Digital Locker Systems to view the details

Name : K.Yuvasri

Register Number: 727723euai134

Class: AI&DS - B

Year: III

Project Overview

- **What it is:** A centralized, cloud-based platform where users can digitally store, manage, and share critical documents securely.
- **Purpose**: To replace traditional physical document management with a secure, reliable, and easily accessible digital solution.
- **Key Features**:
    - Secure authentication (JWT, OAuth2, 2FA).
    - Encrypted document storage (at rest & in transit).
    - Easy upload, search, and retrieval of documents.
    - Role-based access control and dashboards .
    - Full audit trail of all activities (upload, view, revoke).
- **User Benefits**:
    - Eliminates the need to carry or manage physical copies.
    - Saves time in verification processes (education, employment, government services).
    - Reduces risk of document loss, tampering, or unauthorized access.
    - Provides 24/7 access from any device, anywhere.
- **Organizational Benefits**:
    - Simplifies verification workflows for institutions and employers.
    - Enhances trust with digital, verifiable document copies.
    - Reduces administrative costs and manual paperwork.

Proposed System

Centralized Digital Locker System

- Integrates document storage, viewing, sharing, metadata management, and audit trails in one platform.

Detailed Document Model:

- Document categories (ID proofs, certificates, contracts, personal documents).
- Metadata includes title, tags, owner info, upload date, and expiry.
- Version control and document history.

User Engagement:

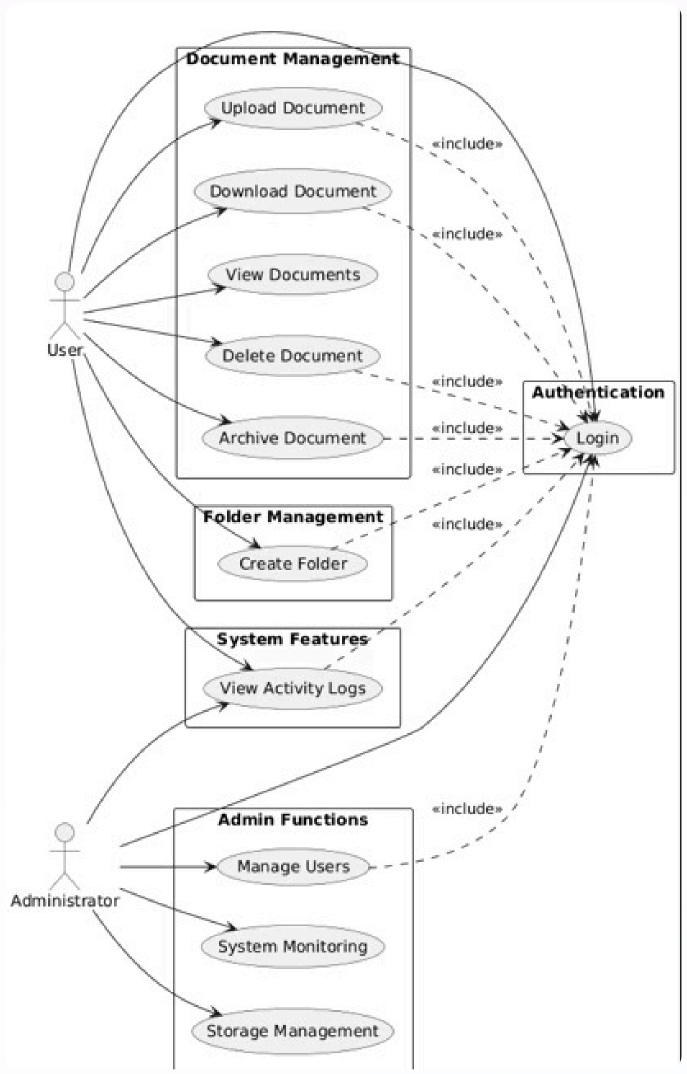
- Upload, edit, and delete documents.
- View documents securely within the system.
- Share documents with time-bound or revocable access links.
- Search and filter by tags, type, or date.

Community & Collaboration Features:

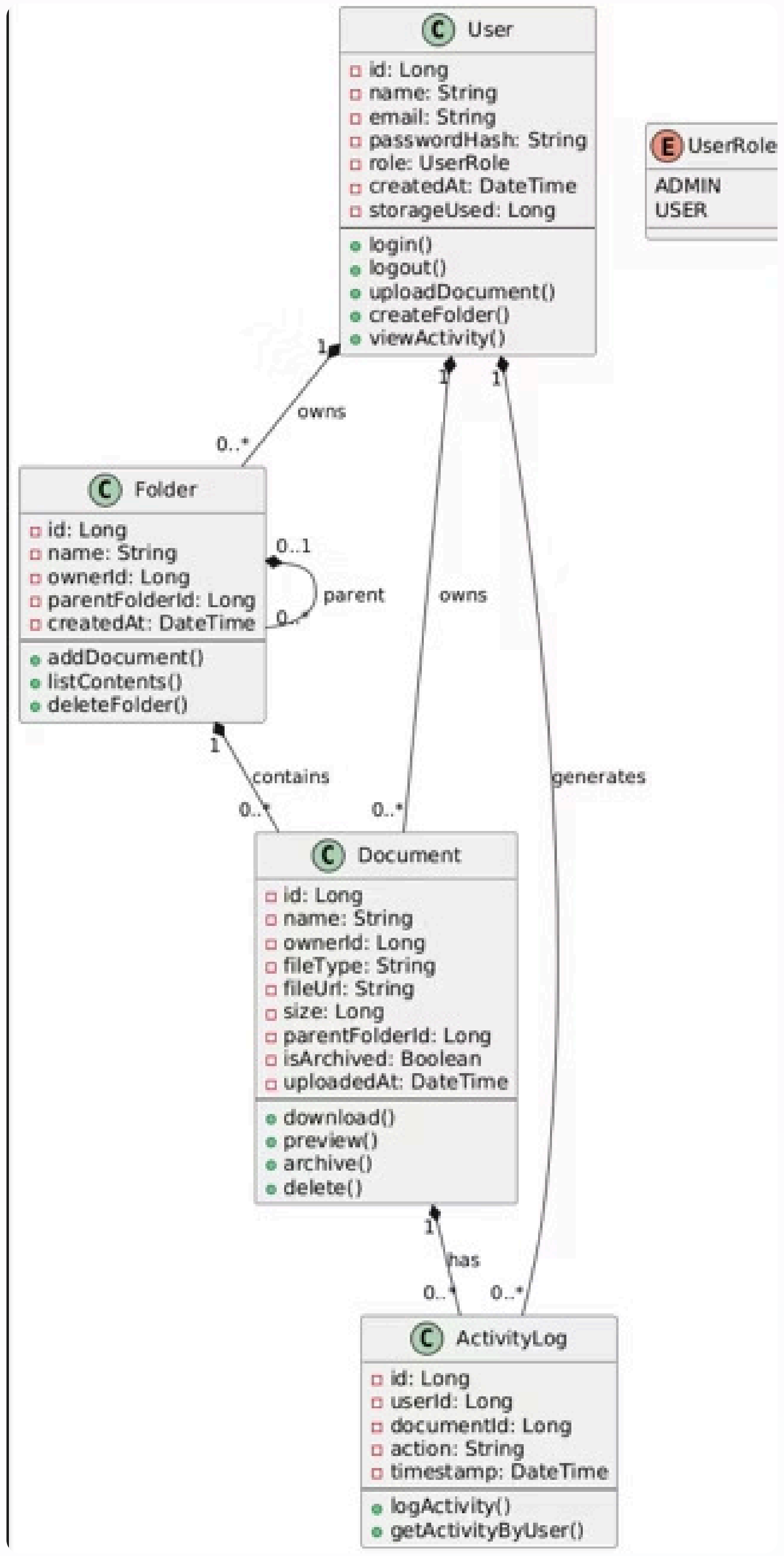
- Optional collaboration with verified third-party verifiers (employers, institutions).
- Consent-based document sharing for specific users or organizations.
- Audit logs visible to owner for tracking document activity.

UML Diagrams

1.Use Case Diagram:

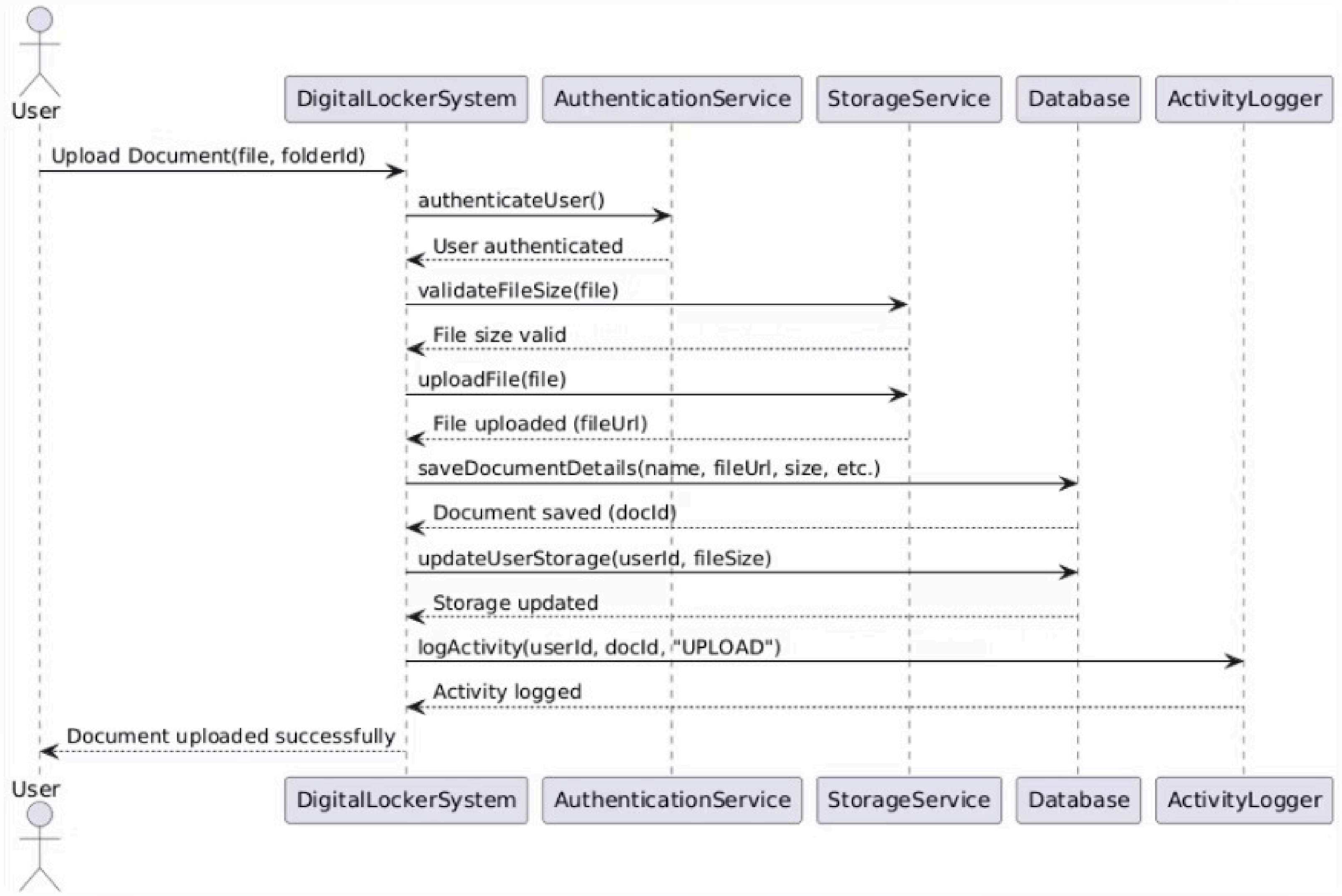


2.Entity Relationship Diagram:

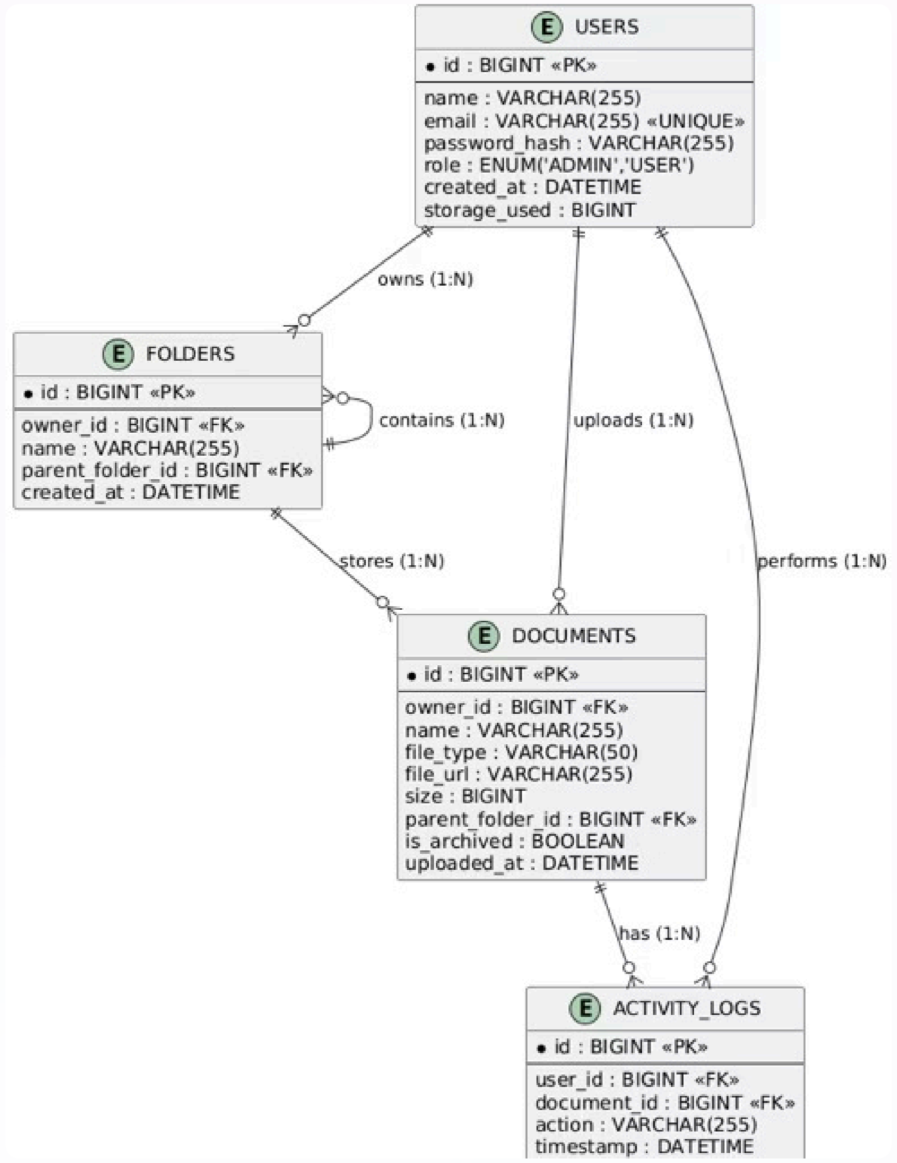


Made with GAMMA

3.Sequence Diagram:



4.Class Diagram:



Api Documentation

Authentication APIs:

	Endpoint	Method	Description	Request Body	Response
1	/auth/login	POST	Authenticate user and get access tokens	{ "email": "user@example.com", "password": "password123" }	{ "accessToken": "...", "refreshToken": "...", "accessTokenExpiresAt": "..." }
2	/auth/refresh	POST	Get new access token using refresh token	{ "refreshToken": "refresh_token_string" }	{ "accessToken": "new_access_token", "refreshToken": "new_refresh_token" }
3	/auth/logout	POST	Revoke refresh tokens	{ "refreshToken": "refresh_token_string", "revokeAll": false }	{ "status": "ok" }
4	/auth/me	GET	Get current authenticated user info	**Header:** Authorization: Bearer <token>	{ "email": "user@example.com", "role": "USER", "name": "John Doe" }

User Management APIs:

	Endpoint	Method	Description	Request Body	Response
1	/users/signup	POST	Register new user	{ "name": "John Doe", "email": "john@example.com", "password": "password123", "role": "USER" }	User object
2	/users	GET	List all users	–	Array of user objects
3	/users/{id}	GET	Get specific user details	–	User object
4	/users/{id}	PUT	Update user information	User object	Updated user object
5	/users/{id}	DELETE	Delete user account	–	204 No Content

Document Management APIs:

	Endpoint	Method	Description	Request Body / Params	Response
1	/documents/upload	POST	Upload file to user's locker	Multipart: file, fileName, fileType, folderId?	{ "id": 1, "name": "document.pdf", "fileType": "PDF", "fileSize": 1024000, "uploadDate": "...", "folderId": 1 }
2	/documents	GET	Get all documents for authenticated user	–	Array of document objects
3	/documents/{id}	GET	Get specific document details	–	Document object
4	/documents/{id}/download	GET	Download document file	–	Binary file stream
5	/documents/{id}	DELETE	Delete document	–	204 No Content

Folder Management APIs:

	Endpoint	Method	Description	Request Body	Response
1	/folders	POST	Create new folder	{ "name": "My Documents", "parentId": null }	{ "id": 1, "name": "My Documents", "parentId": null, "createdAt": "..." }
2	/folders	GET	Get all folders for authenticated user	–	Array of folder objects
3	/folders/{id}	GET	Get specific folder details	–	Folder object
4	/folders/{id}	DELETE	Delete folder and its contents	–	204 No Content

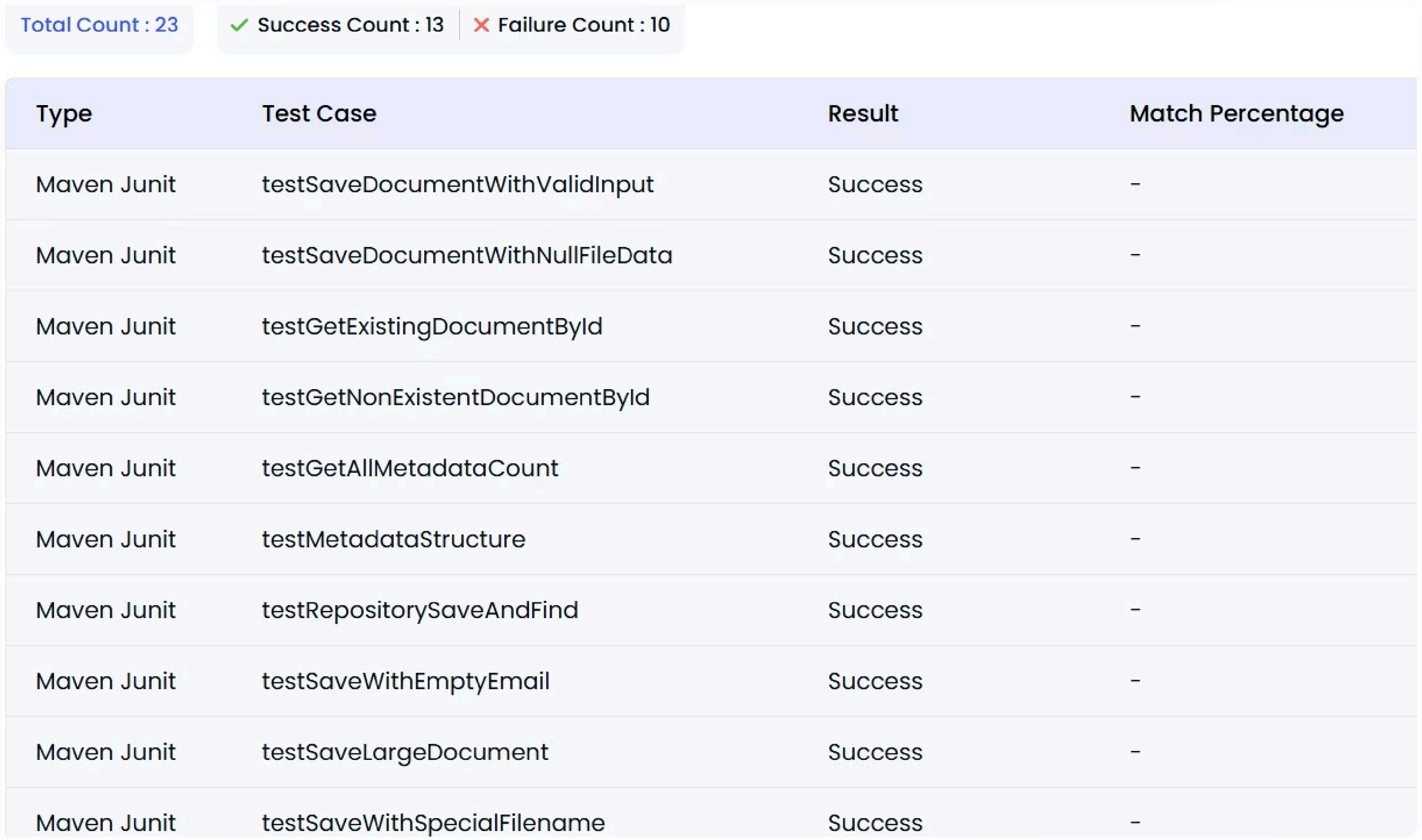
Activity Log APIs:

	Endpoint	Method	Description	Request Body	Response
1	/activity/record	POST	Log user activity	{ "action": "UPLOAD", "details": { "fileName": "document.pdf", "fileSize": "1.2 MB" } }	Activity log object
2	/activity	GET	Get user's activity logs	–	[{ "id": 1, "action": "UPLOAD", "details": {...}, "timestamp": "..." }]

Implementation Details

- **Techstack:**
  - Backend: Spring Boot (Java 17)
  - Database: MySQL 8.0
  - Frontend: React js
  - Storage: AWS S3 for documents
- **Layers & Architecture:**
  a. Presentation Layer: React components, responsive UI.
  b. Controller Layer: REST APIs handling HTTP requests.
  c. Service Layer: Business logic for document management, sharing, and auditing.
  d. Repository Layer: JPA/Hibernate interfaces connecting to MySQL.
  e. Storage Layer: Object storage for encrypted files.

Test Case Result



Total Count : 23 | ✓ Success Count : 13 | ✗ Failure Count : 10

Type	Test Case	Result	Match Percentage
Maven Junit	testSaveDocumentWithValidInput	Success	-
Maven Junit	testSaveDocumentWithNullFileData	Success	-
Maven Junit	testGetExistingDocumentById	Success	-
Maven Junit	testGetNonExistentDocumentById	Success	-
Maven Junit	testGetAllMetadataCount	Success	-
Maven Junit	testMetadataStructure	Success	-
Maven Junit	testRepositorySaveAndFind	Success	-
Maven Junit	testSaveWithEmptyEmail	Success	-
Maven Junit	testSaveLargeDocument	Success	-
Maven Junit	testSaveWithSpecialFilename	Success	-