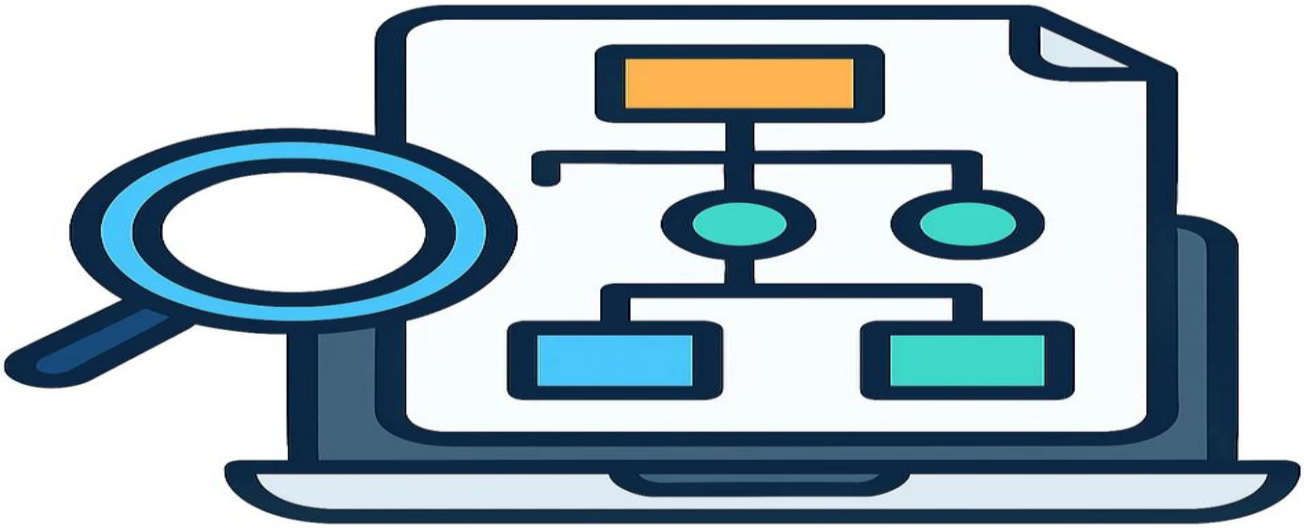
HAQDARSHAK PROJECT REPORT

DISTANCE EDUCATION

Bridging the Distance in Education

Recognized by UGC

Project Title: Scheme Research Tool



Submitted By:

Name: Aditya Singhal

Reg. No.: 12206918

Acknowledgement

I would like to express my sincere gratitude to all those who supported me in the successful completion of this project titled “Scheme Research Tool”.

First and foremost, I would like to thank my project guide, for their constant guidance, valuable suggestions, and encouragement throughout the duration of this work.

I extend my thanks to the faculty and staff of the Computer Science and Engineering, Lovely Professional University, for providing the necessary academic environment and support.

I am also grateful to Haqdarshak for providing a practical and socially relevant problem statement that allowed me to apply my technical skills in a meaningful way.

Special thanks to the creators and maintainers of open-source tools and platforms such as Python, Streamlit, LangChain, HuggingFace, and FAISS, which made it possible to develop this tool without relying on paid APIs.

Lastly, I would like to thank my family and friends for their constant encouragement and support during this project.

Aditya Singhal

12206918

Computer Science and Engineering

Lovely Professional University

Objective

The primary objective of this project is to design and develop a fully automated Scheme Research Tool that simplifies the process of extracting, understanding, and interacting with government scheme documents. The tool is intended to assist researchers, field agents, and citizens by providing quick and accurate access to scheme-related information through a user-friendly web interface.

The Core Objectives of the project are:

1) Accept URLs from government schemes and other sources, including PDF URLs.

This tool lets you pop in one or more web links, including PDF files you often find on government sites. We're talking about things like guidelines, eligibility info, how to apply, and other stuff related to different programs.

2) Extract and Summarize Relevant Content

Uploading the Links of Pdf and government schemes,

Exracts the text from the urls and save it in a form of pickle(faiss_index.pkl)

Splitting the data into chunks

Embedding it into vector representations for efficient storage and retrieval.

This process ensures that the essential information of the document such as scheme benefits, eligibility, required documents, and application processes are preserved in a structured and searchable format.

3) Enable Natural Language Question Answering

Once the Document's Indexing is done then the users can interact with the scheme-research-tool through a simple text input interface. They can ask Question such as:

a) What is the Eligibility Criteria?
b) What documents are required?
c) What are the benefits of the scheme?

4) No Dependence on Paid APIs

In order to make a cost-effective tool, and deployable in low resources environment, I have made it without using any proprietary API's such as OpenAI api.

Instead, I have used:

a) Hugging Face Transformers for local question and answering,
b) Sentence transformers for semantic search embeddings,
c) FAISS for fast-vector based similarity retrieval.

These enable the tool to operate in low-connectivity and offline environments. This makes it the perfect tool for underserved and rural communities.

🧠 Technical Overview

The Application Built using:

- Python
- Streamlit for the web UI
- LangChain for document handling
- HuggingFace Transformers for local Q&A
- FAISS for semantic search
- PyMuPDF for PDF content extraction

It Enables:

- Loading government scheme articles via URL.
- Splitting and embedding documents into chunks.
- Indexing with FAISS for fast similarity-based retrieval.
- Asking Simple and any type of questions related to scheme.
- Getting contextually accurate answers with source URLs.

🧰 Project Components

Files	Description
Main.py	Complete Application logic: loading, indexing, QnA
Requirements.txt	List of the dependencies required
Faiss_index.pkl	Stores the FAISS vector index
.env	Not required (no OpenAI key used)

🛠 Key Features

1) *Flexible URL Input (PDF and Web Pages)*

Users can enter one or multiple URLs pointing to either:

- Government scheme PDF documents, or
- Any Web page

The system identifies the content type and processes it accordingly to ensure seamless document uptake.

2) *Text Extraction*

a) For PDF's, the tool uses Pymupdf to extract clean text from each page of pdf.
b) For Web pages, used UnstructuredURLLoader to load and parse textual content.
c) All extracted content is wrapped into structured Document objects for further processing.

3) *Splitting Text into Small Parts*

a) The full text is split into smaller, overlapping parts.
b) This makes it easier for the tool to search and understand content later when a question is asked.

4) *Converting Text to Embeddings*

a) Each chunk is converted into a vector (embedding) using a free model called MiniLM.
b) These embeddings help the system understand the meaning of the text for smart searching.

5) *Saving and Searching with FAISS*

The utility makes use of the FAISS library to:

a) Save each and every embedding as a.pkl file.
b) When a user asks a query, quickly identify the text's most relevant passages.

6) *Answering Questions Without Internet*

When a query is asked by the user:

a) The system searches the document for the most relevant sections.
b) Then, using the data it discovered, it generates a straightforward response
c) This functions without a paid API or internet connection.

User Flow

User enters URLs in the sidebar :

1) App loads and extracts text from URLs (including PDFs).
2) Text is split and embedded, then indexed with FAISS.
3) User types a question.
4) The app retrieves relevant chunks, runs a local LLM, and displays:
   4.1. The Answer
   4.2. The Source URLs

References

1) **LangChain Documentation** – https://docs.langchain.com
2) **Transformers by HuggingFace** – https://huggingface.co/docs/transformers
3) **Sentence-Transformers (Hugging Face)** – https://www.sbert.net/
4) **FAISS (Facebook AI Similarity Search)** – https://github.com/facebookresearch/faiss
5) **PyMuPDF (fitz)** – https://pymupdf.readthedocs.io/en/latest/
6) **PMSVANidhi Scheme (Test Case)** – https://mohua.gov.in/upload/uploadfiles/files/PMSVANidhi%20Guideline_English.pdf
   Used as a sample URL to test document loading and summarization.