Yadav Reginald Priaram

Student number: ST10356506

prog7311

POE Part 1

1815 words

8 April 2025

**Introduction**

Agri-Energy Connect is a digital platform aimed at farmers and agriculture specialists to bridge the gap between the agriculture sector and the green energy technology providers. This will enable agriculture experts across the globe to connect, exchange ideas, and contribute to sustainable farming initiatives. In this document, I will outline the project's vision, non-functional requirements, and design and architecture patterns.

**Vision**

The purpose of this project is to solve the problem of sustainability in agriculture by providing farmers with a technological platform which will connect them to worldwide sustainable farming hubs. This will benefit farmers as they will be able to communicate, interact and collaborate with other farmers and sustainability experts to share ideas, best practices, and latest news, keeping farmers informed which will promote sustainable farming.

**Non-Functional Requirements**

Non-Functional Requirements based on the FURPS+ model (Ziemek, 2022)

**Usability**

* Usability defines how user-friendly the app is and the overall user experience while using the app (Krüger, 2024), (Bhujbal, 2023). This is important for the Agri-Energy Connect application as it must be user-friendly and for users of different computer literacy skill levels.
* App must be user-friendly and easy to use to cater for users with limited technological experience.
* This will be **Implemented** by **creating a simple and clean UI** which will allow users to interact with the application easily. Example: the UI will have no distracting elements, bigger text, more legible fonts, colour-coding for more understanding. For extra convenience, **tutorials** will also be available when the application is first launched on the device. Example: when the app is first launched, there will be notes and guidance messages to help users navigate through the app.
* This will **Impact** the overall user experience by enhancing user engagement and reducing the applications learning curve.

**Reliability**

* Reliability in non-functional requirements refer to the system's ability to perform its required functions under specific conditions during a period of time(Reddy, 2024). It is important to ensure the Agri-Energy Connect application is reliable so that access to the application remains uninterrupted and avoids downtime.
* The system must be available at all times even during peak traffic and system outages. This will be **Implemented** by using redundant system and failover mechanisms such as load balancing to evenly distribute system load during peak traffic. Example: instead of having a single server, the system can make use of multiple servers to distribute the load to prevent overloading a single server. (Chiaramonte, 2024). Additionally, **redundant backup systems** can be put into place to ensure the application operation is undisrupted. Example: having 2 systems that carry out the same task so that if one system goes down, the other seamlessly takes control of the application.
* This will **Impact** the system reliability by ensuring operational continuity.

**Performance**

* Performance refers to how the platform reacts and behaves when put under load especially during peak traffic (Krüger, 2024). This is necessary to consider when creating the Agri-Energy Connect application as it needs to support real-time data sharing and a handle large number of users at the same time without slowing down or crashing.
* The app must handle platform operations smoothly without compromising performance.
* This can be **Implemented** by using an asynchronous task manager which will allow the system to run different tasks at the same time without compromising the main task.(Janna, 2023) Example: when users upload a post to the app, they immediately get a notification saying their post is uploading. The new post then gets uploaded to a message queue to get uploaded, which allows the main application traffic or thread to remain undisrupted as it is independent of other threads. After that happens, a separate worker method listens for the queue message and to process the post. Once all of this is done, the user gets a notification that their post was uploaded successfully.
* This will **Impact** the Agri-Energy Connect application by improving the app responsiveness and utilize resources more efficiently by running multiple tasks simultaneously.

**Scalability**

* In non-functional requirements, scalability refers to the applications ability to adapt to the growth of the platform without compromising the performance, reliability, or security of the application(Reddy, 2024). This is needed for the Agri-Energy Connect application because as the number of users increase, the system needs to expand its capacity to accommodate more users and maintain system performance.
* The app must be able to accommodate for the growth of the Agri-Energy Farming hub, this includes supporting the growth of the number of users, resources, farming content, and marketplace listings.
* This can be **Implemented** through cloud-based solutions such as Azure cloud storage for its flexible, automatic scaling. This means that as the Farming hub grows, the application will scale itself automatically using Azure(Farooq, Rana, Wanabrahman 2019).
* This will **Impact** and improve the overall quality of the application as it ensures system reliability, performance, and security as the application expands.
* **+ : Security**
* In NFR, security describes how the platform will protect the users sensitive information from data breaches and malicious activity(Krüger, 2024). This is essential for the Agri-Energy Connect application to ensure that user information such as transactional and personal data remains protected at all times.
* The app must securely save sensitive user data such as emails, passwords, contact information, banking information.
* This can be **Implemented** by using **role-based access control** to manage which users have access to sensitive information(Lindemulder, 2024). Example: an individual will have access to their passwords and contact information, no other user will be able to access this information. **Multi-factor authentication** can also be implemented to ensure nobody other than the account owner gains access to the account or account features by providing multiple sources of proof to prove their identity(Kosinski, 2024) . Example: users will have to enter their device password or biometrics, then verify their identity using an email to access their banking information from the app.
* This will **Impact** the safety of the app as user information will be safe from data breaches and theft.

**Design and Architecture Patterns**

For my architecture pattern, I have chosen to go with the Model-View Controller (MVC).

The Model-View Controller architecture is a software design pattern which separates an application into 3 different components(Necula, 2024), (Lasky, Jack, 2024)

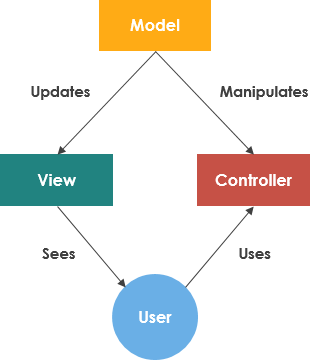
* **Model:** the model contains all the data variables for the application. It is responsible for receiving, storing, reading, and manipulating data from the application.
* **View:** the view is responsible for presenting the user interface (UI) to the user. This is where the users interact with the application. It retrieves data from the model and displays it to the user through the UI.
* **Controller:** the controller acts as a bridge between the model and the view, it processes the user input from the view and sends, updates, or retrieves the data from the model then sends it back to the view.

**Importance of the MVC Architecture Pattern (Owuordove, 2024)**

* By separating the applications components, it makes it easier for developers to code, debug, and manage application logic more efficiently.
* Separating the application components reduces the production time as developers can work on the model, view, and the controller simultaneously, boosting efficiency.

**How it will be integrated:**

* First, I’d create a new ASP.NET MVC project in Microsoft Visual Studio
* I’d then create the models folder and define the data structure by declaring the necessary variables for the Agri-Energy Connect application such as username, email, password, etc.
* Next, I’d create the views for the application by designing the user interface using HTML to present data to the user.
* Lastly, I’d create the controller methods using C# to process user input and communicate with the model to manipulate data and display it to the view.



**Figure 1**

*Figure 1 visually represents the flow of data within the MVC architecture pattern. From the diagram, we can see that the user interacts with the view which sends a request to the controller, the controller then updates the data within the model and is then displayed to the view which is then presented to the user.*

**How will MVC benefit the Agri-Energy Connect application? (Thakur, Pandey, 2019)**

* MVC makes the application easier to develop by separating it into 3 separate components, the model, view, and controller which all link to each other to handle user input. Example: from the view, the user will upload a post to the application which will be sent to the controller to be stored in the model. The model will then update controller and then controller will update the view to display the updated post.
* MVC allows the application to be easily scaled as it separates the application components, this makes it easier to add new features and UI updates without disrupting the other components.

**Conclusion**

In conclusion, the Agri-Energy Connect platform aims to bridge the gap between farmers and the technological farming hub to promote sustainable farming. The app will be created keeping a number of non-functional requirements in mind such as usability, reliability, performance, scalability, and security. The app will then be created using the Model-View Controller architecture pattern in Microsoft Visual Studio. The goal of the Agri-Energy application is to provide a platform for farmers and sustainability experts from around the world to interact with each other with ease.

**Reference list**

Bhujbal, S. (2023) *MVC design pattern*, *Medium*. Available at: <https://medium.com/@sandesh__30_/mvc-design-pattern-ff40d66990e3#:~:text=Implementing%20MVC%20in%20Practice,Models%20and%20Views%20as%20needed>. (Accessed: 08 April 2025).

Chiaramonte, M. (2024) *Application scalability: Ensuring performance and reliability*, *vFunction*. Available at: <https://vfunction.com/blog/application-scalability/#:~:text=Load%20balancing&text=Distributing%20incoming%20traffic%20across%20multiple,experience%20by%20reducing%20overall%20latency> (Accessed: 08 April 2025).

Janna, F. (2023) *Mastering asynchronous tasks and event-driven architectures in backend development*, *Medium*. Available at: <https://medium.com/@fatma_2377/dmastering-asynchronous-tasks-and-event-driven-architectures-in-backend-development-363d657cf1c9> (Accessed: 08 April 2025).

Kosinski, M. (2025) *What is multi-factor authentication?*, *IBM*. Available at: <https://www.ibm.com/think/topics/multi-factor-authentication#:~:text=Multi%2Dfactor%20authentication%20(MFA)%20is%20an%20identity%20verification,a%20temporary%20passcode%2C%20to%20prove%20their%20identity.&text=MFA%20systems%20add%20an%20extra%20layer%20of,of%20evidence%20to%20confirm%20a%20user’s%20identity>. (Accessed: 08 April 2025).

Krüger, G. (2024) *Non-functional requirements: What they do, examples, and best practices*, *Perforce Software*. Available at: <https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples> (Accessed: 08 April 2025).

Lasky and Jack (2024) *Model–view–controller (MVC): EBSCO*, *EBSCO Information Services, Inc. | www.ebsco.com*. Available at: <https://www.ebsco.com/research-starters/architecture/model-view-controller-mvc> (Accessed: 08 April 2025).

Lindemulder, G. (2025) *What is role-based access control (RBAC)?*, *IBM*. Available at: <https://www.ibm.com/think/topics/rbac#:~:text=Role%2Dbased%20access%20control%20(RBAC)%20is%20a%20model%20for,can’t%20touch%20firewall%20settings>. (Accessed: 08 April 2025).

Necula, S. (2024) *(PDF) exploring the model-view-controller (MVC) architecture: A broad analysis of market and Technological Applications*, *Research Gate*. Available at: <https://www.researchgate.net/publication/380197155_Exploring_The_Model-View-Controller_MVC_Architecture_A_Broad_Analysis_of_Market_and_Technological_Applications> (Accessed: 08 April 2025).

Owuordove (2024) *Hands-on guide to model-view-controller (MVC) architecture in python*, *Medium*. Available at: <https://medium.com/%40owuordove/hands-on-guide-to-model-view-controller-mvc-architecture-in-python-ec81b2b9330d> (Accessed: 08 April 2025).

Rana, M.E., Farooq, U. and Wanabrahman, W.N. (2019) *(PDF) scalability enhancement for cloud-based applications using software oriented methods*, *Research Gate*. Available at: <https://www.researchgate.net/publication/336030454_Scalability_Enhancement_for_Cloud-based_Applications_Using_Software_Oriented_Methods> (Accessed: 08 April 2025).

Reddy, A. (2024) *Architecture 101: Top 10 non-functional requirements (NFRS) you should be aware of*, *Medium*. Available at: <https://anjireddy-kata.medium.com/architecture-101-top-10-non-functional-requirements-nfrs-you-should-be-aware-of-c6e874bd57e0> (Accessed: 08 April 2025).

Thakur, R.N. and Panday, U.S. (2019) *(PDF) the role of model-view controller in Object Oriented Software Development*, *Research Gate*. Available at: <https://www.researchgate.net/publication/337128796_The_Role_of_Model-View_Controller_in_Object_Oriented_Software_Development> (Accessed: 08 April 2025).

Ziemek, M. (2022) *Documenting non-functional requirements using FURPS+*, *Marcin Ziemek, blog, article, documenting non-functional requirements using FURPS+*. Available at: <https://www.marcinziemek.com/blog/content/articles/8/article_en.html> (Accessed: 08 April 2025).

Figure 1: <https://www.visual-paradigm.com/servlet/editor-content/guide/uml-unified-modeling-language/what-is-model-view-control-mvc/sites/7/2019/09/model-view-controller.png>

**AI Detector**

**A screenshot of a computer

AI-generated content may be incorrect.**

[**https://www.zerogpt.com/**](https://www.zerogpt.com/)