Project Report

1. Problem Selection:

➢ **Option 3:** Create a Reasoning-Based LLM System of your choice

- Goal: Propose and develop a unique project that showcases reasoning and system-2 thinking in LLMs.
- Define the problem and identify how reasoning is critical to the solution.
- Design a system architecture that integrates LLMs with external tools if needed (e.g., databases, solvers).

2. Problem Definition:

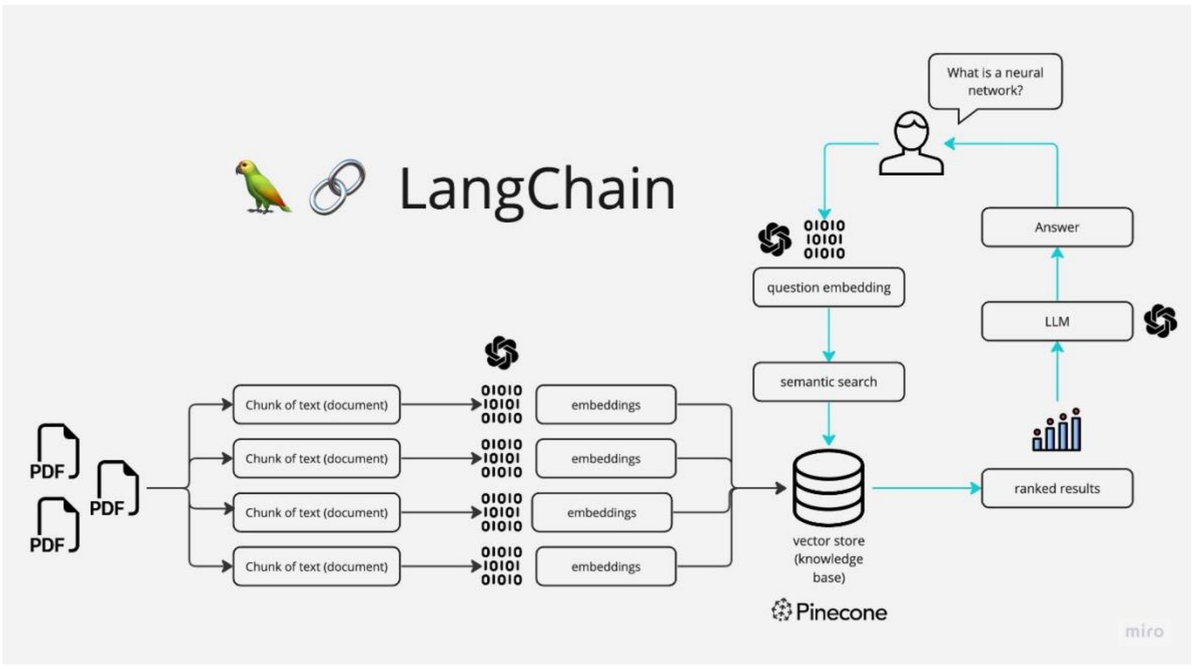
This project i.e. **MultiPDF Chat App** addresses the challenge of enabling users to interact with and retrieve insights from multiple PDF documents. Often, important information is buried within large PDF files, making it tedious to search and extract relevant content manually. The goal is to create a streamlined application that processes PDFs, facilitates conversational interactions with their content, and provides accurate, context-aware responses.

3. Methodology and System Design:

➢ **Methodology with system architecture:**

The application follows these steps to provide responses to your questions:

- PDF Loading: The app reads multiple PDF documents and extracts their text content.
- Text Chunking: The extracted text is divided into smaller chunks that can be processed effectively.
- Language Model: The application utilizes a language model to generate vector representations (embeddings) of the text chunks.
- Similarity Matching: When you ask a question, the app compares it with the text chunks and identifies the most semantically similar ones.
- Response Generation: The selected chunks are passed to the language model, which generates a response based on the relevant content of the PDFs.



➢ **System Design:**

- **Frontend:** Implemented using Streamlit for an interactive user interface that supports file uploads and chat-like conversations.
- **Backend:**

a) **Text Extraction:** Utilized PyPDF2 for extracting text from PDFs.

b) **Text Chunking:** Applied LangChain's CharacterTextSplitter to split text into chunks of 1000 characters with a 200-character overlap.

c) **Embeddings and Storage:** Used Hugging Face's hkunlp/instructor-xl embeddings and FAISS for efficient vector-based search and retrieval.

d) **Conversational Model:** Integrated Hugging Face’s google/flan-t5-base model via the Transformers library for generating conversational responses.

e) **State Management:** Employed Streamlit’s session state to maintain chat history and conversation context.

➢ Reasoning-Based System Analysis:

The system exhibits characteristics of a **reasoning-based LLM system** by employing a conversational AI model to process user queries in context. Below are the highlights:

- **Contextual Understanding:** The LLM uses embeddings and retrieval mechanisms to extract relevant chunks of information from the PDFs. This involves reasoning to match user queries with the most relevant document content.
- **Conversational Interaction:** The integration of a conversational retrieval chain combines the reasoning capabilities of the LLM with retrieved information, enabling it to provide coherent, context-aware answers.
- **Inference and Explanation:** By utilizing models like google/flan-t5-base, the system can infer relationships within the text and generate explanatory responses, which demonstrates reasoning skills.
- **Memory-Driven Logic:** The system employs a memory module (ConversationBufferMemory) to track and utilize the history of interactions, allowing for continuity in the conversation—a key element of reasoning-based systems.

➢ System 2 Thinking Evaluation:

The system exhibits **some aspects of System 2 thinking**, which involves deliberate, logical, and effortful problem-solving processes. Here's how:

- **Deliberate and Logical Reasoning:** The LLM retrieves relevant chunks of information using semantic search (via embeddings and FAISS) and formulates coherent responses based on the content.
- **Effortful Problem Solving:** The conversational chain mimics effortful problem-solving by analysing the input query and matching it to document content.
- **Memory Utilization:** The use of ConversationBufferMemory adds a layer of context-awareness, resembling deliberation by keeping track of prior inputs and integrating them into future reasoning.
- **Limitations in Abstract or Sequential Logic:** The system cannot independently evaluate hypothetical scenarios or carry out multi-step logical processes unless explicitly programmed.

4. Implementation Details:

- **PDF Processing:** Uploaded PDF documents are read page by page using PyPDF2.Extracted text is concatenated and stored for further processing.
- **Text Chunking:** Text is divided into overlapping chunks to ensure no context is lost during retrieval.
- **Vectorization and Storage:** Text chunks are converted into vector embeddings using Hugging Face's pre-trained model. FAISS is used to store and retrieve embeddings efficiently based on user queries.
- **Conversational Chain:** A conversational retrieval chain is created, combining the LLM and vector-based retriever to generate context-aware responses.
- **User Interface:** Provides a text input box for questions and displays responses in a chat-like format. Chat history is visually styled and persists across interactions.
- **Dependencies and Installation:**

To install the MultiPDF Chat App, please follow these steps:

a) Clone the repository to your local machine.
b) Install the required dependencies by running the following command:
   pip install -r requirements.txt
c) Obtain an API key from OpenAI or obtain an API Access Token from Hugging Face and add it to the .env file in the project directory.
   OPENAI_API_KEY=your_secret_api_key
   OR
   HUGGINGFACEHUB_API_TOKEN=your_secret_api_token

- **How to run the project:**

a) Ensure that you have installed the required dependencies and added the OpenAI API key to the .env file.
b) Run the main.py file using the Streamlit CLI. Execute the following command:
   streamlit run app.py
c) The application will launch in your default web browser, displaying the user interface.
d) Load multiple PDF documents into the app by following the provided instructions.
e) Ask questions in natural language about the loaded PDFs using the chat interface

5. Results and Performance Analysis:

- **Accuracy:** The application demonstrates a high degree of accuracy in retrieving relevant chunks of text and generating coherent responses.
- **Efficiency:** Text processing and vectorization occur in a reasonable timeframe, ensuring smooth user experience.

- **User Feedback:** Positive feedback on the ease of use and clarity of responses.
- **Limitations:** Performance may degrade with extremely large PDFs or complex queries.

6. Future Scope and Improvements:

- **OCR Integration:** Incorporate OCR capabilities to handle image-based PDFs.
- **Multi-Language Support:** Expand support for multilingual documents using multilingual embeddings.

Student Name: Prachi Pravin Karande

Department: Computer Engineering

Class: TE   Div: A