Backend Internship Assignment

Objective:
Develop a backend service that allows users to upload PDF documents and ask questions about the content of these documents. The backend will process and analyze these documents using natural language processing (NLP) to provide answers to users' questions. The question-answering functionality will utilize WebSocket communication to allow real-time responses.

Tools and Technologies

- **Backend Framework**: FastAPI with WebSocket support
- **NLP Processing**: LangChain/LLamaIndex
- **Database**: SQLite or PostgreSQL (for storing document metadata)
- **File Storage**: Local filesystem or cloud storage (e.g., AWS S3) for storing uploaded PDFs
- **Testing**: Pytest or Unittest (for API endpoint testing)

Functional Requirements

1. **PDF Upload**:
   - Users can upload PDF documents through an API endpoint.
   - The application should store the PDF and extract text content for further processing.
2. **Real-Time Question Answering**:
   - Users can connect to a WebSocket endpoint to ask questions related to the content of an uploaded PDF.
   - The system should retrieve relevant content from the PDF and use an NLP model to generate real-time responses.
   - Support for follow-up questions on the same document should be implemented, maintaining session-based context.
3. **Data Management**:
   - Metadata about each uploaded document, such as filename and upload date, should be stored in a database.
   - Text content from PDFs should be stored in a form suitable for retrieval and NLP processing.

4. **Rate Limiting**:
    - Implement rate limiting for API endpoints and WebSocket messages to manage server load and prevent abuse.

Backend Specification

1. **FastAPI Endpoints**:
    - **PDF Upload Endpoint**: Accepts PDF uploads and extracts text content for storage and processing.
    - **WebSocket Question Answering Endpoint**:
        - Allows users to connect via WebSocket to ask questions in real-time.
        - Responds with answers based on relevant content from the uploaded PDF.
        - Maintains session-based context for follow-up questions.
2. **PDF Processing**:
    - Use a library like **PyMuPDF** to extract text from uploaded PDFs.
    - Store extracted text in the database or file storage system.
3. **NLP Processing**:
    - Integrate LangChain or LlamaIndex to process questions and generate answers based on the PDF content.
    - Implement logic to support follow-up questions within the WebSocket session.
4. **Rate Limiting**:
    - Use FastAPI's `Limiter` middleware or Redis-based rate limiting to limit requests and WebSocket messages per user, especially for the question-answering endpoint.
5. **Data Storage**:
    - Store extracted text and metadata (e.g., filename, upload date) in a database like SQLite or PostgreSQL.
    - Use file storage (e.g., local filesystem or S3) to store original PDF documents.

Testing Requirements

1. **Endpoint Testing Script**:
    - Create a Python testing script using **pytest** or **unittest** to test all API endpoints and WebSocket functionality.
    - Include tests for:
        - PDF upload functionality
        - WebSocket connection and message exchange for real-time question answering
        - Error handling for unsupported files or malformed requests

        - Rate limiter functionality to restrict excessive requests or WebSocket messages

2. **Rate Limiting Tests**:
    - Verify that the rate limiting returns the appropriate error message or response when the limit is exceeded.
3. **WebSocket Testing**:
    - Write test cases to verify real-time responses and session-based follow-up questions.

Assignment Deliverables

1. **Source Code**:
    - Provide the source code for the backend service, structured and commented appropriately.
2. **Documentation**:
    - Include a `README` file with setup instructions, API documentation, and an architectural overview.
3. **Testing Script**:
    - A comprehensive testing script to verify endpoint functionality, WebSocket message handling, error handling, and rate limiting
4. **Deployed Backend Link**:
    - Provide a link to the deployed backend service so that the endpoints can be tested live.
5. **Demo**:
    - Provide a screencast showcasing the API's functionality, including PDF upload, WebSocket question answering, and rate limiting in action.

Evaluation Criteria

1. **Functionality**: Completeness of API endpoints, WebSocket connection, document processing, and question-answering functionality.
2. **Code Quality**: Organization, clarity, and use of comments.
3. **Testing**: Comprehensive test coverage, especially for WebSocket functionality and rate limiting.
4. **Innovation**: Any additional features that improve usability, performance, or extensibility.