## SmartSDLC – AI-Powered SDLC Automation Platform

### ****1. Introduction :****

* **Project Title**: SmartSDLC – AI-Powered Software Development Lifecycle Automation
* **Team Members**:
* **Vanaja Pulapa** – Team Leader, Full Stack & AI Integration
* **Lakshmidurga Sathi** – Frontend Developer (Streamlit)
* **Sonti Naga Tulasi** – Backend Developer (FastAPI, Model Integration)
* **Yalla Manasa Siri** – PDF Processing & Testing

### ****2. Project Overview :****

**Purpose**:  
SmartSDLC is a Generative AI-based platform that automates various stages of the Software Development Lifecycle (SDLC) such as requirement classification, code generation, test case generation, bug fixing, and code summarization, using IBM Watsonx.

**Features**:

* PDF upload and SDLC phase classification
* AI code generation in multiple languages
* Test case generator for given code
* Bug fixing using AI
* Code summarization
* AI Chat Assistant for developer queries
* User authentication and secure access

### ****3. Architecture :****

* **Frontend**:
* Built using **Streamlit** for interactive UI
* Separate pages for each SDLC stage
* CSS for custom styling and responsive layout
* **Backend**:
* Developed with **FastAPI** (Python)
* Routes for handling SDLC features like /generate-code, /classify-pdf, etc.
* Uses uvicorn server for API hosting
* **Database**:
* Uses **SQLite** for storing user data, login credentials, and task history
* File-based persistent storage using custom history management

### ****4. Setup Instructions :****

* **Prerequisites**:
* Python 3.10+
* pip
* Streamlit
* FastAPI
* Uvicorn
* SQLite3
* **Installation** :
* # Clone the repository :

git clone <https://github.com/pulapa-vanaja/SmartSDLC-AI-Enhanced-Software-Development-Lifecycle> smartsdlc

* # Set up backendcd app :

pip install -r requirements.txt

* # Set up frontendcd ../smart\_sdlc\_frontend

pip install -r requirements.txt

* **Environment Variables**:  
   Create .env with your Watsonx credentials.

### ****5. Folder Structure :****

**Frontend (Streamlit)**:

* **smart\_sdlc\_frontend/**
* **main.py**
* **pages/**
* **Upload\_and\_Classify.py**
* **Code\_Generator.py**
* **...**
* **utils/**
* **history.py**
* **auth\_pages/**
* **login.py**
* **signup.py**

**Backend (FastAPI)**:

* app/
* main.py
* routes/
* code.py
* pdf.py
* services/
* models/
* utils/

### ****6. Running the Application :****

**Start Backend Server**:

cd app

uvicorn main:app --reload

**Start Frontend (Streamlit)**:

cd smart\_sdlc\_frontend

streamlit run main.py

### ****API Documentation :****

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **Method** | **Description** |
| /generate-code | POST | Generate code for given task |
| /generate-test-cases | POST | Generate test cases |
| /fix-bugs | POST | Fix bugs in code |
| /summarize-code | POST | Provide a summary of code |
| /classify-pdf | POST | Upload and classify PDF into SDLC phases |

### ****8. Authentication :****

* **Method**: Custom username-password-based login system
* **Backend**: SQLite DB stores user credentials
* **Frontend**: Login, Sign-up, and Forgot Password pages using Streamlit
* **Security**: Passwords hashed; session state managed via Streamlit

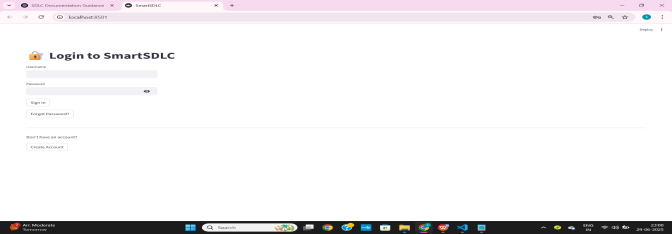
### ****9. User Interface :****

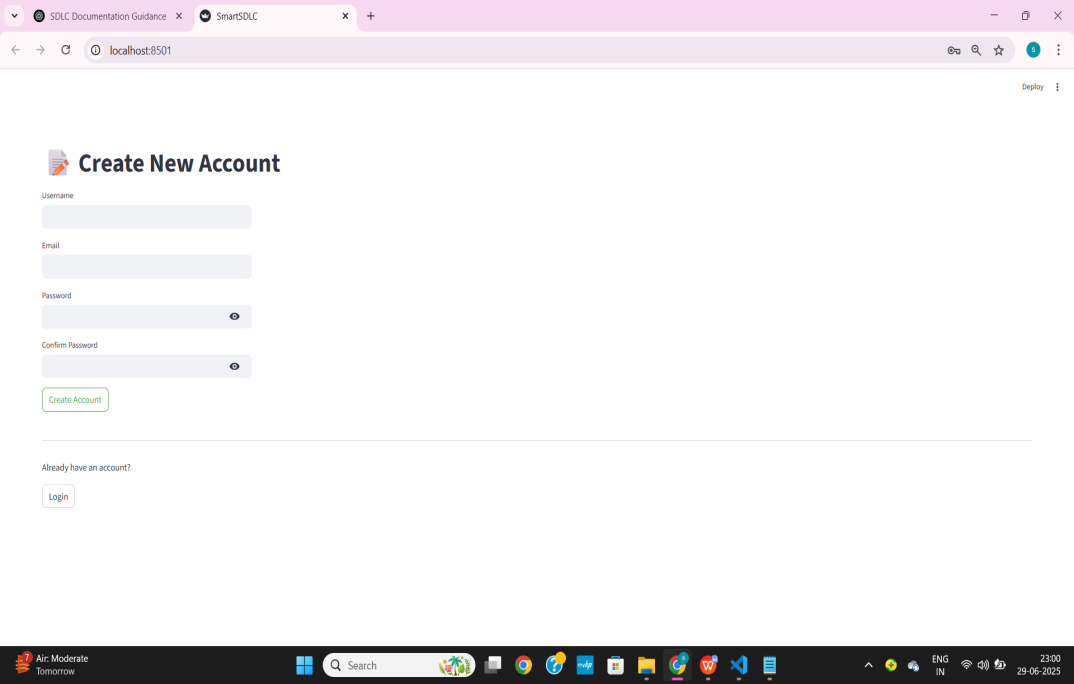
* Sidebar-based navigation
* Role-based features visible after login
* Light theme with custom CSS
* PDF upload, file preview, and code blocks with copy/download buttons

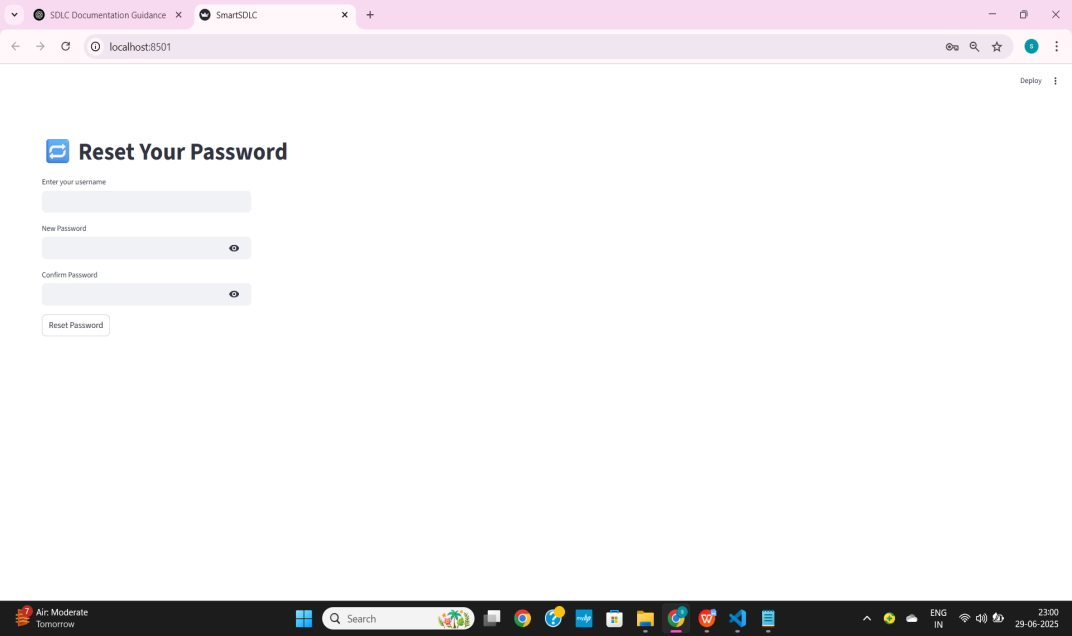
### ****10. Testing :****

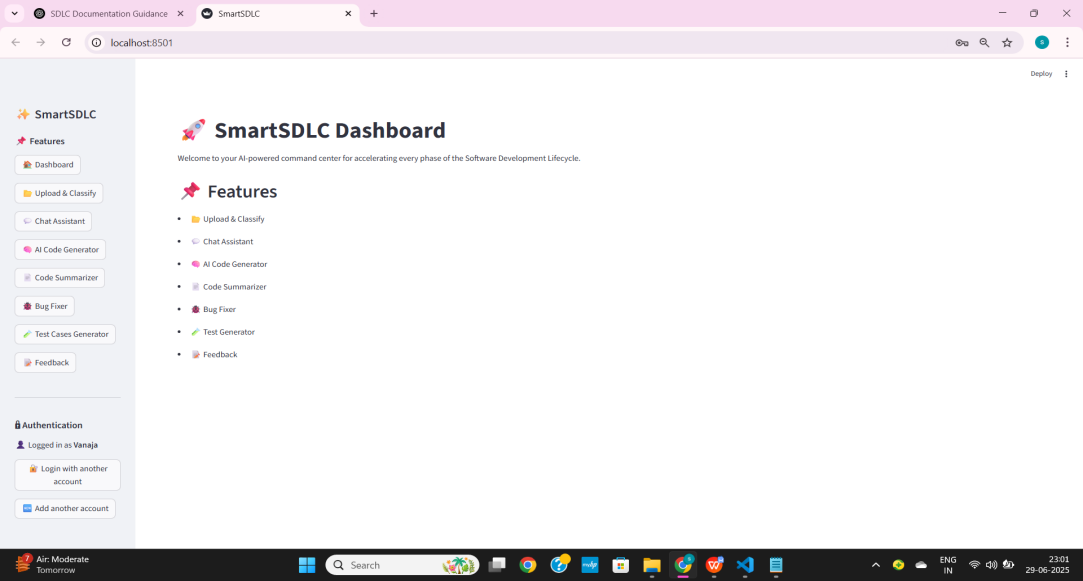
* **Functional Testing**: All features tested with valid and invalid inputs
* **Performance Testing**: Tested under load (5 concurrent users)
* **Tools**: Manual + timers for response metrics
* **UAT**: Verified by team members with sign-off

### ****Screenshots or Demo :****









### ****12. Known Issues :****

* PDF classification may not work with scanned images (OCR pending)
* Minor alignment issues on very small screens
* Requires stable internet for Watsonx API

### ****13. Future Enhancements****

* Add GitHub integration to auto-fetch code
* Implement OCR for image-based PDFs
* Add support for project deployment to cloud (AWS/GCP)
* Improve role-based access control and team collaboration