**FUNewsManagementSystem**

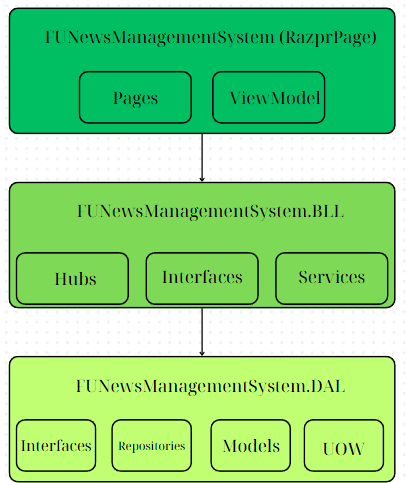
**1. Project**

**1.1 Introduction:**

The **FUNewsManagementSystem** is designed for educational institutions to efficiently manage, organize, and publish news content. It facilitates streamlined news operations, enhancing communication with students and the wider community.

**1.2 Project Structure:**

The **FUNewsManagementSystem** follows the **3-Layered Architecture pattern**, ensuring a clear separation of concerns, maintainability, and scalability. The project consists of three main layers:



**1. Presentation Layer (FUNewsManagementSystem)**

**Description:** This is the **ASP.NET Core Razor Pages** Web Application responsible for handling the user interface, routing, and presentation logic. SignalR is integrated for real-time communication.

**Components:**

* **Pages:** Razor pages for the UI, including login, account management, news management, and category management.
* **wwwroot:** Static assets like CSS, JavaScript, and images.
* **SignalR Hubs:** Enables real-time updates for news articles.
* **appsettings.json:** Configuration file containing database connection strings and default account credentials.

**2. Business Logic Layer (FUNewsManagementSystem.BLL)**

**Description:** Contains the core business logic and rules for the application.

**Components:**

* **Interfaces:** Defines contracts for business services.
* **Services:** Implements business logic, handles data processing, and communicates with the Data Access Layer.

**3. Data Access Layer (FUNewsManagementSystem.DAL)**

**Description:** Responsible for interacting with the database using the Repository pattern.

**Components:**

* **Interfaces:** Contracts for data access methods.
* **Models:** Entity Framework Core models representing the database tables.
* **Repositories:** Implements CRUD operations and data queries using LINQ and Entity Framework Core.

**2. Objectives**

This project aims to:

* Utilize **ASP.NET Core Razor Pages** for web application development.
* Implement **real-time updates using SignalR**.
* Perform **CRUD operations with Entity Framework Core**.
* Leverage **LINQ** for data querying and sorting.
* Apply **3-Layers architecture** and implement **Repository and Singleton patterns**.
* Include **CRUD, search, and data validation** for all fields.
* Ensure **functional user roles and permissions**.

**3. System Architecture**

* **3-Layers Architecture:** Presentation Layer, Business Logic Layer (BLL), Data Access Layer (DAL).
* **Design Patterns:**
  + **Repository Pattern**: For data access abstraction.
  + **Singleton Pattern**: For shared services (e.g., logging, configuration).

**4. Database Design**

* A **news article belongs to one category**.
* **Staff can create multiple news articles** (Roles: Staff = 1, Lecturer = 2, Admin from appsettings.json).
* **Articles can have multiple tags**, and tags can belong to zero or one article.
* **Status values:**
  + News (**Active = 1, Inactive = 0**)
  + Category (**Active = 1, Inactive = 0**)

**5. Functional Requirements**

**5.1 Authentication and Authorization**

* **Login & Logout**: Based on Email and Password.
* **Roles:**
  + **Admin:** Manage account information, create reports (by date range, sorted by date desc).
  + **Staff:** Manage categories, news articles (with tags), own profile, and view personal news history.
  + **Lecturer:** View active news articles.

**5.2 Account Management**

* **Admin** can perform **CRUD and search** actions on accounts.
* **Pop-up dialogs** for **Create and Update** actions.
* **Delete actions require confirmation**.

**5.3 News Article Management**

* **CRUD and search** actions for news articles.
* **Real-time updates** using **SignalR**.
* **Pop-up dialogs** for **Create and Update** actions.
* **Only active news articles are viewable**.
* **Tags management included**.

**5.4 Category Management**

* **CRUD and search** actions for categories.
* **Categories linked to news articles cannot be deleted**.
* **Pop-up dialogs** for **Create and Update** actions.
* **Delete actions require confirmation**.

**6. Non-Functional Requirements**

* **Security:** Role-based authorization, password encryption.
* **Performance:** Fast loading times, efficient querying with LINQ.
* **Scalability:** Modular architecture for future expansion.
* **Usability:** Intuitive UI with Bootstrap for responsiveness.

**7. Development Environment**

* **IDE:** Visual Studio 2019 or above.
* **Framework:** ASP.NET Core Razor Pages (.NET 5/6/7/8).
* **Database:** MS SQL Server 2012 or above.
* **ORM:** Entity Framework Core.
* **Real-Time Communication:** SignalR.
* **Version Control:** GitHub.

**8. Default Admin Account**

* **Email:** admin@FUNewsManagementSystem.org
* **Password:** @@abc123@@ (Stored in appsettings.json)