Full-Stack Developer Assignment

Objective:

Develop a secure, scalable, full-stack application that allows users to upload, store, and interact with any type of documents (pdf, ppt, csv etc.) through advanced natural language processing (NLP) and implement a RAG Agent to do the querying for any question the user has. The application should support document management, user authentication, and efficient RAG agents, and utilize unstructured.io for efficient parsing of document content.

Tools and Technologies:

- Backend: FastAPI
- NLP Processing: LangChain/LLamaIndex
- Agents: Autogen/Crewai (or any)
- Frontend: React.js
- Database: PostgreSQL, Redis
- File Storage: AWS S3 or any other equivalent
- Document Parsing: unstructured.io for advanced document parsing and content extraction
- Authentication: Session based authentication (or any alternatives like Oauth2.0 or JWT)
- Search Engine: Elasticsearch
- Deployment: Docker, Kubernetes

Functional Requirements:

LLD Design:

- DB Schema
    - Tables structure
    - Foreign Keys
    - Normalization of data
- Classes
- Functions
- Attributes of a class
- Interaction between two or more classes.
- OPEN-CLOSE Relationships
- Handling of Dependencies

1. Document Upload and Management

- Objective: Provide a secure and efficient system for users to upload, store, and manage documents of various formats (PDF, PPT, CSV, etc.).
- Technologies:
    - File Storage: AWS S3 for document storage, ensuring durability and accessibility.

        - Document Parsing: Use unstructured.io for extracting text and metadata, which will feed into the NLP processing module.
- Features:
    - Multi-format support with real-time parsing and storage.
    - Metadata extraction for advanced document categorization and retrieval.

2. Advanced NLP Features with RAG Agents

- Objective: Integrate RAG (Retrieve and Generate) agents to provide accurate, context-aware answers to user queries based on the document contents stored within the system.
- Technologies:
    - NLP Processing & Indexing: Utilize LangChain/LLamaIndex for efficient document indexing and search capabilities within the NLP framework.
    - Query Agents: Implement Autogen/Crewai or equivalent technologies for generating responses to user inquiries by retrieving relevant information from the document database.
- Features:
    - Contextual query handling to generate precise answers from the document content.
    - Scalable NLP processing to support complex query interpretations and responses.

Non-Functional Requirements:

- Usability: User friendly UI/UX.
- Performance: Scalable and efficient NLP processing.
- Security: Secure data handling and user authentication.
- Scalability and Reliability: Design for scalability, high availability, and fault tolerance.

Deployment Specification:

- Containerization: Dockerize the frontend, backend, and any other microservice. Provide Dockerfiles and instructions for building images.
- Kubernetes Deployment(Optional): Create Kubernetes manifests or Helm charts for deploying the application components on a Kubernetes cluster. Include instructions for setting up the deployment on a local Kubernetes cluster (e.g., minikube) **or** a cloud-based Kubernetes service (e.g., AWS EKS, Google GKE).

Monitoring (Optional):

- Monitoring: Set up Prometheus for collecting metrics and Grafana for dashboard visualization. Include Kubernetes manifests or configuration for deploying these tools alongside the application. Ensure that key metrics from the application, database, and any middleware are being monitored.

- Logging: Configure the application to emit logs in a structured format. Set up the ELK Stack for log aggregation and visualization. Include setup instructions or configuration as part of the deployment process.

Assignment Deliverables:

- Source Code: Well-commented source code, following best practices.
- Documentation: README with setup instructions, API documentation, architecture diagrams, and deployment guide.
- Demo: A live demo or screencast showcasing application functionality and deployment steps.
- Low Level Design Diagram

Evaluation Criteria:

- System Design: Scalable, secure, and maintainable architecture.
- Code Quality: Clean and efficient code.
- Deployment Proficiency: Successful containerization and deployment on Kubernetes with monitoring and logging.
- Innovation: Novel features or technologies enhancing application value.
- Easy to add features without modifying existing code.

Additional Requirements:

- Document Parsing: Demonstrate the integration and use of unstructured.io for advanced parsing capabilities, handling a variety of document formats beyond PDFs.
- Deployment: Detailed steps for deploying the application in a containerized environment using Kubernetes, including scaling, monitoring, and logging setup.