Flask Web Application for PDF Summarization and Question Answering

- Karthik M

Overview

This Flask web application allows users to upload a PDF file, generate a summary, and ask questions about the content of the PDF. The application leverages Cohere for text summarization and question answering. It uses FAISS for efficient similarity search within the text.

Dependencies

1. Flask
2. PyPDF2
3. Langchain (for text processing and QA chain)
4. langchain_community
5. Cohere API
6. FAISS
7. Cohere
8. Tailwind

Application Structure

Global Variables

knowledge_base: Stores the knowledge base created from the PDF content.
api_key: Stores the API key for accessing Cohere services.

Functions

generate_summary(full_text, api_key)
Generates a summary of the given text using the Cohere API.

Parameters
full_text (str): The complete text extracted from the PDF.
api_key (str): The API key for Cohere.

Returns
summary (str): The generated summary.

Flask Routes
`/ `(GET and POST)

Renders the main page and handles PDF file uploads.

1. GET: Renders the index.html template.
2. POST: Processes the uploaded PDF file, extracts text, generates a summary, and creates a knowledge base.
    - Checks if a file part is present in the request.
    - Extracts text from the uploaded PDF using PyPDF2.
    - Splits the text into chunks using CharacterTextSplitter.
    - Generates a summary of the full text using the generate_summary function.
    - Evaluates the summary length.
    - Creates a knowledge base using FAISS and Cohere embeddings.
3. Returns:
    - JSON response containing the summary and its evaluation.

`/ask` (POST)

1. Handles question answering based on the uploaded PDF content.
    - Parameters (JSON):

question (str): The question to be answered.

- Returns:
  JSON response containing the answer to the question.

Error Handling

The application includes error handling for scenarios such as missing files, errors during summary generation, and errors during question answering.

Usage

1. Upload PDF:
   - Navigate to the main page and upload a PDF file.
   - The application extracts text from the PDF, generates a summary, and evaluates the summary's length.
2. Ask Questions:
   - Use the /ask endpoint to submit questions about the uploaded PDF content.
   - The application uses similarity search to find relevant chunks of text and a QA chain to generate an answer.

Heuristic feedback

The feedback block allows users to rate their experience and leave comments. It includes:

1. Star Rating: Users click on stars to rate. The toggleStar(starNumber) function updates star colors and sets the rating.
2. Comment Textarea: Users can type comments.

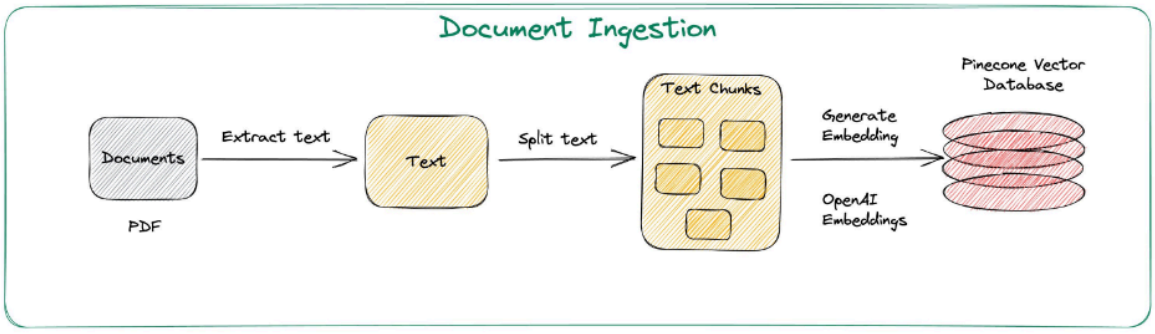
3. Submit Button: On clicking, submitFeedback() disabled inputs, changes the button text to "Submitted," and shows an alert thanking the user with their rating.

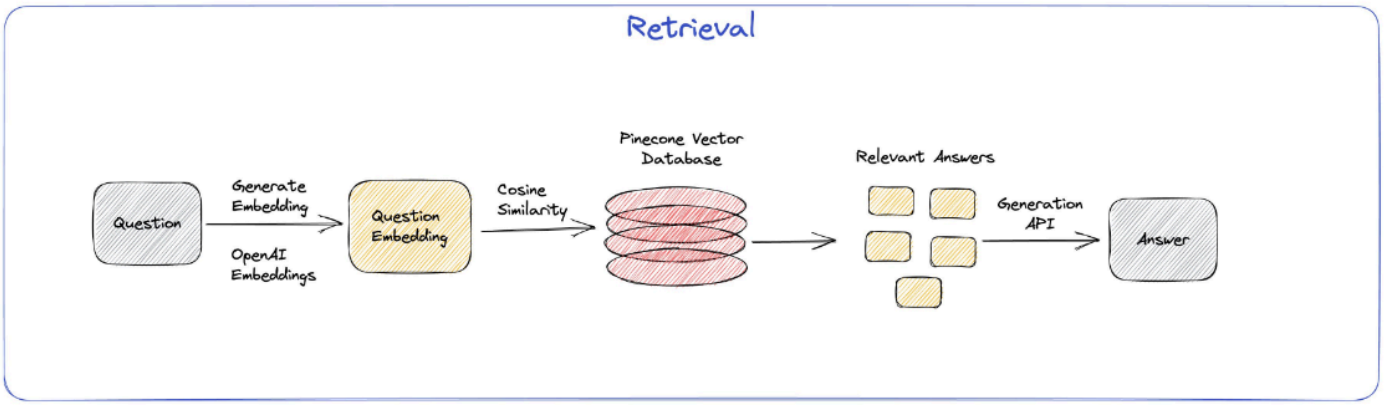
Running the Application

Run the Flask application by executing:

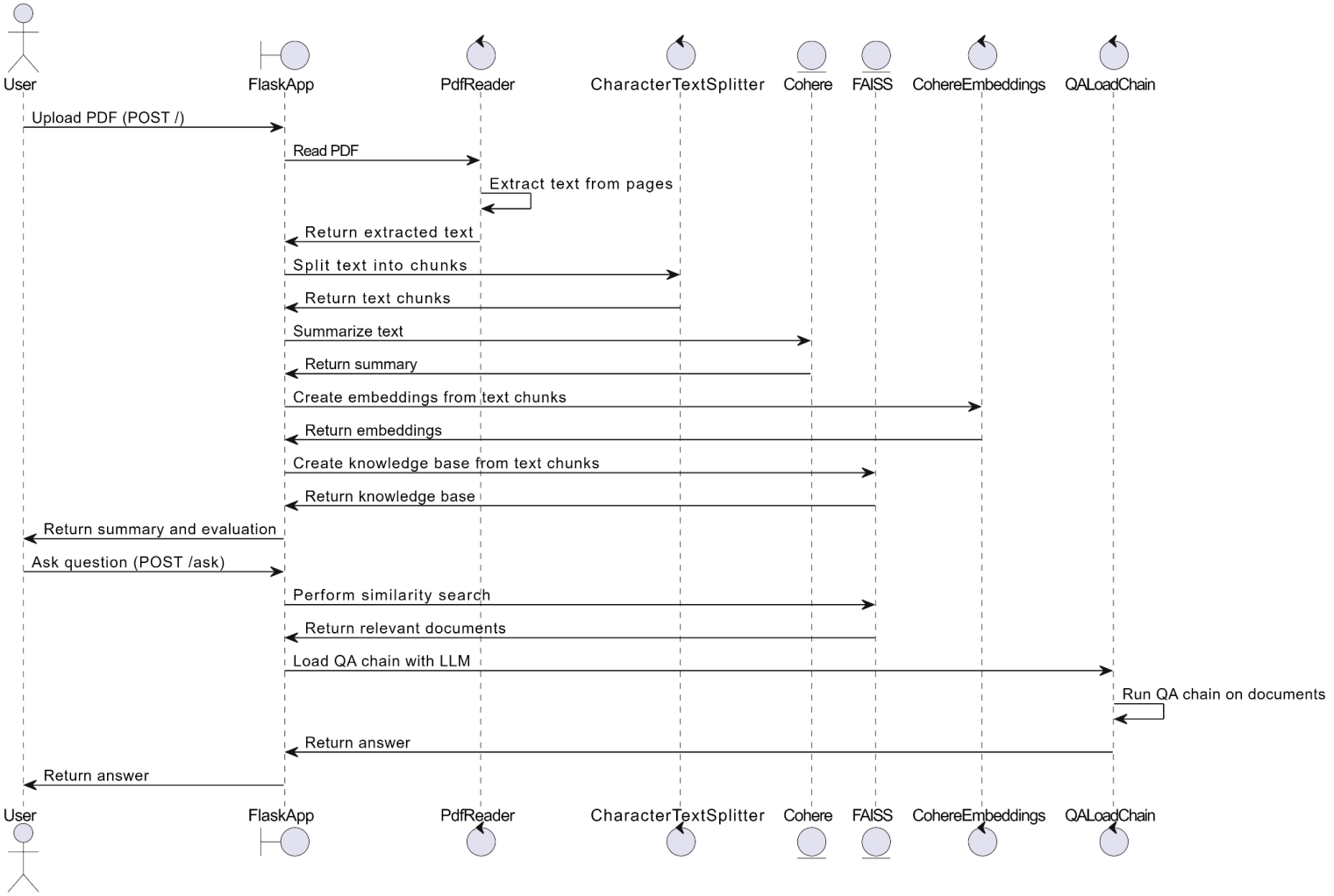
```
if __name__ == '__main__':
    app.run(debug=True)
```

QA SYSTEM WORKFLOW





SEQUENCE DIAGRAM



CLASS DIAGRAM