PROG 7311

README

Sajana Bidesi

ST10249843

LINKS

YOUTUBE: https://youtu.be/dhHuiQ4mKWE

GITHUB: https://github.com/SajanaBidesi25/AgriEnergy.git

ABOUT THE PROJECT

AgriEnergy is a feature-rich e-commerce platform developed to support sustainable farming practices. The application facilitates digital empowerment for farmers, enabling them to list environmentally friendly products, participate in discussions and connect with administrators who oversee the entire platform. The system supports role-based access, providing different functionalities to farmers and employees (administrators).

ARCHITECTURE PATTERN - MODEL-VIEW-CONTROLLER (MVC)

The Model-View-Controller (MVC) architectural pattern separates the application into three interconnected components, each with a distinct responsibility, which enhances scalability, maintainability, and testability:

1. Model:
   - Manages the data, logic, and rules of the application.
   - In AgriEnergy, Models include `User`, `Product`, `Farmer`, and `Employee` classes which define the core data structure.
2. View:
   - The user interface layer that displays data from the Model and sends user commands to the Controller.
   - Razor views (.cshtml) render data and allow user interaction using HTML and Bootstrap in AgriEnergy.
3. Controller:
   - Acts as an intermediary between Model and View.
   - Receives input from users via the UI, processes that data (possibly modifying the model), and returns the appropriate view.
   - E.g., `ProductController`, `UserController` handle user and product operations.

DESIGN PATTERN - FACADE

The Facade design pattern simplifies the interface for a complex system. It provides a unified entry point to

multiple subsystems, hiding implementation details from the client. In AgriEnergy:

- The Facade pattern abstracts data access operations, improving code readability and reducing coupling.
- It also helps to encapsulate service logic for user registration and product management, promoting modular development.
- By using services and a facade structure, the project remains loosely coupled and highly testable.

DATABASE - SQL SERVER MANAGEMENT STUDIO (SSMS)

SQL Server Management Studio (SSMS) is a comprehensive management tool for SQL Server.

In AgriEnergy, SSMS was used to:

- Create and configure the AgriEnergyDB.
- Populate the database with essential data like user roles and an initial admin account.
- Run queries, perform backups, and monitor database health.

The use of SSMS ensures a robust, scalable, and secure relational data storage system.

The database schema includes relationships between Farmers, Products, and Users via foreign key constraints.

TECHNOLOGY STACK

- C# (ASP.NET Core MVC for the backend logic)
- HTML, CSS, JavaScript (for UI)
- Razor Pages (.cshtml) for rendering dynamic content
- SQL Server + SSMS for database
- Visual Studio 2022 IDE

HOW TO USE THE APP

To run AgriEnergy on your local development environment, follow these detailed steps:

1. Clone the GitHub Repository:
   - Open a terminal or Git Bash and run:
   - git clone https://github.com/SajanaBidesi25/AgriEnergy.git
2. Open the Solution in Visual Studio:
   - Launch Visual Studio.
   - Navigate to the `AgriEnergy` directory.
   - Open the `AgriEnergy.sln` solution file.
3. Configure the Database:
   - Ensure that SQL Server and SSMS are installed on your machine.
   - Open SSMS, create a new database called `AgriEnergyDB`, or use the provided `.mdf` file if applicable.
   - Update the connection string in `appsettings.json` (see below).
4. Build the Project:
   - Use `Build > Build Solution` or press `Ctrl + Shift + B`.
5. Run the App:
   - Press `F5` or click the "Start" button to launch the app.
   - The application will open in your default browser.
6. Login or Register:
   - Use the preconfigured employee account to log in.
   - Employees can register new farmers.

DEPENDENCIES

- Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation
- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.VisualStudio.Web.CodeGeneration.Design

RBAC IN AGRIENERGY

AgriEnergy operates with two types of users—Farmers and Employees—implementing Role-Based Access Control (RBAC).

Farmers:

- Can log in and upload products (e.g., solar panels, compost).
- View only their own product listings.

EMPLOYEES:

- Register new farmers via the platform.
- View all products added by any farmer.
- Filter products based on category or production date.

CONNECTION STRING

Ensure your connection string in `appsettings.json` is configured correctly:

Server=lab7L95SR\SQLEXPRESS;Database=AgriEnergyDB;Trusted_Connection=True;TrustServerCertificate=True;

UI DESIGN

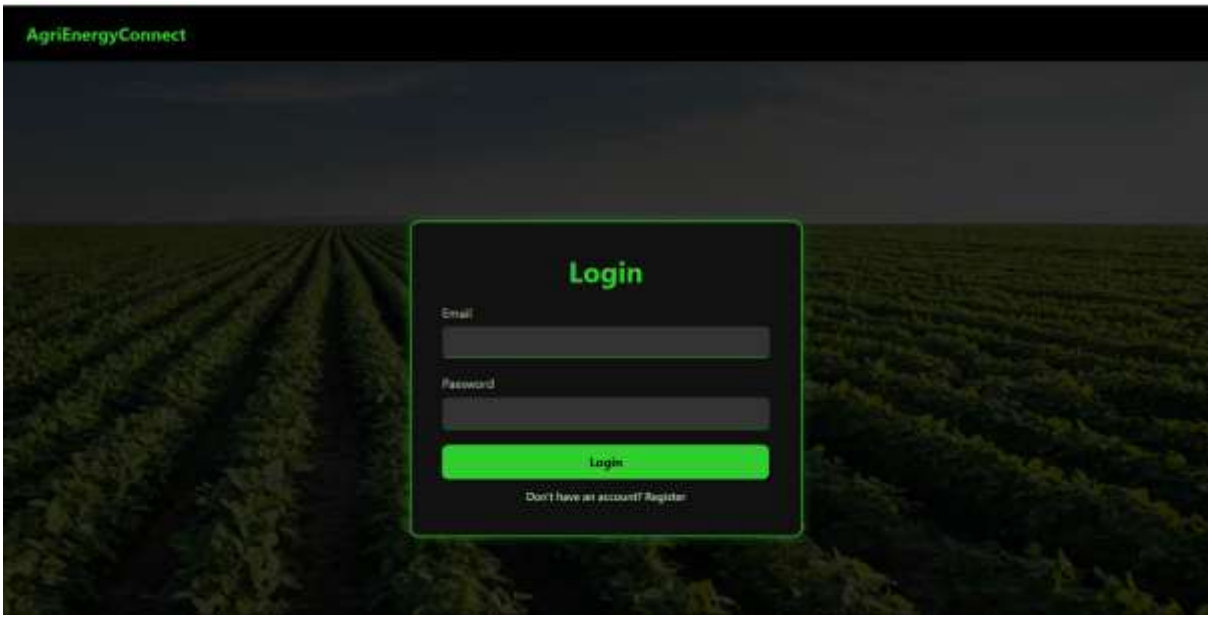
- The application uses a consistent green (#006400) and black color scheme to reflect the theme of sustainability.
- The homepage features a darkened background image of lush crops for aesthetic and thematic effect.
- Cards and buttons are styled using Bootstrap and custom CSS to ensure a responsive, user-friendly interface.

CHALLENGES AND CHANGES

- Role Management: RBAC implementation required thorough testing to ensure access was limited appropriately.
- UI Consistency: Designing uniform cards across views to display user and product information clearly.
- Database Sync: Ensuring the local database reflected the latest schema during development.

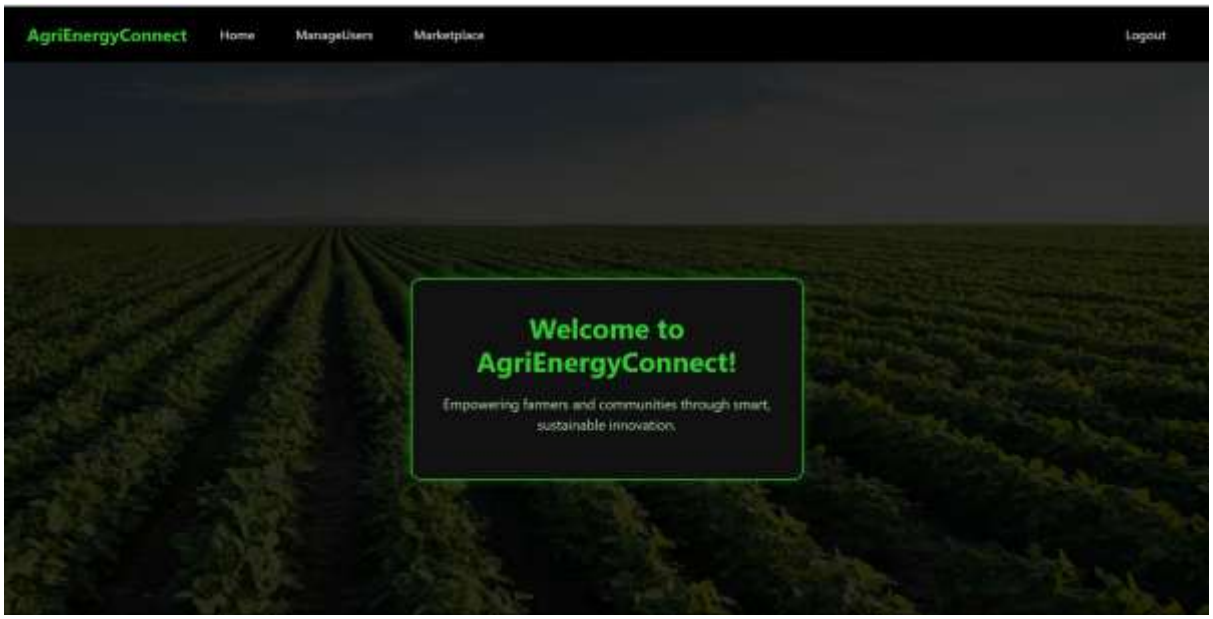
HOW THE APPLICATION WORKS:

LOGIN PAGE:



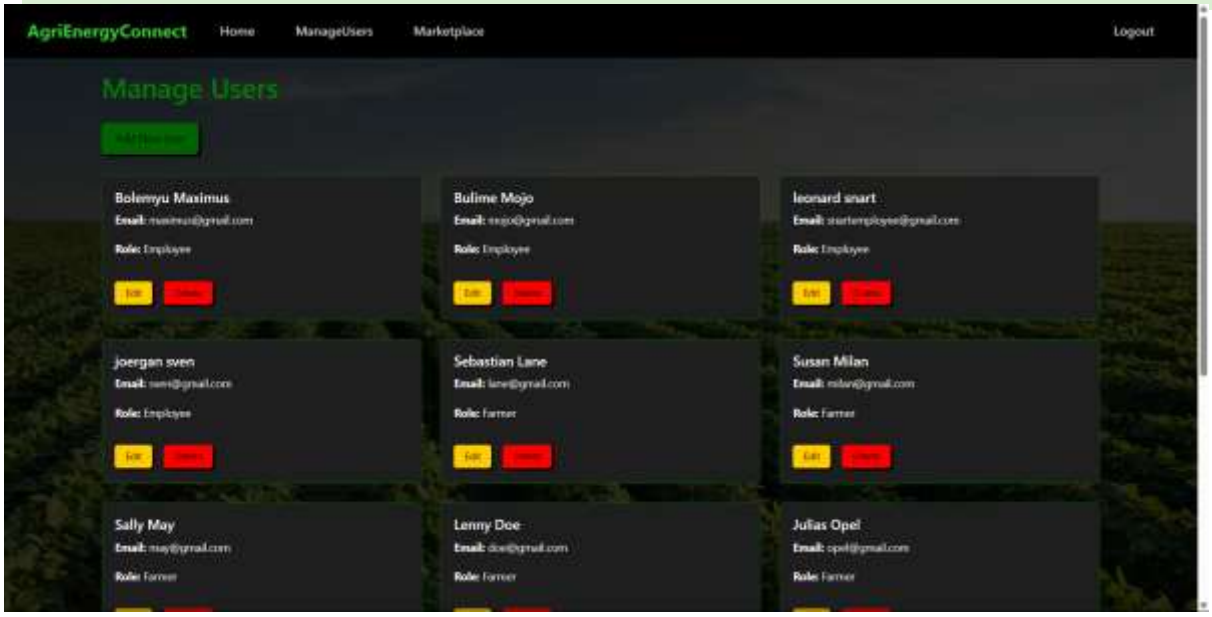
User authentication and authorization for logging in using their email and password. The user details is stored in the database when created by the employee. This view is accessible by both farmers and employees.

HOME PAGE:



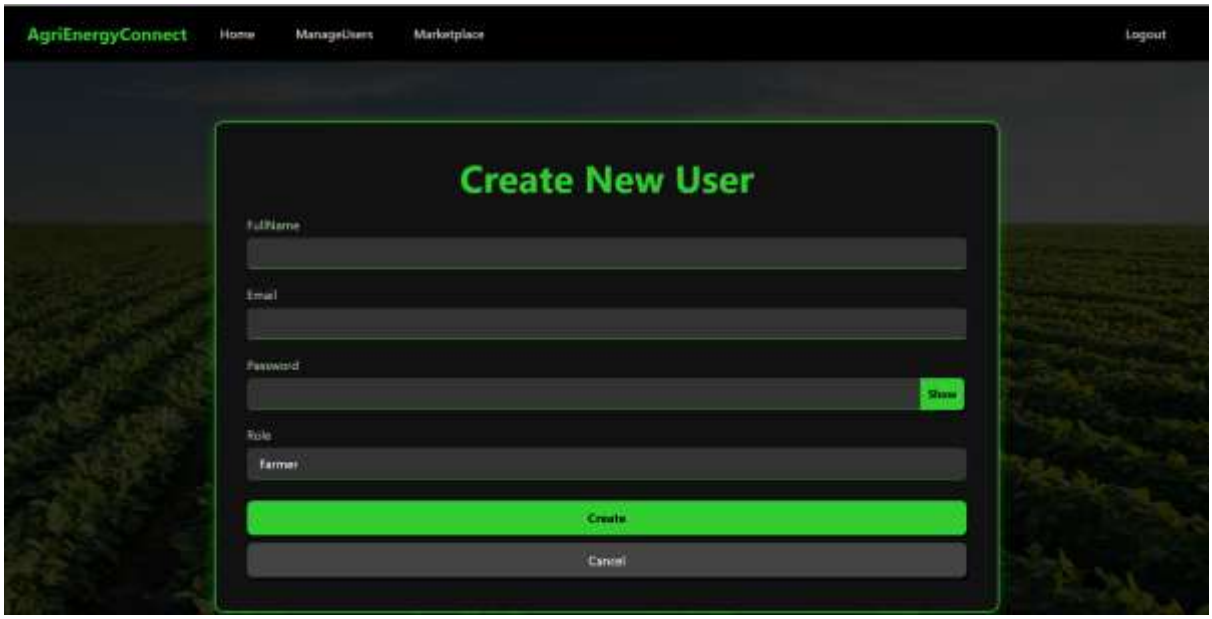
This page welcomes the users to Agri-Energy and is accessible by both the farmers and employees.

MANAGE USERS (EMPLOYEE):



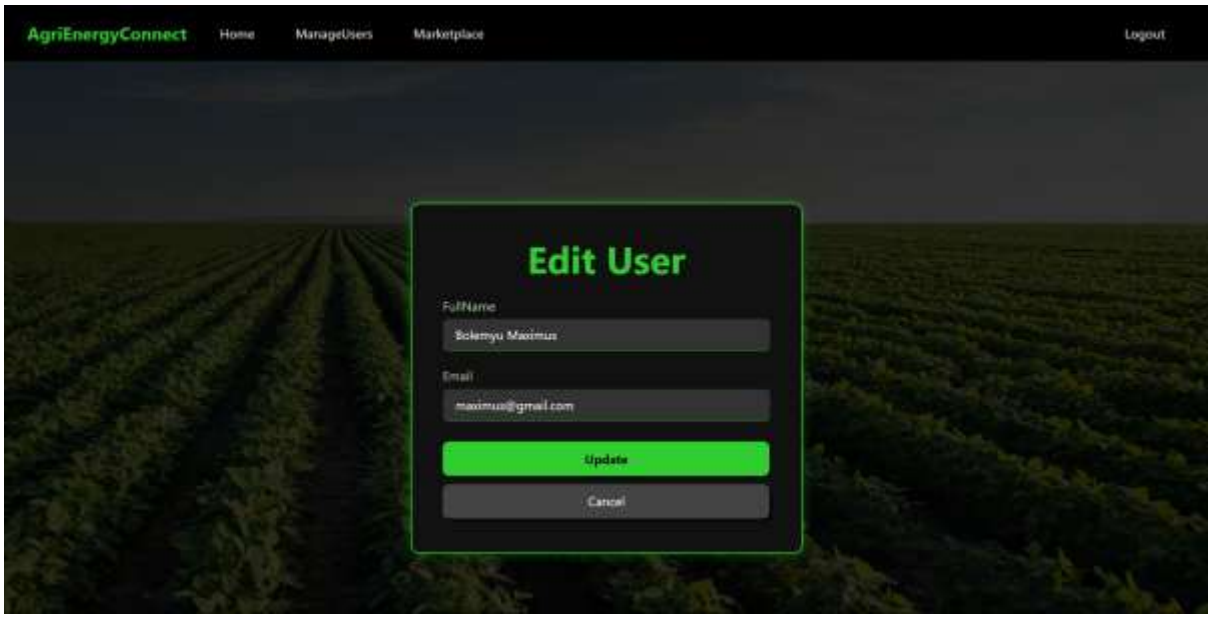
This page allows the employees to view a list of all the users registered in the system. With the option to edit or delete them as well, in doing so, it will update the database accordingly.

CREATE NEW USER (EMPLOYEE):



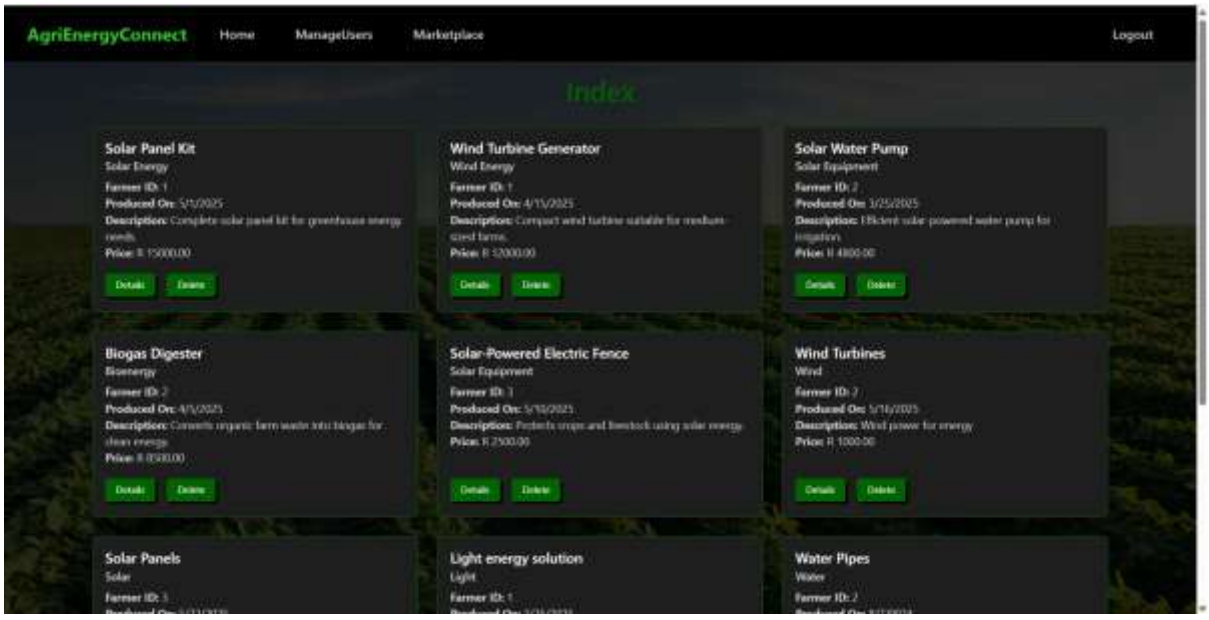
This is a view that will allow the employee to create a new user by their name, email, password, and role. After clicking on ‘Create’, the users data will be stored in the database.

EDIT USER (EMPLOYEE):



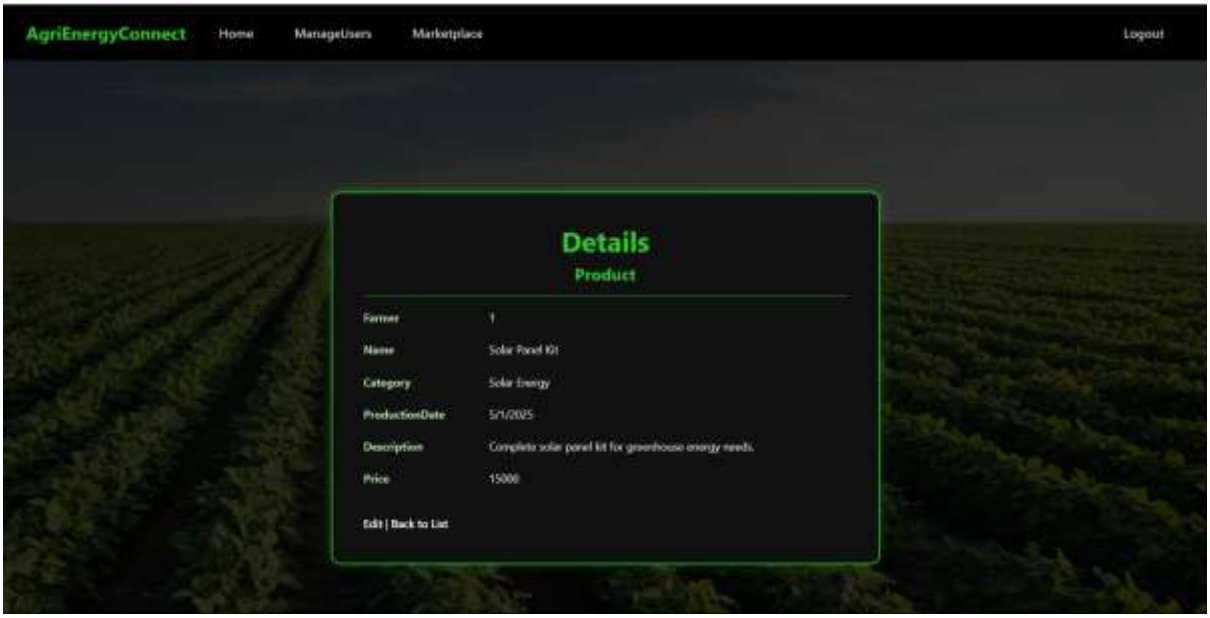
The edit user page allows employees to edit the name or the email of the user, if required or asked to do so by them. By doing thisit will update the database accordingly.

MARKETPLACE (EMPLOYEE):



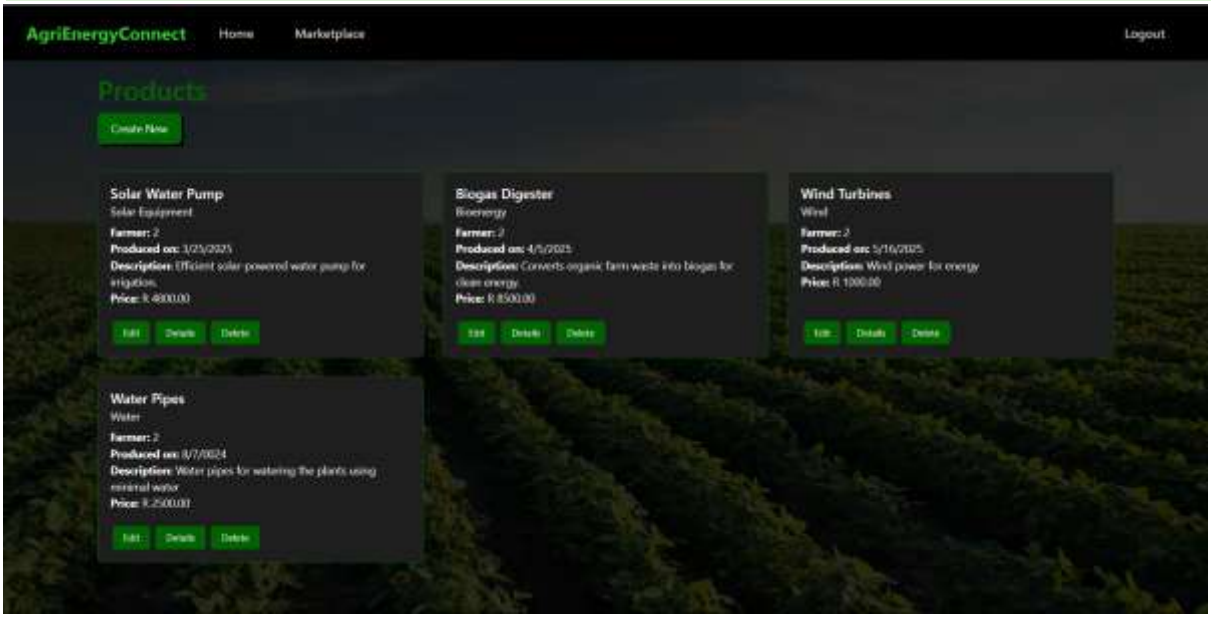
This is also a view only accessible by employees. Here they can view all the products added by all the farmers registered to the system. There is an option to view the details as well as delete.

DETAILS OF PRODUCT (EMPLOYEE):



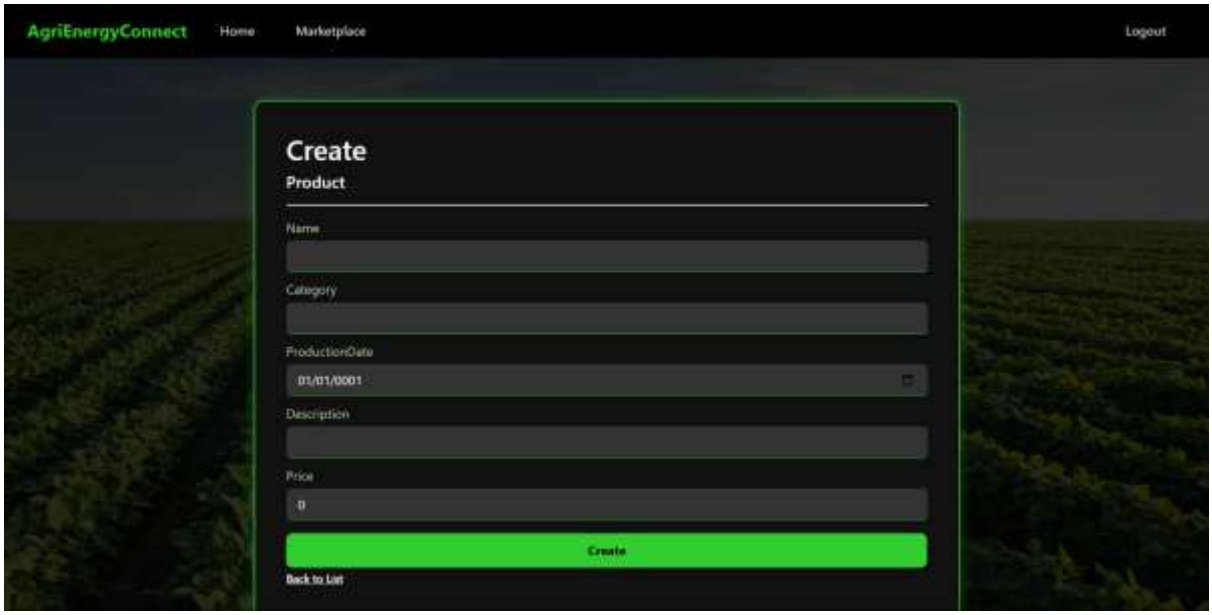
This is the details page which displays the products stored data much clearer.

MARKETPLACE (FARMER):



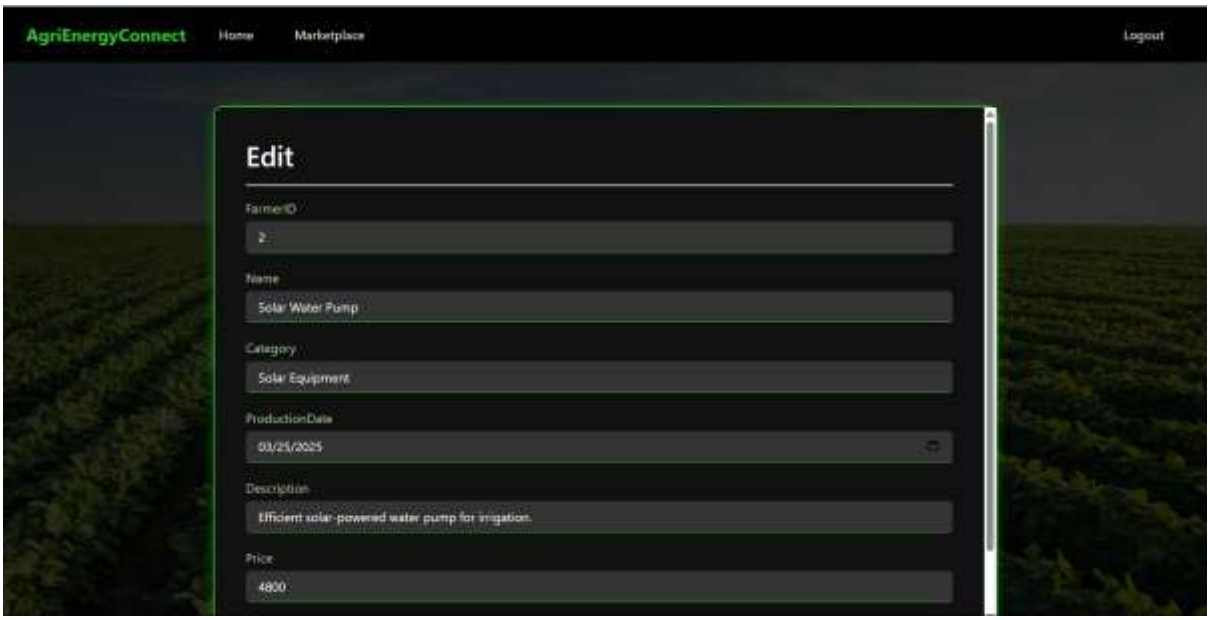
This page is only accessible by the farmers. Here they can only view their product listings via their unique farmer ID. They are also able to edit, view the details, and delete the information of a product.

CREATE PRODUCT (FARMER)



After clicking on ‘Create New’ the farmer will be redirected to the create page where they can fill out a form for a product listing. Clicking on ‘Create’ will store the information in the database in the Products table.

EDIT PRODUCT (FARMER):



This page allows the farmer to edit the details of a product. Since there are a few details, I have decided to make the view scrollable.

REFERENCE LIST

GeeksforGeeks, 2024. *MVC Design Pattern - GeeksforGeeks*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/mvc-design-pattern/>.

ghogen, 2024. *Add new connections - Visual Studio (Windows)*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/visualstudio/data-tools/add-new-connections?view=vs-2022>.

jongalloway, 2025. *Introduction to ASP.NET Identity - ASP.NET 4.x*. [online] learn.microsoft.com. Available at: <https://learn.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity>.

Refactoring Guru, 2014. *Facade*. [online] Refactoring.guru. Available at: <https://refactoring.guru/design-patterns/facade>.

W3Schools, n.d. *How To Create a Card with CSS*. [online] www.w3schools.com. Available at: <https://www.w3schools.com/howto/howto_css_cards.asp>.