CAPSTONE PROJECT

LEX-RAG

PRESENTED BY

STUDENT NAME: ASHUTOSH JENA

COLLEGE NAME: VITAM

DEPARTMENT: CSE

EMAIL ID: ASHUTOSHJE223@GMAIL.COM

AICTE STUDENT ID: STU643e2980cfb3b1681795456

OUTLINE

- Problem Statement
- Proposed Solution
- System Development Approach
- Algorithm & Deployment
- Result
- Conclusion
- Future Scope
- References

PROBLEM STATEMENT

Law firms, legal organizations or corporations, individual prosecutors and citizens face significant challenges when searching through vast repositories of unstructured policies, forums, case studies or files, laws, amendments and other relevant documents. Traditional on-paper or scrolling keyword-based search tools often fail to capture the context and semantics of queries leading to inefficient gathering and increased time spent on research. Even several natives lack exposure to their rights and general laws. The legal system is facing a growing backlog of cases due to increased crime and litigation, reduced resources which leads to complex and lengthy legal procedures.

PROPOSED SOLUTION

Proposed system is designed specifically for legal documents processing and interaction, with strong emphasis on accuracy and context-awareness in legal responses. The RAG implementation ensures that responses are grounded in actual legal documents while maintaining conversational fluidity through the Gemini model. The solution will consist of the following components:

- **Data Collection:**
  - **Utilizes Hugging Face to store PDF documents as parquet structure with proper metadata and manage it using dataset repo-ID**
  - **Implemented a flexible upload system for new legal documents via FastAPI endpoints**
- **Data Preprocessing:**
  - **Converts PDF documents to text using PyMuPDF (fitz library) for processing pipeline**
  - **Implements document chunking using Character Text Splitter: chunk size = 1000 char, overlap = 200 char for context continuity**
- **Vector Database and Embedding System:**
  - **Document Vectorization achieved using HuggingFace Embeddings with "sentence-transformers/all-MiniLM-L6-v2" model**
  - **The system employs state-of –the-art Natural Language Processing (NLP) model**
  - **Implements FAISS vector store for efficient similarity search and contextual responses**

PROPOSED SOLUTION

- **RAG implementation & API layer:**
  - **Context-aware response generation using Google's Gemini model**
  - **Custom Gemini RAG class for handling – Question processing, Context retrieval, Response generation**
  - **Fast API Backend for session management with secure middleware, rate limiting (2 queries per session)**
  - **Endpoints: /chat for user interactions, /upload for document management, Static file serving for frontend**
- **Deployment:**
  - **Containerization using Docker with configured load balancer in environment and set up Session Secret Key for secure implementation**
  - **Container Registry in Azure for model and container versioning**

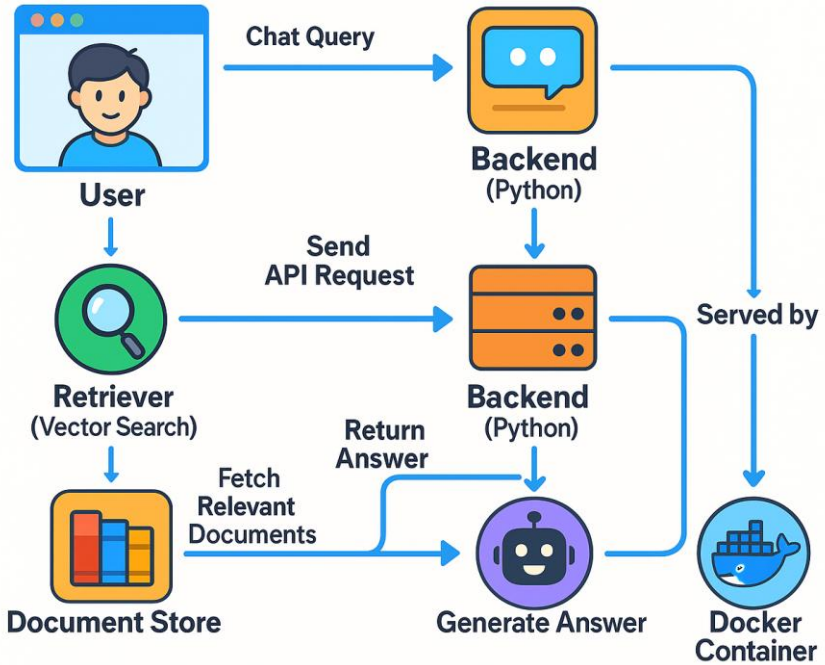
SYSTEM APPROACH

Programming Languages:

- **Python: Core backend API logic, data processing, data retrieval, RAG model and integration with pre-trained NLP models**
- **JavaScript (Vanilla): Interactive front-end components and client-side logic**
- **HTML: User interface and presentation layers**
- **Dockerfile and Git for containerization and version control**

Frameworks and Libraries:

- **FastAPI: deployment and backend services**
- **Hugging Face Transformers: pre-trained language models**
- **FAISS: for efficient vector-based and lexical search indexing**
- **Hugging Face Spaces & CLI: for dataset storage, access and parquet data version**
- **Langchain: for chaining LLMs, integrating retrieval and generation**
- **Uvicorn: ASGI server to run the FastAPI app**
- **Starlette: Middleware and session management**
- **Pydantic: Data validation for API models**
- **PyMuPDF & pdfplumber: for PDF parsing and processing**
- **Google Gemini: Used as the LLM for generating contextual answers**



ALGORITHM & DEPLOYMENT

Core Workflow (RAG pipeline):

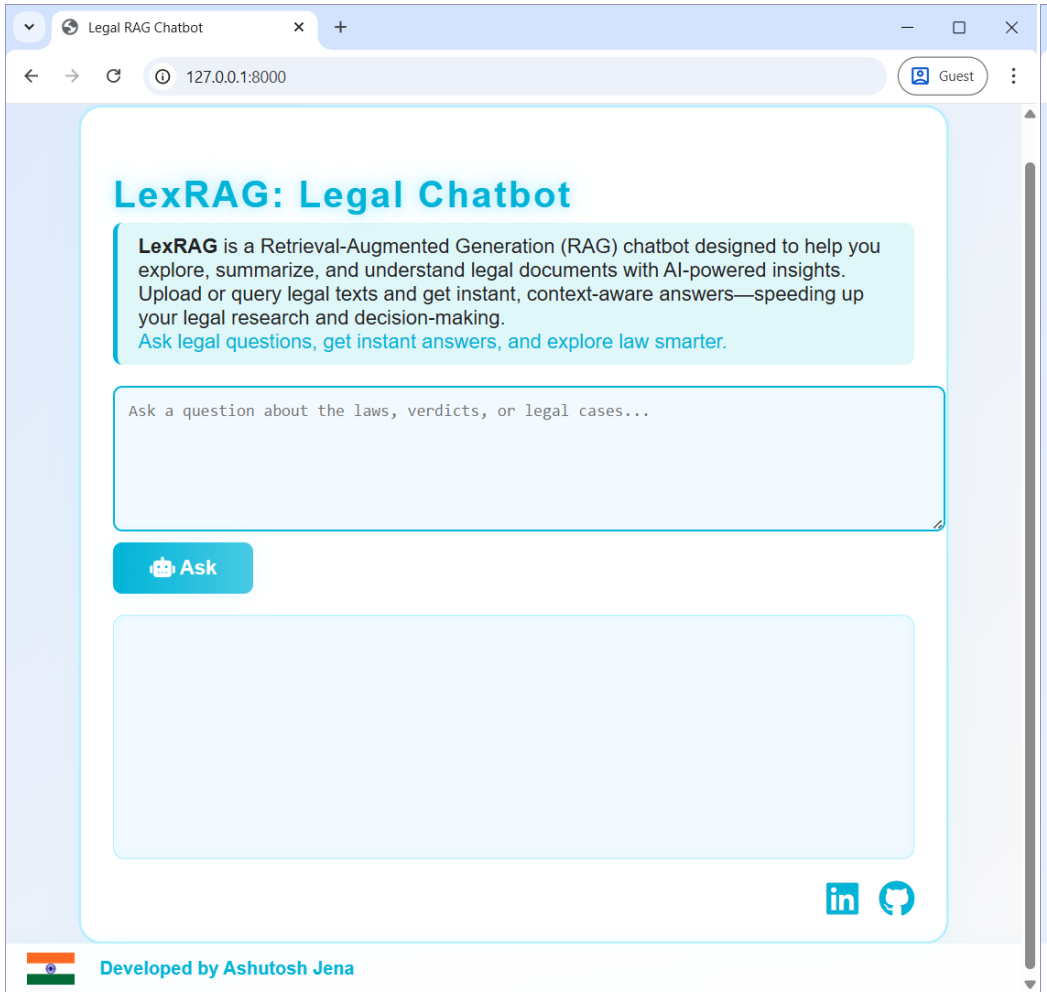
- **Document Ingestion:**
  - **Load legal documents from Hugging Face dataset via API token in integrating with CLI**
  - **Parses PDFs using PyMuPDF**
- **Text splitting and Vectorization:**
  - **Splits large documents into manageable chunks using *CharacterTextSplitter* from LangChain**
  - **Embeds document chunks using *HuggingFaceEmbeddings* with the model *sentence-transformers/all-MiniLM-L6-v2***
- **Retrieval and Generation:**
  - **At query time, retrieves the top relevant chunks using semantic similarity**
  - **Combines the retrieved context with the users' question**
  - **Constructs a prompt for Gemini LLM and calls it via *google-generativeai* to generate the final output ; implemented using python SDK**
  - **Integrated a usage limit up to 2 queries per session**

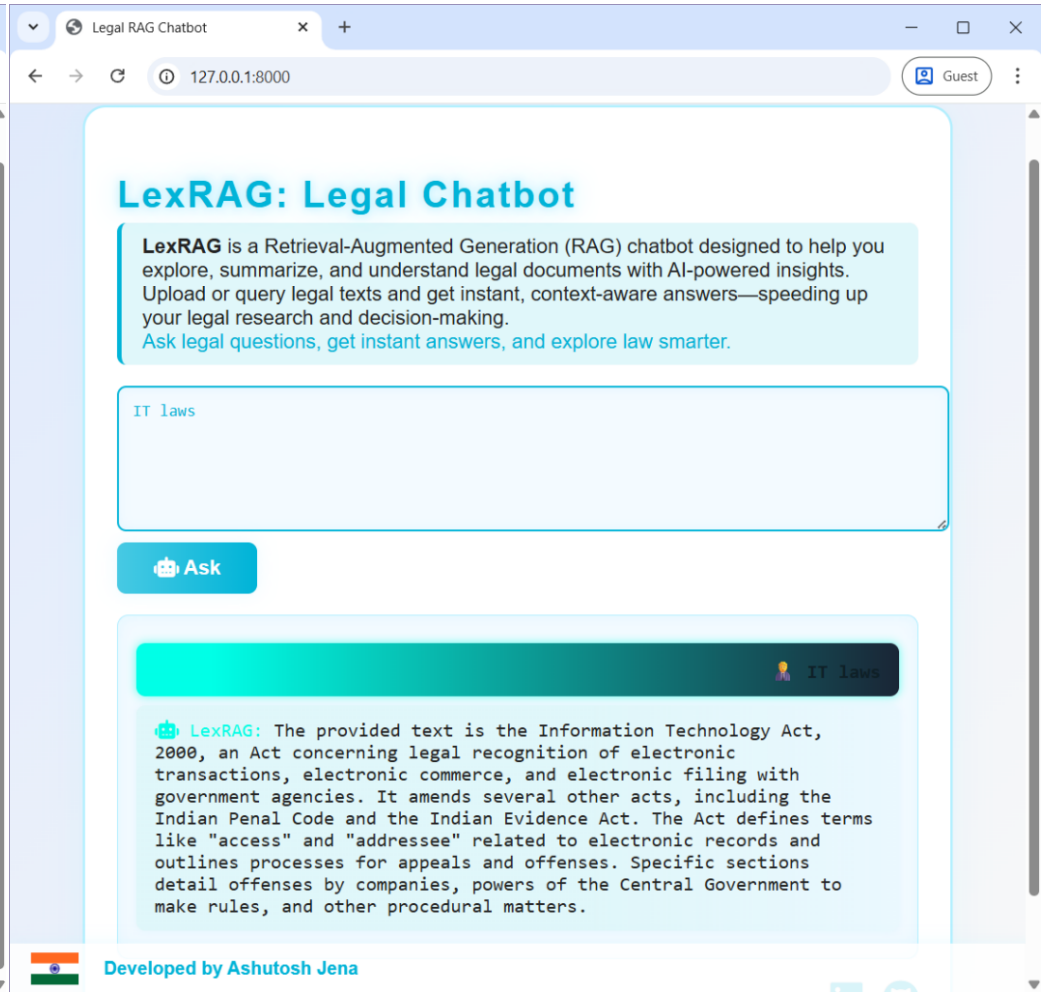
ALGORITHM & DEPLOYMENT

Deployment:

- **Local:**
  - **Clone the repo:** ***git clone*** ***https://github.com/ajverse/LexRAG.git***
  - **Set up** ***.env*** **with** ***GOOGLE_API_KEY*** **and** ***HF_DATASET_ID***
  - **Install dependencies from** ***–r requirement.txt***
  - ***Uvicorn app.main:app –host 0.0.0.0 –port 7860***
- **Docker Deployment:**
  - **The Dockerfile sets up a Python 3.10-slim base**
  - **Build the image** ***docker build –t lexrag***
  - **Run the Container** ***docker run –p 7860 –env-file .env lexrag*** **the app will be available at** ***http://localhost:7860***
  - **Container registry using Azure service**
- **Hugging Face Auth:**
  - **Copy my public dataset from repo ID: ajverse/law-docs**
  - **Authenticate with** ***HUGGINGFACEHUB_API_TOKEN*** **in CLI**

RESULT


Legal RAG Chatbot
127.0.0.1:8000
Guest
LexRAG: Legal Chatbot
LexRAG is a Retrieval-Augmented Generation (RAG) chatbot designed to help you explore, summarize, and understand legal documents with AI-powered insights. Upload or query legal texts and get instant, context-aware answers—speeding up your legal research and decision-making.
Ask legal questions, get instant answers, and explore law smarter.
Ask a question about the laws, verdicts, or legal cases...
Ask
Developed by Ashutosh Jena


Legal RAG Chatbot
127.0.0.1:8000
Guest
LexRAG: Legal Chatbot
LexRAG is a Retrieval-Augmented Generation (RAG) chatbot designed to help you explore, summarize, and understand legal documents with AI-powered insights. Upload or query legal texts and get instant, context-aware answers—speeding up your legal research and decision-making.
Ask legal questions, get instant answers, and explore law smarter.
IT laws
Ask
IT laws
LexRAG: The provided text is the Information Technology Act, 2000, an Act concerning legal recognition of electronic transactions, electronic commerce, and electronic filing with government agencies. It amends several other acts, including the Indian Penal Code and the Indian Evidence Act. The Act defines terms like "access" and "addressee" related to electronic records and outlines processes for appeals and offenses. Specific sections detail offenses by companies, powers of the Central Government to make rules, and other procedural matters.
Developed by Ashutosh Jena

CONCLUSION

LexRAG addresses the limitations of Black-Box models for predictive judgments & traditional search systems by combining lexical and semantic retrieval with advanced generative capabilities. It empowers users to access precise and context-aware information from large-scale document repositories, boosting productivity and decision-making. Through its modular and scalable architecture, LexRAG is adaptable to various organizational needs and data environments.

FUTURE SCOPE

1. **Multilingual Support: Incorporating NLP models for multiple languages to broaden usability**
2. **Real-time Data Ingestion: Enabling dynamic ingestion and indexing of streaming or frequently updated data**
3. **Enhanced Summarization: Integrating more sophisticated generative models for abstractive summarization and report generation**
4. **Personalization: Leveraging user profiles and past queries to deliver personalized results**
5. **Integration with knowledge graphs: Enhancing contextual understanding by linking entities and concepts across documents**

REFERENCES

GitHub Link: https://github.com/ajverse/LexRAG/

HF Transformers Documentation: https://huggingface.co/transformers/

FAISS: https://github.com/facebookresearch/faiss/

Langchain: https://docs.langchain.com/

Docker: https://docs.docker.com/

FastAPI: https://fastapi.tiangolo.com/

BERT: Pre-training of Deep Bidirectional Transformers for Understanding: https://arxiv.org/abs/1810.04805

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks: https://arxiv.org/abs/2005.11401

Thank you