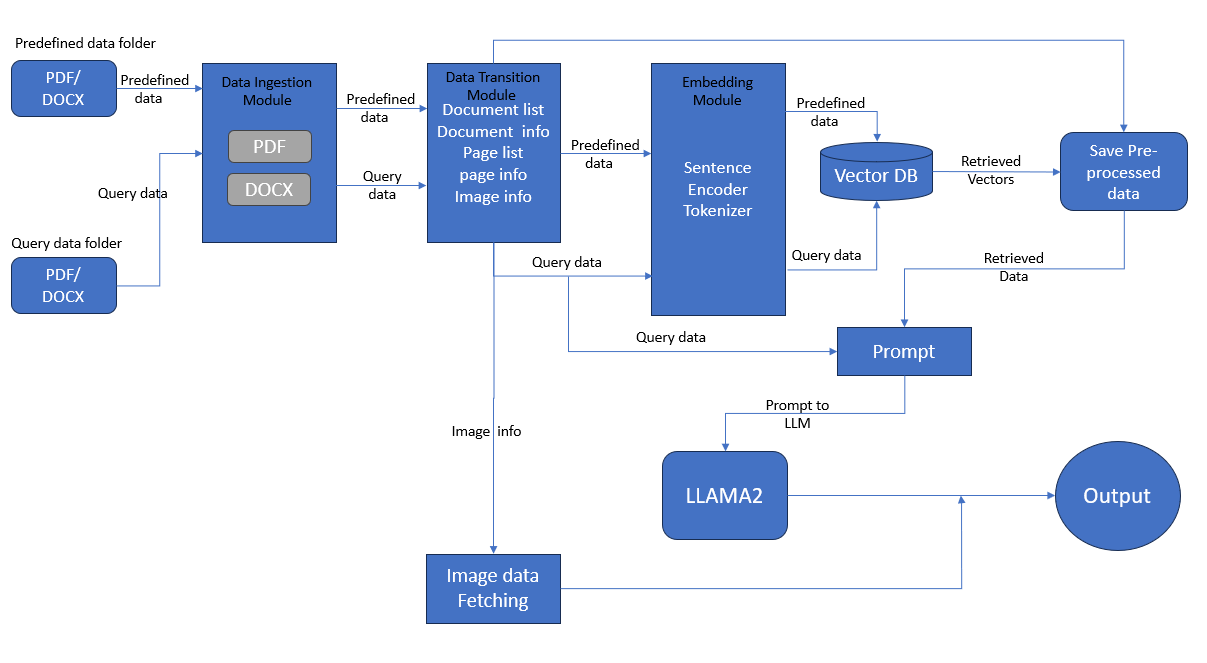
**Technical Documentation**

1. **System Architecture**

****

**Overview:  
The system is designed to classify clinical documents (in PDF and DOCX formats) into predefined clinical sections using a combination of FAISS vector database for similarity matching and the LLAMA2 model for classification. Below is a breakdown of the components:**

**Components:**

1. **Data Ingestion Module:**
   * Purpose: This module handles the intake of documents from both predefined and query datasets, supporting both PDF and DOCX formats.
   * **Process:**
     + Predefined data and query data are passed through this module.
     + The module outputs document lists, document info, page lists, page info, and image info to the Data Transition Module.
2. **Data Transition Module:**
   * Purpose: This module acts as an intermediary, structuring the data for further processing.
   * **Process:**
     + Takes inputs from the Data Ingestion Module and processes them to generate structured information about documents, pages, and images.
     + Passes this structured data to the Embedding Module for further processing.
3. **Embedding Module:**
   * Purpose: Converts text data into vector embeddings that can be queried against the FAISS database.
   * **Components:**
     + Sentence Encoder: Encodes the text data into sentence embeddings.
     + Tokenizer: Breaks down text data into tokens suitable for the LLAMA2 model.
   * **Process:**
     + Generates embeddings for both predefined and query data, which are then sent to the FAISS Vector Database**.**
4. **FAISS Vector Database:**
   * Purpose: Stores predefined vector embeddings and allows querying to find the most similar vectors to the query data.
   * **Process:**
     + Predefined vectors are stored and queried when a new document is processed.
     + The database retrieves the closest matching vectors for further classification.
5. **Prompt Module:**
   * Purpose: Prepares and structures the query data for submission to the LLAMA2 model.
   * **Process:**
     + Combine the query data with the data retrieved from the FAISS database and format it into a prompt suitable for the LLAMA2 model.
     + Ensures that the prompt is aligned with the classification tasks required by the model.
6. **LLAMA2 Model:**
   * Purpose: Performs the final classification of the document into one of the 10 predefined clinical sections.
   * **Process:**
     + Takes the prompt from the Prompt Module and processes it to output the classification.
     + Links image data fetched earlier to the corresponding text sections.
7. **Image Data Fetching:**
   * Purpose: Retrieves image data associated with the documents, which is used to ensure that images are contextually linked to the correct clinical sections.
   * **Process:**
     + Images are extracted and aligned with their corresponding text content.
     + Sent to the LLAMA2 model for contextual understanding and final output.
8. **Output:**
   * Purpose: Delivers the final classified sections, ensuring that images and text are correctly linked.
   * **Process:**
     + The final classified output is generated, which includes both the text and any related images categorized into the appropriate clinical sections.

**2. Data Preprocessing Steps**

**Text Preprocessing**:

* **PDF Processing**:
  + Use PyMuPDF (fitz) to extract text and images.
  + Tokenize the text for embedding generation.
* **DOCX Processing**:
  + Use docx2pdf to convert DOCX to PDFs.
  + Tokenize text using the tokenizer from the Embedding Module.

**Image Preprocessing**:

* **Image Extraction**:
  + Extract images from PDFs files during the ingestion phase.

**Feature Extraction**:

* **Text Embedding**:
  + Use the Sentence Encoder in the Embedding Module to generate text embeddings for both predefined and query documents.

**3. Model Training and Validation**

**FAISS Vector Database**:

* **Data Preparation**:
  + Generate embeddings from predefined clinical sections.
  + Store these embeddings in the FAISS database for quick retrieval.
* **Indexing**:
  + Index the embeddings for fast similarity searches when querying the database with new documents.

**LLAMA2 Model**:

* **RAG**:
  + **Dataset**: Utilize a dataset categorized into the 10 clinical sections, incorporating RAG to enhance the LLAMA2 model's process by retrieving relevant contextual information.

**4. Usage Instructions**

* **Download and Install OLLAMA**:
  + Visit the [Ollama official website](https://ollama.com/) or their GitHub page.
  + Look for the Windows installer or binary for the latest version.
  + If you downloaded an installer (usually an .exe file), double-click it to start the installation process.
  + Follow the on-screen instructions to complete the installation.
* **Verify Installation**:
  + Open a new Command Prompt .
  + Type ‘**ollama --version**’ and press Enter to check if Ollama is installed correctly and to see the installed version.
* **Download LLAMA2 7B model file:**
  + Open a new Command Prompt.
  + ollama run llama2:7b
    - llama2:7b model will get downloaded and now close the Command Prompt
* **Set-up local environment using conda** 
  + **Download the Anaconda Installer:**
    - Go to the Anaconda Distribution page.
    - Download the Windows installer for Python 3.x (64-bit or 32-bit depending on your system).
    - Double-click the downloaded .exe file to start the installer.
    - Follow the on-screen instructions.
  + Execute below commands to create a conda environment in your pc
    - Open a new Command Prompt and navigate to the directory where you want to set-up git and conda environment.
    - Execute the following steps to fetch data github and setup conda env
      * git clone https://github.com/HIRANSHA/LLM-Classification.git
      * Navigate to the Cloned Repository
      * conda create -p venv python==3.10 -y
      * pip install -r requirements.txt
* **Copy Sample Data**
  + Copy sample data in two folder named “pre-defined-data2” and “query\_data”.
    - pre-defined-data2 folder contains reference data (data from different clinical section)
    - query\_data folder contains data to classify
  + Folder path:
    - ./artifacts/ pre-defined-data2
    - ./artifacts/ query\_data
* To generate embeddings of pre-defined data and store it in vector db
  + Run “python populatind\_db.py”
  + **To classify single or each documents provided in ./artifacts/ query\_data**
    - Run “python main.py”
    - Result will be displayed in command prompt
  + **To classify each page and images in a documents provided in ./artifacts/ query\_data**
    - Run “python main.py --pagewise”
    - Result will be displayed in command prompt