Project Report: Multilingual NCERT Doubt Solver

1. Project Overview

This project is a **local, offline, RAG-based (Retrieval-Augmented Generation)** AI system designed to answer student questions based **strictly** on NCERT textbooks. It supports multilingual queries, provides precise citations, and formats answers with clear "word equations" and structured explanations suitable for school students (Grades 5-10).

Key Features:

- **Strict RAG**: Answers are grounded purely in the provided PDF textbooks. Hallucination is minimized.
- **Multilingual Support**: Can process and answer queries in multiple languages (English/Hindi) using cross-lingual embeddings.
- **Smart Formatting**:
  - **Bold Keywords**: Highlights key concepts.
  - **Word Equations**: Formats chemical/biological processes clearly on separate lines.
  - **Clean Citations**: Removes inline clutter and provides a consolidated "Source" footer.
- **Hybrid Ingestion**: Handles both standard digital PDFs and scanned images using OCR (Tesseract).
- **Performance Benchmarking**: Integrated tools to measure latency and generation speed.

---

2. Technical Architecture & Component Breakdown

A. Tech Stack

- **Language**: Python 3.10+
- **UI Framework**: Streamlit
- **LLM Orchestration**: LangChain
- **Vector Database**: FAISS (Facebook AI Similarity Search) - CPU optimized.
- **Embeddings**: `sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2` (HuggingFace).
- **LLM Inference**: `llama-cpp-python` running a Quantized (GGUF) **Mistral-7B-Instruct** model.
- **OCR Engine**: Tesseract OCR (via `pytesseract` and `pdf2image`).

B. How It Works (The Pipeline)

1. **Data Ingestion (`src/ingestion.py`)**:
   - **Loading**: Recursively scans `data/raw/` for PDF files.
   - **OCR processing**: If a PDF is scanned (text not extractable), it converts pages to images and uses **Tesseract** to extract text.
   - **Chunking**: Splits text into 500-character chunks with 50-character overlap using `RecursiveCharacterTextSplitter`.
   - **Vectorization**: Converts chunks into numerical vectors using the Embedding Model and saves them to a local FAISS index (`data/vectorized/`).
2. **Retrieval (`src/retrieval.py`)**:
   - When a user asks a question, the system converts the question into a vector.
   - It searches the FAISS index for the top `k` (default 5) most similar chunks of text from the textbooks.
3. **Generation (`src/generation.py`)**:
   - **Prompt Engineering**: A strictly engineered prompt instructs the LLM to format the answer (bolding, spacing, equations) and ignore inline citations.
   - **Inference**: The context and question are sent to the local Mistral 7B model.
   - **Post-Processing**: A Regex cleaner aggressively strips any hallucinated inline citations from the text.
   - **Footer Generation**: The system programmatically builds a "Source" list based on the metadata of the retrieved chunks.
4. **UI (`app.py`)**:
   - Provides a chat interface.
   - Manages session state (chat history).
   - Displays the final formatted answer and source citations.

---

3. Installation & Setup

Prerequisites

1. **Python 3.10+**
2. **Tesseract OCR**: Must be installed and added to System PATH.
3. **Poppler**: Required for processing PDF images (OCR).
4. **C++ Build Tools**: Required for compiling `llama-cpp-python` (Visual Studio Build Tools on Windows).

Installation Commands

Run these in your terminal within the project directory:

```
# 1. Install Python Dependencies
pip install -r requirements.txt

# 2. Download the Model
# (Ensure 'mistral-7b-instruct-v0.1.Q4_K_M.gguf' is placed inside 'models/' directory)
```

---

4. How to Run

Step 1: Ingest Data (Prepare the Brain)

If you add new PDFs to `data/raw/`, run this command to update the database:

```
python src/ingestion.py
```

Output: This will create `index.faiss` and `index.pkl` in `data/vectorized/`.

Step 2: Launch the App

To start the Chat Interface:

```
streamlit run app.py
```

Access the app at: `http://localhost:8501`

Step 3: Run Benchmarks (Optional)

To test system performance (latency/speed) across 50 questions:

```
python benchmark_50.py
```

Results will be saved to `benchmark_50_results.csv`.

Step 4: Stopping the Application

To stop the application or any running script:

1. Click inside the terminal window where the app is running.
2. Press **`Ctrl + C`** on your keyboard.
3. The process will terminate, and you will see the command prompt again.

---

5. File Structure

```
Project_Root/
├── data/
│   ├── raw/                # Place your NCERT PDFs here
│   └── vectorized/         # Generated FAISS index files
├── models/                 # Place GGUF LLM models here
├── src/
│   ├── ingestion.py        # ETL Pipeline (PDF -> Vector DB)
│   ├── retrieval.py        # Search Logic
│   ├── generation.py       # LLM & Formatting Logic
│   ├── pipeline.py         # Orchestrator
│   └── utils.py            # Helpers (Language detection)
├── app.py                  # Main Streamlit Application
├── config.py               # Configuration (Paths, Prompts, Constants)
├── requirements.txt        # Python dependencies
├── benchmark_50.py         # Automated Stress Test
└── PROJECT_REPORT.md       # This Documentation
```