# Overview

This project is a Blazor WebAssembly application designed to manage information about farmers and their products. It includes functionalities for different user roles (Farmers and Employees) and integrates various modern development practices and tools.

# Development Environment Setup

## Prerequisites

.NET SDK 8.0 or higher: Download and install from [here](https://dotnet.microsoft.com/en-us/download).

Visual Studio 2022: Ensure you have the Blazor WebAssembly template installed.

Service Based Database: Local Database.

## Steps

1. Clone the Repository:  
   <https://github.com/ST10134012/ST10134012-AgriConnect>

Opening from VCLearn

1. Download the project and unzip
2. If there is a 0x80010135 file path too long error, shorten the name of the zipped project e.g. proj or agri.

# Building and Running the Prototype

## Build the Project

1. Open the solution in Visual Studio.
2. Restore NuGet packages by right-clicking on the solution and selecting "Restore NuGet Packages".
3. Build the solution by pressing Ctrl+Shift+B.

## Setup Database

1. In the solution open AgriConnect\AgriConnect\AgriConnect.Infrastructure\Migrations\AgriConnect.mdf.
2. In Server Explorer, under Data Connections right-click on AgriConnect.mdf and select Properties.
3. Copy the Connection String.
4. Open apssettings.json in AgriConnect.WebApp and update the two connection strings.
5. If the project is from Github, you will have to create a service-based database.
6. Add the database in the AgriConnect\AgriConnect\AgriConnect.Infrastructure\Migrations folder. Right click on Migrations, select Add New Item. Go to Data and select Service Based Database.
7. Name your database AgriConnect then follow Steps 2-4.

## Install Dependencies

1. Go to Tools then NuGet Package Manager and open the Package Manager Console
2. Enter the following commands: ‘***dotnet restore***’ then ‘***Add-Migration InitialCreate***’ then ‘***Update-Database -Context ApplicationDbContext***’ then ‘***Update-Database -Context AuthenticationDbContext***’.
3. Rebuild the project.

## Run the Project

1. Set the AgriConnect.WebApp project as the startup project.
2. Run the application by pressing F5 or clicking the "Run" button.

# System Functionalities

## Farmers

1. **Register and Login**:
   * **Registration**: Farmers can sign up by providing essential details such as name, email address, and a secure password.
   * **Login**: Farmers can log in using their email and password.
   * **Remember me**: Options for login details to be saved so details are auto populated for next log in.
2. **Add Products**:
   * **Product Details**: Farmers can add new products with fields such as name, category (e.g., vegetables, fruits, grains) and production date.
3. **View Products**:
   * **List View**: Farmers can view a list of all their products.
   * **Sorting and Filtering**: Ability to sort products by date, category, and other criteria.

## Employees

1. **Register and Login**:
   * **Registration**: Employees can sign up by providing necessary details such as name, email address, and a secure password.
   * **Login**: Employees can log in using their email and password.
   * **Remember me**: Options for login details to be saved so details are auto populated for next log in.
2. **Add Farmer Profiles**:
   * **Farmer Details**: Employees can add new farmers by entering essential details such as name, email address and password.
3. **View and Filter Products**:
   * **Comprehensive Product List**: Employees can view a comprehensive list of all products from all farmers.
   * **Filtering Options**: Employees can filter products based on criteria such as date range, category, and farmer name.

# User Roles

## Farmer:

* Add Products: Ability to add new products to their profile.
* View Products: Can view and manage their own product listings.

## Employee:

* Add Farmer Profiles: Can add new farmer profiles with all necessary details.
* View All Products: Can view products from all farmers.
* Filter and Search Products: Can use various filters to search for specific products based on criteria.

# Technologies Used

## Blazor WebAssembly:

* Interactive UIs: Provides a responsive and dynamic user interface for both farmers and employees.
* Single Page Application (SPA): Ensures a seamless user experience without full page reloads.

## Entity Framework Core:

* Database Interactions: Manages database operations such as CRUD (Create, Read, Update, Delete) seamlessly.
* Migrations: Handles database schema changes through migrations.

## Service-Based Database:

* Relational Data Storage: Uses a relational database to store farmer profiles, products, and other related data.
* Scalability: Supports scalable data storage solutions to handle growing data.

## Caching:

* Performance Improvement: Reduces database calls by caching frequently accessed data.
* Cache Invalidation: Ensures cache is updated or invalidated when underlying data changes.

## Dependency Injection:

* Manage Dependencies: Helps manage and inject dependencies across the application to ensure modularity and testability.

## MediatR:

* Command and Query Handling: Manages commands, queries, and events for better separation of concerns and maintainability.

## Repositories:

* Data Access Abstraction: Abstracts the data access logic, making the codebase cleaner and more maintainable.

## Exception Handling:

* Graceful Error Handling: Catches and manages exceptions to prevent system crashes and data corruption.
* User-Friendly Messages: Provides meaningful error messages to users.

# Additional Features

## Responsive Design:

* Multi-Device Accessibility: Ensures the application is accessible on various devices such as desktops, tablets, and smartphones.
* Adaptive Layouts: Adjusts the layout dynamically based on screen size and resolution.

## Data Validation:

* Client-Side Validation: Validates data on the client side to provide instant feedback.
* Server-Side Validation: Ensures data integrity by validating on the server side as well.

## Error Handling:

* Error Logging: Logs errors for debugging and monitoring purposes.
* User Notifications: Notifies users of errors and guides them on how to proceed.