Full-Stack view of Smart SDLC

1. Introduction

- **Project Title:** SmartSDLC – AI-Enhanced Software Development Lifecycle

  LTVIP2025TMID59401
- **Team Members:**

  1. Rajolu Gowtham

  2. Pedalanka Jaswanth

  3. Perabathula Sujitha Naga Sesha Lakshmi

  4. Orsu Naga Yashaswini

---

2. Project Overview

- **Purpose:**
  This project provides an AI-powered chatbot that integrates IBM Watsonx with FastAPI and Streamlit to allow natural language interactions, text generation, and question answering in a simple, secure, and scalable way.
- **Features:**
  - Streamlit-based user interface
  - FastAPI backend for routing and token management
  - Integration with IBM Watson/Watsonx APIs
  - .env-based secure token handling
  - Token caching for improved performance
  - Error handling and JSON response parsing
  - Easily customizable prompt input

---

3. Architecture

- **Frontend (Streamlit):**
  - Input box for prompts
  - Output display of generated responses
  - Uses query params for prompt passing
  - Lightweight and runs in-browser with no extra setup

Full-Stack view of Smart SDLC

- **Backend (FastAPI):**
  - Exposes a /chatbot POST endpoint
  - Handles requests to IBM Watson APIs
  - Includes token caching using Python dictionaries/memory
  - Structured for easy route addition
- **Database:**
  - Not applicable (currently stateless).
  - Optional: MongoDB or Redis can be added for token storage/prompt history.

---

4. Setup Instructions

- **Prerequisites:**
  - Python 3.9+
  - IBM Cloud account with Watsonx API access
  - pip for dependency management
- **Installation:**

bash

CopyEdit

```
1. Clone the repository
git clone https://github.com/gowtham-rajolu/ibm.git
cd ibm


 2. Create a virtual environment and activate
python -m venv venv
source venv/bin/activate  # or venv\Scripts\activate (Windows)


 3. Install dependencies
pip install -r requirements.txt


 4. Set environment variables in .env
```

Full-Stack view of Smart SDLC

touch .env

Add your IBM API_KEY and other required values

5. Run the backend

uvicorn api.main:app --reload

6. Run the frontend

streamlit run frontend/mainpg.py

---

5. Folder Structure

bash

CopyEdit

```
ibm/
 ├── api/              # FastAPI backend
 |   └── main.py
 ├── frontend/         # Streamlit app
 |   └── mainpg.py
 ├── .env              # API keys and sensitive config
 ├── requirements.txt    # Dependencies
 ├── .gitignore         # Excludes __pycache__, .env etc.
```

- **Client (Streamlit):**
  Contains the UI logic, prompt capture, and output rendering.
- **Server (FastAPI):**
  Handles POST requests and communication with IBM APIs.

---

6. Running the Application

- **Frontend:**

bash

CopyEdit

Full-Stack view of Smart SDLC

cd frontend

streamlit run mainpg.py

- **Backend:**

bash

CopyEdit

cd api

uvicorn main:app --reload

---

7. API Documentation

- **POST /chatbot**
    - **Request Body:**

json

CopyEdit

```
{
  "prompt": "Explain quantum computing"
}
```

-
    - **Response:**

json

CopyEdit

```
{
  "response": "Quantum computing uses..."
}
```

- **Authentication:** IBM IAM token (handled internally)

---

8. Authentication

- **Method:**
  IAM Token from IBM Cloud generated via POST call
- **Stored:**
  In-memory (or can be stored in DB for persistence)

Full-Stack view of Smart SDLC

- **.env Example:**

env

CopyEdit

API_KEY=your_ibm_api_key

- **Token Caching:**
  Tokens are reused until expired to save authentication calls

---

9. User Interface

- Streamlit-based UI
- Input box for user prompt
- Output area to display AI-generated response
- Responsive and browser-friendly

---

10. Testing

- **Manual Testing:**
  - Tested with multiple prompt types (FAQs, code generation, summaries)
  - Verified error handling for invalid tokens and empty input
- **Future:**
  - Add unit tests using pytest or unittest
  - API testing with Postman or Swagger

---

11. Screenshots / Demo

(Add real screenshot or GIF)

---

12. Known Issues

- Token cache resets if the server restarts
- Only one user supported at a time (stateless)
- Frontend doesn’t yet support file uploads or history

Full-Stack view of Smart SDLC

13. Future Enhancements

- Add MongoDB for chat history and token persistence
- Enable multi-user sessions
- Enhance UI with chat-style interaction
- Add model selection or prompt templates
- Deploy using Docker or CI/CD pipeline (GitHub Actions)