

PROG POE FINAL

ST10091991



June 28, 2024

VC Waterfall

Midrand

**Table of Contents**

PROG POE PART 1……………………………………………………….2

PROG POE PART 1 Feedback and Improvements……………………12

PROG POE PART 2……………………………………………………….24

PROG POE PART 2 Feedback and Improvements……………………25

PROG POE PART 3……………………………………………………….25

Introduction & Background………………………………………………..25

1. Performance optimization………………………………………………25

2. Preferred Development Methodology…………………………………28

3. Implementation of DevOps……………………………………………..29

4. Recommended Framework…………………………………………….30

5. Rundown…………………………………………………………………32

Reference List………………………………………………………………33

***PROG POE PART 1:***  
**Introduction & Background**

According to Sihlobo from The Conversation South African farmers face multiple issues that affect the agriculture industry that are applicable in South Africa such as:  
  
1. Energy cuts / Power cuts  
2. Land reformation  
3. Lack of progress on key regulations  
4. The financing of the sector

(Sihlobo, 2024)  
  
With these real-world issues stated and present in South Africa the agriculture industry demands a solution thus is where IT steps in (Sheldon, 2023). The production of a multi-functional agricultural records and processing system that is capable of sharing information amongst farmers, green energy experts, and enthusiasts to tackle the current issue found in South Africa, this solution is called the Agri-Energy Connect Platform. This is a multi-functional platform that uses the MVC (Model view control) architectural pattern which is easily scalable to any laptop-based operating system and with the optionality of an android and IOS operating system capability which allows the application to be scalable to mobile cellular devices (Sheldon, 2023).

**Vision and Objectives**

The vision of the Agri-Energy Connect Platform is to create a digital ecosystem that allows a group or community within the Agriculture and environment industry to share resources and data with regards to the land conditions and additional data that helps collaborate, innovate and sustain in the agriculture and renewable energy sectors in Sout Africa (Sharma, 2023).  
  
The given challenge will not have a simple solution however requires complex system integration and multi-functional application layers as well as security and scalability that allows multi access to many forms of devices (Sharma, 2023). The complexity of this system has been broken down into smaller sections to ensure better handle-meant of the development of this application by making critical objectives which aim to target critical success criteria which are used to build the application piece by piece until it is fully complete (Sharma, 2023). These Objectives are as followed:  
  
1. Facilitating knowledge exchange and resources between farmers, green energy experts and enthusiasts via multiple data storing options such as databases, files and dashboards that allows a user to comment on a topic or item (Sheldon, 2023).  
  
2. Teaching and learning sustainable agricultural practices as well as adopting green energy solutions to ensure health and global safety practices (Sheldon, 2023).

3. Ensure the system has special access to educators and trainers as well as having special access to training resources that help educate all necessary individuals. The system will also have special access points based of the specific user’s role that is assigned to the profile account which allows each user to be able to view exclusive features and shared features (Sheldon, 2023).

4. Projects will be connected to funding assets which allow a specific project to be viewed by potential investors which can decide to donate to the project to ensure that the projects have feasible funds at its disposal to ensure it gets implemented (Sheldon, 2023).

**Understanding the non-functional and functional needs**

**Non-functional needs:**  
1. **Security** - The platform needs to guarantee customer data and transactions are available, secure, and secret. To reduce security threats, steps like access restriction, encryption, and frequent security audits are crucial. Multiple applicable solutions such as hash password, user verification questions and additional features that allow security, encryption, and protection for the system.  
2. **Scalability** - Over time, the platform needs to be able to handle an expanding user population as well as rising content and transaction volumes. A few scalability factors are load balancing, database query optimization, and horizontal scaling. The application design will allow the system to be cross platform and compactible.  
3. **Usability** - The platform ought to be user-friendly and intuitive, accommodating users with different degrees of technical proficiency. Enhancing usability and user experience should be the goal of interactive components, navigation frameworks, and user interface design. With this functional need it is important to keep the system simple and not complex.  
4. **Performance** - Even with high loads, the platform should have quick reaction times and excellent responsiveness. To increase system performance, performance optimization strategies including database indexing, code profiling, and caching may be used. The application will be using a hybrid architecture pattern that will allow data to be pulled from the database and locally added to the machine to allow local debugging. The database will be a relational SQL database due to its high performance linking.  
5. **Reliability** - The platform must be stable and dependable, with little downtime and interruptions in service. To guarantee service continuity, disaster recovery plans, redundancy, and fault tolerance techniques must be in place. This is easily achievable as if the database is offline the system goes into an offline mode which allows the user to still be able to use the software but with limited functionality.  
6. **Accessibility** - Users with disabilities should be able to access the platform, and it should adhere to all applicable accessibility standards and guidelines. This could entail supporting keyboard navigation, offering alternative text for images, and being screen reader compatible.  
7. **Compliance** - The platform must abide by all applicable laws and regulations regarding consumer rights, privacy, and data protection. GDPR compliance, PCI-DSS compliance (for payment processing), and conformity to industry standards are a few examples of compliance measures.  
  
**Functional needs:**1. **User authentication** - It should be safe for users to create accounts, log in, and manage them. Different user roles may have different access levels to features and material, such as farmers, specialists in green energy, and fans. Via the system the specific user account will be assigned a designated role that will allow the user to view and have access to role specific content and features.  
2. **Content Management** -Administrators should be able to add, modify, and oversee a variety of content kinds on the platform, including project listings, articles, forums, and resources. Additionally, users ought to be able to submit content, but this could need to be moderated. Content will be able to be viewed, added, edited, or removed based of the role designated system.  
3. **Communication and collaboration** - It should be possible for users to communicate with one another via forums, comments, and messaging. Project workspaces and document sharing are examples of collaboration technologies that should promote knowledge sharing and teamwork. The specific features associated will allow all users to be able to interact with one another, follow projects and other additional aspects as this will be an open ground with no role restrictions.  
4. **Marketplace Functionality** - The platform ought to facilitate all aspects of the green energy marketplace's transactions, such as order administration, payment processing, and product listings. It should be safe for customers to peruse merchandise, evaluate costs, and complete transactions. This function specifically targets the marketing functionality which allows users to donate or follow the project via the in-built marketplace.  
5. **Educational Resources** - A wide variety of instructional resources, such as articles, webinars, videos, and courses, ought to be available to users. Features for classifying and arranging resources as well as monitoring user progress and interaction should be available on the platform. This functionality will be open to all roles however higher up roles will have special functions associated with them.  
6. **Project Management** - Projects pertaining to sustainable agriculture and renewable energy should be easy for users to create, oversee, and work together on. Task delegation, monitoring of progress, and milestone management are examples of project management capabilities. Designated project managers will be assigned to each project where an assignment system will be put in place.  
 **Role of the design and architecture patterns.**



(Horta, 2023)

* **Relevance of Design Patterns:** Because they encourage modularization, reusability, and maintainability in software development, design patterns like MVVM (Model-View-ViewModel) and MVC (Model-View-Controller) are extremely pertinent to the project (Horta, 2023).
* **Integration Strategies:** By organizing our software in accordance with best practices, dividing up our concerns, and utilizing frameworks and libraries that provide out-of-the-box support for these patterns, we will incorporate design patterns into the project (Horta, 2023).
* **Justification**: By encouraging code structure, lowering complexity, and fostering developer collaboration, these design principles improve the platform. We may increase the codebase's readability, maintainability, and extensibility by using accepted patterns (Horta, 2023).

**High level Plan**

A diagram of a diagram

Description automatically generated

The diagram provided above is a general plan that illustrates the individual components and how they are dependant on one another and the flow of data as well, breaking down the diagram it is separated into 3 main modules where the application module (Middle block) is where the users will be interacting from. The system will then branch out into the special features module if the specific user is an administrator or they system allows the specific user to add, remove or edit content / data from the system. The entire application will be using a database to store data and the data will be imported per user from the database to minimize connectivity to the database however certain features won’t be imported as that is an online database handling.

**Conclusion**

Selecting the Model-View-Controller (MVC) architecture for the Agri-Energy Connect platform has certain advantages that correspond well with the needs and goals of the project:   
  
1. **Modularity and Concern Separation**: Model-View-Controller (MVC) encourages the modular construction of applications by separating the presentation (view), data (model), and application logic (controller) into discrete parts. The better code organization, maintainability, and reusability that results from this separation of concerns makes it simpler to manage and grow the program over time (Saafan, 2023).  
  
2. **Scalability**: Because MVC is modular, it allows for the independent scaling of separate components, which promotes scalability (Saafan, 2023). It is possible to add new features to an application without affecting already-existing components as the platform develops, allowing for a smooth expansion to handle a larger user base, more content, and more transactions (Saafan, 2023).

3. **Flexibility and extensibility**: MVC offers an architecture that is both adaptable to future improvements and flexible enough to change as needs do (Saafan, 2023). By adding or changing controllers, models, and views, new functionality can be added without requiring significant alterations to the application's other components or the underlying architecture (Saafan, 2023).   
  
4**. Support for Several User Interfaces**: Model-View-Controller (MVC) enables the creation of several UIs that share the same application logic and data at the core (Saafan, 2023). The Agri-Energy Connect platform benefits greatly from this flexibility since it can accommodate a variety of users (such as farmers and experts in green energy) who access the platform through different devices and interfaces (like web browsers and mobile apps) (Saafan, 2023).

5. **Community Support and Ecosystem**: MVC is a popular architectural pattern that has a thriving ecosystem of tools, libraries, and frameworks in addition to broad community support (Saafan, 2023). Using MVC frameworks, like Python's Django, PHP's Laravel, or Java's Spring MVC, can speed up development, give access to best practices, and make routine chores like data binding, routing, and validation easier (Saafan, 2023).   
  
All things considered, the MVC architecture provides a strong base upon which to build the Agri-Energy Connect platform (Saafan, 2023). This allows developers to build a digital ecosystem that is scalable, maintainable, and extensible, thereby encouraging innovation, sustainability, and teamwork in the agricultural and renewable energy sectors (Saafan, 2023).  
  
**PROG POE PART 1 Feedback and Improvements:**  
**Introduction & Background**

According to Sihlobo from The Conversation, South African farmers face numerous challenges that significantly impact the agriculture industry. These issues include energy cuts, land reformation, lack of progress on key regulations, and inadequate financing of the sector (Sihlobo, 2024).

In response to these pressing issues, the agriculture industry is in search of effective solutions (Sheldon, 2023). One promising development is the Agri-Energy Connect Platform, a multi-functional system designed to share information among farmers, green energy experts, and enthusiasts. This platform addresses the current challenges in South Africa by providing a comprehensive solution for managing agricultural records and processes.

The Agri-Energy Connect Platform utilizes the MVC (Model-View-Controller) architectural pattern, ensuring scalability and compatibility with various laptop-based operating systems. Additionally, it offers optional support for Android and iOS, making it accessible on mobile devices (Sheldon, 2023). This scalability and accessibility are crucial for enhancing collaboration and knowledge sharing across the agriculture and green energy sectors in South Africa.  
**Vision and Objectives**

The vision of the Agri-Energy Connect Platform is to create a digital ecosystem that allows a group or community within the Agriculture and environment industry to share resources and data with regards to the land conditions and additional data that helps collaborate, innovate and sustain in the agriculture and renewable energy sectors in Sout Africa (Sharma, 2023).  
  
The given challenge will not have a simple solution however requires complex system integration and multi-functional application layers as well as security and scalability that allows multi access to many forms of devices (Sharma, 2023). The complexity of this system has been broken down into smaller sections to ensure better handle-meant of the development of this application by making critical objectives which aim to target critical success criteria which are used to build the application piece by piece until it is fully complete (Sharma, 2023). These Objectives are as followed:  
  
1. Facilitating knowledge exchange and resources between farmers, green energy experts and enthusiasts via multiple data storing options such as databases, files and dashboards that allows a user to comment on a topic or item (Sheldon, 2023).  
  
2. Teaching and learning sustainable agricultural practices as well as adopting green energy solutions to ensure health and global safety practices (Sheldon, 2023).

3. Ensure the system has special access to educators and trainers as well as having special access to training resources that help educate all necessary individuals. The system will also have special access points based of the specific user’s role that is assigned to the profile account which allows each user to be able to view exclusive features and shared features (Sheldon, 2023).

4. Projects will be connected to funding assets which allow a specific project to be viewed by potential investors which can decide to donate to the project to ensure that the projects have feasible funds at its disposal to ensure it gets implemented (Sheldon, 2023).

**Understanding the non-functional and functional needs**

In the development of the Agri-Energy Connect Platform, it's essential to distinguish between non-functional and functional requirements. Non-functional requirements (NFRs) describe the system's operational attributes such as security, scalability, and performance, while functional requirements (FRs) define specific behaviors or functions of the system. Below, we delve into each of these requirements, linking them to the Agri-Energy Connect system and explaining their implementation and impact.

**Non-Functional Requirements:**

Security**:** The Agri-Energy Connect Platform must ensure that customer data and transactions are secure, private, and accessible only to authorized users. This is achieved through:

* **Access Restriction**: Role-based access control (RBAC) assigns specific permissions to users based on their roles (e.g., farmers, green energy experts).
* **Encryption**: Data encryption in transit and at rest protects sensitive information.
* **Security Audits**: Regular security audits and vulnerability assessments help identify and mitigate potential threats.
* **Implementation Impact**: These measures prevent unauthorized access and data breaches, enhancing user trust and compliance with regulations.

Scalability: The platform must handle an increasing number of users and transactions over time. Key strategies include:

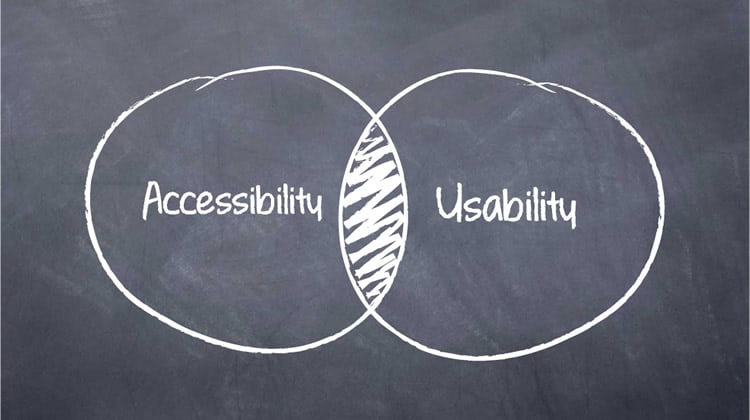
* **Load Balancing**: Distributing incoming network traffic across multiple servers to ensure no single server is overwhelmed.
* **Horizontal Scaling**: Adding more servers to handle increased load.
* **Database Optimization**: Efficient query handling and indexing to speed up data retrieval.
* **Implementation Impact**: Ensures the platform can grow and handle more users without performance degradation.



[This Photo](https://www.pngall.com/cybersecurity-png/) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

Usability: The platform should be easy to use for people with varying technical skills. This involves:

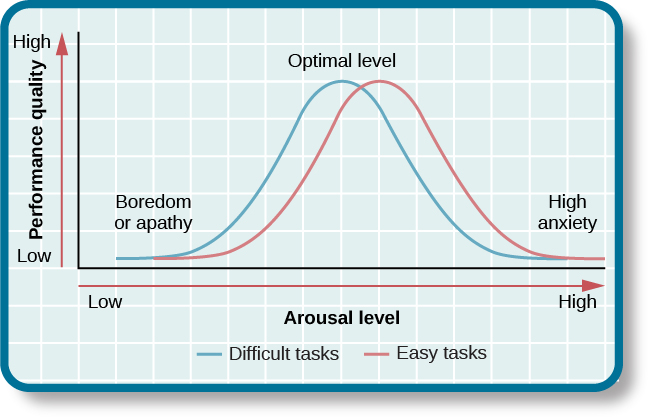
* **Intuitive Interface**: Simple, user-friendly design and clear navigation.
* **Interactive Components**: User feedback mechanisms to improve the interface based on actual use.
* **Implementation Impact**: Enhances user satisfaction and engagement, reducing the learning curve.



[This Photo](https://arcanusrandomstuffs.wordpress.com/2015/04/20/diez-directrices-para-mejorar-la-usabilidad-y-la-accesibilidad-de-tu-sitio/) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

Performance: The platform must remain responsive under high loads. Techniques include:

* **Code Profiling**: Regular analysis to identify and fix performance bottlenecks.
* **Caching**: Storing frequently accessed data in memory for faster retrieval.
* **Database Indexing**: Efficiently organizing data to speed up queries.
* **Implementation Impact**: Provides a fast, responsive user experience, crucial for retaining users.



[This Photo](https://courses.lumenlearning.com/wsu-sandbox/chapter/motivation/) by Unknown Author is licensed under [CC BY](https://creativecommons.org/licenses/by/3.0/)

Reliability: The system must be dependable with minimal downtime. This is ensured by:

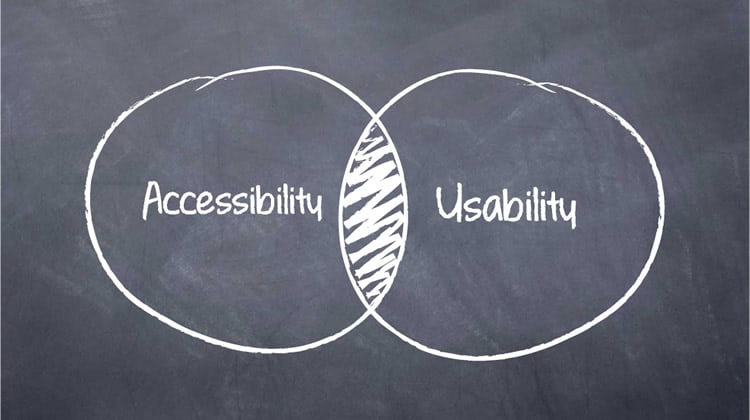
* **Disaster Recovery Plans**: Strategies for data backup and recovery.
* **Fault Tolerance**: Redundant systems to maintain service during failures.
* **Implementation Impact**: Guarantees continuous service availability, enhancing user trust.



[This Photo](https://usq.pressbooks.pub/socialscienceresearch/chapter/chapter-7-scale-reliability-and-validity/) by Unknown Author is licensed under [CC BY-SA-NC](https://creativecommons.org/licenses/by-nc-sa/3.0/)

Accessibility: The platform must be accessible to users with disabilities, adhering to standards such as WCAG. This includes:

* **Keyboard Navigation**: Ensuring all functionality can be accessed via keyboard.
* **Alternative Text**: Providing text descriptions for images.
* **Screen Reader Compatibility**: Ensuring content is readable by screen readers.
* **Implementation Impact**: Makes the platform usable for a wider audience, promoting inclusivity.



[This Photo](https://arcanusrandomstuffs.wordpress.com/2015/04/20/diez-directrices-para-mejorar-la-usabilidad-y-la-accesibilidad-de-tu-sitio/) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

Compliance: The platform must comply with legal standards for data protection and privacy, such as GDPR and PCI-DSS. This involves:

* **Data Protection Policies**: Implementing policies and procedures to protect user data.
* **Regular Audits**: Ensuring ongoing compliance through regular reviews.

**Implementation Impact**: Ensures legal compliance, avoiding fines and legal issues.  
A gavel and a book

Description automatically generated  
  
**Role of the design and architecture patterns.**In software development, design and architecture patterns are crucial because they support maintainability, reusability, and modularization. Model-View-ViewModel (MVVM) and Model-View-Controller (MVC) patterns are especially pertinent to our project because they facilitate efficient code organisation and the distinct division of responsibilities. By following these best practices, we may strategically employ frameworks and libraries that enable patterns out-of-the-box to integrate them into the project. This method lowers complexity, encourages developer collaboration, and improves readability and maintainability of the code, all of which contribute to a more reliable and scalable platform (Horta, 2023).



**Figure 1.1:** Model-View-Control Web architecture design (Horta, 2023)

* **Relevance of Design Patterns:** Because they encourage modularization, reusability, and maintainability in software development, design patterns like MVVM (Model-View-ViewModel) and MVC (Model-View-Controller) are extremely pertinent to the project (Horta, 2023).
* **Integration Strategies:** By organizing our software in accordance with best practices, dividing up our concerns, and utilizing frameworks and libraries that provide out-of-the-box support for these patterns, we will incorporate design patterns into the project (Horta, 2023).
* **Justification**: By encouraging code structure, lowering complexity, and fostering developer collaboration, these design principles improve the platform. We may increase the codebase's readability, maintainability, and extensibility by using accepted patterns (Horta, 2023).

**High level Plan**

A diagram of a diagram

Description automatically generated

The diagram provided above is a general plan that illustrates the individual components and how they are dependant on one another and the flow of data as well, breaking down the diagram it is separated into 3 main modules where the application module (Middle block) is where the users will be interacting from. The system will then branch out into the special features module if the specific user is an administrator or they system allows the specific user to add, remove or edit content / data from the system. The entire application will be using a database to store data and the data will be imported per user from the database to minimize connectivity to the database however certain features won’t be imported as that is an online database handling.

**Conclusion**

Selecting the Model-View-Controller (MVC) architecture for the Agri-Energy Connect platform has certain advantages that correspond well with the needs and goals of the project:   
  
1. **Modularity and Concern Separation**: Model-View-Controller (MVC) encourages the modular construction of applications by separating the presentation (view), data (model), and application logic (controller) into discrete parts. The better code organization, maintainability, and reusability that results from this separation of concerns makes it simpler to manage and grow the program over time (Saafan, 2023).  
  
2. **Scalability**: Because MVC is modular, it allows for the independent scaling of separate components, which promotes scalability (Saafan, 2023). It is possible to add new features to an application without affecting already-existing components as the platform develops, allowing for a smooth expansion to handle a larger user base, more content, and more transactions (Saafan, 2023).

3. **Flexibility and extensibility**: MVC offers an architecture that is both adaptable to future improvements and flexible enough to change as needs do (Saafan, 2023). By adding or changing controllers, models, and views, new functionality can be added without requiring significant alterations to the application's other components or the underlying architecture (Saafan, 2023).   
  
4**. Support for Several User Interfaces**: Model-View-Controller (MVC) enables the creation of several UIs that share the same application logic and data at the core (Saafan, 2023). The Agri-Energy Connect platform benefits greatly from this flexibility since it can accommodate a variety of users (such as farmers and experts in green energy) who access the platform through different devices and interfaces (like web browsers and mobile apps) (Saafan, 2023).

5. **Community Support and Ecosystem**: MVC is a popular architectural pattern that has a thriving ecosystem of tools, libraries, and frameworks in addition to broad community support (Saafan, 2023). Using MVC frameworks, like Python's Django, PHP's Laravel, or Java's Spring MVC, can speed up development, give access to best practices, and make routine chores like data binding, routing, and validation easier (Saafan, 2023).

All things considered, the MVC architecture provides a strong base upon which to build the Agri-Energy Connect platform (Saafan, 2023). This allows developers to build a digital ecosystem that is scalable, maintainable, and extensible, thereby encouraging innovation, sustainability, and teamwork in the agricultural and renewable energy sectors (Saafan, 2023).  
  
***PROG POE PART 2:***  
Original Link : <https://drive.google.com/file/d/11INOsidXdKyt1GwQ_Mv1lyt4du_v1aeT/view?usp=sharing>   
**PROG POE PART 2 Feedback and Improvements :**New Link : <https://drive.google.com/file/d/11INOsidXdKyt1GwQ_Mv1lyt4du_v1aeT/view?usp=sharing>  
  
**N.B. there will be a folder named Part 2 in which it will have the original folder containing the original part 2 code then the Updated folder containing the corrections. These links provided are backup links for in case the folders are corrupted or any other issues with the submission folder. In the case of the submission folder being too large to submit the report will only be sent and you may use the links to extract the source codes**  
***PROG POE PART 3:***

**Introduction & Background**  
The Agri-Energy Connect Platform connects agricultural and renewable energy, promoting resource sharing, cooperation, and education in a time when sustainable practices are essential. To guarantee the platform's success and scalability, this study describes optimisation tactics for the prototype, suggests an appropriate software development process, and talks about integrating DevOps and pertinent frameworks. It also offers a fair assessment of the technical solution put in place, emphasising both the business and technical values.  
  
**1. Performance optimization**  
To optimize the performance of the Agri-Energy Connect Platform prototype (ORA), we will consider the following improvements to be implemented:

|  |  |
| --- | --- |
| Improvement Index | Reason |
| Database Indexing | Optimising the database for efficient indexing is essential to accelerating data retrieval processes (C. Surabhi, 2020). At the expense of more storage and slower writes, indexes expedite data retrieval operations on a database table by constructing a data structure (C. Surabhi, 2020). Performance can be improved overall by ensuring that the columns that are most frequently requested are indexed, as this can drastically cut down on query response times (C. Surabhi, 2020). |
| Caching | By putting frequently visited data in memory, caching techniques can minimise database load and improve response times (C. Surabhi, 2020). By storing the outcomes of costly searches or computational operations in memory, techniques like in-memory caching (using programmes like Redis or Memcached) can significantly speed up subsequent requests. To guarantee data consistency and freshness, appropriate cache management strategies need to be implemented (C. Surabhi, 2020). |
| Load Balancing | Incoming network traffic is uniformly distributed among several servers when load balancing is used. This keeps any one server from acting as a bottleneck, improving the dependability and availability of the application. To maintain continuous service availability, load balancers can also keep an eye on the health of the servers and divert traffic from malfunctioning ones (C. Surabhi, 2020). |
| Efficient Query handling | Writing effective, organised, and resource-efficient SQL queries is the first step in optimising them. Using the right JOINs, avoiding SELECT \*, filtering data early using WHERE clauses, and making advantage of database-specific features like stored procedures are some of the techniques. Effective query optimisation and indexing can drastically lower database load, enhancing system performance (C. Surabhi, 2020). |

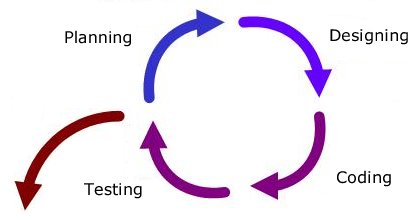
(C. Surabhi, 2020)  
Additional guidelines that can be followed for the final software development:

|  |  |
| --- | --- |
| Guideline | Reason |
| Security | Protecting consumer information and transactions requires the implementation of strong security measures (Hume, 2018). This includes vulnerability assessments, frequent security audits, encryption (in transit and at rest), and access controls. To avoid common vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF), secure coding techniques should be adhered to (Hume, 2018). |
| Scalability | Scalability in platform design guarantees that the system can support a growing number of users (Hume, 2018). Methods like vertical scaling—adding additional resources to already-existing servers—and horizontal scaling—adding new servers—should be used. Scalability can also be attained by utilising cloud-native technologies and microservices architecture (Hume, 2018). |
| Usability | It is essential to design an intuitive and user-friendly interface to support users with varying degrees of technical expertise (Hume, 2018). Usability testing ought to be carried out to get input and make the required modifications (Hume, 2018). A strong user experience is mostly dependent on accessible information, straightforward navigation, and consistent design principles (Hume, 2018). |
| Reliability | Having disaster recovery strategies and fault tolerance methods in place is necessary to guarantee that the platform is reliable and has little downtime (Hume, 2018). It is advisable to use methods like failover solutions, automated backups, and data replication. These ideas are regularly tested to make sure they work well in real-world situations (Hume, 2018). |

(Hume, 2018)  
**2. Preferred Development Methodology**  
For the further future development of the application a suggestion of adopting the **Agile Software Development Methodology** is recommended due to the following reasons:

|  |  |
| --- | --- |
| Factor | Reason |
| Flexibility | Agile development methodology facilitates iterative development and ongoing feedback, both of which are essential for integrating modifications and enhancements during the development process (Johnson, 2024). Teams can adjust to shifting priorities and requirements using agile approaches (such as Scrum or Kanban), which keeps the project in line with business requirements (Johnson, 2024). |
| Risk Management | Early in the development cycle, risks can be identified and addressed because to Agile's iterative nature (Johnson, 2024). Regular reviews, continuous integration, and testing all assist in reducing risks and making sure the project stays on schedule (Johnson, 2024). |
| Improved Quality | Agile's continuous testing and integration procedures guarantee that quality is upheld across the whole development process (Johnson, 2024). Pair programming, code reviews, and automated testing are techniques that improve code quality and lower the risk of errors (Johnson, 2024). |
| Client Collaboration | Agile places a strong emphasis on working together with stakeholders to ensure that their needs and expectations are met by the finished product (Johnson, 2024). Stakeholders are included in the development process through regular meetings and feedback loops (like sprint reviews and retrospectives), which improves transparency and alignment (Johnson, 2024). |

(Johnson, 2024)  
  
**Figure 1.3a**: Image of a simplified Agile Software Development Life Cycle (Microsoft, 2024)

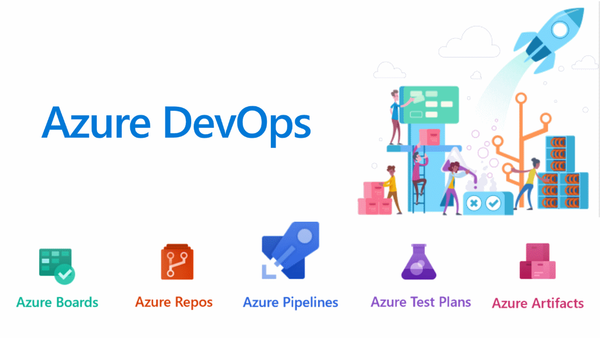


[This Photo](https://baravalle.com/presentations/cn4103/topic6.html) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

**3. Implementation of DevOps**  
DevOps **is recommended** to be implemented in this project due to the following reasons:

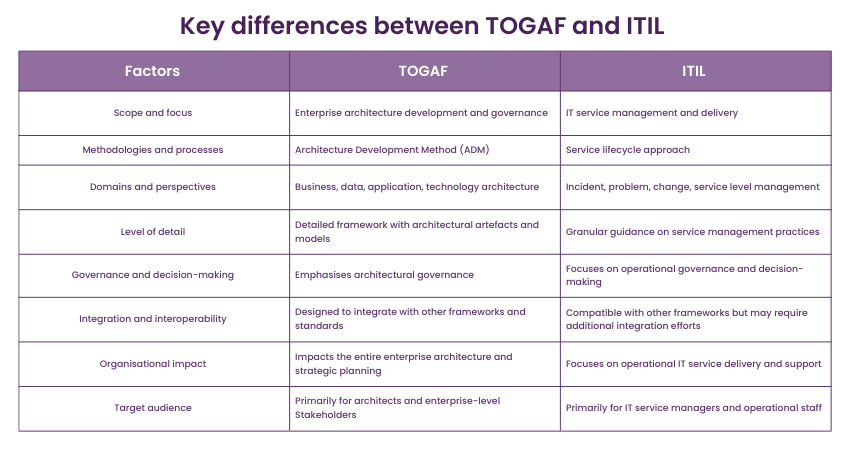
|  |  |
| --- | --- |
| Factor | Reason |
| Enhanced Collaboration | Improved communication and cooperation between the development and operations teams are fostered by DevOps (Patel, 2023). This change in culture ensures a shared responsibility for the success of the product and breaks down organisational silos, which is in line with Agile's collaborative character (Patel, 2023). |
| Improved Efficiency | Repetitive task automation and monitoring contribute to increased productivity and less human error (Patel, 2023). Teams may concentrate on more important activities by automating build, test, and deployment procedures with the help of tools like CircleCI, Jenkins, and GitLab CI (Patel, 2023). |
| Scalability and Flexibility | The system can manage higher loads and scale well thanks to DevOps principles (Patel, 2023). Automation and consistency in infrastructure management are made possible by Infrastructure as Code (IaC) tools such as Terraform and Ansible, which guarantee that the system can adjust to fluctuating demands (Patel, 2023). |
| Continuous Integrations and enhancements | Code updates are tested and deployed automatically thanks to DevOps techniques like CI/CD (Patel, 2023). Whereas CD automates the release process, continuous integration (CI) entails regularly integrating code changes into a shared repository (Patel, 2023). As a result, releases happen more quickly and dependably, and the effects of modifications are felt right away (Patel, 2023). |

(Patel, 2023)  
  
**Figure 1.3b**: Image of Microsoft Azure DevOps (Microsoft, 2024)  
  
**4. Recommended Framework**  
A recommendation of **both** the **ITIL** and **TOGAF Frameworks** are to be used in the application due to the following reasons:



[This Photo](https://www.hugobarona.com/a-success-story-of-improved-productivity-and-quality-of-work-processes-using-azure-devops/) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

|  |  |
| --- | --- |
| ITIL (Information Technology Infrastructure Library) | TOGAF (The Open Group Architecture Framework) |
| IT service management is the focus of ITIL, which makes sure that the platform offers top-notch IT services that satisfy business requirements (Simplilearn, 2024). It consists of procedures for overseeing service delivery, making certain that services are dependable, effective, and in line with corporate goals (Simplilearn, 2024). | An organised method for creating, organising, putting into practice, and overseeing a corporate information architecture is offered by TOGAF (Simplilearn, 2024). It guarantees that every facet of the architecture is considered, offering a thorough understanding of the IT environment within the company (Simplilearn, 2024). |
| Best practices for managing IT services are provided by ITIL, guaranteeing dependability, effectiveness, and ongoing development (Simplilearn, 2024). To guarantee reliable and superior service delivery, it contains recommendations for incident, issue, change, and service-level management (Simplilearn, 2024). | The platform's scalability and ability to adapt to future expansion and changes are ensured by TOGAF (Simplilearn, 2024). It contains instructions for designing scalable architectures that are adaptable to changing business requirements (Simplilearn, 2024). |
|  | By addressing every facet of enterprise architecture, TOGAF guarantees that the platform is in line with corporate objectives and procedures (Simplilearn, 2024). It contains procedures for creating, putting into practice, and keeping up the architecture, making sure it is efficient and up to date (Simplilearn, 2024). |

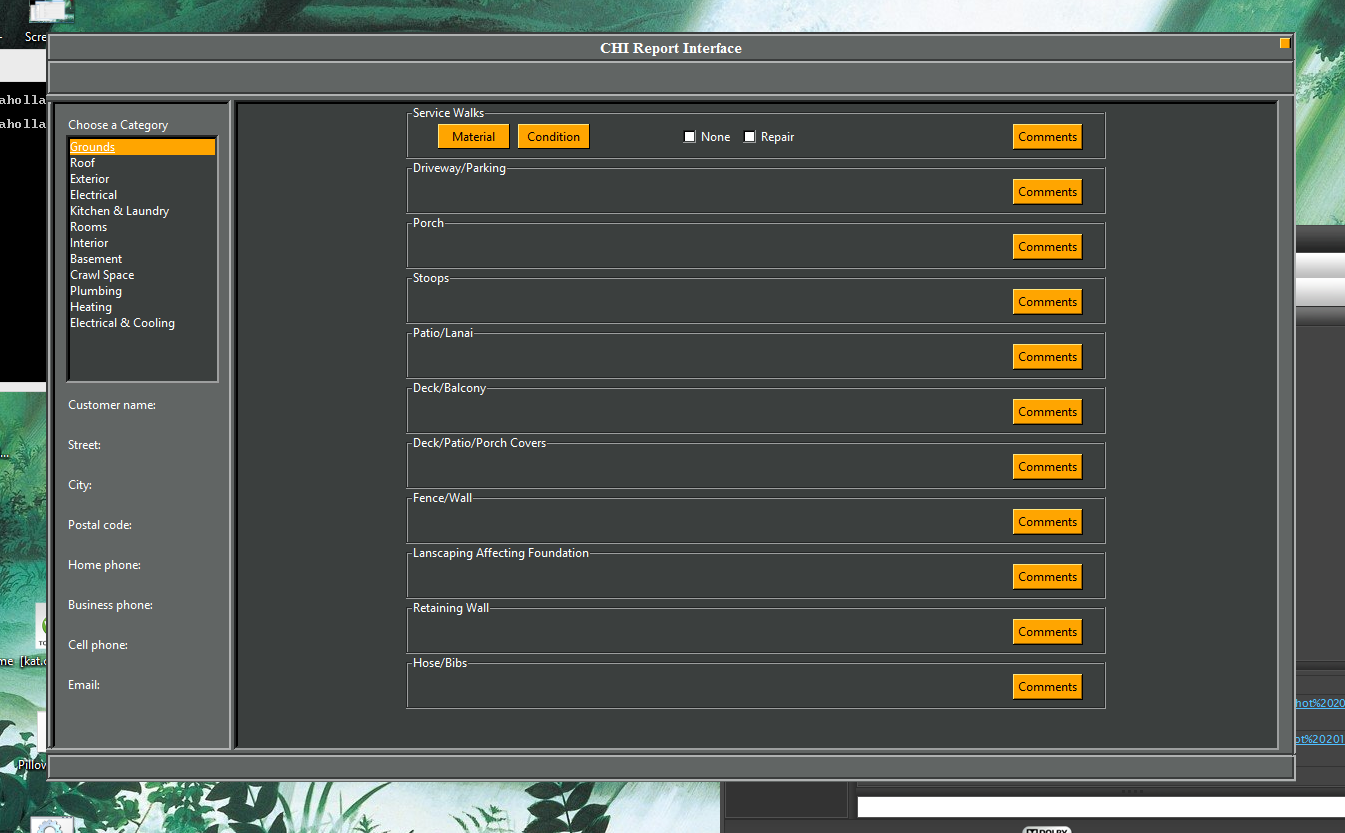
(Simplilearn, 2024)  
**Figure 1.3c:** Table comparison between ITIL and TOGAF (TheKnowledgeAcademy, 2024)  
  
A solid, scalable, and well-managed platform that supports corporate goals and offers top-notch IT services will be ensured by combining TOGAF with ITIL (Simplilearn, 2024).

**5. Rundown**  
The Agri-Energy Connect Platform is intended to serve as a multipurpose digital ecosystem for the renewable energy and agriculture industries. It employs the architectural pattern known as Model-View-Controller (MVC), which improves maintainability, scalability, and modularity. Farmers, experts in green energy, and enthusiasts can exchange resources and information on sustainable methods and land conditions on this platform.   
  
**Prototype**  
The application prototype that was developed illustrated the core functionalities of the application; it demonstrates the simplicity of the application with minimum complexities to properly illustrate the flow of operations the application offers.  
  
**Current Prototype Characteristics**

The following table provides the key functional requirements of the application that the prototype has provided:

|  |  |
| --- | --- |
| Important characteristics | Reason |
| User authentication | Safe account setup and administration with feature-specific access determined by role. |
| Material management | Users and administrators can add, edit, and manage material, including forums, articles, and project listings. |
| Collaboration and Communication | Message, comment, and forum features encourage conversation and the exchange of knowledge. |
| Marketplace Features | The green energy marketplace allows users to peruse, assess, and do business. |
| Educational Resources | Users have access to a vast array of educational resources that promote lifelong learning and growth. |
| Project management tools | These are programmes for planning, directing, and working together on renewable energy and sustainable agriculture initiatives. |

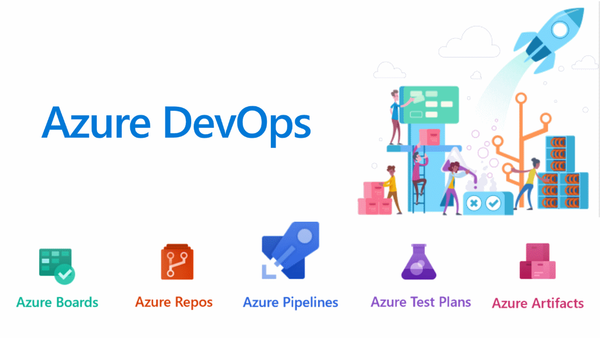
**Enhancements to the Prototype**  
The following provides a list of additional improvements that can be implemented into the application:  
  
1. **Enhance GUI** (Graphical User Interface) - To ensure a much more friendlier user interface and easier manoeuvrability  
  
**Figure 1.4**: Image of an enhanced GUI (Microsoft, 2024)  
2. **Enhance Database security and authentications** – This ensures that the current sensitive information must be saved via encryption algorithms.  
  
**Figure 1.5** : Image of a System Access Control Console (Microsoft, 2024)  
3. **Enhance the general functionality** – Small updates can be implemented into the system such as profile deleting and editing of data.  
  
**Figure 1.6** : Image of Microsoft Azure DevOps (Microsoft, 2024)  
  
The scalability of the platform to several platforms, such as laptops, Android, and iOS, guarantees accessibility and usability for diverse user groups. Both online and offline functionalities are made possible by the hybrid architecture, which guarantees constant availability even in the event of connectivity problems.



[This Photo](http://stackoverflow.com/questions/36685835/how-to-create-an-impressive-gui-in-python-using-tkinter) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)



[This Photo](https://www.anupamanatarajan.com/2019/12/data-platform-tips-15-advanced-data.html) by Unknown Author is licensed under [CC BY](https://creativecommons.org/licenses/by/3.0/)



[This Photo](https://www.hugobarona.com/a-success-story-of-improved-productivity-and-quality-of-work-processes-using-azure-devops/) by Unknown Author is licensed under [CC BY-NC](https://creativecommons.org/licenses/by-nc/3.0/)

**Reference List**

C. Surabhi (2020) *5 ways to tune high performance to your ASP.NET MVC applications*, *Medium*. Available at: https://surabhigwl.medium.com/5-ways-to-tune-high-performance-to-your-asp-net-mvc-applications-fc25c6a3a382 (Accessed: 26 June 2024).

Horta, J.Y.M. (2023) *The importance of good architecture and Design Patterns for high-performance software*, *LinkedIn*. Available at: https://www.linkedin.com/pulse/importance-good-architecture-design-patterns-software-jos%C3%A9-yadiel#:~:text=Good%20architecture%20and%20design%20patterns%20also%20help%20ensure%20that%20the,and%20design%20patterns%20is%20reliability. (Accessed: 17 April 2024).

Hume, D. (2018) *Improving the performance of your MVC Application*, *Dean Hume’s Blog*. Available at: https://deanhume.com/improving-the-performance-of-your-mvc-application/ (Accessed: 26 June 2024).

Johnson, J. (2024) *Agile web design process*, *O8*. Available at: https://www.o8.agency/blog/marketing-team/agile-web-design#:~:text=Agile%20web%20design%20inherently%20reduces,keeping%20the%20project%20on%20track.&text=Agile%20web%20design%20can%20also,flexible%20and%20adaptable%20process%20structure. (Accessed: 26 June 2024).

Patel, M. (2023) *Understand key role of devops in web development*, *WPWeb Infotech*. Available at: https://wpwebinfotech.com/blog/role-of-devops-in-web-development/#:~:text=DevOps%20significantly%20enhances%20the%20speed,time%20from%20development%20to%20deployment. (Accessed: 26 June 2024).

Saafan, A. (2023) *What are the benefits of using the model-view-controller (MVC) framework in ASP.NET development?* *LinkedIn*. Available at: https://www.linkedin.com/pulse/what-benefits-using-model-view-controller-mvc-framework-amr-saafan-rjmdf/ (Accessed: 17 April 2024).

Sharma, N. (2023) *Top challenges of developing cross-platform mobile applications*, *kellton*. Available at: https://www.kellton.com/kellton-tech-blog/top-challenges-of-developing-cross-platform-mobile-apps (Accessed: 16 April 2024).

Sheldon, R. (2023) *What is model-view-controller (MVC)?: Definition from TechTarget*, *WhatIs*. Available at: https://www.techtarget.com/whatis/definition/model-view-controller-MVC#:~:text=In%20programming%2C%20model-view-,a%20specific%20set%20of%20tasks. (Accessed: 15 April 2024).

Sihlobo, W. (2024) *Farming in South Africa: 6 things that need urgent attention in 2023*, *The Conversation*. Available at: https://theconversation.com/farming-in-south-africa-6-things-that-need-urgent-attention-in-2023-197772 (Accessed: 15 April 2024).

Simplilearn (2024) *Itil vs. TOGAF: Which framework is better in 2024?*, *Simplilearn.com*. Available at: https://www.simplilearn.com/itil-vs-togaf-article (Accessed: 26 June 2024).

TheKnowledgeAcademy (2024) *Togaf vs ITIL: Which one is better?*, *The Knowledge Academy - Online certification training courses provider*. Available at: https://www.theknowledgeacademy.com/blog/togaf-vs-itil/ (Accessed: 26 June 2024).