

Document Management System (DMS)

To successfully complete this .NET Full stack project, we will develop a system with a set of features that integrate with SQL server database, handling user authentication, workspace management, document handling, and more. Below is a detailed outline for each task and feature requirement of the project.

Project Overview

This project is built using the .NET stack (Entity framework, WebAPI, Asp Identity, Angular) and incorporates with relational database by SQL server. The application will enable user registration and login, manage workspaces, and handle documents with various operations for multiple user roles (Admin - User)

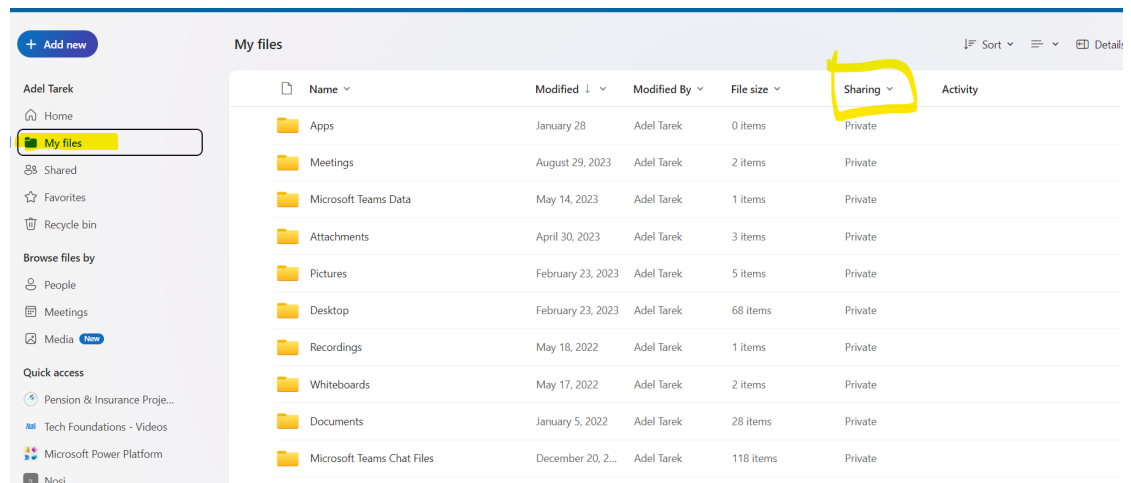
Task 1: Login & Registration (Asp Identity)

- **Objective:** Implement user authentication using Asp Identity and doing the required DB migrations to store **user** information.
- **Requirements:**
 - **Registration:**
 - Endpoint to register users by collecting email, password, NID, and personal information.
 - Ensure data validation.
 - Each registered user must input their Workspace name (it should be automatically created as part of registration cycle and unique).
 - **Login:**
 - Endpoint to authenticate users via email and password.
 - Implement JWT or session-based authentication for session management.
 - **Security:**
 - Implement custom action filter to check user authorizations over directories.

Task 2: Document Workspace Structure

- **Objective:** Create a workspace directory structure that is user specific to store workspace and documents data.
- **Requirements:**
 - **Database Setup:**
 - Use SQL server to store workspace and document information.
 - Example Tables: Workspaces, Directories, Documents. Etc...
 - **Workspace Structure:**

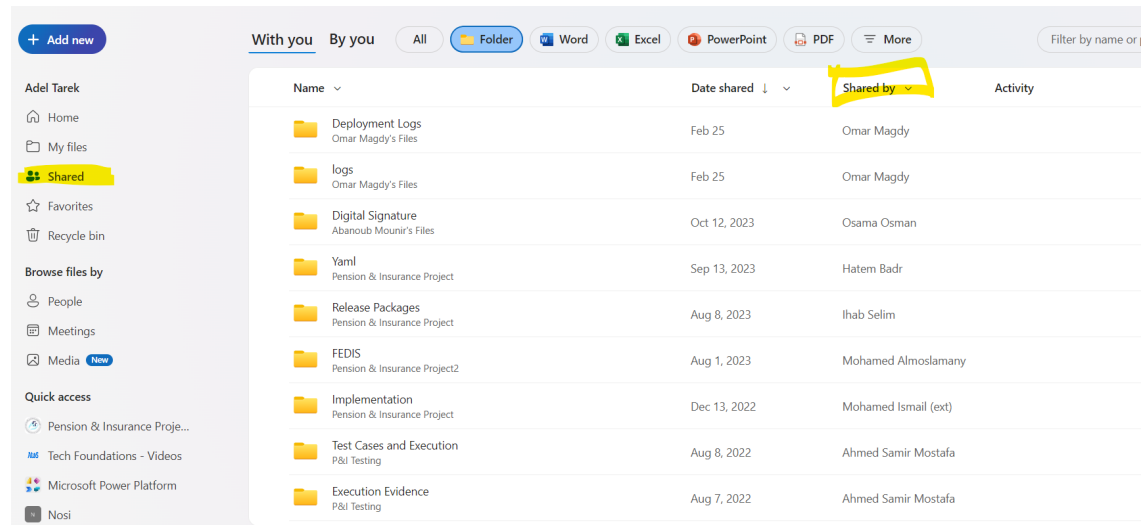
- Each user has One workspace with multiple directories
- Link workspace structure to user profiles using the user Id.
- All users should have access to public directories in a separate tab/page
 - In this page user should have **read only** access public directories Documents
- User can modify their workspace
 - Create directory
 - Soft delete directory
 - Make directory public/private “default is private”
 - All document CRUD operations in their workspace
- APIs:
 - Create endpoints to create, retrieve, update, and delete workspaces.
 - Ensure workspaces are uniquely tied to user accounts.
- UI Example :
 - My Workspace



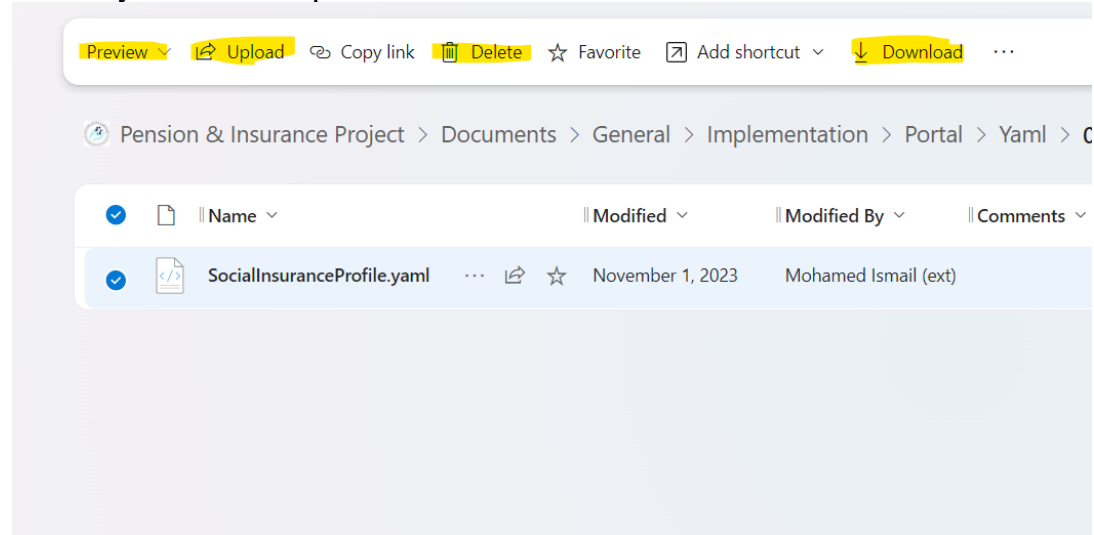
The screenshot shows a user interface for managing files. On the left is a sidebar with navigation options: Home, My files (selected), Shared, Favorites, Recycle bin, Browse files by (People, Meetings, Media), and Quick access (Pension & Insurance Proje..., Tech Foundations - Videos, Microsoft Power Platform, Nosi). The main area is titled 'My files' and contains a table of folders. The table has columns: Name, Modified, Modified By, File size, Sharing, and Activity. The 'Sharing' column is highlighted with a yellow box. The folders listed are: Apps, Meetings, Microsoft Teams Data, Attachments, Pictures, Desktop, Recordings, Whiteboards, Documents, and Microsoft Teams Chat Files.

Name	Modified	Modified By	File size	Sharing	Activity
Apps	January 28	Adel Tarek	0 items	Private	
Meetings	August 29, 2023	Adel Tarek	2 items	Private	
Microsoft Teams Data	May 14, 2023	Adel Tarek	1 items	Private	
Attachments	April 30, 2023	Adel Tarek	3 items	Private	
Pictures	February 23, 2023	Adel Tarek	5 items	Private	
Desktop	February 23, 2023	Adel Tarek	68 items	Private	
Recordings	May 18, 2022	Adel Tarek	1 items	Private	
Whiteboards	May 17, 2022	Adel Tarek	2 items	Private	
Documents	January 5, 2022	Adel Tarek	28 items	Private	
Microsoft Teams Chat Files	December 20, 2...	Adel Tarek	118 items	Private	

- Shared Directories



• Directory Action Example



Task 3: Document List

- **Objective:** List documents based on workspace and profile.
- **Requirements:**

- **API Development:**
 - Create endpoints to list documents for a specific workspace.
 - Create endpoint to retrieve shared directories.
- **Filtering:**
 - Implement filters to sort and search documents by various criteria using documents metadata.

Task 4: Uploading Document (Multi-part API [[FromForm](#)])

- **Objective:** Implement a multi-part API for document upload.
- **Requirements:**
 - **API Development:**
 - Create an endpoint to handle multi-part form data for document uploads.
 - Store documents in a file storage system.
 - **Data Handling:**
 - Save document metadata and file storage path in the `Documents` Table.
 - Ensure metadata includes document name, type, owner, and workspace association.

Task 5: Download Document (Multi-part API)

- **Objective:** Enable document download functionality.
- **Requirements:**
 - **API Development:**
 - Create an endpoint to download documents based on document ID.
 - Ensure proper authentication and authorization checks.
 - **Integration:**
 - Retrieve documents from storage and serve them to authenticated users.

Task 6: Delete Document (Soft Deletion)

- **Objective:** Implement soft deletion of documents.
- **Requirements:**
 - **API Development:**
 - Create an endpoint to mark documents as deleted without removing them from storage.
 - Add a `deleted` flag in the document metadata.
 - **Data Integrity:**

- Ensure soft-deleted documents are not visible in the user's active workspace.

Task 7: Preview Document

- **Objective:** Provide a preview of documents as Blob data.
- **Requirements:**
 - **API Development:**
 - Create or reuse an endpoint to return document data as a Blob.
 - Preview the Blob data in an <Iframe>
 - **Security:**
 - Ensure that only authorized users can preview documents.

Task 8: Document Metadata

- **Objective:** Manage document metadata efficiently.
- **Requirements:**
 - **Metadata Structure:**
 - Ensure metadata includes document name, type, owner, versioning info, tags, and access controls.
 - **API Development:**
 - Create endpoints to update and retrieve document metadata.

Admin Features

- List all users
- Lock user and ensure the locked user cannot log in
- See user details (info – workspaces and its documents) in a separate component.

Additional Features

Admin Action logs notifications

- Implement an action log mechanism that log each user action inside workspace, with action type enumeration that contains the main actions (upload, download, preview) using middleware and RabbitMQ messaging queue to store actions and send notification using SignalR to admin to view those actions.

Implementation Notes

- **Frontend:** Use Angular to create an intuitive user interface with appropriate components for each feature.
- **Backend:** Develop APIs using .NET WebAPI to handle requests and interact with databases using ORM (Entity Framework).