PDF -langchain & Groq

Purpose:

The script integrates natural language processing capabilities with the GROQ API to create an interactive application for analyzing PDF documents and providing AI-driven responses in real-time.

Key Components:

1. **Imports and Setup**:
    - Imports necessary libraries from LangChain and Streamlit.
    - Sets the GROQ API key for authentication.
2. **Initialization Functions**:
    - `initialize_groq_client()`: Initializes the GROQ client using the API key.
    - `create_directories()`: Creates directories for storing uploaded PDF files and persistent data.
    - `initialize_session_state()`: Sets up session variables for managing chat history, memory, and vector stores using Streamlit's session state.
3. **Main Application (`main()` function)**:
    - Sets up the Streamlit user interface with a title and file uploader for PDF documents.
    - Displays chat history if any messages are stored.
    - Processes uploaded PDF files:
        - Reads and analyzes the PDF using `PyPDFLoader`.
        - Splits the document into smaller chunks using `RecursiveCharacterTextSplitter`.
        - Stores embeddings of each chunk in a `Chroma` vector store for efficient retrieval.
4. **User Interaction**:

    - Provides a chat interface for users to input queries.
    - Retrieves relevant context from stored document embeddings based on user queries.
    - Sends user queries to the GROQ API (`llama3-8b-8192`) for generating AI-driven responses.
    - Displays assistant responses in real-time within the chat interface, with typing animation for user experience.

Additional Features:

- **Error Handling**: Manages file uploads and ensures proper directory creation.
- **Contextual Responses**: Utilizes stored document embeddings (`Chroma`) to enhance the relevance of AI responses based on user queries.

Use Case:

The application is designed to assist users in analyzing and interacting with PDF documents, leveraging advanced natural language processing techniques and AI models via the GROQ API.