Mini RAG System with WebSocket Streaming (Laravel)

Objective

The goal of this assignment is to evaluate your ability to design and implement a **real-world backend system** using **Laravel**, with a focus on **authentication, file handling, WebSockets, and AI integration**.

You will build a **Mini Retrieval-Augmented Generation (RAG) system** that allows authenticated users to upload PDF files and chat with an LLM that answers questions based on the uploaded content, with **real-time streamed responses**.

Functional Requirements

1. Authentication

- Implement authentication using Laravel (**Sanctum or Passport preferred**).
- Provide a **login API endpoint**.
- All REST APIs and WebSocket connections **must be protected**.
- Any unauthenticated attempt to connect to the WebSocket must be **rejected immediately**.

2. PDF Upload & Indexing

- Create a versioned endpoint: /api/v1/pdf/upload
- Validate uploaded files:
    - PDF files only
    - Reasonable file size limit
    - Reject empty or corrupted PDFs

- Extract text from the PDF.
- Chunk and preprocess the text for retrieval.
- Store embeddings in a **vector database** (FAISS, Chroma, Qdrant, or similar).
- Ensure all uploaded content is **scoped to the authenticated user**.

---

3. WebSocket Chat (RAG + LLM)

- Implement a **dedicated WebSocket endpoint** for chat.
- Only authenticated users can connect.
- After connection:
    - Receive user queries
    - Retrieve relevant context from the indexed PDF data
    - Send (query + context) to an LLM
    - **Stream the response** back to the client in real time via WebSocket

You may use any LLM provider (OpenAI, HuggingFace, or local models).

Technical Constraints

- Unauthenticated users must never access the WebSocket.
- Unauthorized connection attempts must:
  - Be rejected instantly
  - Be logged

- All API errors must return:
  - Clear messages
  - Structured JSON responses

Documentation (Required)

The repository **must include a README.md** covering:

- System architecture and design decisions
- End-to-end data flow
- Local setup and run instructions
- How to:
  - Authenticate
  - Upload a PDF
  - Connect to the WebSocket and chat
- Required environment variables and API keys
- Dependencies and libraries used
- A working example or sample flow

Engineering Guidelines

- Use **API versioning** (/api/v1/...)
- Follow **Clean Code and SOLID principles**
- Write modular, extensible, and readable code
- Follow RESTful API best practices
- Handle edge cases properly:
    - Invalid or empty PDFs
    - Empty queries
    - Unauthorized access
- Add **basic logging** for easier debugging

Submission Instructions

- Submit the assignment as a **GitHub repository**
- The repository must contain:
    - A complete, runnable Laravel project
    - Clear documentation
    - Example usage
- Provide full repository access for review

Evaluation Criteria

Candidates will be evaluated based on:

- Code quality and structure

- Security and authentication handling
- Correctness of RAG implementation
- WebSocket streaming implementation
- Error handling and logging
- Clarity of documentation