**High-Level Design**

**1. Architecture Overview**

* **Frontend**: React (with Tailwind CSS for styling) to create user interfaces for different personas (Admin, User, Super Admin).
* **Backend**: Node.js and Express to handle API requests and server-side logic.
* **Database**: MongoDB to store user data, documents, rules, and activity logs.
* **Cloud Services (Azure)**:
* **Azure Blob Storage** for document storage.
* **Azure Cognitive Services** for document analysis (optional, e.g., content scanning).
* **Azure Active Directory (AAD)** for user authentication and role-based access control.
* **Azure Functions** for background processing, such as running sensitive content checks.
* **Middleware**: Authentication and authorization middleware based on user roles.

**2. Core Components**

* **User Interfaces**:
  + **Admin Panel**: Allows admins to define content rules, view flagged documents, and manage user permissions.
  + **User Dashboard**: Enables users to create, view, edit, share, and check documents against DLP rules.
  + **Super Admin Interface**: Used for adding sensitive content types and performing overall system configuration.
* **Document Management**:
  + CRUD operations for documents.
  + Flagging functionality if document content violates admin-defined rules.
* **DLP Rule Management**:
  + Admin-defined rules to scan documents for sensitive information (e.g., regex matching for credit card patterns).
* **Metrics and Analytics**:
  + Real-time metrics on documents, users, and rule violations.

**3. High-Level API Design**

* **Auth & User Management**: User registration ,User login, Role assignment and verification.
* **Document Management**: Create, read, update, delete and share documents with others.
* **Rule Management**: Provide CRUD operations for document for managing and compliance with DLP rules.
* **Analytics**: Fetch metrics for Super Admins.

**4. Data Models**

**User:**

* userId: String (primary key)
* username, email, password: String
* role: Enum (Admin, User, Super Admin

**Document:**

* docId: String (primary key)
* content: Text or Blob
* owner: Reference to a User
* sharedWith: Array of User references
* status: Enum (Compliant, Non-compliant)

**Rule:**

* ruleId: String (primary key)
* description: String
* pattern: Regex for detecting sensitive content
* action: Enum (Block, Flag))

**5. Workflow & Use Cases**

* **Admin Defined Rules**:
  + Admins can create rules with specific patterns (e.g., regex for credit card detection).
  + Rules are stored and automatically applied to relevant documents upon creation or sharing.
* **Document Creation & Validation**:
  + When a user creates or edits a document, the backend checks the content against admin-defined DLP rules.
  + If the document contains sensitive content, it is flagged, and sharing is restricted based on the rule's action.
* **Real-time Monitoring & Reporting**:
  + Super Admins can monitor metrics and system status, such as the count of flagged documents or blocked shares.

**6. Implementation on Azure**

* **Azure Blob Storage** for document storage, with access restrictions based on compliance status.
* **Azure Functions** for asynchronous processing, such as checking documents for sensitive data.
* **Azure Cognitive Services** (optional) for advanced text analysis, such as named entity recognition.
* **Azure Active Directory** for identity and access management, handling user authentication and permissions.

**7. Security and Compliance**

* **Encryption**: Encrypt sensitive data both at rest and in transit (API calls).
* **Role-based Access Control**: Ensure only Admins and Super Admins can define rules or manage user roles.
* **Data Compliance**: Regular audits and logs to monitor rule adherence and document compliance.

**8. Deployment and DevOps**

* **CI/CD Pipeline** using GitHub Actions for automated builds and deployments to Azure.
* **Monitoring and Alerts**: Azure Monitor for real-time alerts on rule violations or system issues.