Low-Level Design (LLD)

1. Frontend

- Key Interfaces:
    - Admin Panel UI: Tailored for administrators with interactive forms for rule creation, visualization of flagged documents, and permission controls.
    - User Dashboard UI: Document upload/download, content preview, compliance checks, and sharing features.
    - Super Admin Interface: Advanced features like real-time analytics dashboards and overall rule settings.
- Data Models:
    - UI State Management:
        - Use `Redux` or `Context API` for shared states (e.g., user authentication, role definitions).
        - Store flags for document compliance results locally for smooth UX.
- Non-functional Requirements:
    - Lightweight UI libraries (`Tailwind CSS`) for consistent, responsive, and efficient styling.
    - Optimal frontend loading time: < 2 seconds.
    - Memory optimization: Lazy loading React components.

2. Backend (Node.js with Express)

- Key Classes/Modules:
    - AuthService:
        - Methods: `signUp`, `login`, `validateRole`.
    - DocumentService:
        - Methods: `create`, `read`, `update`, `delete`, `checkCompliance`.
    - RuleService:
        - Methods: `addRule`, `updateRule`, `deleteRule`, `checkPattern`.
    - AnalyticsService:
        - Methods: `getMetrics`.
- Key Data Models:

User:

```
 const User = {
  userId: String,
  username: String,
  email: String,
  password: String,
  role: { type: String, enum: ['Admin', 'User', 'Super Admin'] }
};
```

    -

Document:

```
const Document = {
 docId: String,
 content: Buffer, // Blob data
 owner: String, // userId reference
 sharedWith: [String], // userId array
 status: { type: String, enum: ['Compliant', 'Non-compliant'] }
};
```

Rule:

```
const Rule = {
 ruleId: String,
 description: String,
 pattern: String, // Regex
 action: { type: String, enum: ['Block', 'Flag'] }
};
```

- Non-functional Requirements:
  - Ensure backend latency < 200ms.
  - Memory usage capped under 512 MB per instance.
  - Use libraries such as `Mongoose`, `Regex`, and `jsonwebtoken`.

3. Database (MongoDB)

- Structure:
  - Collections: `users`, `documents`, `rules`, `activityLogs`.
  - Optimize indexes for frequent queries (e.g., `docId`, `userId`).
- Non-functional Requirements:
  - Query response time < 50ms.
  - NoSQL schema flexibility for scaling rules.

4. Cloud Services (Azure Implementation)

- Key Components:
  - Azure Blob Storage:
    - Interfaces: Upload/download documents via APIs.
    - Secure access using Shared Access Signatures (SAS).
  - Azure Functions:
    - Perform compliance checks asynchronously on document upload.
  - Azure Active Directory:
    - Authentication for users with JWT tokens.
  - Azure Cognitive Services:

- Optional advanced features like Named Entity Recognition (NER).

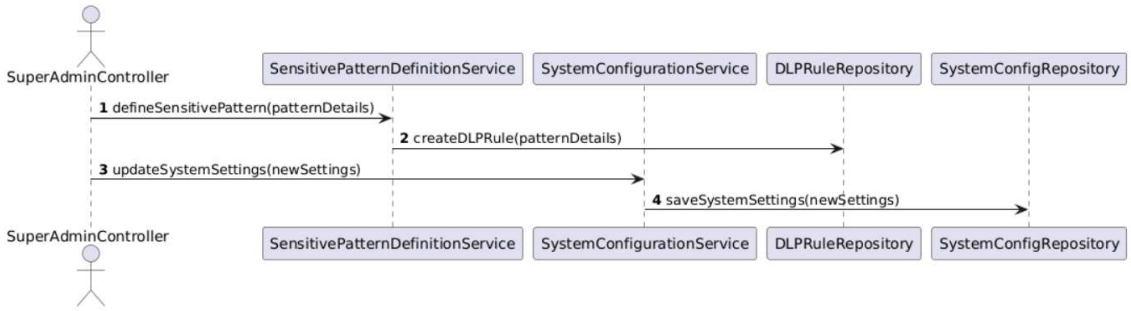
5. Middleware

- Authentication:
  - Verify tokens in headers for each request.
- Authorization:
  - Check user roles for admin/super admin privileges.
- Performance:
  - Cache results of compliance checks using `Redis`.

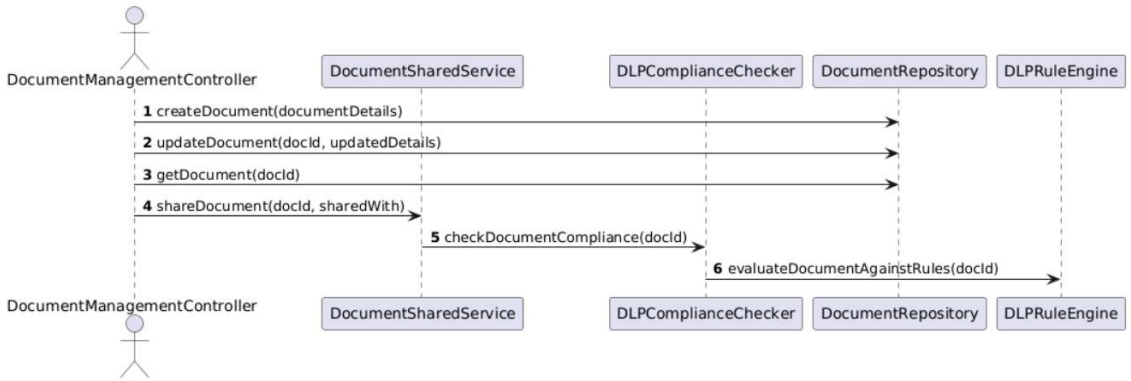
6. Deployment and DevOps

- CI/CD:
  - Use GitHub Actions for pipeline automation (build → test → deploy).
- Real-time Monitoring:
  - Azure Monitor for flagged anomalies.

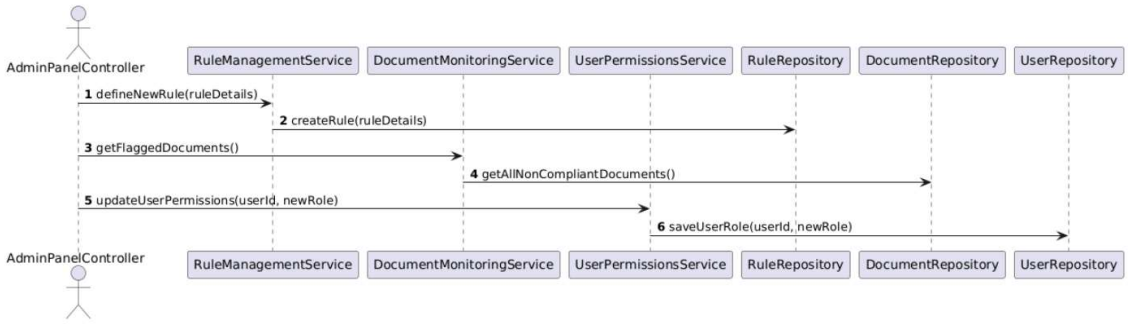
a.User Interfaces: a. **Admin Panel**: Sequence Diagram:



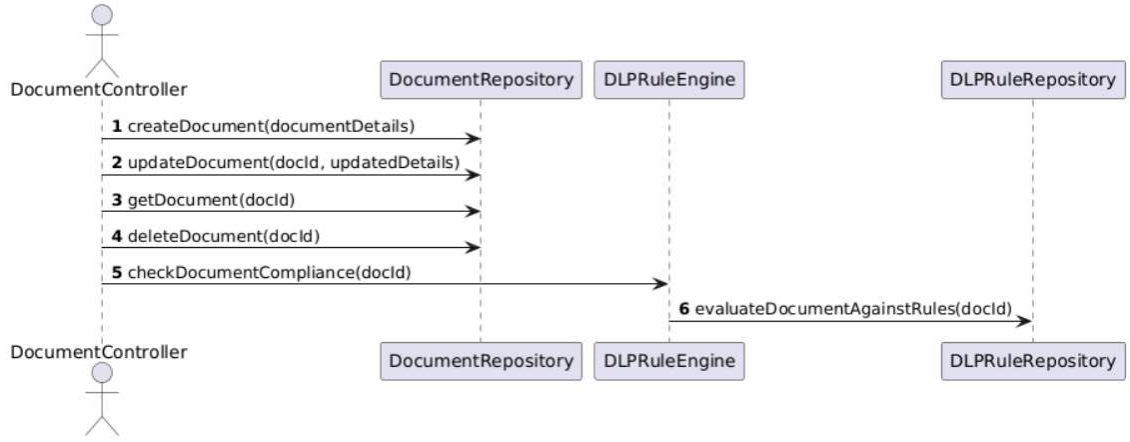
b. **User Dashboard**: Sequence Diagram:



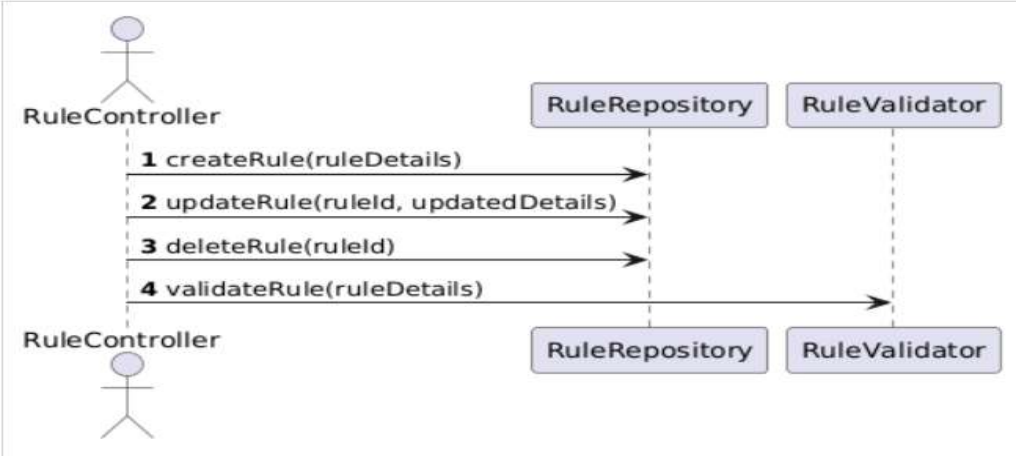
c. **Super Admin Interface**: Sequence Diagram:



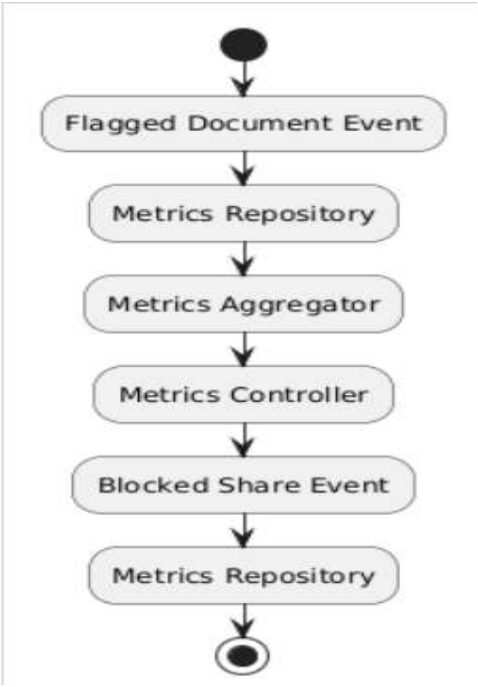
2.Document Management: Sequence Diagram:



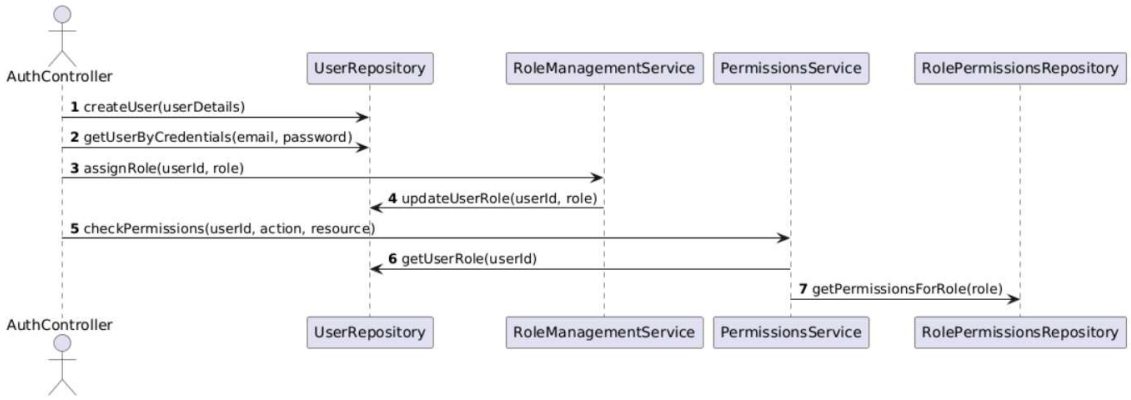
3. **DLP Rule Management**: Sequence Diagram



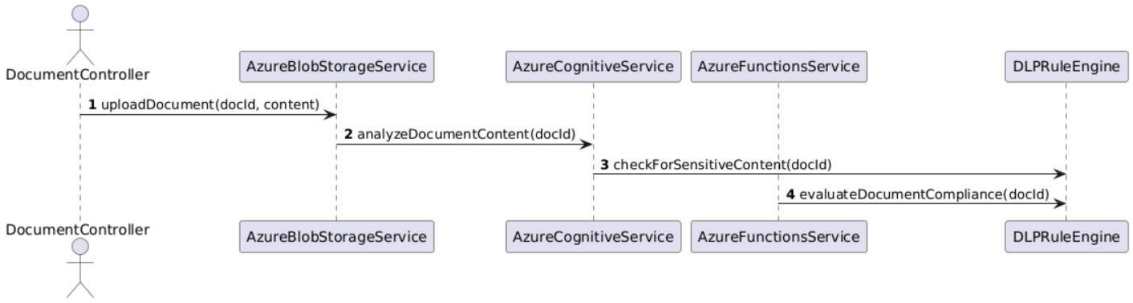
4.**Metrics and Analytics**: Flow Chart:



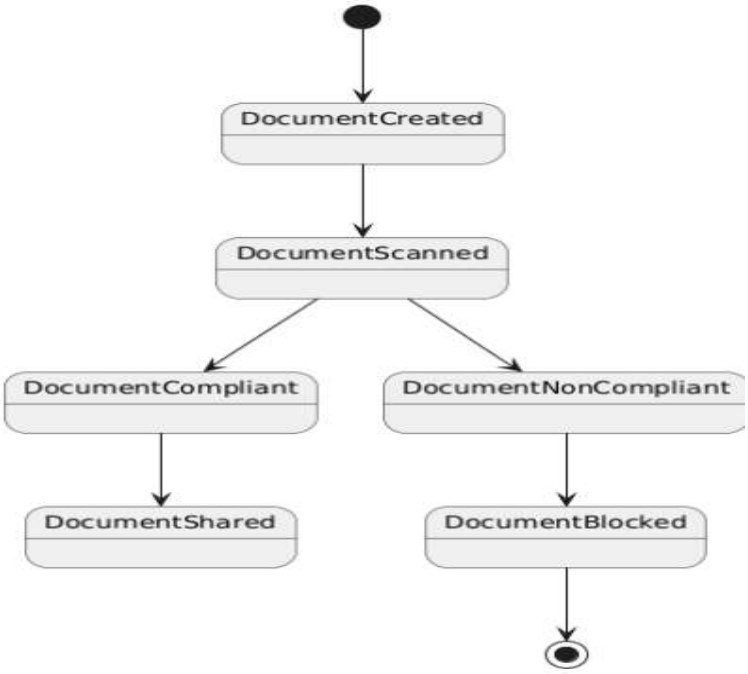
5.**Authentication and Authorization**: Sequence Diagram:



6. Azure Integration: Sequence Diagram:



7.Security and Compliance: State Machine Diagram:



8. **Deployment and DevOps**: Flow Chart: