ST10358162

PROG 7311

Zahra Karan

POE  PART 1

Contents

[Agri-Energy Connect Proposal 2](#_Toc195455883)

[Features of the System 2](#_Toc195455884)

[Non-functional requirements 4](#_Toc195455885)

[Design and Architecture Patterns 5](#_Toc195455886)

[Design Patterns 5](#_Toc195455887)

[System Architecture 7](#_Toc195455888)

[References 8](#_Toc195455889)

[Table 1 - Non-functional Requirements for Agri-Energy Connect 2](#_Toc195018301)

[Table 2 - Design Pattern Application 3](#_Toc195018302)

[Figure 1 - Use Case Diagram 3](#_Toc195455768)

[Figure 2 - MVC Figure 7](#_Toc195455769)

# Agri-Energy Connect Proposal

South Africa continues to face widespread electricity supply issues due to Eskom’s instability, resulting in frequent blackouts. This ongoing energy crisis, combined with a rising focus on environmental sustainability, highlights the urgent need for alternative energy approaches—especially in agriculture. These conditions have created a strong demand for digital tools that promote greener solutions.

Agri-Energy Connect is a planned online platform that will bring together farmers, renewable energy experts, and sustainability professionals. It will help users share knowledge, work together on green projects, and find practical ways to use clean energy in farming. The platform will promote new ideas, make sustainable technologies more accessible, and support environmentally friendly farming practices.

## Features of the System

Key features of the system will include a knowledge hub on sustainable farming techniques, a green energy marketplace tailored to agricultural needs, interactive forums for communication and support, educational resources such as webinars and workshops, and tools for project collaboration and access to funding opportunities.

The system will offer the following tools and services to support knowledge-sharing, collaboration, and access to resources.

Key features:

* **Sustainable Farming Knowledge Hub**  
  A section with tips and examples to help farmers learn about eco-friendly farming methods that work well in their area.
* **Green Energy Marketplace**  
  A specialised digital marketplace where users can discover, compare, and acquire renewable energy technologies and services suitable for agricultural operations.
* **Interactive Chat Portal**  
  A real-time chat space where farmers, employees, consultants, and energy experts can ask questions, share advice, and exchange knowledge. The portal will also provide helpful links to educational resources and updates on available green funding, including grants and subsidies.

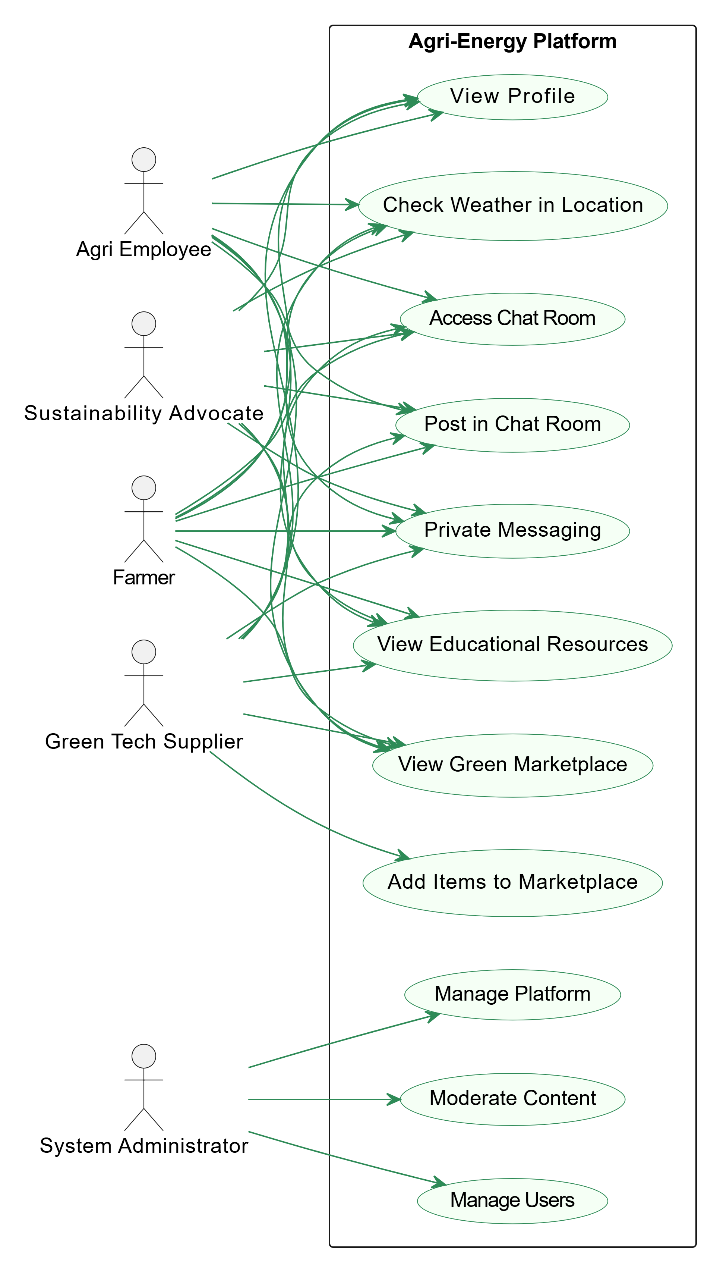


Figure - Use Case Diagram

Source: <https://app.diagrams.net/>

This use case diagram shows how different types of users interact with the Agri-Energy Platform. Farmers, agricultural employees, green tech suppliers, and sustainability advocates all have similar access to the system. They can view their profiles, check the weather in their area, join the public or private chat rooms to share ideas, view educational resources, and browse the green marketplace. Green tech suppliers have one extra function — they can also add items to the marketplace. The system administrator has a different role and is responsible for managing users, moderating content in the system, and maintaining the overall platform. This setup helps users collaborate, learn, and access useful tools while keeping the system secure and well-managed.

# 

# Non-functional requirements

To ensure Agri-Energy Connect delivers a seamless and reliable user experience, non-functional requirements must be prioritised. As Krüger explains, “Non-functional requirements are the criteria that define how a system should behave, rather than what it is supposed to do” (Krüger, 2024). These are crucial in a system aimed at farmers, consultants, and energy professionals across rural and urban settings, especially given varied levels of digital literacy.

This means the platform must be highly usable, with a simple and intuitive interface that supports users who may not be familiar with complex digital tools. It should also be accessible, working smoothly across devices and low-bandwidth internet connections often found in rural areas. Additionally, the system should demonstrate reliability and performance efficiency, ensuring fast loading times, minimal downtime, and consistent responsiveness. Other key non-functional requirements include security, to protect sensitive user data and communications, and scalability, so the platform can grow and handle increased usage over time. These aspects together will help build trust, encourage adoption, and support long-term success of the platform.

(Digital Adoption, 2025).

Table - Non-functional Requirements for Agri-Energy Connect

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement | Description | Quantifiable Goal | Impact & Implementation |
| Scalability | The system should grow smoothly as users and data increase (Digital Adoption, 2025). | Support 1000 users and scale. | The system will be hosted locally on a MongoDB database with replication to scale horizontally as the number of users and data increases. |
| Security | The platform must protect sensitive data. | Secure login, password requirement, and security tests passed. | Role-based authentication will limit access to data based on user roles and ensure privacy. |
| Usability | The platform must be user-friendly (AccessiBe, 2025). | Complete core tasks within 5 minutes. | A simple user interface (UI), clear labels, and visuals will guide users, especially those less tech-savvy. |
| Performance | Must always remain responsive (Metis, 2023). | 95% of pages must load in under 5 seconds. | The system will use efficient SQL queries with indexes on frequently accessed fields, retrieving only necessary data. Image compression and lazy loading will improve load times, while minifying and combining CSS and JavaScript files reduce browser requests keeping the platform responsive (Metis, 2023). |

# Design and Architecture Patterns

## Design Patterns

Design and architecture patterns are important for building a platform that is easy to manage, update, and grow over time. These patterns help developers organise their code in a clear and efficient way, making it easier to fix problems and add new features. Using proven patterns helps keep the system well-structured and reduces the chance of errors. They also support code reusability and modularity, which means parts of the code can be reused and updated without affecting the whole system (Out Systems, 2024). This is especially useful for Agri-Energy Connect, which may need to connect to other tools, such as databases for funding or smart farming devices.

Design patterns also make the platform more reliable, easier to maintain, and more secure. They help developers work together more effectively by following shared best practices (Linjanja, 2023; Teamhub, 2024). By using these patterns, Agri-Energy Connect can stay strong and flexible as it grows and changes in the future.

This table outlines specific design patterns selected for the Agri-Energy Connect platform and their role in meeting key system requirements.

Table - Design Pattern Application

|  |  |
| --- | --- |
| **Requirement** | **Design Pattern** |
| **Scalability** | Singleton Pattern will be applied to manage the database connection pool, this avoids duplication of components by managing shared resources (Curate Partners, 2025). |
| **Security** | The Factory Pattern generates role-based security features, such as custom login systems (e.g., farmer, employee, energy expert) (Patel, 2024). |
| **Usability** | The MVC architecture separates the UI from data logic, enhancing navigation. The Builder Pattern guides users through their profile. (Tutor, 2023). |
| **Performance** | |  | | --- | |  |  |  | | --- | | The Observer Pattern provides real-time updates, increasing engagement, reducing manual checks. Minification boosts loading speed (Glushenkov, 2023). | |

Non-functional requirements will guide key decisions:

* **Scalability** will affect the choice of database design (e.g. using MongoDB with replication).
* **Security** will focus on building secure login systems and access controls.
* **Usability** will ensure a simple and user-friendly interface.
* **Performance** will prioritise fast load times and efficient data handling.  
  Design patterns like MVC, Singleton, and Factory will help organise the code, making it easier to update and scale the platform as it grows.

(Krüger, 2024)

System Architecture

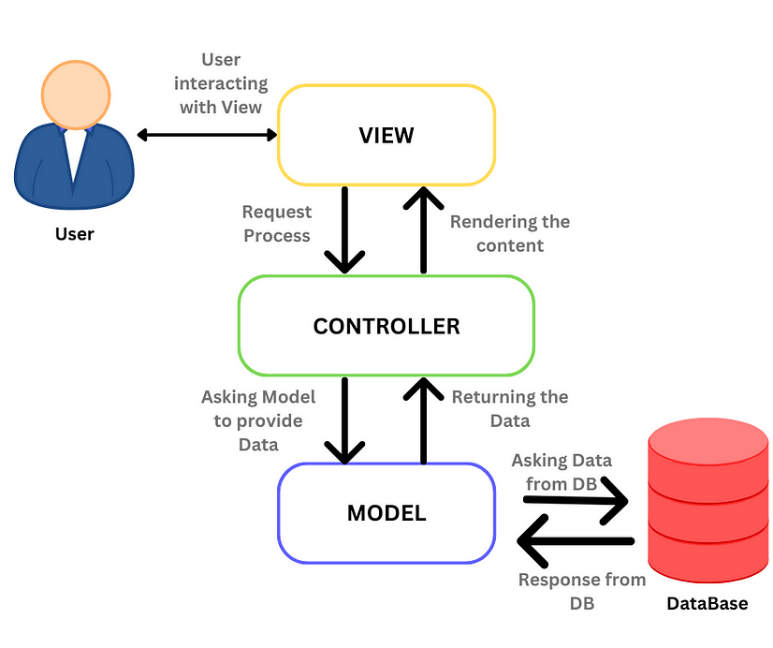


Figure - MVC Figure

Source: [The MVC Architecture. MVC (Model-View-Controller) is a… | by Sadika | Medium](https://medium.com/@sadikarahmantanisha/the-mvc-architecture-97d47e071eb2) (Sadika, 2023)

MVC (Model-View-Controller) divides the platform into three layers:

* **Model** (handles data),
* **View** (responsible for the user interface),
* **Controller** (manages user input and updates the view).

Using MVC ensures that changes in one part of the platform do not impact other parts, making the system easier to maintain and update (MDN, 2025). The user interacts with the view, which requests a process from the controller. The controller asks the model to provide the data from the database (Sadika, 2023).

The design and architecture patterns chosen for the Agri-Energy Connect platform are vital for ensuring the system is scalable, maintainable, and adaptable to future changes. The MVC pattern will allow for clear separation between the user interface and business logic, making it easier to modify and extend the platform.

The Agri-Energy Connect platform aims to address South Africa's energy and farming challenges through scalable, secure, and user-friendly software. By prioritising non-functional requirements and employing effective design and architecture patterns such as MVC, Singleton, and Factory, the system will be able to grow with its users and provide a reliable, efficient experience. The combination of these strategies ensures that the platform will meet the evolving needs of farmers and energy experts, contributing to a sustainable future.

# References

AccessiBe. (2025) Usability. AccessiBe [Online] Available at: <https://accessibe.com/glossary/usability#:~:text=Ensuring%20digital%20platforms%20are%20usable,usable%20for%20a%20broader%20audience> [Accessed 8 April 2025].

Amazon Web Services. (2025) What are Microservices? Amazon Web Services. [Online] Available at: <https://aws.amazon.com/microservices/#:~:text=With%20a%20microservices%20architecture%2C%20an,service%20performs%20a%20single%20function> [Accessed 8 April 2025].

Curate Partners (2025) The Singleton Pattern: Ensuring Efficient Resource Management and Global Access in Software Architecture. Curate Partners. [Online] Available at: <https://curatepartners.com/blogs/skills-tools-platforms/mastering-the-singleton-pattern-in-software-architecture-efficiency-and-global-access/#:~:text=In%20many%20systems%2C%20resources%20such,%2C%20or%20I%2FO%20resources>. [Accessed 8 April 2025].

Digital Adoption (2025) Scalability. Digital Adoption. [Online] Available at: <https://www.digital-adoption.com/glossary/scalability/> [Accessed 6 April 2025].

Glushenkov, A. (2023) Notifying the Masses: The Benefits and Challenges of the Observer Pattern in Software Development. Medium. [Online] Available at: <https://medium.com/@alexglushenkov/notifying-the-masses-the-benefits-and-challenges-of-the-observer-pattern-in-software-development-5decd1c008e3> [Accessed 6 April 2025].

Indeed. (2025) What is an enterprise solution? Definition and advantages. Indeed. [Online] Available at: <https://www.indeed.com/career-advice/career-development/what-is-an-enterprise-solution> [Accessed 6 April 2025].

Krüger, G. (2024) Non-functional Requirements: What They Do, Examples, and Best Practices. Perforce. [Online] Available at: <https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples#:~:text=Non%2Dfunctional%20requirements%20are%20the,usability%2C%20reliability%2C%20and%20scalability> [Accessed 7 April 2025].

Linjanja, P. (2023) Design Patterns for Modern Backend Development – with Example Use Cases. FreeCodeCamp. [Online] Available at: <https://www.freecodecamp.org/news/design-pattern-for-modern-backend-development-and-use-cases/> [Accessed 8 April 2025].

MDN. (2025) MVC. MDN Web Docs. [Online] Available at: <https://developer.mozilla.org/en-US/docs/Glossary/MVC> [Accessed 8 April 2025].

Metis. (2023) 8 Proven Strategies to Improve Database Performance. Metis. [Online] Available at: <http://metisdata.io/blog/8-proven-strategies-to-improve-database-performance> [Accessed 7 April 2025].

Out Systems. (2024) Code Reusability and Modularity. Out Systems. [Online] Available at: <https://success.outsystems.com/documentation/10/developing_an_application/reuse_and_refactor/code_reusability_and_modularity/> [Accessed 7 April 2025].

Patel, R. (2024) Understanding the Factory Method Design Pattern in C#. Medium. [Online] Available at: <https://medium.com/@ravipatel.it/understanding-the-factory-method-design-pattern-in-c-5cedf15c5616> [Accessed 8 April 2024].

Sadika. (2023) The MVC Architecture. Medium [Online] Available at: <https://medium.com/@sadikarahmantanisha/the-mvc-architecture-97d47e071eb2> [Accessed 7 April 2025].

Tutor, L. (2023) C# Builder Pattern: Constructing Objects Step-by-Step. Medium. [Online] Available at: <https://medium.com/@lexitrainerph/c-builder-pattern-constructing-objects-step-by-step-97b583246599> [Accessed 8 April 2024].