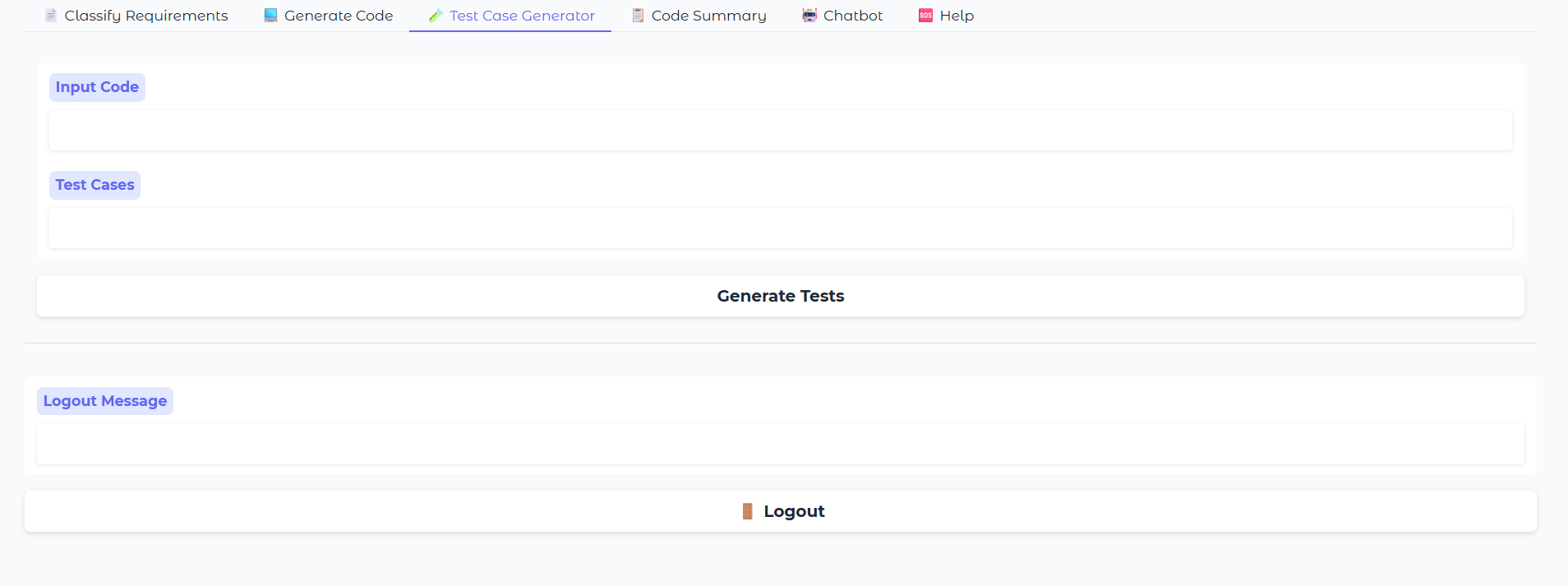
**Generate code page:**

****

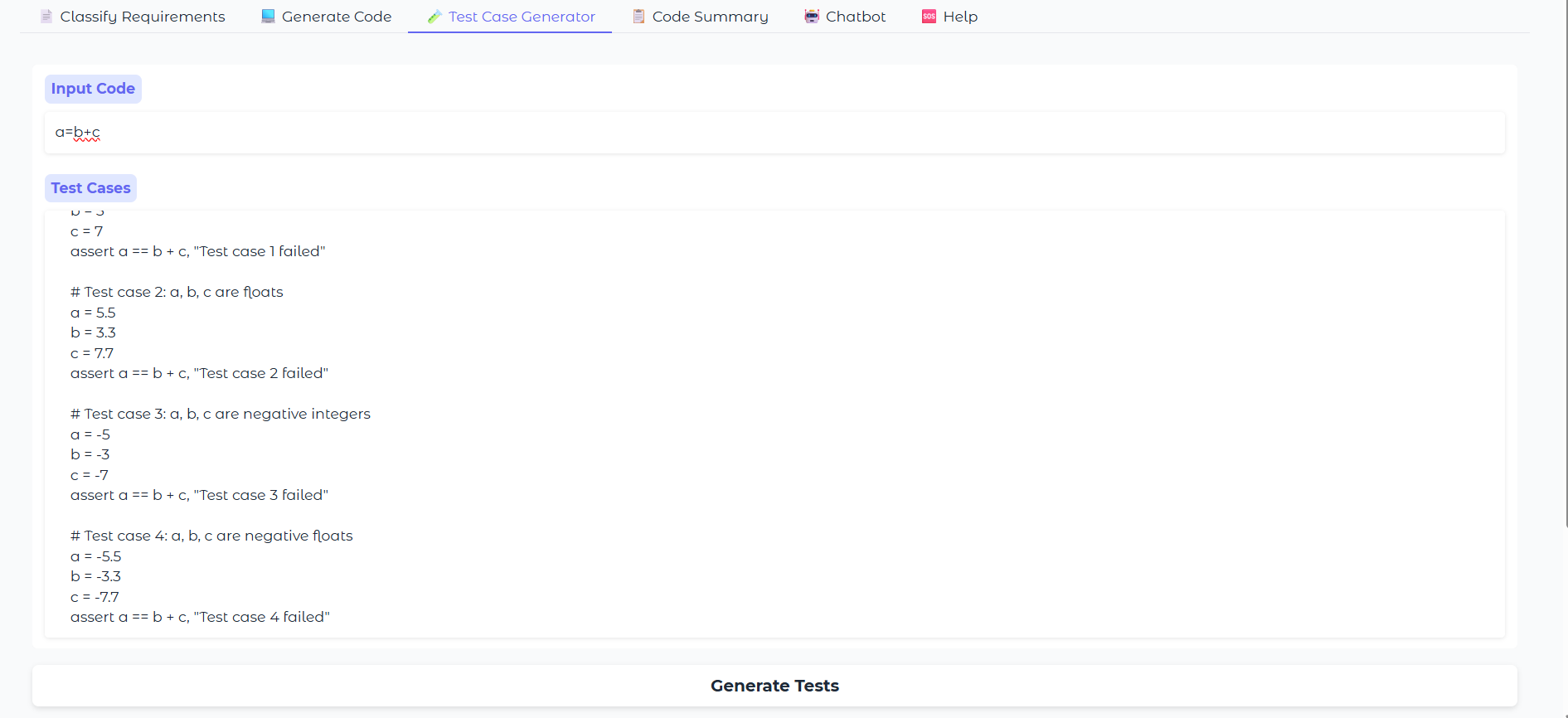
**Output:**

****

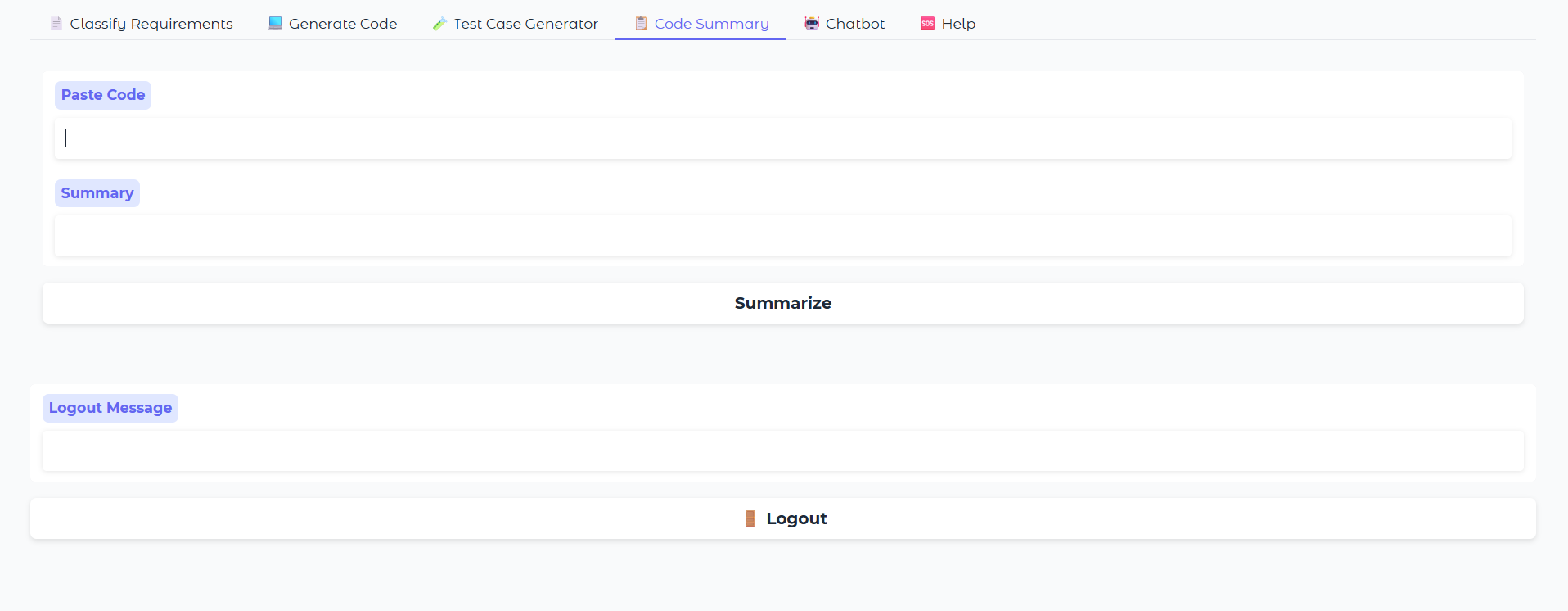
**Test case Generator Page:**

****

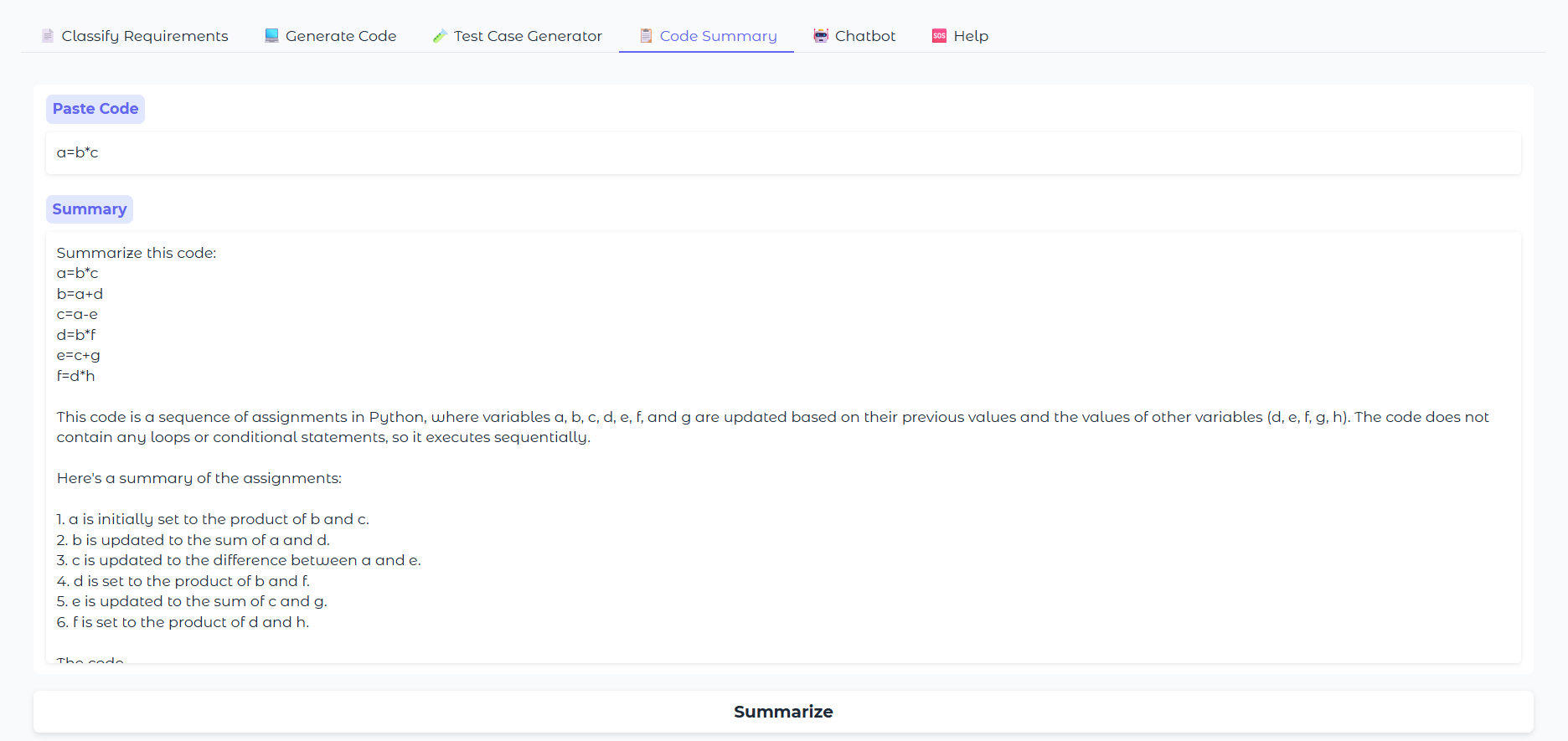
**Output:**

****

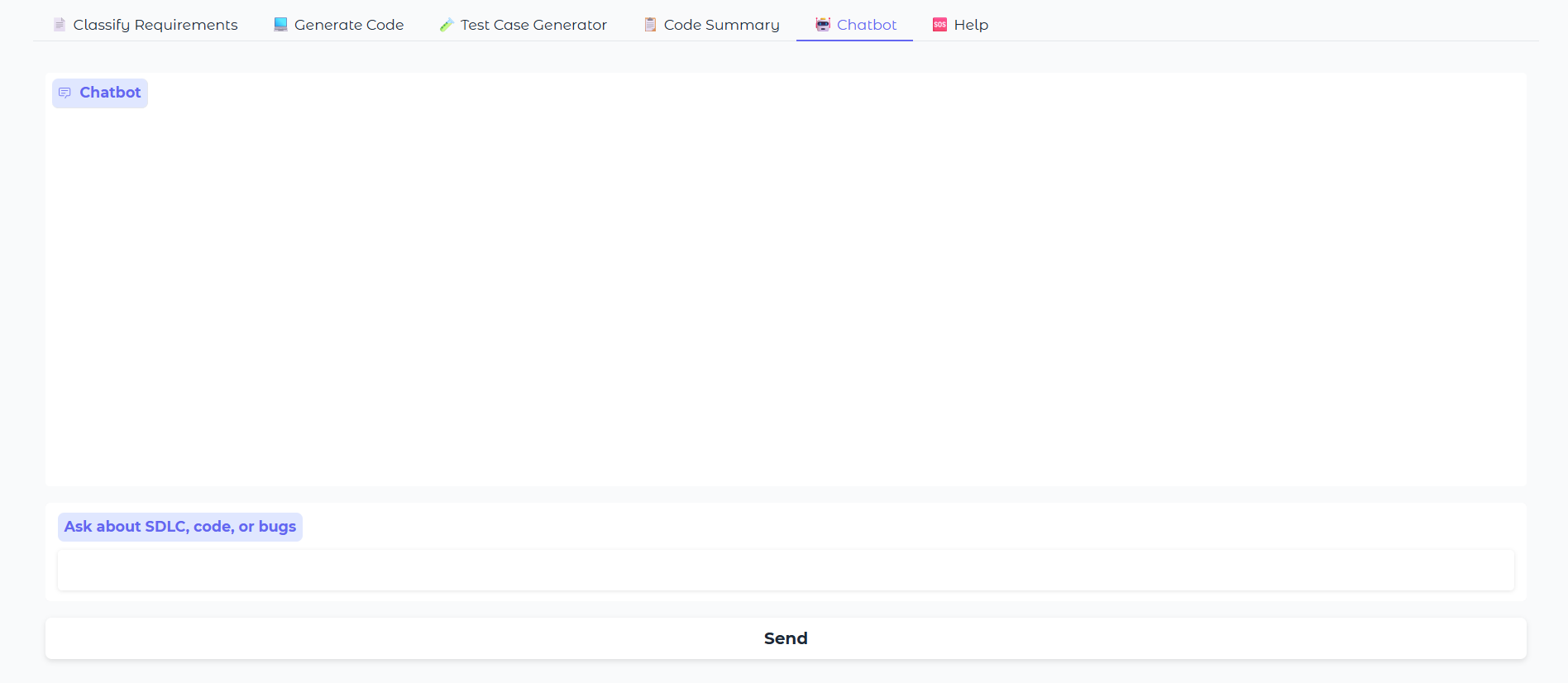
**Code Summary page:**

****

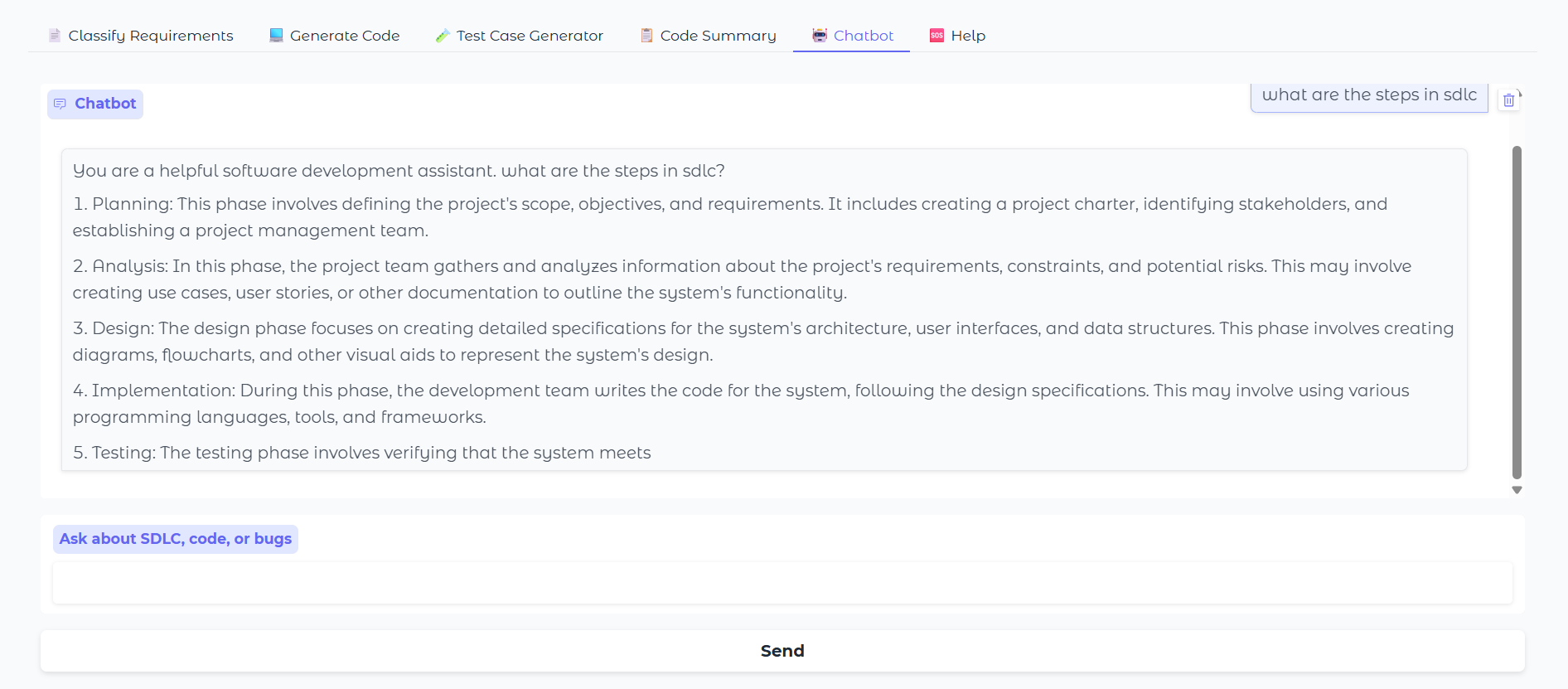
**Output:**

****

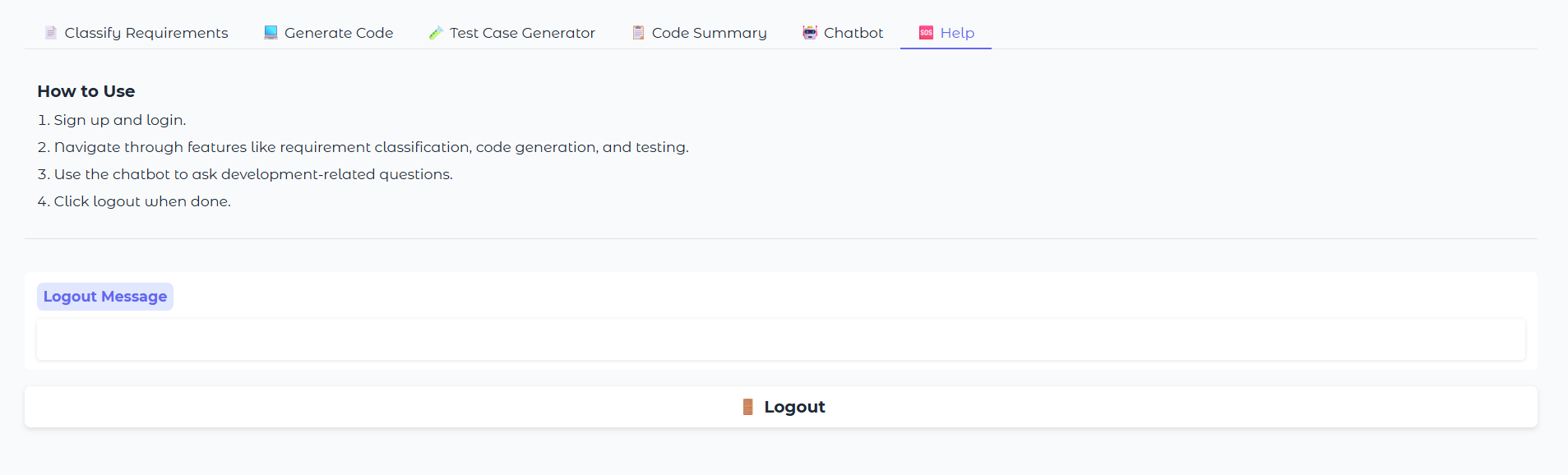
**Chatbot Page:**

****

**Output:**

****

**Help page:**

****

**12. Known Issues**

While Smart SDLC delivers a wide range of AI-powered SDLC functionalities, there are a few known limitations in its current version. Firstly, the system uses in-memory data handling, meaning user sessions and generated outputs are lost once the application is closed or refreshed. It does not yet support persistent storage or user account tracking. Secondly, the accuracy of the AI-generated code, bug fixes, or classifications may vary depending on prompt clarity and model behavior—manual review is still required before real-world deployment. The image upload feature is present but not fully functional for visual SDLC artifacts like UML diagrams. Additionally, since the system is hosted in Google Colab or a local FastAPI server, it requires a stable internet connection and may experience temporary downtime if not deployed on a cloud service. Despite these limitations, Smart SDLC functions well as a prototype and educational tool, with room for robust expansion in future versions.

**13. Future Enhancements**

Smart SDLC has strong potential for future enhancements that can transform it into a fully intelligent development assistant. One key direction is the integration of persistent storage using databases like MongoDB or Firebase to allow users to save their inputs, generated code, and progress history. The platform can also be extended to support multilingual prompts, enabling non-English-speaking users to benefit from the system. Another major enhancement is the integration of CI/CD pipelines and DevOps tools (such as Jenkins or GitHub Actions), allowing the AI-generated code to be tested, versioned, and deployed automatically. Real-time code suggestions via IDE plugins (e.g., for VS Code) could make AI assistance available directly inside developers’ editors. Further improvements include training or fine-tuning custom AI models for better accuracy in requirement classification and bug detection. With these advancements, Smart SDLC could evolve into a scalable AI-powered platform suitable for students, startups, and software enterprises.