# Assignment: Document Management System with Microservices and Chunked File Upload/Download

## Objective

Develop a Document Management System (DMS) using a microservices-based architecture with Java (Spring Boot) in the backend, Angular in the frontend, and MySQL as the database. The system should support CRUD operations on document metadata, and allow large file upload and download using chunked streaming.

## System Overview

The application will enable users to:  
1. Create, Read, Update, and Delete (CRUD) document metadata.  
2. Upload large files (e.g., PDFs, images, videos) in chunks to handle size limitations.  
3. Download files in chunks to ensure efficient streaming for large files.  
4. Search and filter documents based on metadata fields (title, description, upload date, etc.).  
5. View file upload progress in the Angular UI.

## Core Requirements

### 1. Backend

Architecture:  
- Use Spring Boot to implement microservices:  
 • Document Metadata Service (CRUD for document info)  
 • File Storage Service (handle chunk upload/download)  
 • Gateway Service (API gateway, routing requests to microservices)  
 • Discovery Service (Eureka/Consul for service registry)  
  
Document Metadata Service:  
- Fields: id, title, description, fileName, fileSize, uploadDate, uploadedBy  
- Endpoints:  
 • Create metadata (POST)  
 • Get metadata (GET by ID, all, with search & filter)  
 • Update metadata (PUT)  
 • Delete metadata (DELETE)  
  
File Storage Service:  
- Upload in chunks:  
 • Endpoint to upload file chunks (POST /upload-chunk)  
 • Endpoint to finalize upload after all chunks are received (POST /finalize-upload)  
- Download in chunks:  
 • Endpoint to download a specific chunk (GET /download-chunk/{fileId}/{chunkNumber})  
- Store files in local storage or cloud (AWS S3, etc. — optional)  
  
MySQL Database:  
- Store document metadata  
- Maintain file chunk information if needed for recovery  
  
Communication:  
- Services should communicate via REST (can extend to use message queues for extra credit)

### 2. Frontend (Angular)

Pages:  
- Document List:  
 • Display all documents with pagination and search filters  
 • Buttons for edit, delete, view  
- Document Upload:  
 • Form for metadata (title, description, uploadedBy)  
 • File selector with chunk upload progress bar  
- Document Detail:  
 • Show metadata and download button  
- Edit Document:  
 • Update metadata form  
  
Features:  
- Implement chunked file upload with progress bar  
- Handle chunked download and save to user’s device  
- Show success/failure notifications

## Additional Requirements

- Security:  
 • Basic authentication or JWT-based authentication for all API endpoints.  
- Error Handling:  
 • Return meaningful error messages for failed uploads, missing files, etc.  
- Validation:  
 • Validate metadata fields (title not empty, file size limits, etc.)  
- Performance:  
 • Ensure chunk size is configurable (e.g., 1MB per chunk).  
- Testing:  
 • Unit tests for microservices (JUnit + Mockito)  
 • API testing with Postman

## Bonus Features (Optional)

- Resume interrupted file uploads  
- File preview in Angular (for images/PDFs)  
- Tagging system for documents  
- Role-based access control (admin/user)

## Deliverables

1. Backend:  
 - Java Spring Boot code for microservices  
 - API documentation (Swagger/OpenAPI)  
 - MySQL schema script  
  
2. Frontend:  
 - Angular application code  
 - Screenshots of main pages  
  
3. Deployment Instructions:  
 - Steps to run services locally (Docker Compose is a plus)  
 - API base URLs and test credentials  
  
4. Report:  
 - Architecture diagram  
 - Chunk upload/download mechanism explanation  
 - Known limitations & future improvements