HackLLM

A Premier LLM Hackathon under ESYA'25

Team Name: TeamX

Challenge Accepted: Medical Summarization Challenge

Project Title: ClearMed

Team Details: Rishabh Gupta (Team Leader)
Sachin Gupta
Rishabh Dubey
Ranjan Yadav

From College: ABES Engineering College, Ghaziabad

Problem statement:

- Medical documents are often long and complex.
- Doctors need clinically relevant summaries, while patients need easy-to-understand summaries.
- Existing tools rely on LLMs with APIs, but here we must build a custom lightweight summarization model.

Solution:

ClearMed (**AI System**) is an AI-powered solution that can read long medical reports or Q&A documents and generate two types of summaries:

- **Clinician Focused Mode →** Gives a summary with correct medical terms and detailed clinical information.
- **Patient Friendly Mode →** Gives a summary in simple, clear language without medical jargon.

The **AI** makes sure:

- Every line in the summary is linked to the original report, so users can verify the source.
- It checks for risky or sensitive statements (like dosages or absolute instructions) and adds a safety disclaimer.
- In Patient Mode, it automatically translates complex medical terms into simple words while keeping the meaning correct.

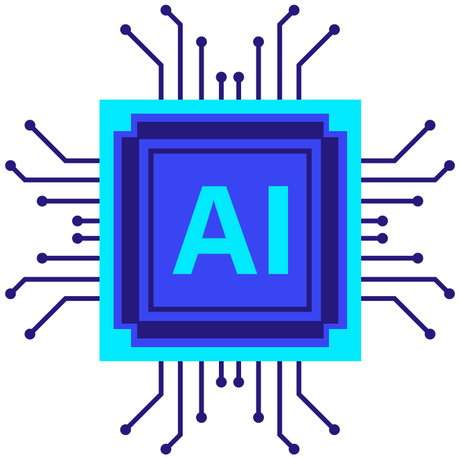
Why Our AI is Unique

Simple language converter: Complex medical terms are automatically replaced with easy-to-understand words for patients.

Two modes (Doctor & Patient): The AI has two separate outputs – one keeps full medical details for doctors, the other simplifies terms for patients.

Runs locally: The model is small and fast, so it works on a normal laptop without cloud or external APIs.

Trustworthy summaries: Every line in the summary is linked back to the original report so nothing is made up.



Safe to use: A built-in checker scans for risky advice (like dosages) and adds warnings/disclaimers

System Architecture

Ingest (Input Handling)

- If the PDF is already searchable → parse text directly.
- If the PDF is scanned → run OCR to extract text and preserve layout.

Sectionizer

- Automatically detects key sections: History, Symptoms, Diagnosis, Treatment, Follow-up.

Medical NER + Ontology Linker

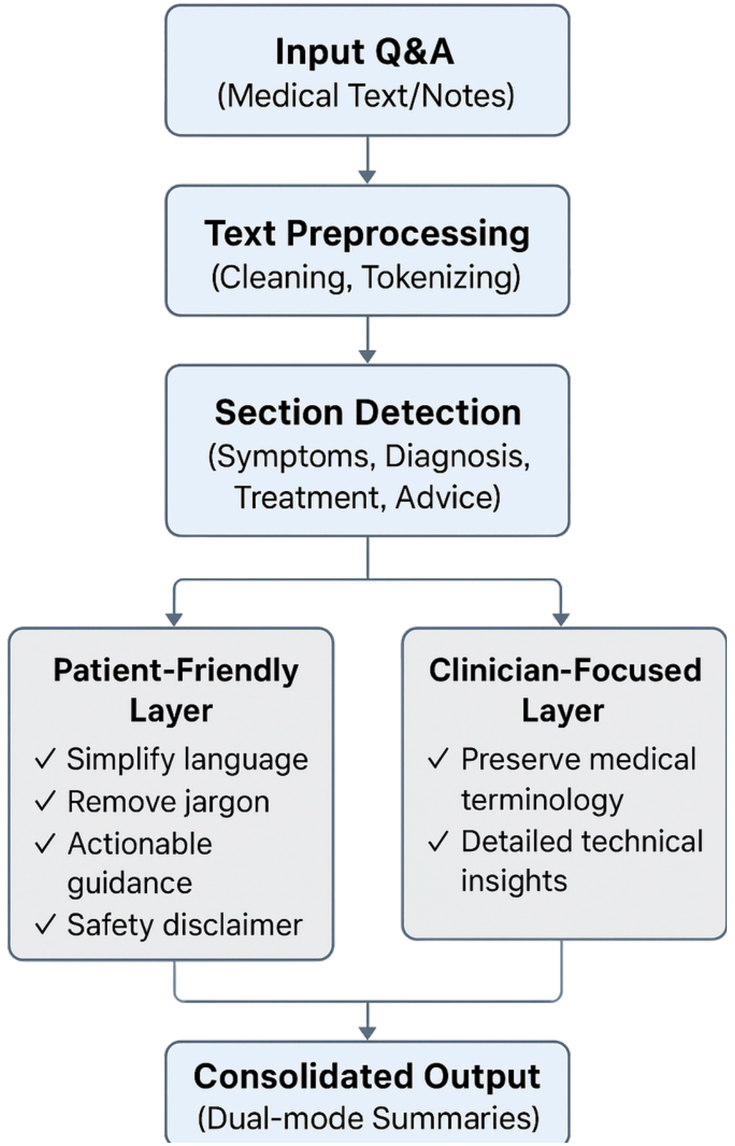
- Identifies medical terms and links them to standard IDs (UMLS).
- Builds a glossary for simpler explanation later.

Summarizer (Dual-Head Model)

- Uses one shared encoder with two decoders:
  - Clinician view (Decoder-C): detailed, technical summary.
  - Patient view (Decoder-P): easy-to-read summary.

Web UI

- Side-by-side clinician & patient summaries.
- Evidence highlighting, disclaimers, and PDF export option.



Technical blueprint

Pre-processing
- PDF & OCR: PyMuPDF / pdfminer, Tesseract
- Text Cleaning & NLP: spaCy

Section & Entity Detection
- Models: Lightweight Transformer / BiLSTM
- NER + Ontology: Transformer-based NER, UMLS mapping

Evidence Retrieval
- Search: BM25 (sparse) + Dense embeddings (MiniLM)

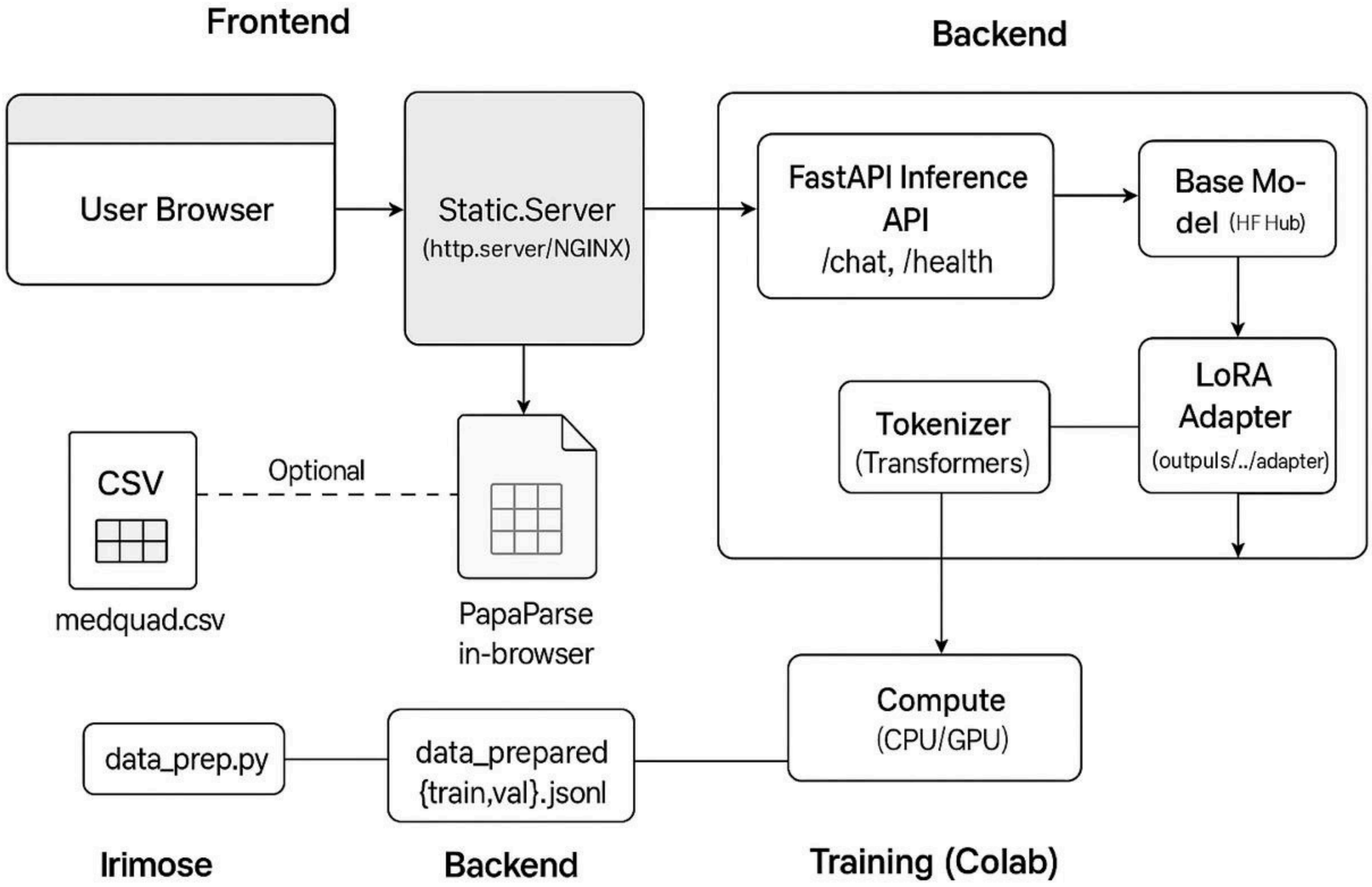
Summarization Model
- Framework: PyTorch
- Architecture: Distilled Transformer encoder + Dual decoders (Clinician & Patient)

Post-processing
- Glossary-based jargon simplifier
- Risk classifier for unsafe claims

Application Layer
- Frontend: Streamlit or Flask (with JS for highlighting)
- Export: PDF reports

Optimization
- Model distillation & quantization for efficiency

Work Flow


Frontend
Backend
User Browser
Static.Server
(http.server/NGINX)
FastAPI Inference API
/chat, /health
Base Model (HF Hub)
LoRA Adapter
(outputs/../adapter)
Tokenizer
(Transformers)
CSV
Optional
medquad.csv
PapaParse
in-browser
Compute
(CPU/GPU)
data_prep.py
data_prepared
{train,val}.jsonl
Irimose
Backend
Training (Colab)