**Thato Sebelemetja**

**Student Number: 10067544**

**PATHWAY: PROG7311**

**Lecturer: Handsome Mpofu**

**28 June 2024**

**Programming 3A – POE Part 1 (Amended)**

Table of Contents

[**1.** **INTRODUCTION:** 3](#_Toc170500622)

[**2.** **ANALYSIS OF NON-FUNCTIONAL REQUIREMENTS:** 3](#_Toc170500623)

[**2.1.** **How Non-Functional Requirements Impact Our Development Approach:** 4](#_Toc170500624)

[**3.** **ROLE OF DESIGN AND ARCHITECTURE PATTERNS:** 5](#_Toc170500625)

[**3.1.** **Integration of Design and Architecture Patterns:** 5](#_Toc170500626)

[**4.** **CONCLUSION:** 6](#_Toc170500627)

[**REFERENCES:** 7](#_Toc170500628)

# **INTRODUCTION:**

Agriculture and renewable energy are both essential industries for tackling worldwide issues like food security and climate change. Nevertheless, a gap commonly exists between these areas, resulting in inefficiencies and overlooked chances for creativity.

The goal of the Agri-Energy Connect Platform is to close this divide by establishing a digital environment that encourages cooperation between farmers and energy specialists. This platform aims to make it easier to share knowledge, resources, and innovations to support sustainable agriculture and renewable energy solutions.

This proposal therefore outlines the vision, objectives, and plan for the Agri-Energy Connect Platform of which the aim is to bridge the gap between agriculture and green energy through a digital ecosystem which will help facilitate collaboration and innovation amongst farmers and energy experts.

# **ANALYSIS OF NON-FUNCTIONAL REQUIREMENTS:**

* Scalability: The system will utilize horizontal scaling with cloud-based infrastructure for adaptable scalability. As the system expands, the workload will be spread out among various servers in the cloud, enabling smooth growth without any decrease in performance. See *Figure 1* below for a visual diagram of this.

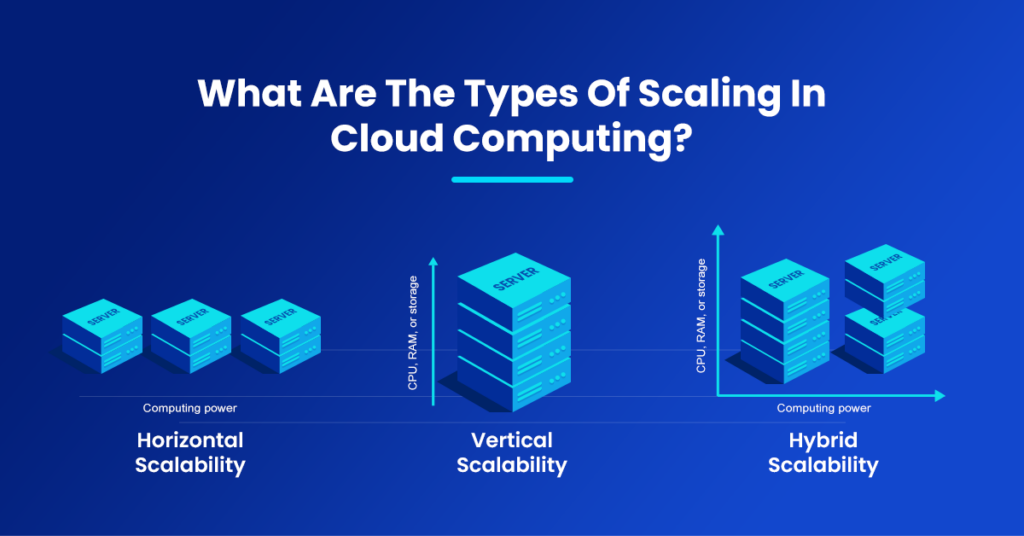


Figure 1: nOps. 2024. What Are The Types Of Scaling In Cloud Computing? (nOps, 2024)

* Security: Protecting user data and financial transactions is crucial, so I will be implementing encryption & authentication protocols, and role-based user access control to prevent unauthorized access.
* Usability: I will be implementing a user-friendly UI design with help guides as the user interacts with the platform to enhance usability. It will also be a responsive web design to ensure accessibility across most devices.
* Performance: Since this agricultural platform will require real-time data sharing to monitor crop growth, health etc., this will require above average performance. Utilizing aspects such as caching, I will be ensuring low latency and high responsiveness by caching static files and distributing files across multiple servers close to the users.

## **How Non-Functional Requirements Impact Our Development Approach:**

* In terms of scalability, it just means that during development, we must ensure that our system can smoothly handle any expansion of more users and data without slowing down or crashing.
* In terms of security, it just pushes us in the direction of ensuring that throughout development, we constantly check for any potential security risks and set up rules to ensure that only authorized individuals can access sensitive data.
* In terms of usability, it pushes us to ensure that during development, we test how easy it is for people to use our platform and make changes to improve it based on their feedback as we keep building.
* In terms of performance, during development, we must plan and build the platform in a way that makes the most efficient use of resources. We must also monitor how quickly our platform responds to certain actions and fix any issues that might slow it down.

# **ROLE OF DESIGN AND ARCHITECTURE PATTERNS:**

Design and architecture patterns are crucial as they ensure efficient scalability & maintainability. In this project, I will be using the Model-View-Controller (MVC) pattern and the Microservices architecture.

The MVC pattern is meant to help us write code that's easier to understand, maintain, and reuse. So I will be using it to write the code into separate sections for proper data handling, display, and managing user interactions for code maintainability and minimizing errors.

The microservices architecture however is a pattern that controls how different parts of our system interact with each other. Since our project will need to include aspects such as an online store and a forum for collaboration, instead of building a full application where all functionality is tightly built together, I will break it down into micro, independent services. This will allow for the platform to adapt to changes and increased demand more effectively.

## **Integration of Design and Architecture Patterns:**

I will be using the **MVC design pattern** to implement the code. Each component will have specific responsibilities:

* + Model: Since this component represents the data and logic of the platform, It will manage the data related to the farming practices, green energy solutions, and user profiles.
  + View: Since this component is responsible for the user interface, I will use it to create views with smooth UI’s to search for sustainable energy solutions, participate in forums etc.
  + Controller: Since this component is the bridge between the model and the view, it will be used to process user interactions and update the databases

Then regarding the **microservices design pattern**, I will use it to build the platform as separate services such as:

* + User profile service: This will manage all the profiles, process all new registrations and user authentication.
  + E-commerce service: This will manage all transactions within the green energy marketplace.
  + Forum service: This will be responsible for the forum discussions & project collaborations amongst the users. Refer to *Figure 2* below for a graphical representation of the microservices layout.

A diagram of a computer

Description automatically generated

Figure 2: Example of a Microservice Pattern

Both these patterns add a lot of value to the Agri-Energy Connect Platform as MVC promotes code reusability and maintainability, thus making it easier to manage and maintain the long-term growth of the platform, while microservices architecture allows for each service to be developed and updated individually, which helps with error isolation because if one service fails, it will not affect the entire platform, therefore minimizing the platform’s downtime.

# **CONCLUSION:**

This project is revolutionary, bringing sustainable farming and green energy solutions together. With my robust and scalable proposal which implements clever design and architecture strategies, the Agri-Energy Connect platform will lead the way in advancing agriculture into the digital era.

# **REFERENCES:**

nOps (2024). *nOps*. Available at: <https://www.nops.io/wp-content/uploads/2023/03/What-Are-The-Types-Of-Scaling-In-Cloud-Computing-nOps-1024x536.png> [Accessed 18 Apr. 2024].

nOps (2023). *What is Scalability in Cloud Computing? Types, Benefits, and Practical Advice*. [online] nOps. Available at: <https://www.nops.io/blog/cloud-scalability/> [Accessed 18 Apr. 2024].

altexsoft (2023). *Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices*. [online] AltexSoft. Available at: <https://www.altexsoft.com/blog/non-functional-requirements/> [Accessed 18 Apr. 2024].

Phelps, O. (n.d.). *Guide to Non-Functional Requirements: Types and Examples | Insight | Box UK*. [online] boxUK. Available at: <https://www.boxuk.com/insight/guide-to-non-functional-requirements-types-and-examples/> [Accessed 18 Apr. 2024].

Molstud (2024). *Web Development for Agriculture: Digital Solutions for Farmers*. [online] Molstud. Available at: <https://moldstud.com/articles/p-web-development-for-agriculture-digital-solutions-for-farmers> [Accessed 18 Apr. 2024].

Woke, G. (2023). *Optimizing Your Web Application Performance*. [online] Pieces. Available at: <https://code.pieces.app/blog/optimize-web-application-performance> [Accessed 18 Apr. 2024].

Walker, V. (2022). *14 software architecture design patterns to know*. [online] Red Hat. Available at: <https://www.redhat.com/architect/14-software-architecture-patterns> [Accessed 18 Apr. 2024].

Butani, A. (2020). *5 essential patterns of software architecture*. [online] Red Hat. Available at: <https://www.redhat.com/architect/5-essential-patterns-software-architecture> [Accessed 18 Apr. 2024].

Thorne, J. (2022). *7 Microservice Design Patterns to Use*. [online] Open Replay. Available at: <https://blog.openreplay.com/7-microservice-design-patterns-to-use/> [Accessed 18 Apr. 2024].