# Assignment: Individualised Chatbot with RAG Personalization

## Objective

Build a personalized chatbot that can dynamically adapt its responses based on a set of individual parameters (such as tone, goal, personality traits, preferred style of answers, etc.) and optionally augment its answers using user-uploaded documents via a Retrieval-Augmented Generation (RAG) mechanism. This assignment is aimed at evaluating the effectiveness of combining LLMs with a RAG pipeline tailored for personalization use cases.

## Assignment Description

In this task, you are required to:

### 1. Develop an Individualized Chatbot

Build a chatbot interface that allows users to define personal settings such as:

- Tone of conversation (e.g., formal, friendly, humorous)
- Communication goal (e.g., educate, summarize, advise, entertain)
- Preferred length of responses (e.g., short, detailed)
- Response style (e.g., storytelling, bullet-points, step-by-step)
- Language preference (if multilingual)
- User persona traits (e.g., a beginner learner, domain expert, etc.)

These preferences should influence how the chatbot generates its responses using the LLM.

### 2. Implement a RAG (Retrieval-Augmented Generation) Module

Integrate a document upload option where users can provide reference material (PDFs, text files, etc.) for the chatbot to use as background knowledge. This should include:

- A way to upload and parse multiple documents.
- Extract and chunk relevant content from uploaded documents.
- Embed and store these chunks in a retrievable format (e.g., using FAISS, Chroma, or similar vector DB). - On receiving a user query, retrieve relevant context from the uploaded documents and integrate it into the

LLM prompt.

## Assignment: Individualised Chatbot with RAG Personalization

### 3. Ensure Personalization + Contextual Accuracy

Combine user-defined parameters and document-based context to generate high-quality, customized responses.

### 4. Optimize and Test the Chatbot

Ensure the chatbot can handle a variety of scenarios:

- Personality-specific queries (e.g., "Explain this in a fun way for a 10-year-old")
- Fact-based queries enhanced by uploaded documents
- Context-switching with different personas
- Switching tones dynamically (e.g., based on mood: serious, motivational, sarcastic)

### Deliverables

1. Code Repository (GitHub or zip file):

- End-to-end working code for the chatbot with personalization and RAG setup.
- Include clear folder structure and comments for easy understanding.

2. Example Testcases:

- At least one sample user configuration (persona settings).
- One or two documents uploaded as RAG sources.
- Sample query and chatbot response.
- (Optional but preferred) A screenshot or short screen recording of the working chatbot in action.

3. Documentation (Markdown or PDF):

- A short write-up (1-2 pages) explaining:
  - How the chatbot handles personalization.
  - How uploaded documents are used for retrieval.
  - Steps to set up and run the chatbot locally.

## Evaluation Criteria

## Assignment: Individualised Chatbot with RAG Personalization

| Criteria | Description |
|---|---|
| Personalization Depth | How well does the chatbot adapt to user-defined tone, style, and goals? |
| RAG Integration | Quality of context integration from uploaded documents into LLM answers | |
| Code Quality | Modularity, clarity, efficiency, and reusability of the code |
| Flexibility & UX | Ease of switching between personas or uploading documents | |
| Documentation & Demos | Completeness of documentation and clarity of demo examples |

## Expected Skills

- Proficiency in Python and working with LLMs (e.g., GPT, LLaMA, Mistral, etc.)
- Familiarity with RAG architecture and document embedding techniques
- Experience with vector databases (FAISS, Chroma, Weaviate, etc.)
- Ability to design user-driven personalization systems
- Experience with front-end libraries (e.g., Streamlit, Gradio) is a plus