1. Components.

The system is divided into the following logical component blocks:

1.1 Presentation Layer.

- User Interface (UI): Utilizes a UI library/framework to display web pages based on the design, manage UI states, and handle user interactions.

1.2 Application & Business Logic Layer.

- Application Server: Utilizes an application server framework to handle requests and execute business logic. The entire server is deployed on a serverless deployment platform and processes requests via the platform's serverless functions.

1.3 AI Service Layer.

- AI Service (LLM Service): A third-party Large Language Model (LLM) service to provide natural language generation and embedding creation (data vectorization).
- Data Ingestion Pipeline: A process (typically a Serverless Function) to automatically process internal documents when they are uploaded.

1.4 Data & Storage Layer.

- Database: A NoSQL database is used to store application data (e.g., chat history, user information).
- File Storage: A cloud file storage service to store original internal documents (PDF, DOCX...) uploaded by administrators.
- Vector Database: A vector database solution to store document "embeddings" for Retrieval-Augmented Generation (RAG).

1.5 Support & Security Layer.

- Authentication Service: An authentication service is used to manage access and user authentication.

2. Relationships & Data Flows

2.1 Basic Communication Flow.

- Client ⇔ Server: The website (Client) communicates with the application server via restful API calls. After the server loads data and handles logic, it returns the rendered web pages to the client.
- Server ⇔ Database/Services: The application server uses the cloud provider's SDKs to connect directly to the NoSQL database and authentication service.
- Security Constraint: The client never connects directly to the cloud provider's services.

2.2 Data Ingestion Flow (Admin).

· An administrator uploads a document via the UI.

· The UI sends this file to File Storage.

· The file upload triggers the Data Ingestion Pipeline.

· This pipeline: a. Retrieves the file from File Storage. b. Splits the file into small chunks. c. Calls the AI Service to create "embeddings" (vectors) for each chunk. d. Stores these vectors in the Vector Database.

2.3 Chatbot Query Flow (User).

· A user submits a query via the UI.

· The UI sends the query to the Application Server.

· The server calls the AI Service to create an embedding for the user's query.

· The server uses this embedding to query the Vector Database and find the most relevant internal document chunks.

· The server constructs a complete prompt (including the original query + relevant document chunks).

· The server sends this complete prompt to the AI Service.

· The AI Service returns the final answer.

· The server returns this answer to the UI for display.

3. Principles & Constraints.

- Serverless Architecture (FaaS): The entire system requires no physical server management and is deployed on a serverless platform and cloud service provider.
- Security by Layer: All sensitive data and business logic are handled by Server Components. The client (Client Component) is never exposed to API keys or database credentials.
- Technology Stack: The system is JavaScript & Typescript based for both backend and frontend logic.
- Data Privacy Constraints: As this is internal data, strict rules on data residency and ensuring third-party AI APIs do not use this data for their own training are required.
- Performance Constraints: Serverless functions may experience "Cold Starts," which can affect the chatbot's initial response time.
- Cost Constraints: The "Pay-as-you-go" model of the serverless platform and AI APIs requires close cost monitoring to stay within budget.