Intelligent Clinical Trials Search

Problem Overview

Clinical trial data is vast, complex, and highly structured, making it difficult for researchers, doctors, and patients to find relevant studies using simple keywords. Traditional search interfaces often fail to capture the nuance of a request like *"Show me ongoing Phase 3 lung cancer trials in the USA for adults."* Users are forced to manually navigate complex filter sidebars or learn specific query syntaxes.

The problem is the absence of an intuitive interface that allows users to query this rich dataset using everyday language.

Core Challenge

Participants must build a full-stack solution that:

1. **Ingests** the provided clinical trials dataset into Elasticsearch.
2. **Translates** natural language user queries (e.g., *"recruiting diabetes trials near Boston"*) into structured Elasticsearch queries.
3. **Displays** the results in a user-friendly interface with relevant metadata.

Constraints & Stack

Parameter	Requirement
Language	Python (Backend) - Flask/Fast API
Database	Elasticsearch (or OpenSearch)
NLP	Open choice (Spacy, Transformers, LLM APIs)
Frontend	React
Dataset	Provided `clinical_trials.json`

Technical Details

Data Ingestion

Write a Python script that will ingest the given clinical trials into an Elasticsearch index.

- **Source Data**: *`clinical_trials.json`* (List of JSON documents).

- **Task**:
  - Setup Elasticsearch index locally.
  - Indexer script to read the JSON data.
  - Define an appropriate Elasticsearch Index Mapping (schema) to handle fields like *`brief_title`*, *`official_title`*, *`condition`*, *`phase`*, *`status`*, *`sponsors`*, *`enrollment`*, *`completion_date`*, etc.
  - Index all documents into a local Elasticsearch instance.

Natural Language Processing (The "Brain")

Develop the API layer using Flask or FastAPI that will be used as a server to handle user requests, convert it into Elasticsearch query, and return the expected results.

Required API Endpoints

1. Route: Search
   a. Method: GET
   b. Endpoint: /search/<natural language query>
   c. Behavior: Returns a paginated list of all the trials from Elasticsearch index.

- The system must analyze user input to identify intent and entities.
- **Sample user query**: *"List all the Phase 2 trials for Breast Cancer associated with BRCA1 gene."*
- **Expected Entities/Filters**:
  - **Phase**: `PHASE2`
  - **Condition**: `Breast Cancer`
  - **Keyword/Gene**: `BRCA1`
- Dynamically build an Elasticsearch query DSL based on the extracted entities.
- Must handle cases where:
  - Filters are exact (e.g., `Phase: PHASE3`).
  - Filters are text-based matches (e.g., Title contains "Vaccine").
  - Ranges are implied (e.g., "large trials" -> `enrollment > 500` - *optional advanced feature*).
- The search must return relevant documents ranked by score.

Frontend UI

- A clean search interface with:
  - **Search Bar**: For natural language input.

   - **Interpretation Display**: “We understood: Condition=X, Phase=Y…” (Show the user how their query was parsed).
   - **Results List**: Cards showing Title, Status, Conditions, Phase, Sponsor.

Evaluation Criteria

How will this solution be judged?

1. **Entity Extraction Accuracy**: Does the system correctly identify ‘Phase’, ‘Condition’, ‘Status’, and ‘Location’ from a complex sentence?
2. **Query Precision**: Does the generated Elasticsearch query accurately reflect the user’s intent? (e.g., avoiding false positives).
3. **Result Relevance**: Do the top results match the criteria?
4. **Error Handling**: How does it handle ambiguous queries or queries with no matches?
5. **UX/UI Quality**: Is the interface intuitive? Does it provide feedback on how the query was interpreted?

Bonus Challenges (Nice to Have)

1. **Clarification Questions**: If the user’s query is ambiguous, misspelled, or too broad, the system should ask a clarifying question to refine the search (e.g., “Did you mean ‘Lung Cancer’?” or “I found trials for both Adults and Children; which group are you interested in?”).
2. If users provide queries such as “List open clinical trials for melanoma”, the system should understand that “open” means “recruiting” in clinical trials.
3. Implement auto-suggestion feature so that the system can recommend words/spellings as the user types in the query.