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

Category: Generative AI with IBM

Project Description:

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-friendly interface** that allows users to upload PDFs, generate structured requirements, and transform natural language prompts into functional code.

**Scenario 1: Requirement Upload and Classification**

**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.

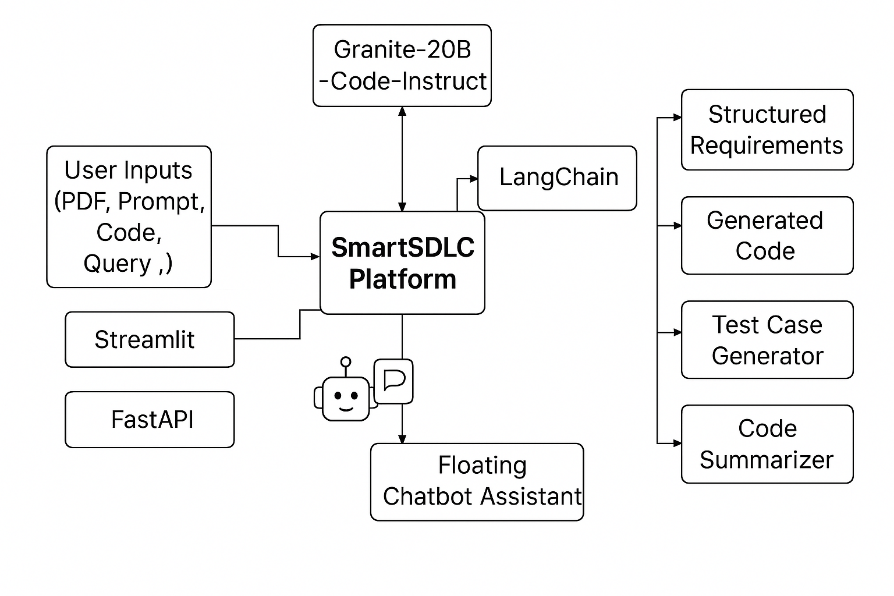
**Scenario 2: AI Code Generator**

**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.

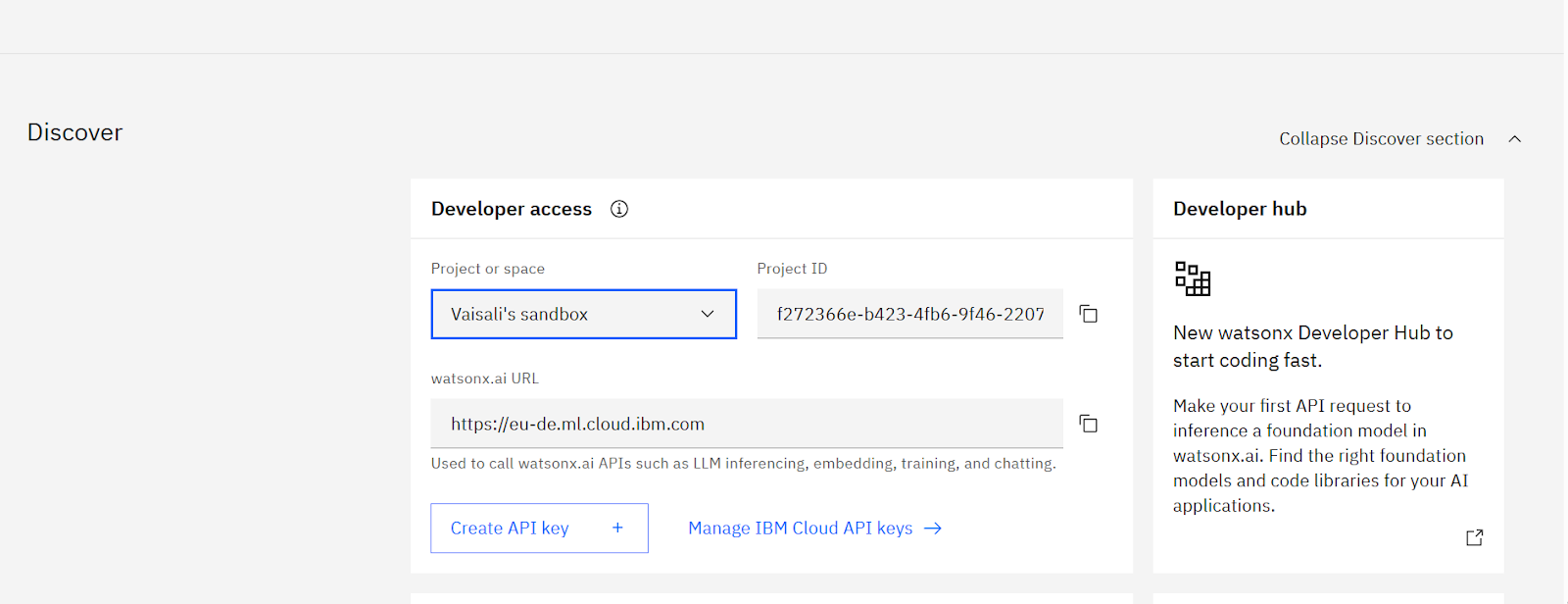
**Scenario 3: Bug Fixer**

**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.

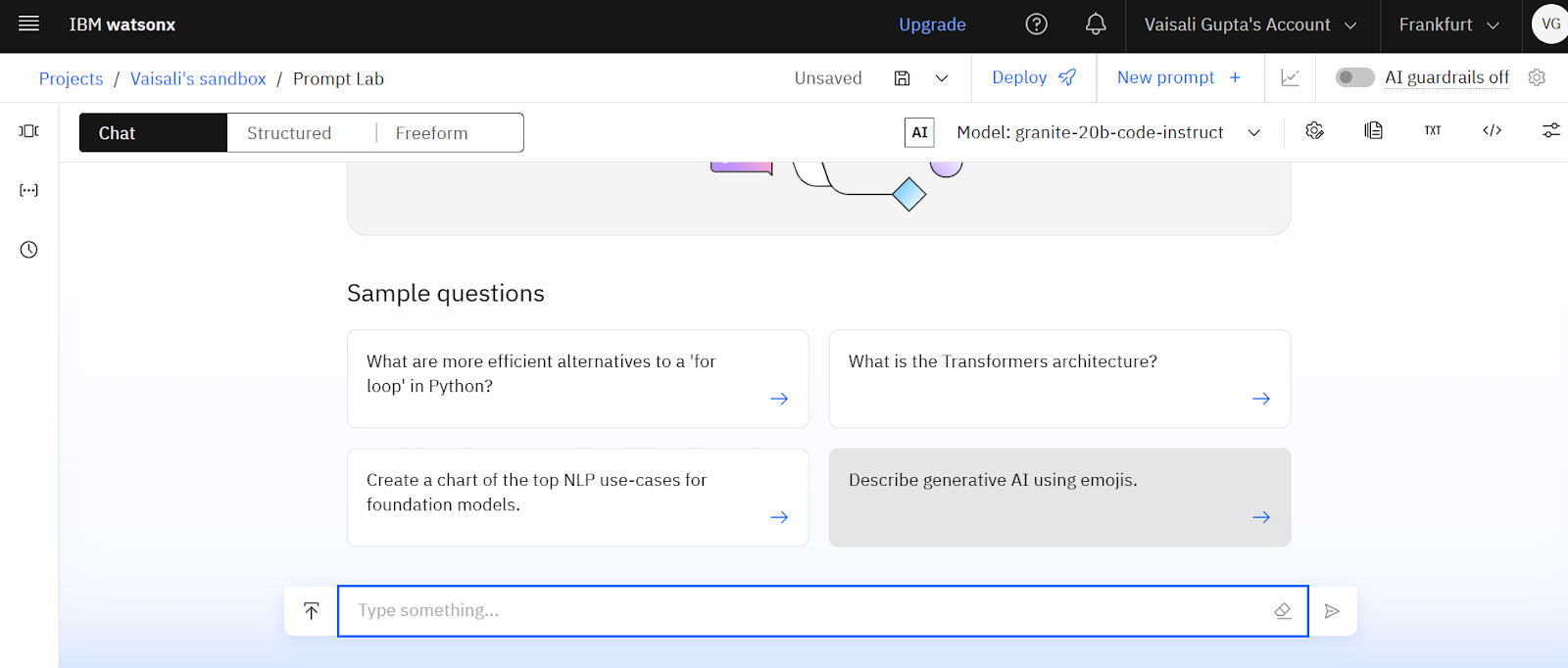
**Architecture**

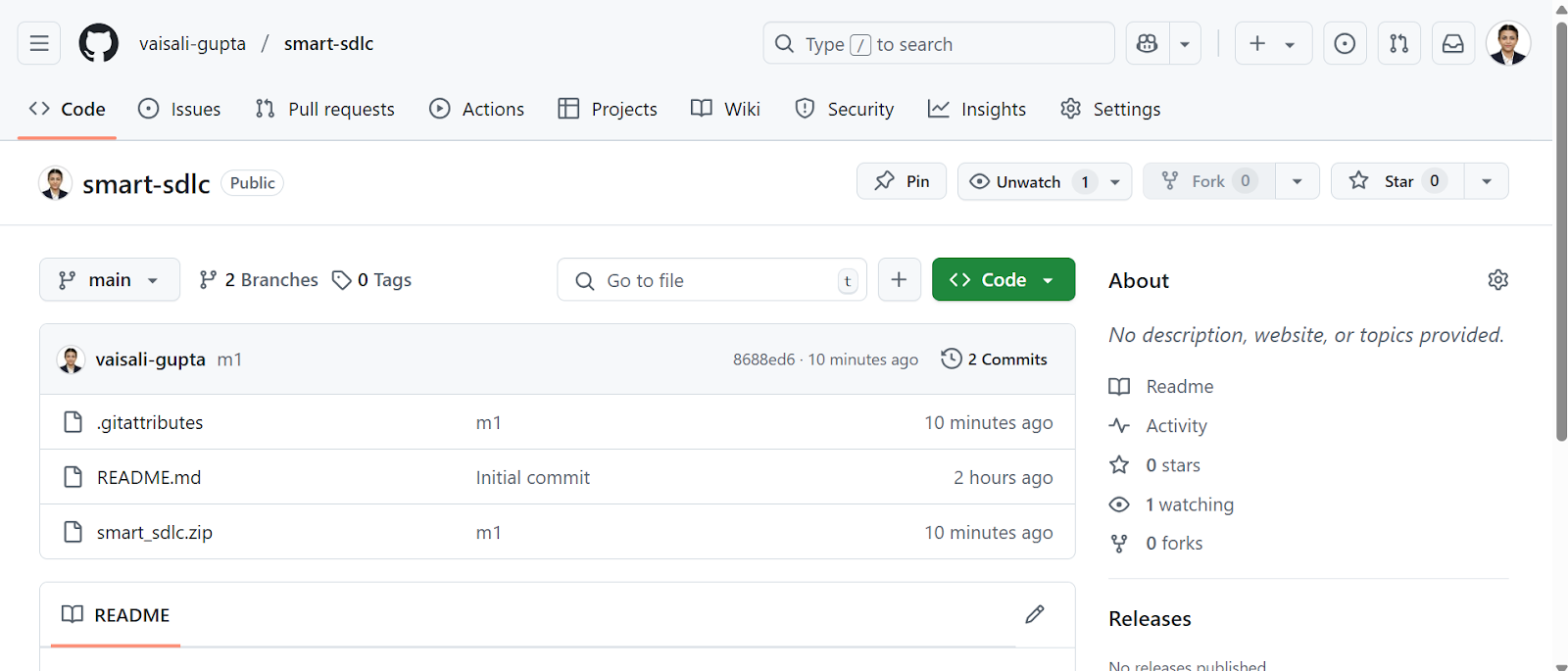


**Generating Watsonx ai API key**

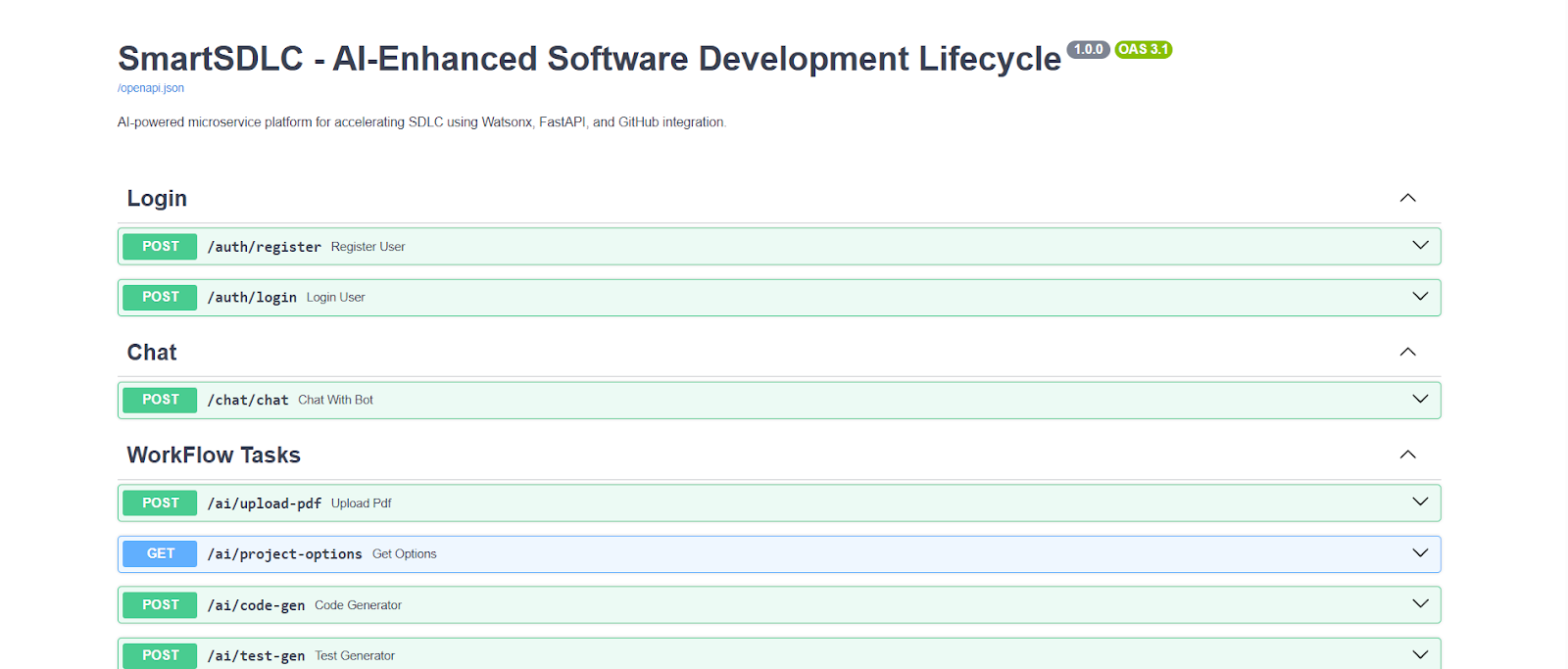


**Selecting the model**

****



**Output: Fast API Backend**



#### 1: Set Up a Virtual Environment

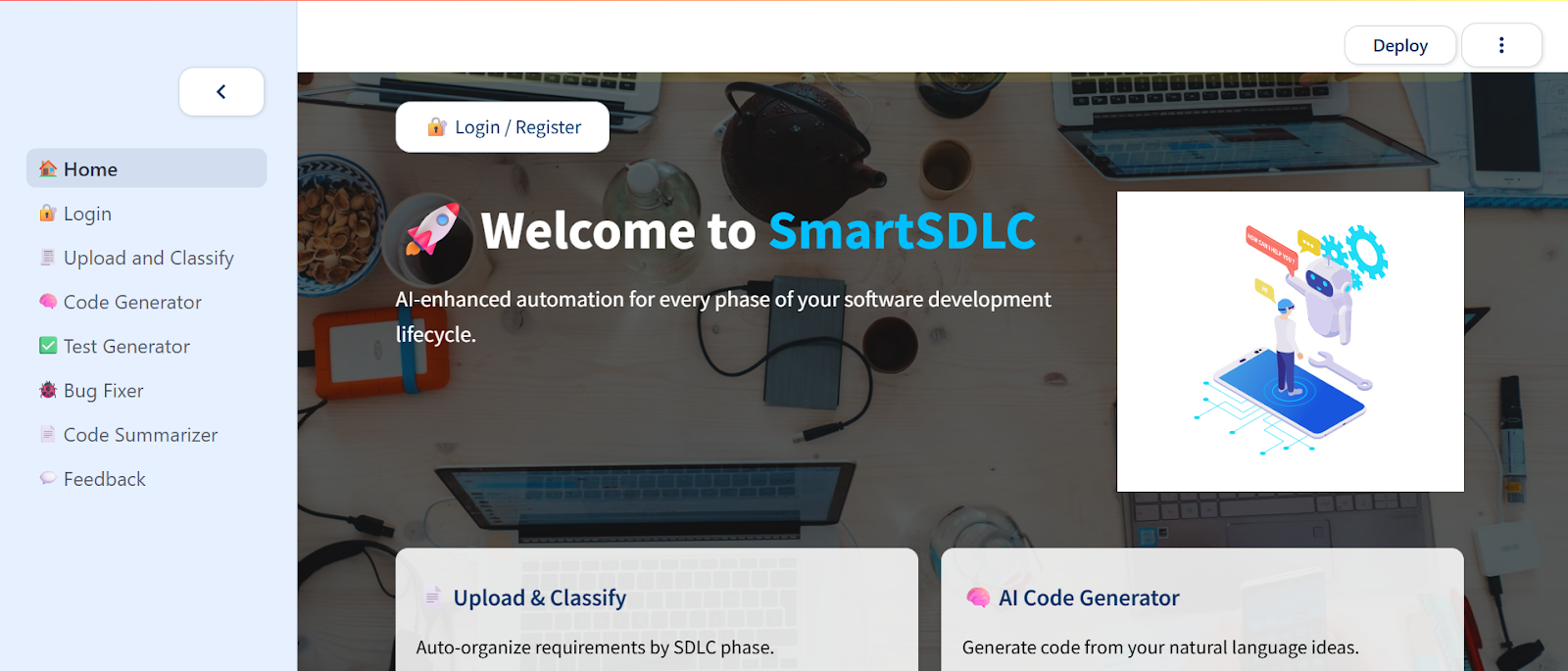
* Create a Python virtual environment to manage dependencies and avoid conflicts with other projects.
* Activate the environment and install dependencies listed in requirements.txt to ensure all libraries (FastAPI, Streamlit, Watsonx SDK, etc.) are available.

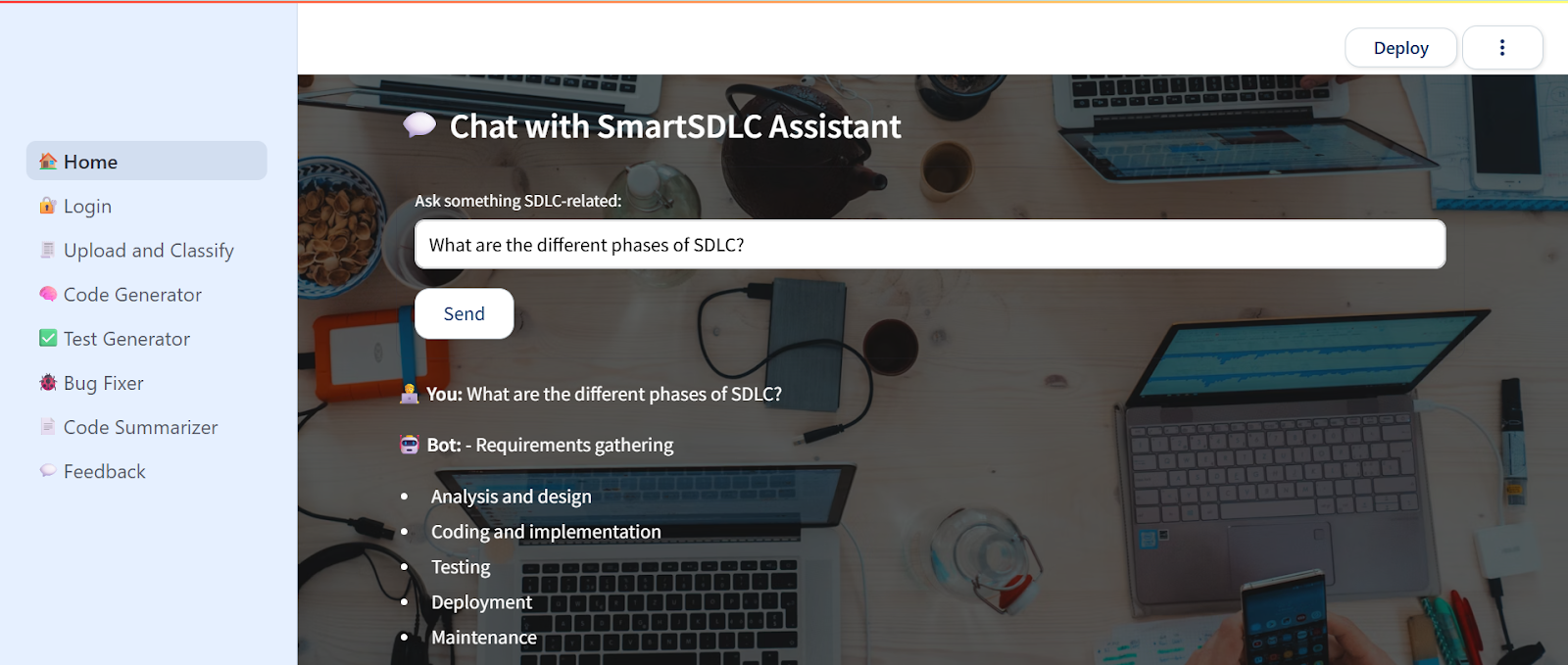
#### 2: Configure Environment Variables

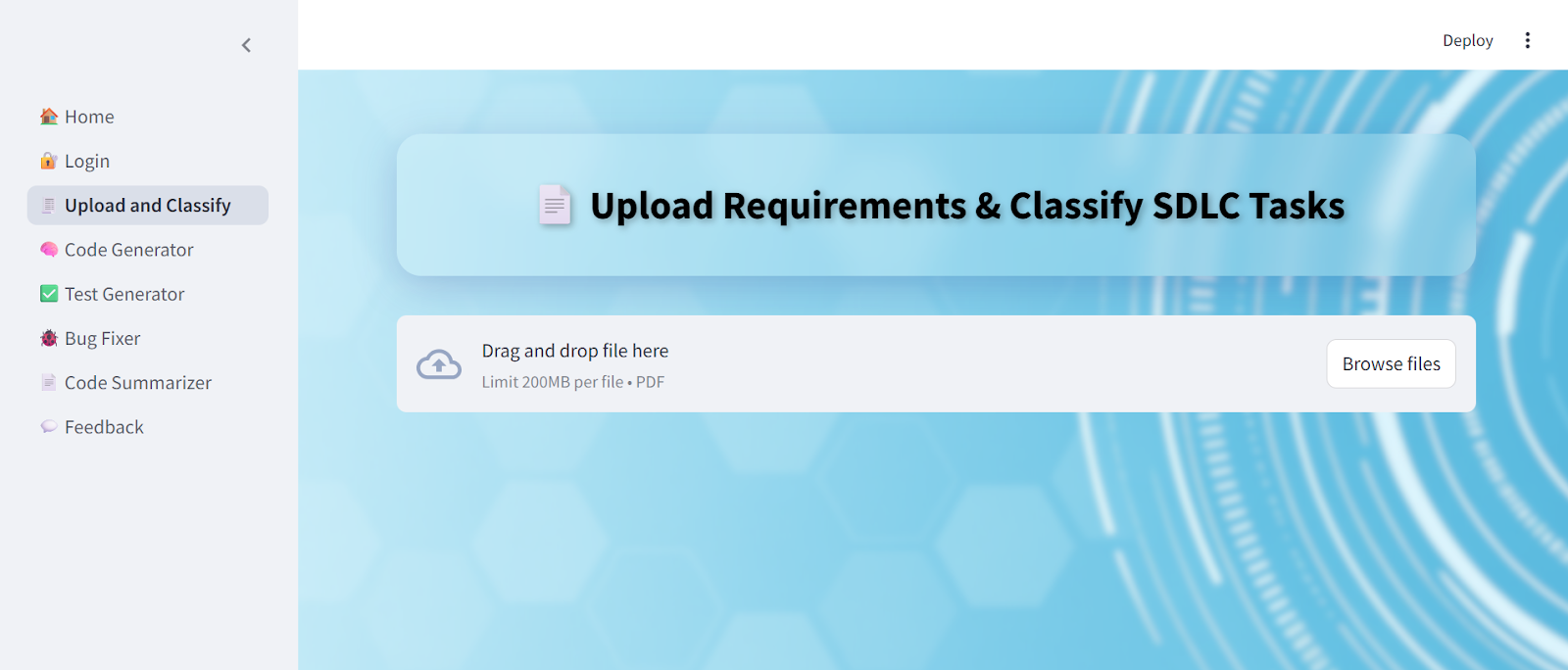
* Set environment variables for sensitive data such as **IBM Watsonx API key**, **model IDs**, and database URLs.
* Create a .env file in your project root to securely store and load these settings during runtime.
  + WATSONX\_API\_KEY=your\_ibm\_key\_here
  + WATSONX\_PROJECT\_ID=your\_project\_id
  + WATSONX\_MODEL\_ID=granite-20b-code-instruct

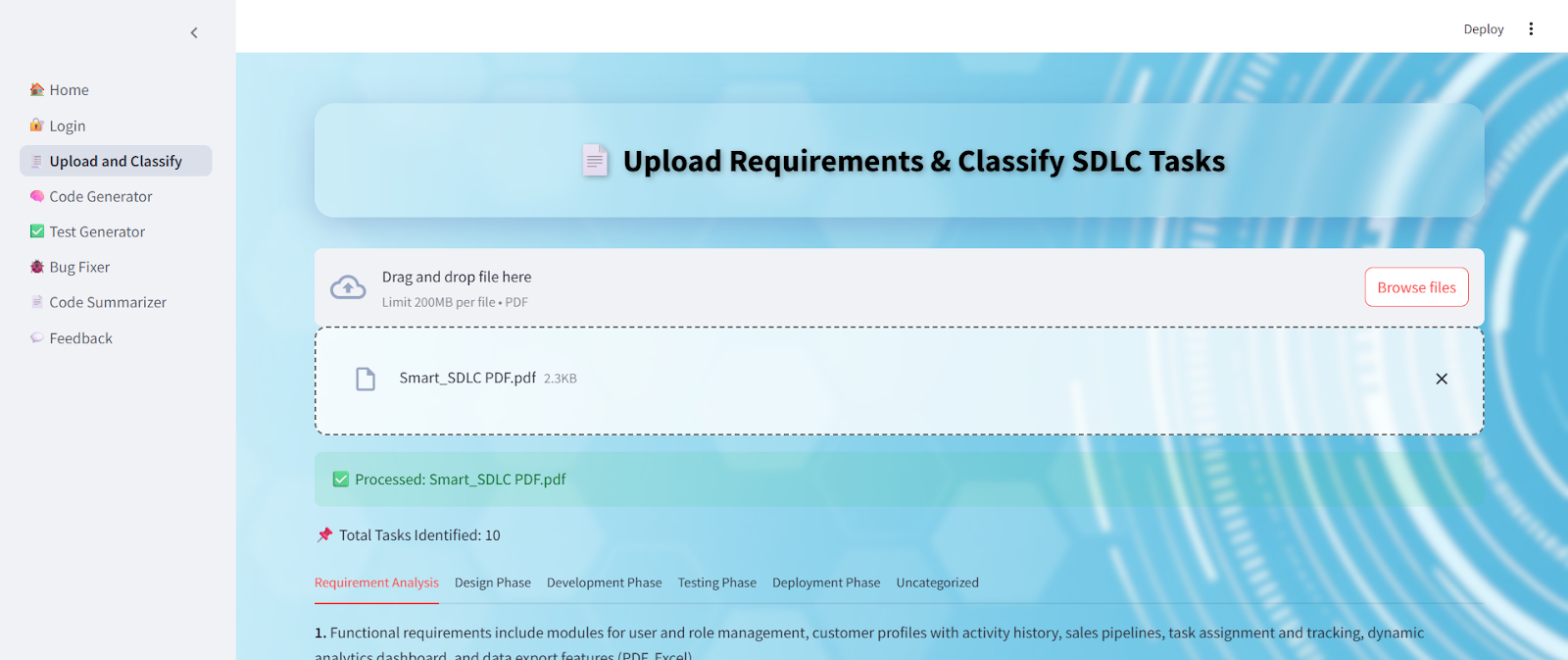
These values are loaded using python-dotenv inside your backend

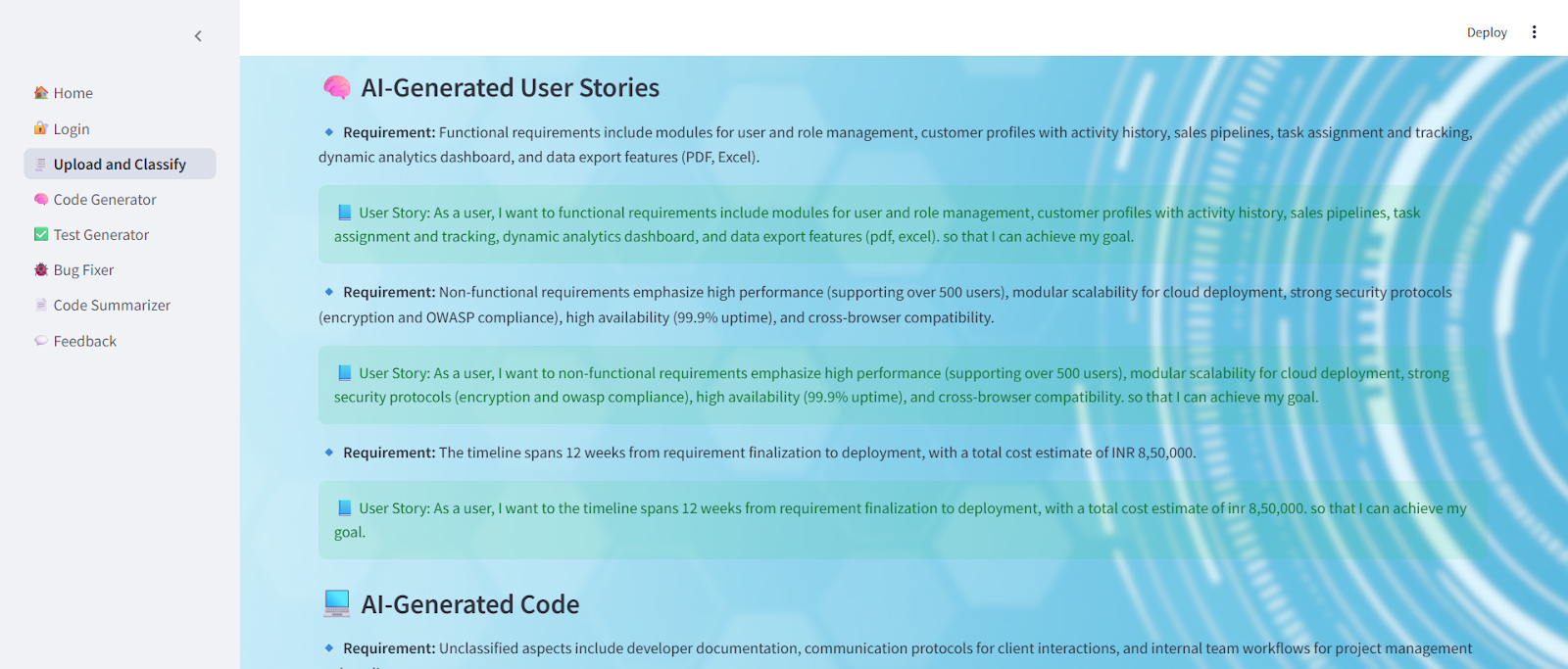
**The web application will open in the web browser:**



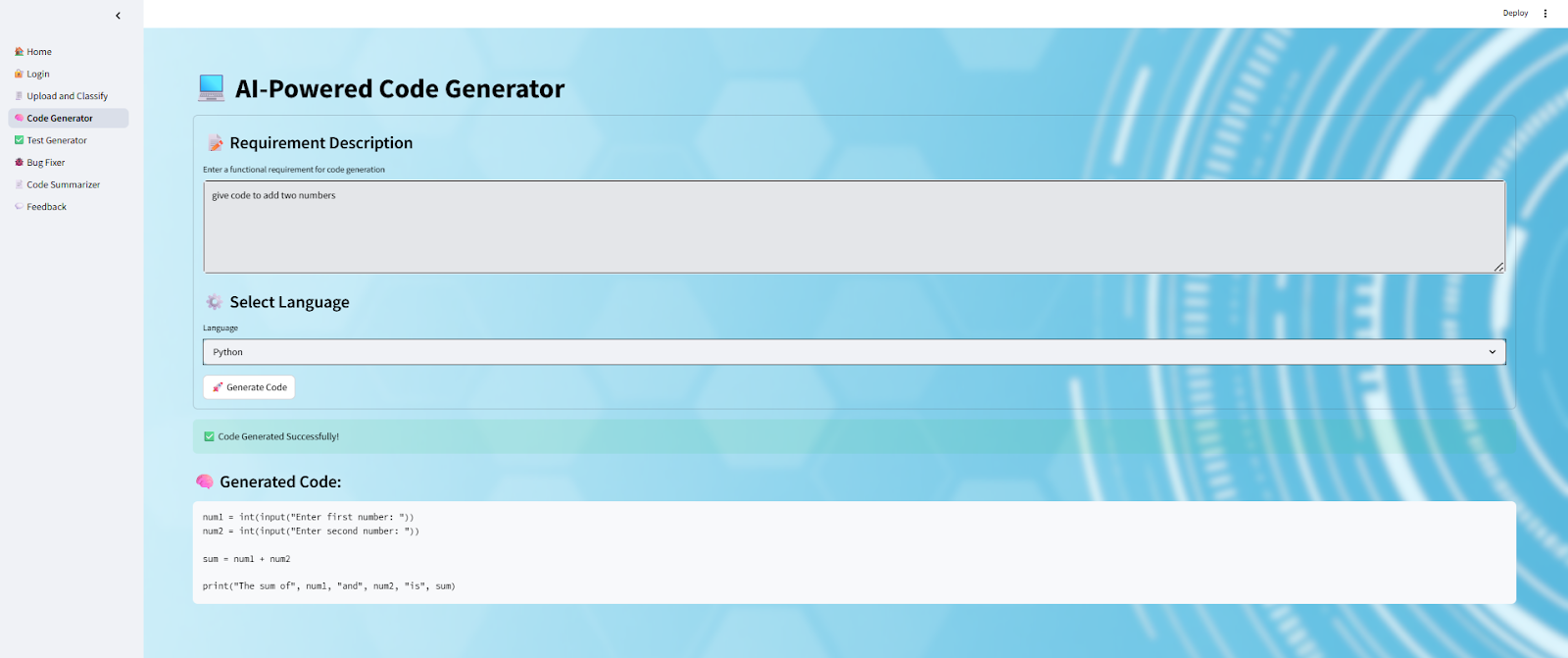


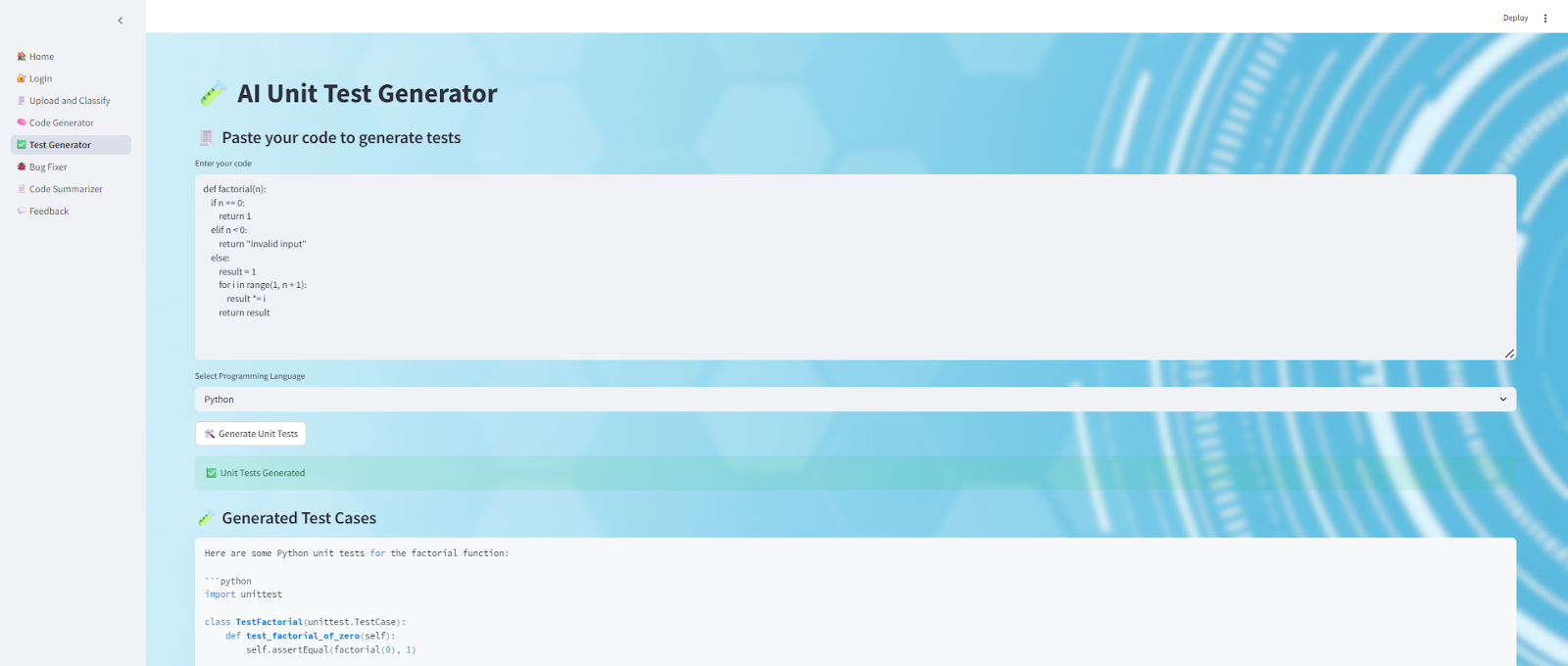


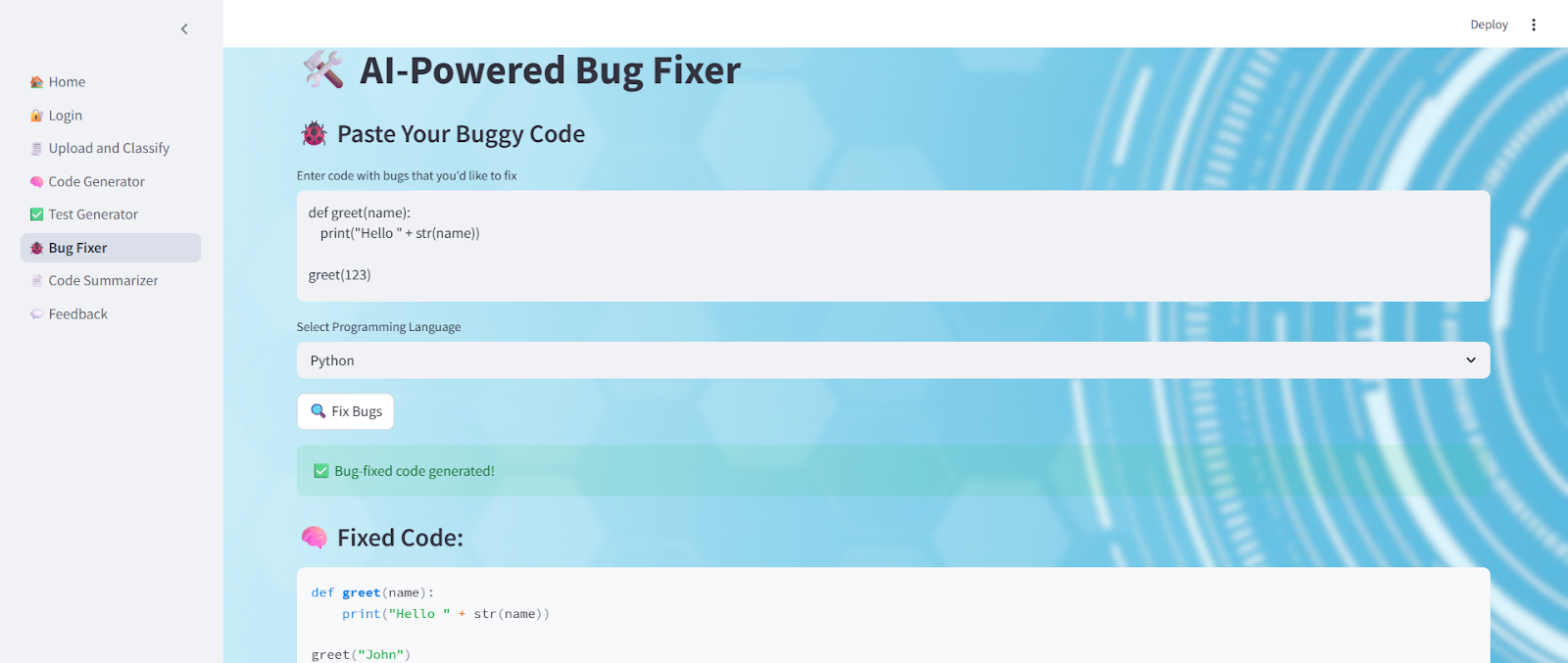


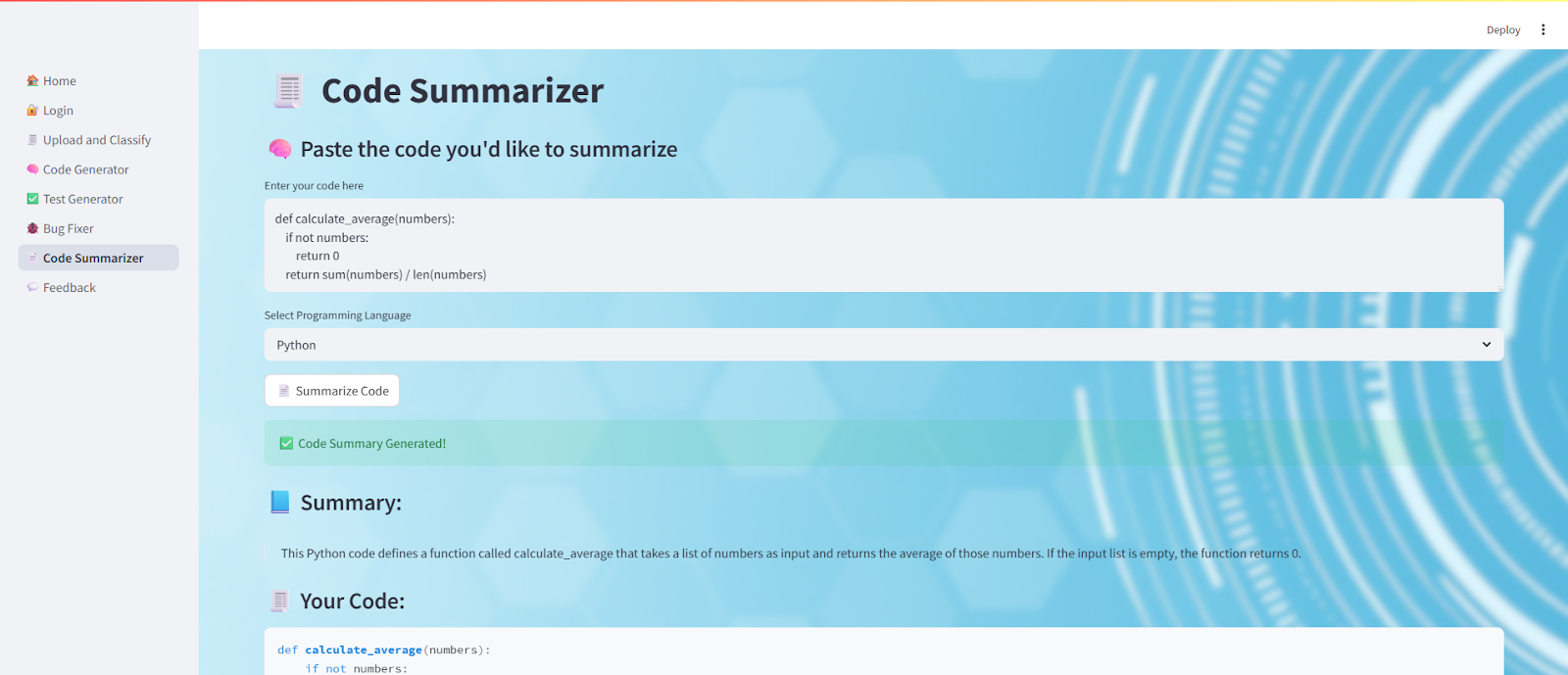


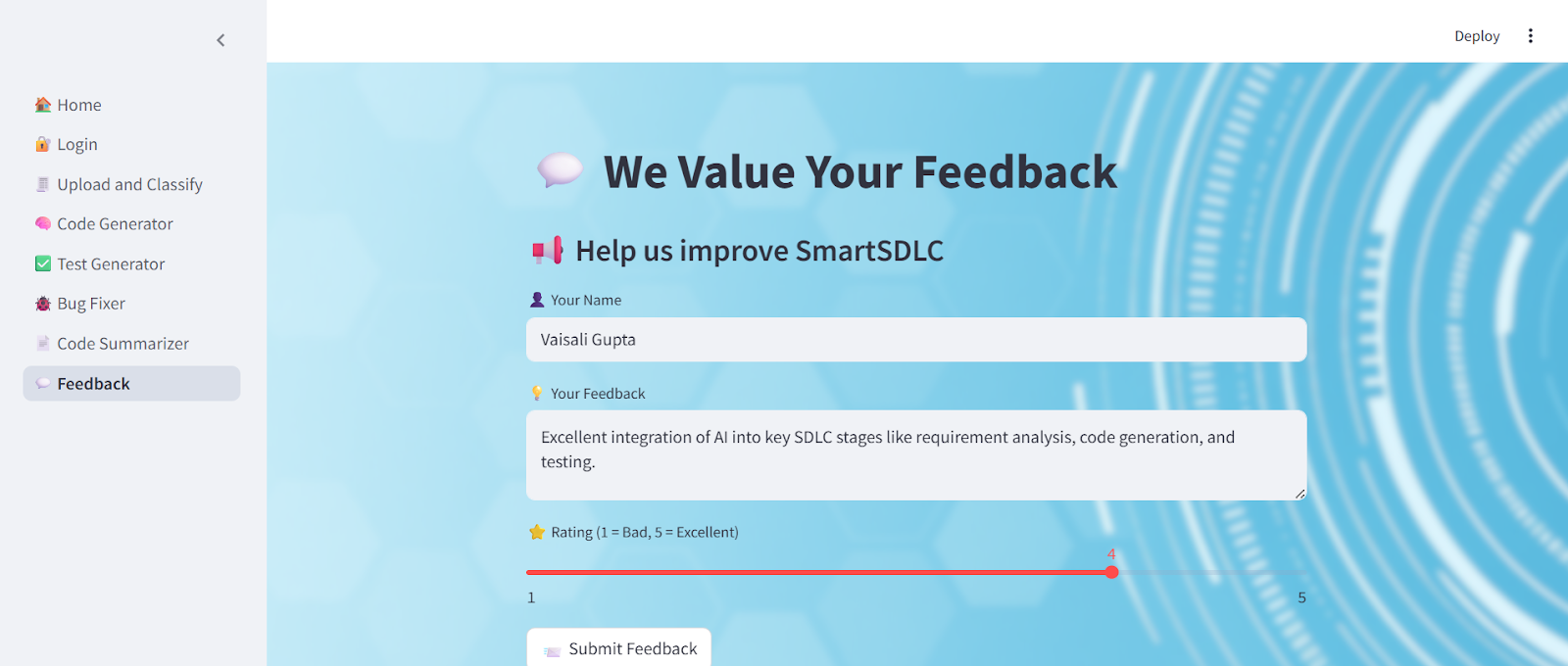


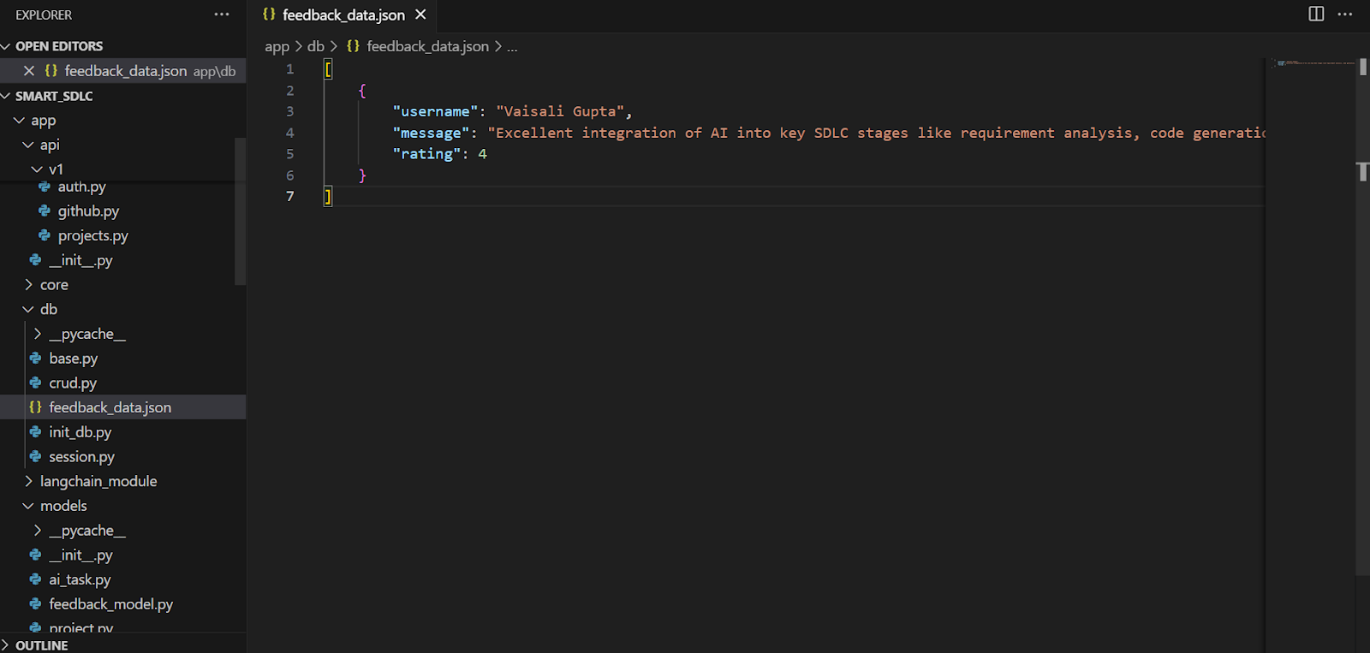


****









*.*