SmartSDLC – AI-enhanced Software Development Lifecycle

❖ Project Documentation Format

1. Introduction

- **Project Title: SmartSDLC - AI – Enhanced Software Development Lifecycle (Generative AI with IBM Cloud)**
- **Team Members:**
  - **Jaya Adithya Nagalla (Project Lead & Developer):**
    Managed overall planning, led development, and handled deployment, Coordinated tasks and meetings, Setup project structure & Streamlit UI , Integrated all Features, Final deployment and GitHub.
  - **Paramata Poojith Rajkamal (Feature Developer):**

    Developed Requirement Assistant and Test Case Generator modules, Handled text-based content logic
  - **Syed Mohammad Saad(UI & UX Designer):**

    Designed the app interface and performed feature testing,

    Tested all features for accuracy
  - **Anudeep Gudala(Documentation and Reporting):**

    Wrote README and report content, Collected screenshot

    Helped with final submission and presentation

Project Overview

- **Purpose:**
- SmartSDLC is an intelligent software development lifecycle assistant designed to optimize and automate each phase of SDLC using Generative AI. It provides guidance, templates, auto-generated content, and checklists across all stages of software development. The tool uses simulated AI responses (mock LLM) to show how AI could enhance planning, design, development, and testing.
- **Features:**
    - Simulate AI-powered support for every phase of SDLC
    - Streamline project documentation and task handling
    - Build a modular full-stack application with mock AI
    - Enable future upgrades with real LLM integration

Project Planning

- **Problem Statement:**
  Traditional SDLC processes are manually driven, time-consuming, and repetitive. SmartSDLC solves this by introducing AI-powered automation to assist developer s and project managers in faster, error-free execution.
- **Proposed Solution:**

  Create a web-based system where users interact with an AI module that provides guidance, templates, and suggestions for requirements gathering, system design, testing, and deployment using intelligent prompts

Implementation :

- **Technologies Used:**
  - Python 3.10
  - Streamlit for UI
  - No extrenal API’s
  - OPENAI

- **Documents:**

```
# SmartSDLC - AI-enhanced Software Development Lifecycle

SmartSDLC is a simplified AI-powered assistant to support key stages of the Software Development Life Cycle (SDLC). This tool is designed for students and developers who want quick assistance with requirement gathering, test cases, design suggestions, user stories, and even code generation — all without needing an API key.

---

## 💡 Features

1. **Requirement Gathering Assistant**
   Generate use cases from a software idea description.

2. **Test Case Generator**
   Automatically suggest standard test cases for a given feature.

3. **Architecture Design Suggestions**
   Basic system architecture and design proposals for your project.

```

```
•  4. **User Story Generator**
•     Auto-generate user stories for modules and features.
•
•  5. **Sample Code Snippets**
•     Provides example login logic using Python Flask.
•
•  ---
•
•  ## 🛠️ Tech Stack
•
•  - Python
•  - Streamlit
•
•  ---
•
•  ## 🚀 How to Run the Project
•
•  ### 🔧 Step 1: Clone the Repository
•  ```bash
•  git clone https://github.com/your-username/SmartSDLC.git
•  cd SmartSDLC
•
```

- **CODE:**

```
•  import streamlit as st
•
•  st.set_page_config(page_title="SmartSDLC - AI-enhanced SDLC",
   layout="wide")
•
•  st.title("SmartSDLC - AI-enhanced Software Development Lifecycle")
•
```

```
# Feature 1: Requirement Gathering
st.subheader("1. Requirement Gathering Assistant")
req_input = st.text_area("Describe your software idea")

if st.button("Generate Use Cases"):
    if req_input.strip():
        st.success("Sample Use Cases:")
        st.markdown("""
        - User Login and Authentication
        - Add to Cart and Checkout
        - Search Products or Services
        - View Order History
        - Admin Dashboard for Management
        """)
    else:
        st.warning("Please enter your software idea.")

# Feature 2: Test Case Generator
st.subheader("2. Test Case Generator")
test_input = st.text_area("Describe a feature to generate test cases")

if st.button("Generate Test Cases"):
    if test_input.strip():
        st.success("Sample Test Cases:")
        st.markdown("""
        - Verify user can register with valid data
        - Check validation on empty form
        - Ensure cart updates after adding item
        - Validate response when payment fails
        """)
    else:
        st.warning("Please describe a feature.")

# Feature 3: Design Suggestion
st.subheader("3. Design Suggestion")
if st.button("Generate Architecture Design"):
    st.success("Suggested Design:")
    st.markdown("""
    **Frontend:** React or HTML/CSS
    **Backend:** Node.js / Python Flask
    **Database:** MongoDB or MySQL
    **Architecture:**
    - Client sends requests to backend
    - Backend processes and interacts with DB
    - Response sent to frontend
    - Admin Dashboard available for analytics
    """)

# Feature 4: User Story Generator
st.subheader("4. User Story Generator")
story_input = st.text_area("Enter module/feature name")

if st.button("Generate User Stories"):
    if story_input.strip():
        st.success("Sample User Stories:")
        st.markdown(f"""
        - As a user, I want to use the **{story_input}** so that I can complete my task faster
        - As an admin, I want to manage **{story_input}** efficiently
        - As a tester, I want to validate **{story_input}** under edge cases
        """)
    else:
```

```
•           st.warning("Please enter a feature/module.")
•
•   # Feature 5: Sample Code Snippet
•   st.subheader("5. Sample Code Generator")
•   if st.button("Generate Login Code (Python Flask)"):
•       st.success("Sample Code:")
•       st.code("""
•   from flask import Flask, request
•
•   app = Flask(__name__)
•
•   @app.route('/login', methods=['POST'])
•   def login():
•       username = request.form['username']
•       password = request.form['password']
•       if username == 'admin' and password == '123':
•           return "Login Successful"
•       else:
•           return "Login Failed"
•
""", language='python')
```

- **INSTALLATION:**

 - Clone repo, install requirements, run Streamlit app.

```bash
git clone https://github.com/your-username/SmartSDLC.git
cd SmartSDLC
pip install -r requirements.txt
streamlit run app.py
```

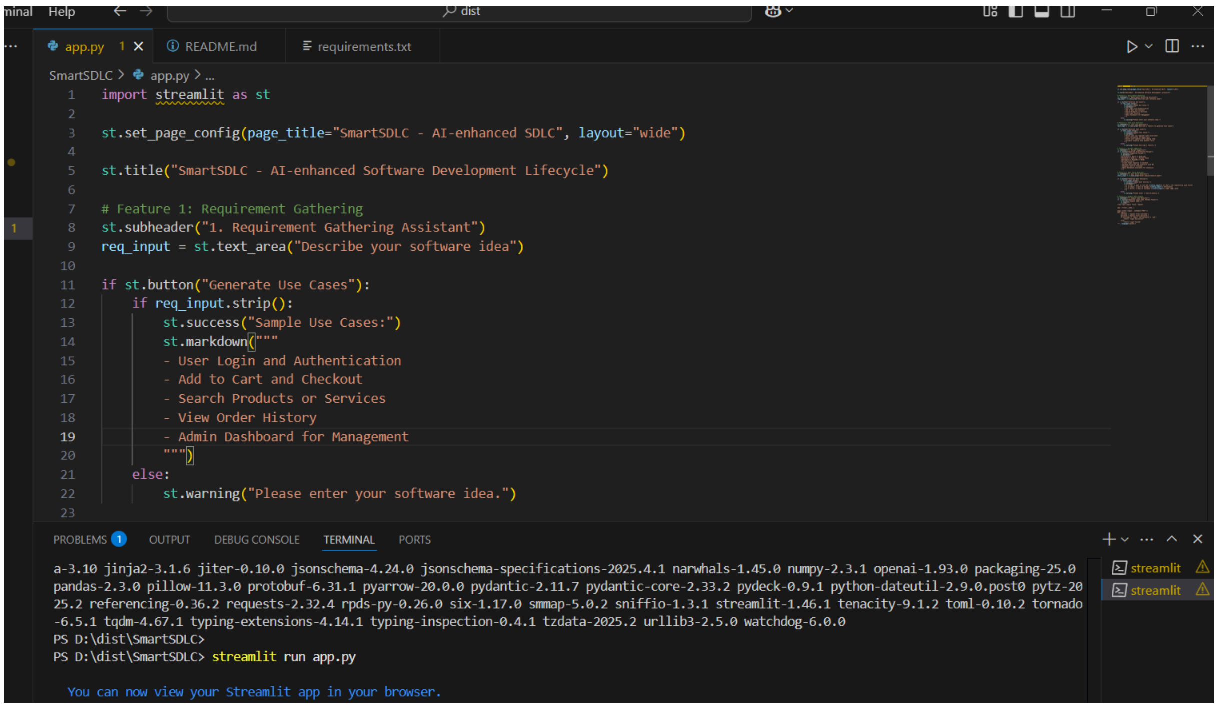
- **Testing**

  ✓ Manual testing across all modules

  ✓ Model tested with varied prompts and edge cases

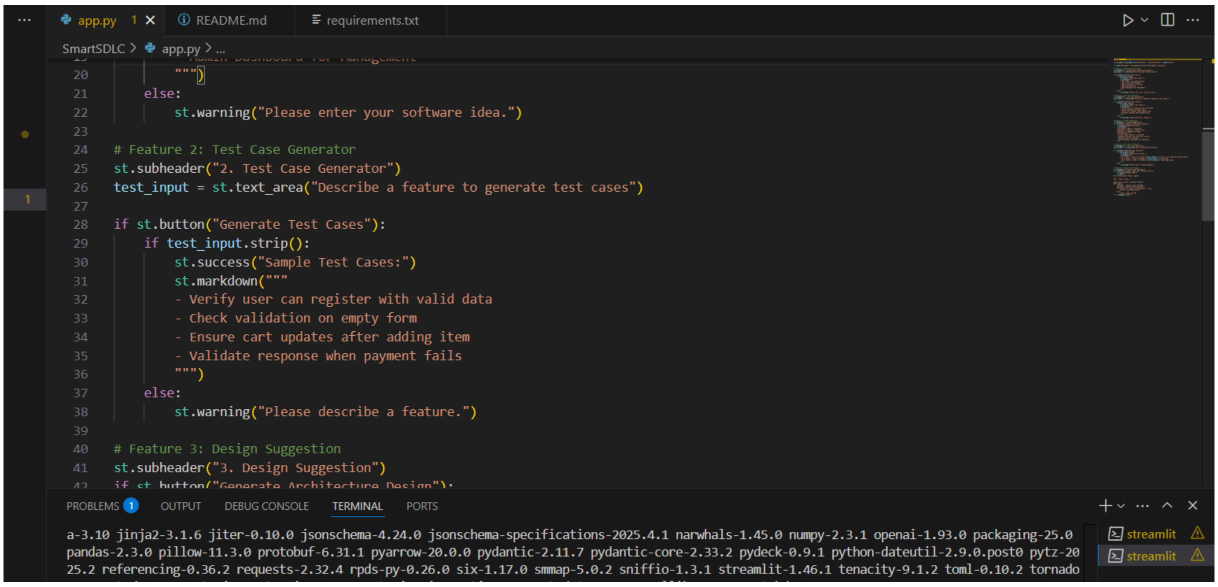
  ✓ Handled errors for invalid inputs and model timeouts

- **INPUTS (CODES) :**



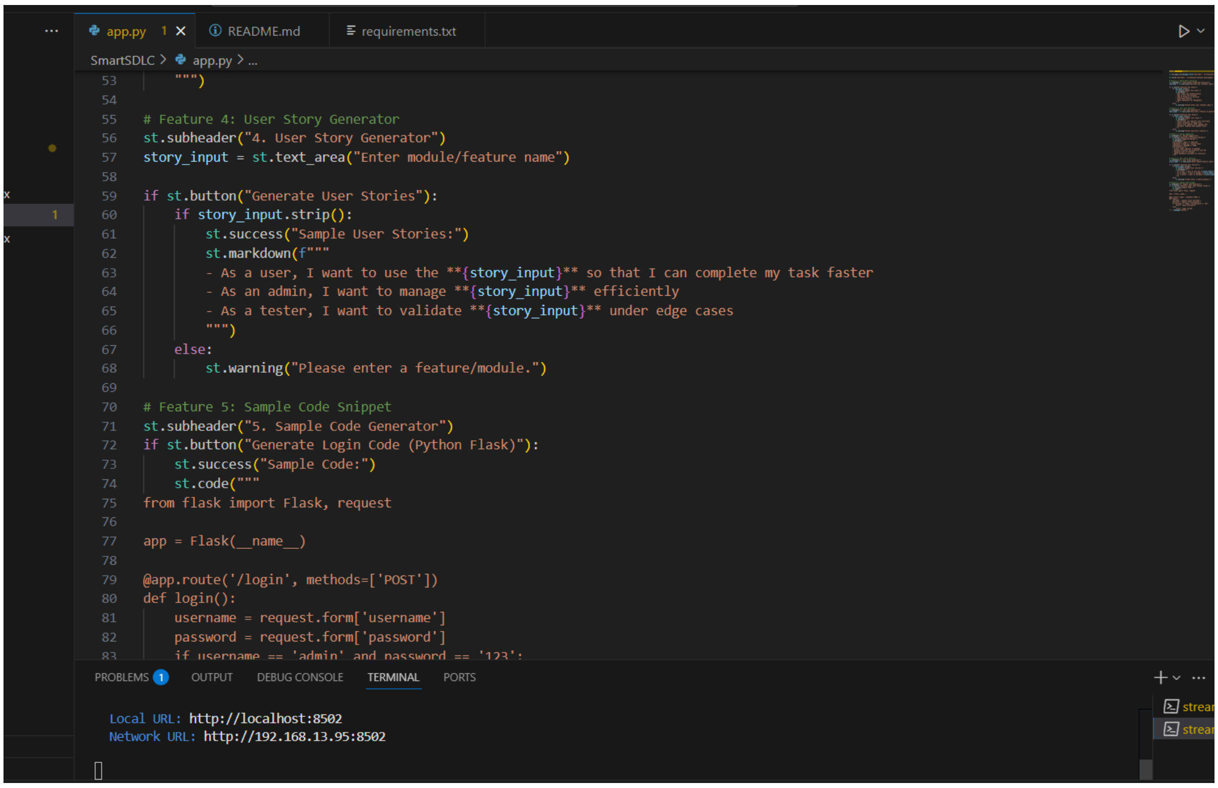
```python
import streamlit as st

st.set_page_config(page_title="SmartSDLC - AI-enhanced SDLC", layout="wide")

st.title("SmartSDLC - AI-enhanced Software Development Lifecycle")

# Feature 1: Requirement Gathering
st.subheader("1. Requirement Gathering Assistant")
req_input = st.text_area("Describe your software idea")

if st.button("Generate Use Cases"):
    if req_input.strip():
        st.success("Sample Use Cases:")
        st.markdown("""
        - User Login and Authentication
        - Add to Cart and Checkout
        - Search Products or Services
        - View Order History
        - Admin Dashboard for Management
        """)
    else:
        st.warning("Please enter your software idea.")
```

```
a-3.10 jinja2-3.1.6 jiter-0.10.0 jsonschema-4.24.0 jsonschema-specifications-2025.4.1 narwhals-1.45.0 numpy-2.3.1 openai-1.93.0 packaging-25.0
pandas-2.3.0 pillow-11.3.0 protobuf-6.31.1 pyarrow-20.0.0 pydantic-2.11.7 pydantic-core-2.33.2 pydeck-0.9.1 python-dateutil-2.9.0.post0 pytz-20
25.2 referencing-0.36.2 requests-2.32.4 rpds-py-0.26.0 six-1.17.0 smmap-5.0.2 sniffio-1.3.1 streamlit-1.46.1 tenacity-9.1.2 toml-0.10.2 tornado
-6.5.1 tqdm-4.67.1 typing-extensions-4.14.1 typing-inspection-0.4.1 tzdata-2025.2 urllib3-2.5.0 watchdog-6.0.0
PS D:\dist\SmartSDLC>
PS D:\dist\SmartSDLC> streamlit run app.py

  You can now view your Streamlit app in your browser.
```



```
            """)
    else:
        st.warning("Please enter your software idea.")

# Feature 2: Test Case Generator
st.subheader("2. Test Case Generator")
test_input = st.text_area("Describe a feature to generate test cases")

if st.button("Generate Test Cases"):
    if test_input.strip():
        st.success("Sample Test Cases:")
        st.markdown("""
        - Verify user can register with valid data
        - Check validation on empty form
        - Ensure cart updates after adding item
        - Validate response when payment fails
        """)
    else:
        st.warning("Please describe a feature.")

# Feature 3: Design Suggestion
st.subheader("3. Design Suggestion")
```

```
a-3.10 jinja2-3.1.6 jiter-0.10.0 jsonschema-4.24.0 jsonschema-specifications-2025.4.1 narwhals-1.45.0 numpy-2.3.1 openai-1.93.0 packaging-25.0
pandas-2.3.0 pillow-11.3.0 protobuf-6.31.1 pyarrow-20.0.0 pydantic-2.11.7 pydantic-core-2.33.2 pydeck-0.9.1 python-dateutil-2.9.0.post0 pytz-20
25.2 referencing-0.36.2 requests-2.32.4 rpds-py-0.26.0 six-1.17.0 smmap-5.0.2 sniffio-1.3.1 streamlit-1.46.1 tenacity-9.1.2 toml-0.10.2 tornado
```

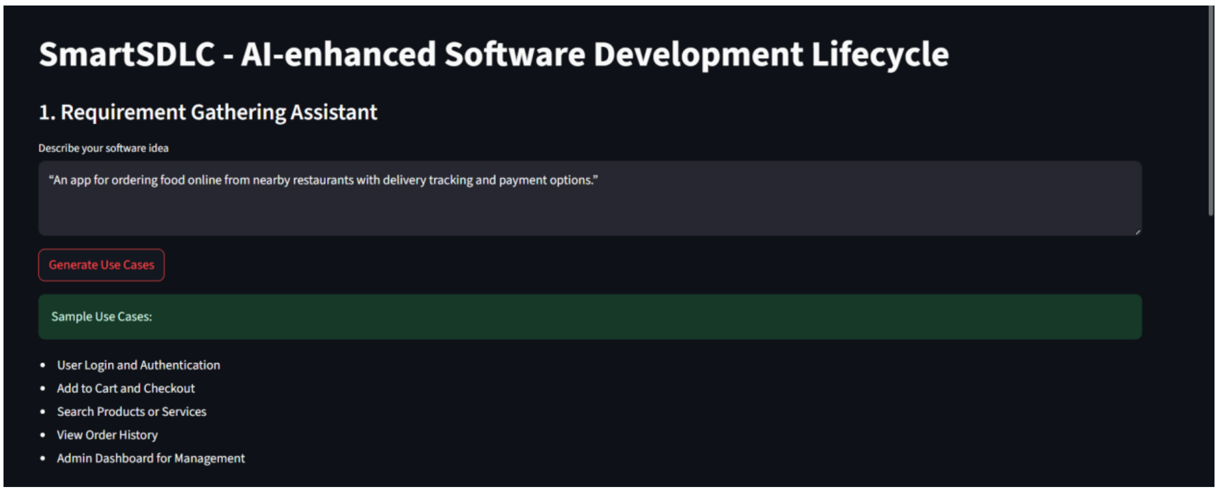


```
    """)

# Feature 4: User Story Generator
st.subheader("4. User Story Generator")
story_input = st.text_area("Enter module/feature name")

if st.button("Generate User Stories"):
    if story_input.strip():
        st.success("Sample User Stories:")
        st.markdown(f"""
        - As a user, I want to use the **{story_input}** so that I can complete my task faster
        - As an admin, I want to manage **{story_input}** efficiently
        - As a tester, I want to validate **{story_input}** under edge cases
        """)
    else:
        st.warning("Please enter a feature/module.")

# Feature 5: Sample Code Snippet
st.subheader("5. Sample Code Generator")
if st.button("Generate Login Code (Python Flask)"):
    st.success("Sample Code:")
    st.code("""
from flask import Flask, request

app = Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
```

```
  Local URL: http://localhost:8502
  Network URL: http://192.168.13.95:8502
```

SMARTBRIDGE
Let's Bridge the Gap

- **OUTPUT :**


SmartSDLC - AI-enhanced Software Development Lifecycle

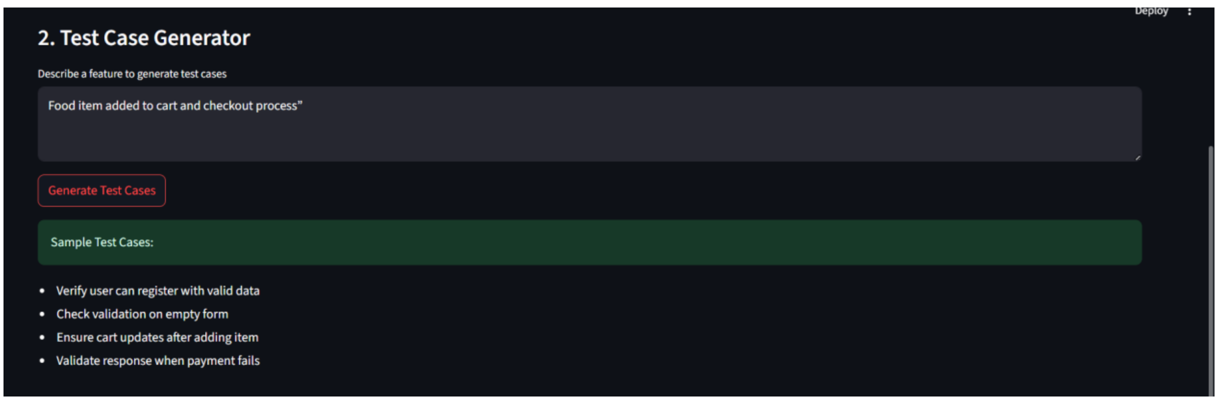
1. Requirement Gathering Assistant

Describe your software idea

"An app for ordering food online from nearby restaurants with delivery tracking and payment options."

Sample Use Cases:

- User Login and Authentication
- Add to Cart and Checkout
- Search Products or Services
- View Order History
- Admin Dashboard for Management



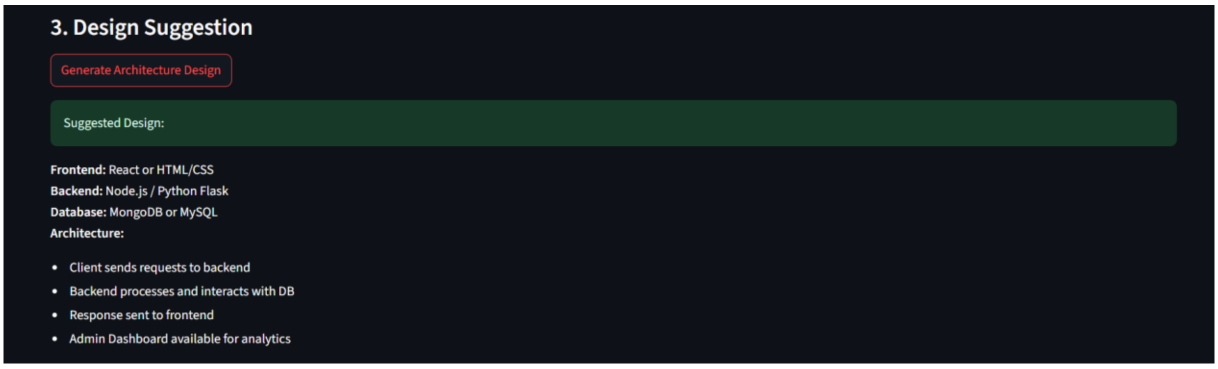
2. Test Case Generator

Describe a feature to generate test cases

Food item added to cart and checkout process"

Sample Test Cases:

- Verify user can register with valid data
- Check validation on empty form
- Ensure cart updates after adding item
- Validate response when payment fails

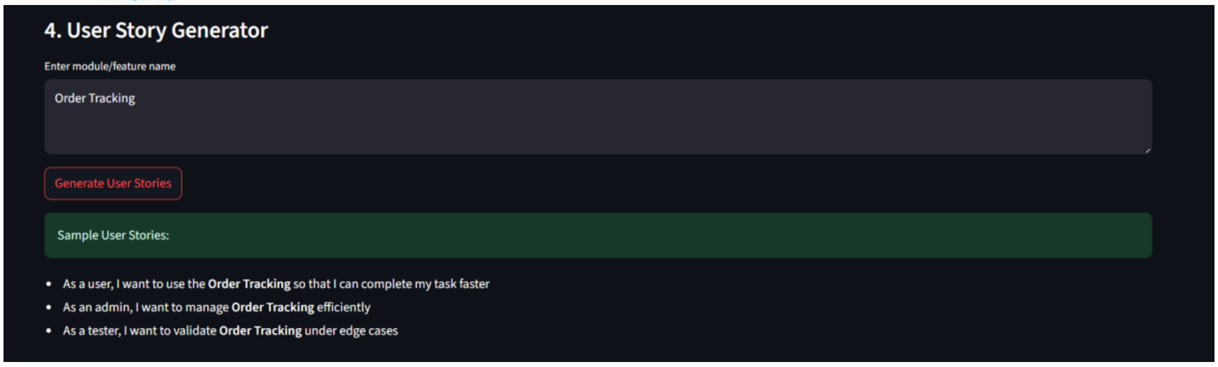

3. Design Suggestion

Suggested Design:

Frontend: React or HTML/CSS
Backend: Node.js / Python Flask
Database: MongoDB or MySQL
Architecture:

- Client sends requests to backend
- Backend processes and interacts with DB
- Response sent to frontend
- Admin Dashboard available for analytics

SMARTBRIDGE
Let's Bridge the Gap

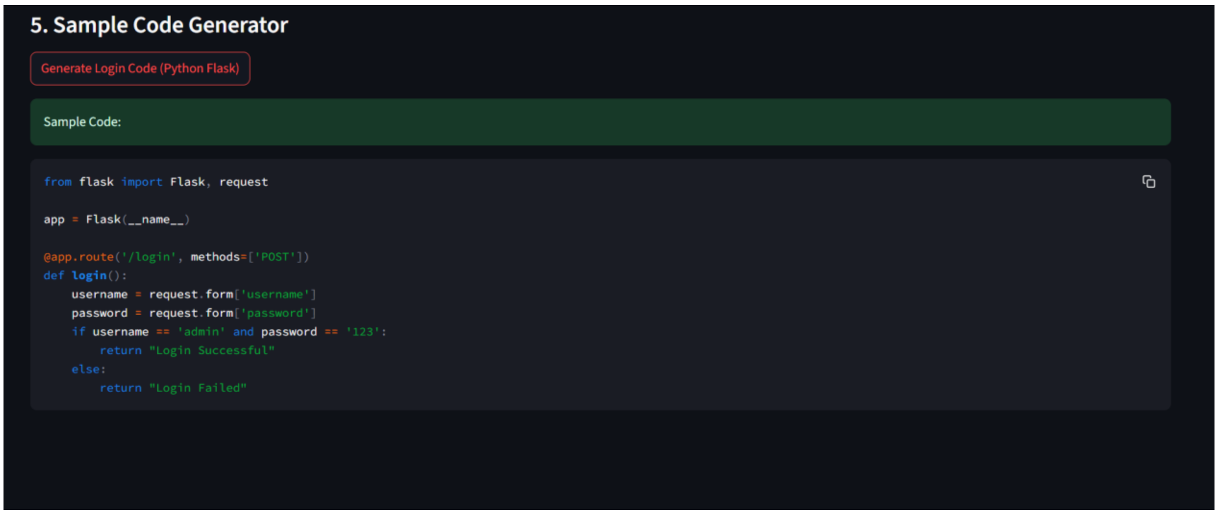

4. User Story Generator

Enter module/feature name

Order Tracking

Sample User Stories:

- As a user, I want to use the **Order Tracking** so that I can complete my task faster
- As an admin, I want to manage **Order Tracking** efficiently
- As a tester, I want to validate **Order Tracking** under edge cases


5. Sample Code Generator

Sample Code:

```python
from flask import Flask, request

app = Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    if username == 'admin' and password == '123':
        return "Login Successful"
    else:
        return "Login Failed"
```

5.Future Enhancements

- ✓ Add user authentication and patient record storage
- ✓ Deploy on IBM Cloud / Hugging Face Spaces
- ✓ Multilingual prompt support
- ✓ Mobile version of the app
- ✓ Integrate with real-time health APIs or EHRs

7. Conclusion

SmartSDLC provides a simple simulation of AI support for software engineering projects, making SDLC tasks faster and more efficient