WO AI/ML Task

Document Intelligence RAG Chatbot System

Problem Statement

Build an intelligent document processing and querying system that can handle various document formats and provide accurate, contextual responses with proper citations.

Time Limit

24 Hours

Core Requirements

Model Selection and justification

You can choose any LLM, embedding and re-ranker model of your choice, but you need to justify why did you used that particular model.

Document Processing Pipeline

1. **Document Ingestion**
    - Support for multiple document formats:
        - PDF (including scanned documents with OCR)
        - Microsoft Word (DOCX with scanned images with OCR)
        - Text files (TXT)
    - Document metadata extraction
    - OCR processing for scanned documents
2. **Text Processing & Chunking**
    - Implement contextual chunking with metadata:
        - Page number
        - Document name
    - Justify your chunking strategy and size selection

Document Processing API

1. Embedding API

This API is used to embed documents. The embedded documents can then be queried using the Query API.

Endpoint: POST /api/embedding

Sample Request:

```
{
 "document": "sample_document.pdf"
}
```

Successful Response:

```
{
 "status": "success",
 "message": "Document embedded successfully.",
 "document_id": "12345"
}
```

Unsuccessful Response:

```
{
 "status": "error",
 "message": "Failed to embed document.",
 "error_details": "Document content is empty."
}
```

2. Query API

This API allows users to query embedded documents, receive citations, and track conversation history.

Endpoint: POST /api/query

For the First Message in a New Conversation

When initiating a new conversation, give conversation_id in the request. The API will generate and return a new conversation_id in the response.

Sample Request:


```
{
  "query": "What is the main argument in the document?",
  “document_id”: “12345,
  "require_citations": true
}
```


Successful Response with New Conversation ID:


```
{
  "status": "success",
  "response": {
    "answer": "The main argument is about the impact of technology on education.",
    "citations": [
      {
        "page": 12,
        "document_name": "sample_document.pdf"
      }
    ]
  },
  "conversation_id": "abc123xyz"
}
```

For Subsequent Messages in the Same Conversation

Include the conversation_id in each follow-up query to maintain context.

Sample Request with Conversation ID:


```
{
  "query": "How does this relate to student engagement?",
  "require_citations": true,
  “document_id”: “12345,
  "conversation_id": "abc123xyz"
}
```


Successful Response:


```
{
  "status": "success",
  "response": {
    "answer": "It suggests that technology can enhance student engagement by offering interactive learning tools.",
    "citations": [
      {
        "page": 15,
        "document_name": "sample_document.pdf"
      }
    ]
  }
}
```


Unsuccessful Response:


```
{
  "status": "error",
  "message": "Invalid conversation ID. Please start a new session."
}
```

Test Documents

Implement and test your solution with at least one document from each category:

1. **Academic/Technical**
   - Research paper (PDF)
   - Technical documentation (DOCX)
2. **Literary/Historical**
   - Public domain book (TXT)
   - Religious or philosophical text
3. **Business/Legal**
   - Annual report or policy document
   - Legal contract or agreement

Deliverables

1. **Implementation**
   - Complete source code
   - Dependencies and setup instructions
2. **Technical Documentation**
   - Detailed technology choice justifications
   - API documentation

Submission Requirements

1. GitHub repository with complete code
2. README with:
   - Setup instructions
   - Technology choice justifications
   - Performance metrics
   - API Documentation

Remember: Focus on justifying your technology choices with data and reasoning rather than just implementing the solution. Your ability to make and explain architectural decisions is key to this assessment.