Digital Asset Management System - Capstone Project Report

Developer: Mukesh Yadav

Training: Web Development Capstone

Frontend: Angular

Backend: ASP.NET Core Web API (.NET 8.0)

Database: SQL Server

Storage: Local File System

Authentication: JWT-based Authentication

Authorization: Role-based Access (User / Admin)

1. Project Overview

The Digital Asset Management System (DAMS) is designed to manage the lifecycle of digital files such as images, videos, documents, and other file types. Users can upload, tag, categorize, and manage their digital content, while admins can moderate the content, manage users, track system activities, and view analytics. The system supports real-time notification to admins on new uploads and enables users to view only approved files.

2. Functional Modules

2.1 User Module

- User Registration & Login
- Role: "User"
- Upload digital assets with metadata (title, description, tags, file type)
- View uploaded assets

- Submit feedback on other users' approved assets

2.2 Admin Module

- Role: "Admin"
- View all user uploads (pending/approved/rejected)
- Approve or reject uploaded files
- Manage user status (block/unblock)
- View activity log of users
- Real-time notification on new uploads

2.3 Asset Upload & Management

- Users can upload images, videos, PDFs, and documents
- Files stored in a local folder
- Each upload contains metadata: title, description, tags, category
- Files must be approved by an admin before being accessible

2.4 Notification System

- Admin receives notification on every new upload
- User receives notification on file approval/rejection
- Notifications stored in a table and shown in frontend UI

2.5 Feedback Module

- Users can view and submit feedback on approved assets
- Feedback includes rating, comment, timestamp

2.6 Analytics Dashboard

- Pie charts for:

  - Total images, videos, documents uploaded
  - Category-wise distribution
  - Number of approved, rejected, pending assets

2.7 Activity Logging

- Every user/admin action logged (upload, approval, feedback)
- Admin can view activity logs in tabular form with filters

3. Technical Architecture

Backend (ASP.NET Core Web API)

- Follows clean architecture
- Uses Entity Framework Core for DB operations
- Controllers:
  - UserController, AssetController, AdminController
  - NotificationController, AnalyticsController, ActivityLogController

Frontend (Angular)

- Modular structure
- Components:
  - Login, Register, Asset Upload, Asset List
  - Admin Dashboard, Approval List, Analytics, Notifications

Database Design (Key Tables)

- Users (Id, Name, Email, PasswordHash, Role)
- Assets (Id, UserId, Title, FilePath, FileType, Status, Category)
- Tags, AssetTags (many-to-many)

- Feedbacks (AssetId, UserId, Rating, Comment)
- Notifications (UserId, Message, IsRead, CreatedAt)
- Activities (UserId, Action, Timestamp)

4. Authentication and Authorization Report

Authentication

- JWT-based system
- Token is issued after login and stored in localStorage
- Every protected API checks token validity using middleware

Authorization

- Roles: User and Admin
- Protected endpoints:
  - Users can only manage their own uploads
  - Admin-only routes for approval, analytics, and activity log
- Frontend route guards for role-based access

Report Metrics:

- Total uploads by type: Images, Videos, Documents
- Status report: How many are Approved, Rejected, Pending
- Category-wise asset count (e.g., Education, Events, Legal)

5. File Handling & Storage

- All files are stored in a local folder
- On upload, file is renamed uniquely using GUID
- DB stores metadata + file path

- Files are served via static hosting from the local directory

6. Real-time Notifications

- On file upload, admin receives notification
- On file approval/rejection, user gets notification
- Notifications are polled every 30s via Angular service
- Notification icon shows unread count

7. Analytics and Reporting

- Dashboard includes:
  - Pie chart of file types: Images, Videos, Documents
  - Pie chart of status: Approved, Rejected, Pending
  - Pie chart showing distribution per category
- Data is shown directly on the frontend dashboard page

8. Project Execution Plan

Phase	Task
Phase 1	Setup Angular and ASP.NET Core projects
Phase 2	Implement Authentication and Authorization
Phase 3	Implement Upload + File Storage Logic
Phase 4	Admin Approval Workflow and Notifications
Phase 5	Feedback, Activity Log, and Analytics
Phase 6	Frontend Polish and Integration

9. Conclusion

This Digital Asset Management System provides a secure and scalable platform for users to manage digital content efficiently, with admin oversight, feedback mechanisms, and live visual analytics. The system demonstrates strong backend structure, role-based authentication, real-time updates, and modern Angular UI capabilities.

Prepared By: Mukesh Yadav

Capstone Title: Digital Asset Management System