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

**project documentation**

**1.Introduction:**

* **Project tittle**:smartSDLC-AI-Enhanced Software Development Lifecycle
* **Team Member**: M.Hemalatha
* **Team Member**: E.Maheshyazhini
* **Team Member**: A.Kaviyalakshmi
* **Team Member**: M.Roshini

**2.Project overview:**

**Goal:** Provide accessible, AI-powered educational assistance for students, teachers, and self-learners.

**Core Features:**

* Detailed concept explanation
* Automatic quiz generation

**Tech Stack:**

* **Model**: ibm-granite/granite-3.2-2b-instruct
* **Backend:** Python (PyTorch, Transformers)
* **Frontend:** Gradio

**3.Architecture:**

**+-----------------------+**

**| User Interface |**

**| (Gradio Frontend) |**

**+----------+------------+**

**|**

**v**

**+----------+------------+**

**| Application Logic |**

**| - Input validation |**

**| - Prompt generation |**

**| - Response parsing |**

**+----------+------------+**

**|**

**v**

**+----------+------------+**

**| Language Model API |**

**| (IBM Granite 3.2 2B) |**

**| via Hugging Face Hub |**

**+-----------------------+**

**4.Setup instruction:**

**Prerequisites :**

Make sure you have the following installed:

* Python 3.8+
* pip

**📥 Install dependencies**

pip install torch transformers gradio

If using a GPU, make sure the appropriate CUDA version of PyTorch is installed.

**5. Folder structure:**

educational-ai-assistant/

│

├── app.py # Main application script

├── requirements.txt # Python dependencies

├── README.md # Project overview

├── assets/ # (Optional) UI assets like images or icons

└── docs/ # (Optional) Extended documentation or screenshots

**6.Running the application:**

**To launch the Gradio app locally:**

python app.py

Gradio will display a **local URL** and optionally a **public shareable link** (if share=True is used).

**7.API Documentation:**

Even though Gradio is UI-focused, here’s how the backend works:

#### ➤ generate\_response(prompt: str, max\_length: int) -> str

Generates a response using the language model based on a prompt.

#### ➤ concept\_explanation(concept: str) -> str

Explains a concept with examples using a predefined prompt template.

#### ➤ Quiz\_generator(concept: str) -> str

Generates 5 quiz questions in multiple formats and provides answers.

**8.Authentication:**

Currently, the app **does not require authentication**. However, you can add:

* **Gradio login/auth hooks** for admin use.
* **API key checks** before model access.
* **Password-protected sharing links** (via app.launch(auth=...)).

**Example (basic auth):**

app.launch(auth=("username", "password"))

**9.User Interface:**

The Gradio UI has two main tabs:

c**oncept Explanation**

* Textbox: Input a concept like "machine learning"
* Button: "Explain"
* Output: Detailed explanation with examples

#### ****Quiz Generator****

* Textbox: Input a topic like "photosynthesis"
* Button: "Generate Quiz"
* Output: 5 questions (MCQ, True/False, Short Answer) + answers

**10.Testing:**

#### Manual Testing :

 Run the app locally.

 Use different concepts and topics.

 Confirm:

* Output is coherent and accurate.
* Errors (like empty input) are handled.