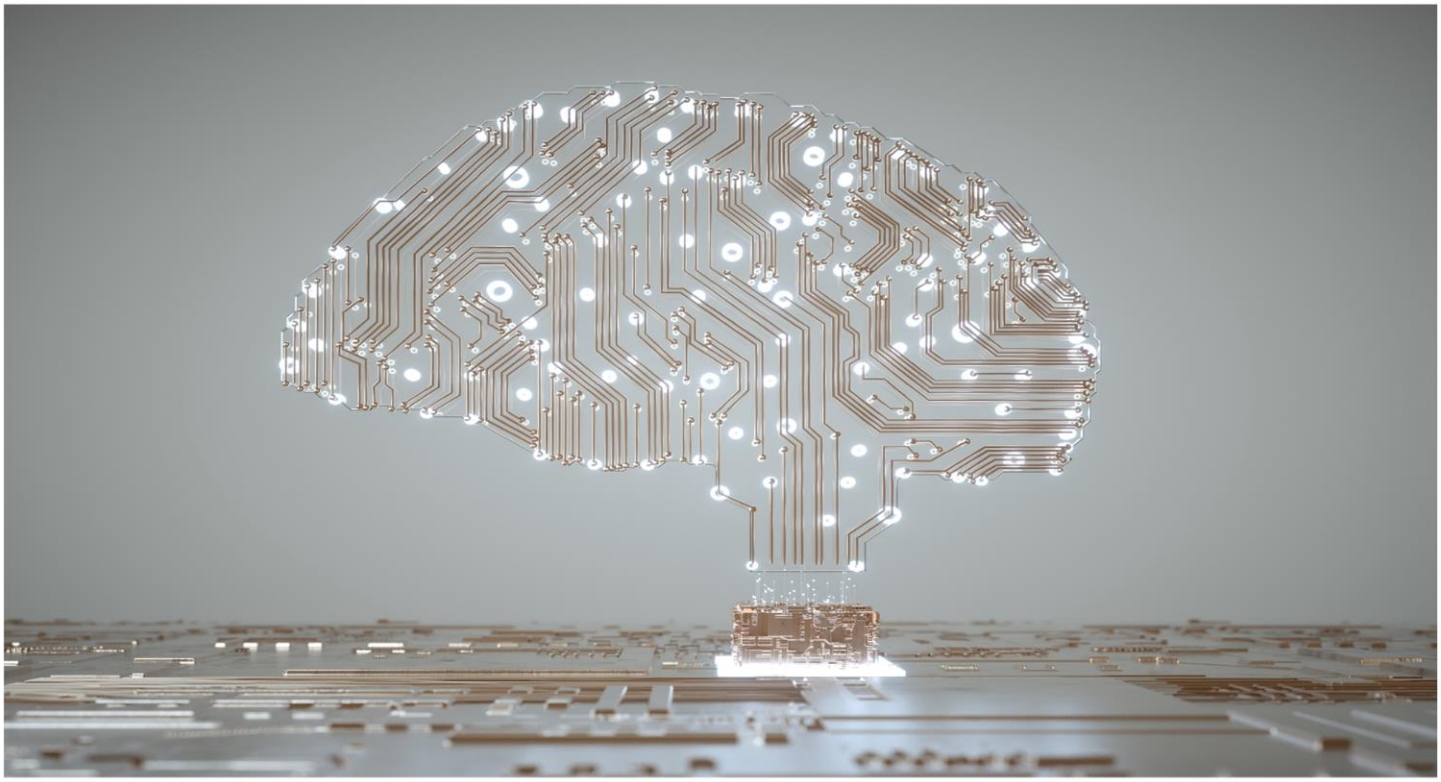
Prajna AI

Wizzify Your Data

HACKATHON

PROBLEM STATEMENT



Problem Overview:

In the era of digital information, managing and querying vast amounts of PDF documents is crucial for businesses and researchers alike. However, users often struggle to extract insights and validate facts from unstructured data contained within PDF files. This hackathon challenge focuses on building a **comprehensive PDF ingestion and querying system** that allows users to upload PDF documents, automatically generate embeddings, suggest relevant questions, enable user queries, and provide precise citations for validation. The solution should feature an intuitive, interactive frontend and be deployable in a cloud environment.

Objective:

Develop an **Intelligent PDF Querying System (IPQS)** with the following capabilities:

1. **PDF Document Ingestion**: Support seamless uploading and parsing of PDF documents.
2. **Embedding Generation and Data Persistence**: Create semantic embeddings for the content within the PDFs and store them efficiently for querying.
3. **Question Suggestion Engine**: Automatically suggest relevant questions based on the content of the uploaded PDFs.
4. **User Querying Interface**: Allow users to submit natural language queries against the PDF content.
5. **Citation and Validation**: Provide citations for the information returned in response to user queries.
6. **Interactive Frontend**: Design a user-friendly web interface for document interaction.
7. **Deployment**: Ensure the system is deployable to a cloud environment for public access.

Detailed Requirements and Criteria:

1. PDF Document Ingestion

10 MARKS

- **Criteria**: Implement functionality to upload and parse PDF documents accurately, preserving the structure of the content (headings, paragraphs, lists, etc.).
- **Functionality**: Users can upload a PDF file, which is processed to extract textual content and metadata.
- **Evaluation**: Ability to handle various PDF structures without losing information or context.

2. Embedding Generation and Data Persistence

30 MARKS

- **Criteria**: Use a state-of-the-art model to generate embeddings that capture the semantics of the PDF content. Store these embeddings in a vector database (e.g., ChromaDB).
- **Functionality**: Ensure efficient retrieval of embeddings for querying.
- **Evaluation**: Performance of similarity-based search and how well embeddings represent the document's content.

3. Question Suggestion Engine

50 MARKS

- **Criteria**: Develop an engine that generates 3-5 insightful questions related to the uploaded PDF's content.
- **Functionality**: Automatically generate diverse questions that cover key themes, concepts, and points of interest in the document.
- **Evaluation**: Quality, relevance, and diversity of generated questions.

4. User Querying Interface

40 MARKS

- **Criteria**: Create a responsive interface that allows users to submit natural language queries related to the content of the PDF.
- **Functionality**: Users receive concise, accurate answers to their queries.
- **Evaluation**: Accuracy and relevance of responses to user queries, as well as robustness against varied query structures.

5. Citation and Validation

70 MARKS

- **Criteria**: For each answer generated in response to a user query, provide a citation that indicates the source within the PDF.
- **Functionality**: Allow users to trace back responses to specific pages or sections in the PDF.
- **Evaluation**: Accuracy and clarity of citations, enhancing the user's ability to validate information.

6. Interactive Frontend

40 MARKS

- **Criteria**: Build an intuitive, user-friendly web interface that facilitates document uploads, question suggestions, and query entry.
- **Functionality**: The interface should support features like viewing question suggestions, entering queries, and displaying responses with citations.
- **Evaluation**: Usability, responsiveness, and design appeal of the frontend.

7. Deployment

60 MARKS

- **Criteria**: Deploy the entire system on a cloud platform (e.g., AWS, Vercel) with considerations for scalability, reliability, and accessibility.
- **Functionality**: Ensure users can access the system through a public URL with minimal downtime and fast response times.
- **Evaluation**: Functionality of the deployment, ease of access, and performance under user load.

Additional Points:

1. **Cost-Effective API Usage:** Ensure the use of APIs and external services is cost-effective while maintaining a focus on accuracy. If a free API or service is available that meets the accuracy requirements, prioritize its use to minimize operational costs for the company. Consider the trade-off between cost and performance to provide a balance that optimizes both financial and functional aspects of the system.
2. **Additional Document Types:** Support additional document formats like HTML or spreadsheets (CSV, Excel) to enhance the versatility of the system, allowing users to work with a wider range of data sources and increasing the utility of the platform.
3. **Customizable User Dashboard**:
   Create a customizable dashboard where users can manage their uploaded documents, view query history, and access frequently asked questions.

Note ->

Reference Implementation: For inspiration and guidance on implementing document querying systems, consider examining Notebook LM by Google, which offers insights into handling document-based interactions.