**Full Stack Development with MERN**

**Project Documentation format**

**1. Introduction**

**Project Title:** SmartSDLC: AI-Enhanced Software Development Lifecycle using IBM Granite  
**Team Members:**

* Seelam Maha Lakshmi (Team Leader)
* Poorna Sri
* Penke Mani Kanta

**2. Project Overview**

**Purpose:**  
SmartSDLC aims to streamline and automate various phases of the Software Development Lifecycle using AI technologies. It empowers developers and project managers by turning natural language requirements into actionable code, test cases, bug fixes, and documentation — all through a simple and interactive interface.

**Features:**

* Upload and analyze PDF/Word requirement documents
* AI-generated backend code from natural language
* Bug detection and auto-suggestions
* Unit test case generation and export
* Code summarization for documentation
* AI Chatbot for SDLC support and Q&A

**3. Architecture**

**Frontend:**  
Built with **Gradio** for interactive UI, enabling users to upload files, type queries, view code/test/doc outputs, and interact with a chatbot powered by IBM Watson Assistant.

**Backend:**  
Implemented in **Python (FastAPI)**, responsible for routing, AI model orchestration (Watsonx / Hugging Face), file handling, and user session management.

**Database:**  
Uses **SQLite** (dev) or **MySQL/PostgreSQL** (production) via **SQLAlchemy** to store user data, file history, generated outputs, and logs.

**4. Setup Instructions**

**Prerequisites:**

* Python 3.9+
* Optional: SQLite or MySQL/PostgreSQL
* IBM Cloud credentials for Watson Assistant & Watsonx.ai

**Installation:**

1. Clone the repository: git clone <repo-url>
2. Navigate into the project folder
3. Create virtual environment: python -m venv venv && source venv/bin/activate
4. Install dependencies: pip install -r requirements.txt
5. Create .env file with required keys (IBM\_API\_KEY, DB\_URL, etc.)

**5. Folder Structure**

**Client (Gradio UI):**

* app.py: main UI entrypoint
* components/: reusable UI blocks (upload, chatbot, viewer)

**Server (Python backend):**

* main.py: FastAPI app
* models/: AI-related logic and wrappers
* services/: IBM Watsonx / Hugging Face APIs
* db/: database models and ORM logic
* tests/: unit and endpoint testing scripts

**6. Running the Application**

* **Frontend:** python app.py
* **Backend:** uvicorn main:app --reload

**7. API Documentation**

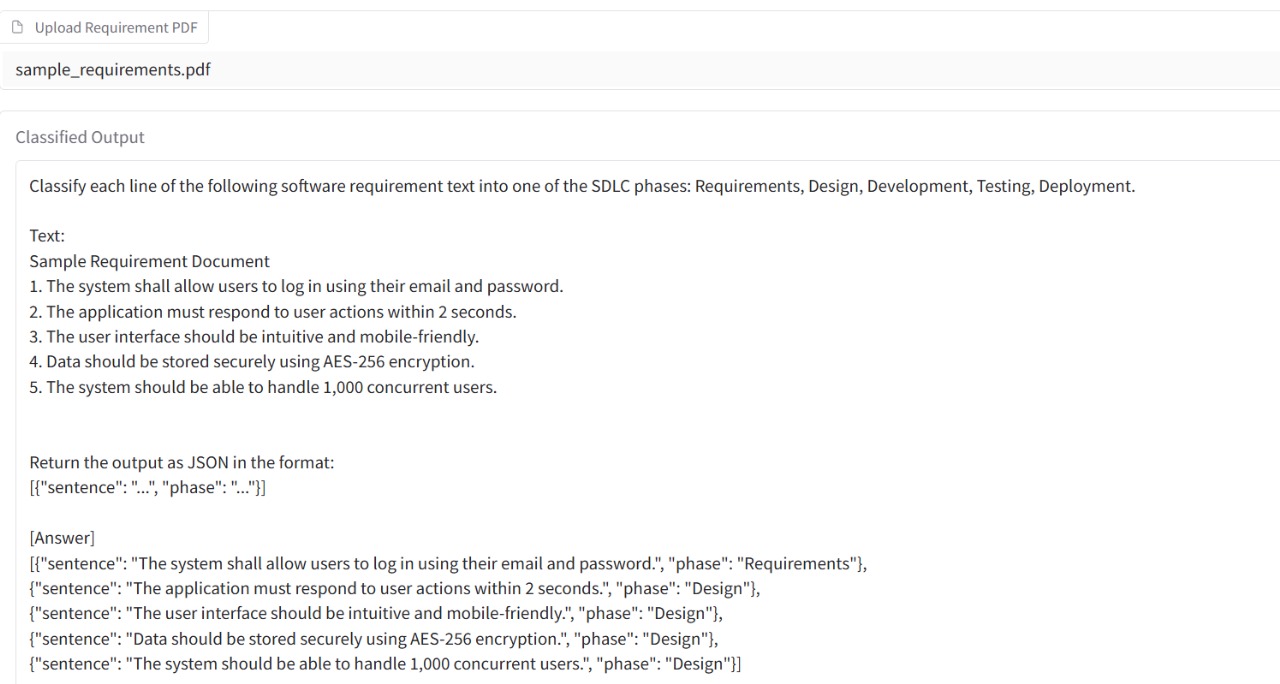
| **Endpoint** | **Method** | **Request Parameters** | **Response** |
| --- | --- | --- | --- |
| /register | POST | email, password | Success/status message |
| /confirm-email | GET | token | Account activation confirmation |
| /login | POST | email, password | Auth token + profile |
| /upload | POST | file | Extraction summary, file ID |
| /generate-code | POST | requirement ID, language | Generated backend code |
| /test-cases | POST | code | Unit test cases output |
| /bug-fix | POST | buggy code | Suggested bug fixes |
| /summarize-code | POST | code | Code summary |
| /chatbot | POST | question | AI-powered SDLC support answer |

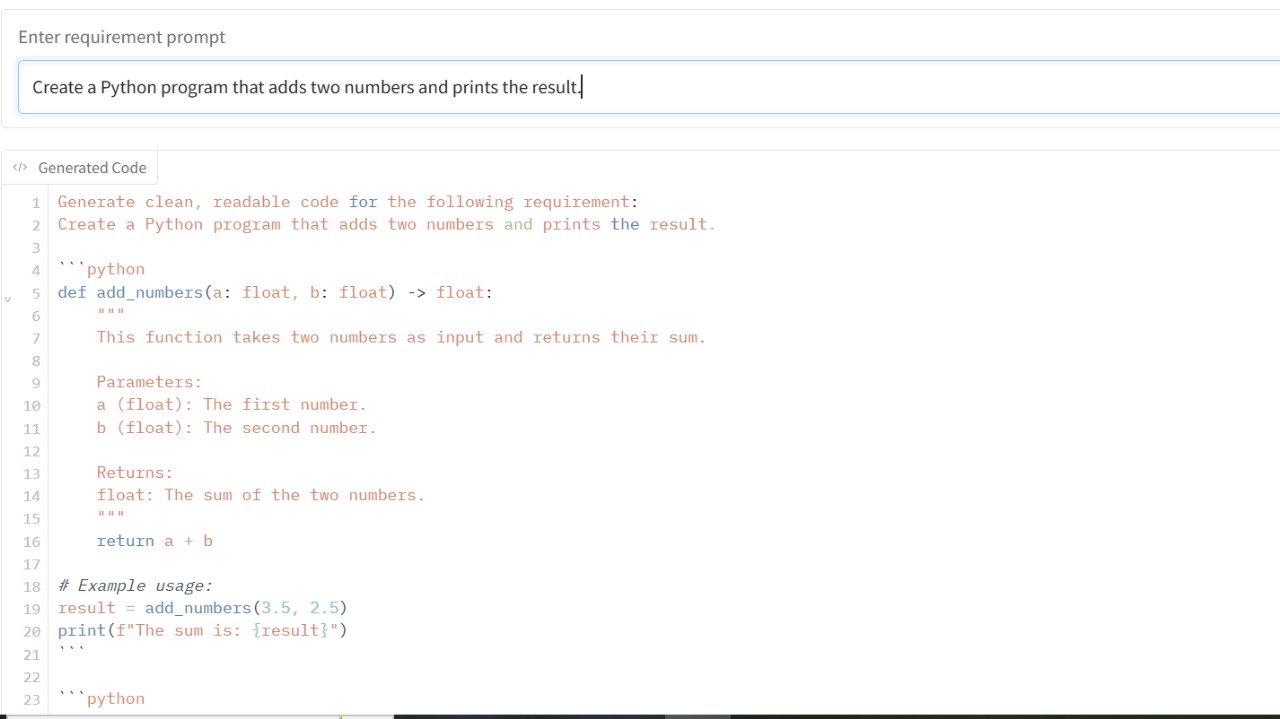
**8. Authentication**

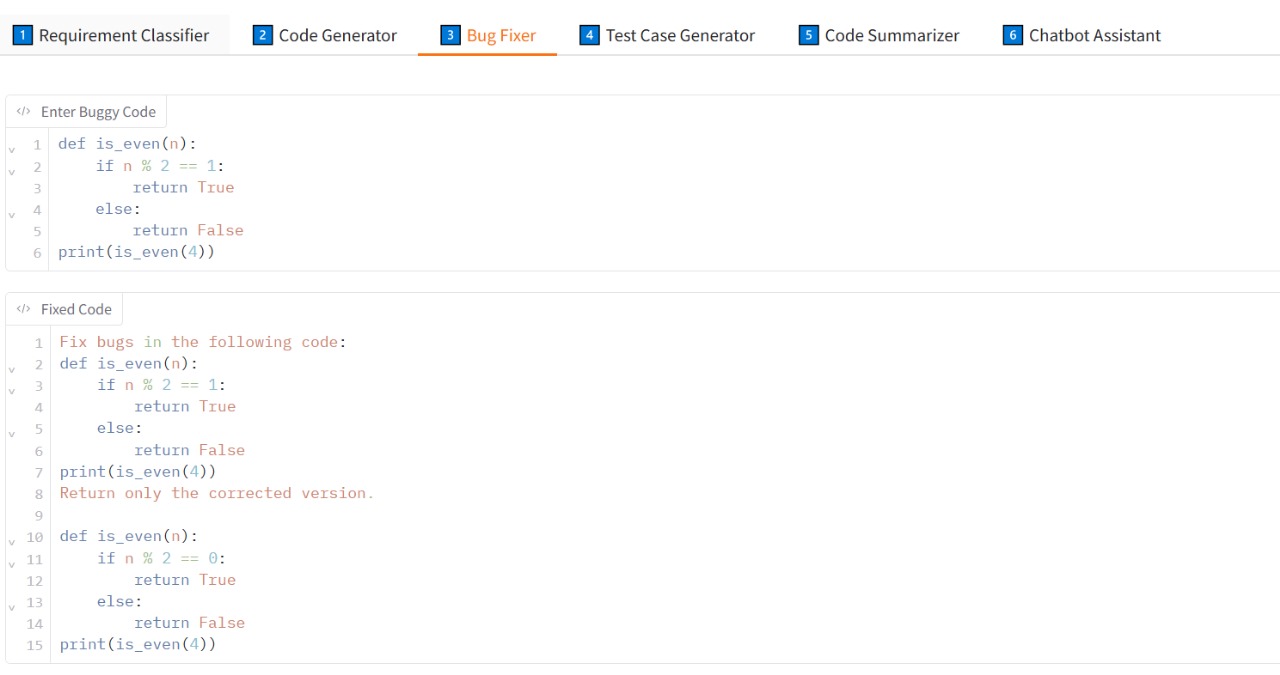
* Uses **JWT** tokens generated after login
* Middleware validates tokens on protected routes
* Token expiry: 1 hour; refresh enabled for long sessions

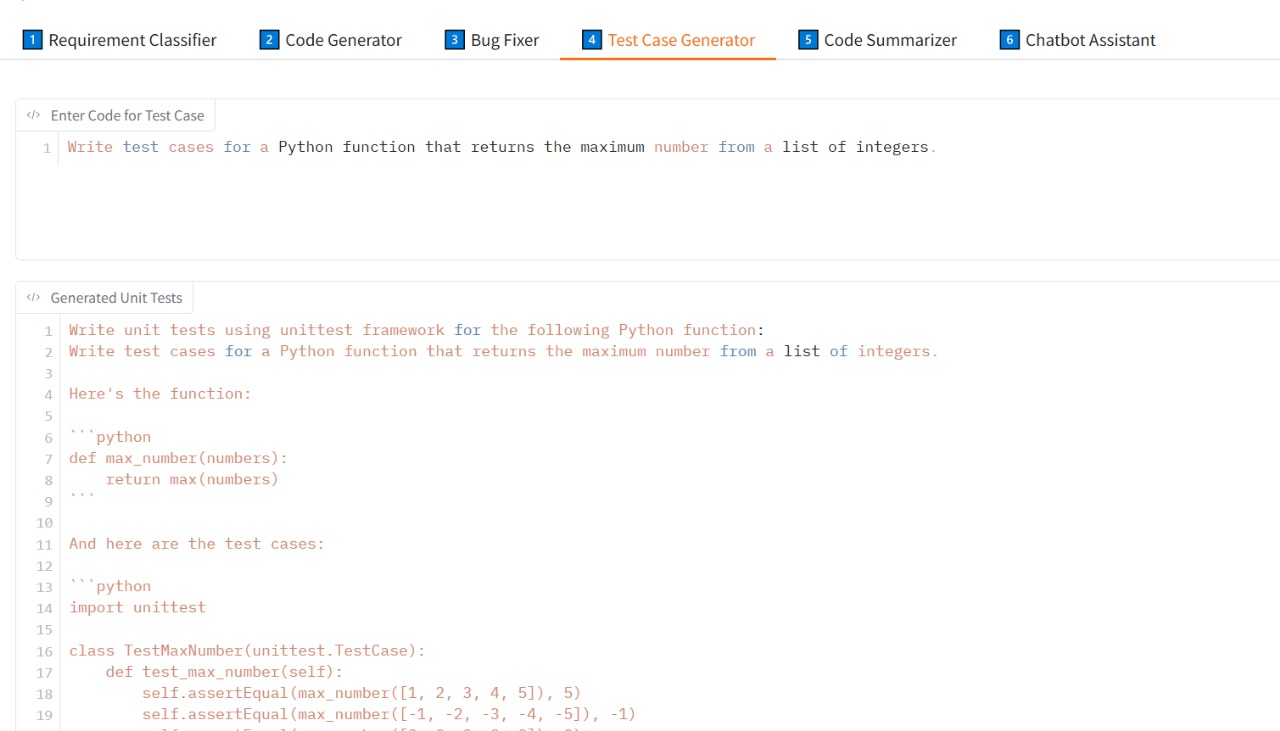
**9. User Interface**

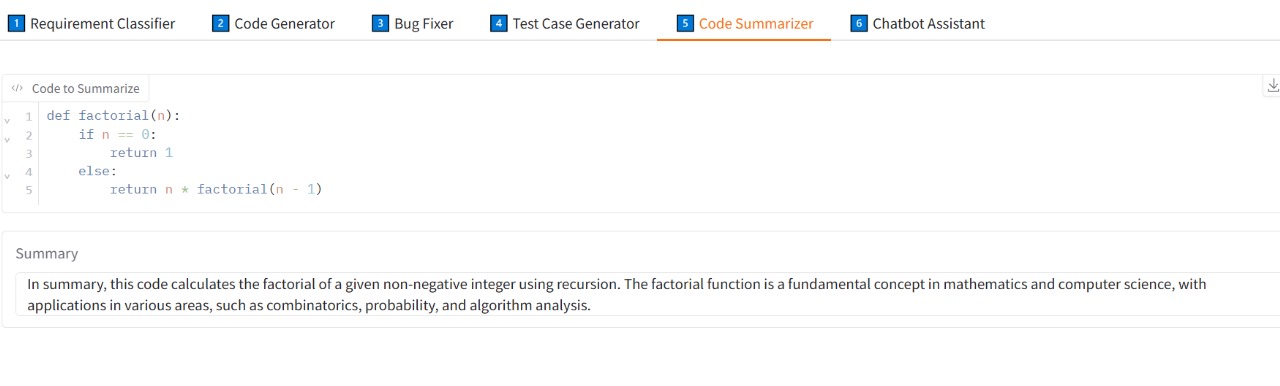
**Screenshots/Demos:**











**10. Testing**

**Strategy & Tools:**

* Unit Testing: pytest for service/model logic
* API Testing: FastAPI TestClient
* UI Testing: Manual QA via Gradio interface

**11. Screenshots or Demo**

* Screenshots and user walkthrough videos (add in appendix)
* GitHub Repository: [GitHub Link Placeholder]
* Demo: [Live Demo Link Placeholder]

**12. Known Issues**

* Minor delay in response (~2s) from large code inputs
* Summary generation may truncate for very large files
* Model access limited by IBM Cloud API quotas

**13. Future Enhancements**

* Add multi-language code generation (e.g., Java, Node.js)
* CI/CD auto-deploy integration
* Voice-based SDLC chatbot using STT/TTs
* Admin dashboard with analytics & logs
* AI model fine-tuning for domain-specific projects