Problem Statement 6:

AI-Powered Document Classification and Intelligent Indexing

Organizations and institutions handle large volumes of unstructured data such as documents, reports, contracts, and emails. Traditional methods like manual tagging and keyword-based search are inefficient, error-prone, and slow. There is a pressing need for AI-driven solutions that can classify, index, and retrieve documents automatically, while ensuring security and data privacy.

Challenge:

In 36 Hours, Design and implement an AI-powered Document Classification and Indexing System that can:

1. **Classify Documents:**
   - Automatically categorize documents into predefined classes (e.g., Finance, HR, Technical Reports, Contracts, Invoices).
2. **Extract Metadata & Summaries:**
   - Capture essential details such as title, author, key topics, and entities to build an intelligent index.
3. **Enable Semantic Search:**
   - Provide fast and accurate retrieval of documents based on meaning rather than just keywords.
4. **Ensure Security:**
   - Leverage open-source AI/LLM models (e.g., Hugging Face transformers) with optional integration to private/secure LLM deployments to preserve confidentiality.
5. **Provide User Interface:**
   - Develop a frontend dashboard to upload, classify, and search documents seamlessly.

Backend Requirements

1. Document Upload & Storage

- Accept: **PDF, DOCX, TXT**, Store files **locally**, Save metadata (**filename, date, uploader**) in **SQLite DB**

2. Document Classification

- Categories: **Finance, HR, Legal, Contracts, Technical Reports, etc**

3. Metadata Extraction

- **Title**: First bold line / sentence
- **Author**: Look for "Author:", "By:", emails
- **Date**: Regex-based detection

- **Entities**: Use **spaCy** or regex for names, orgs, amounts

4. Summarization

- **Extractive** approach
- Sentence scoring via **TF-IDF / word frequency**
- Pick **top 3–5** sentences

5. Semantic Search

- Generate embeddings using **SentenceTransformers**
- Store in **FAISS** or in-memory
- Use **cosine similarity** for ranking results

6. Security & Access Control

- **Login system** (hardcoded/simple DB)
- **Role-based access**:
  - HR → HR documents only
  - Finance → Finance documents only
- Store **access logs** (uploads, views)

Frontend Requirements

1. Dashboard

- Upload button
- Document list with: **Title, Category, Author, Upload Date**

2. Document View

- Show:
  - Metadata (Title, Author, Date)
  - Extracted **summary**
  - **Download** original file

3. Search & Filters

- **Search bar** for semantic search
- Filters: **Category, Author, Date**
- Results ranked by **relevance**

4. User Authentication

- **Login page** with username/password
- Show documents based on **user role**