Joshua Ryan Pillay

St10175525

a Report about the Prototype, Performance, and Methodologies For Agri-Energy Connect

## Table of Contents

1. Introduction …………….Page 2
2. Performance Optimization of the Prototype …………….Page 2
   * 2.1 Strategies for Optimizing Prototype Performance …………..Page 2
   * 2.2 Guidelines for Final Software Performance ……………..Page 3
3. Recommended Software Development Methodology …………….Page 5
   * 3.1 Agile Methodology ……………Page 5
   * 3.2 Rationale for Agile …………. Page 5
4. Recommendation for DevOps Implementation …………… Page 7
   * 4.1 Benefits of DevOps …………… Page 7
   * 4.2 Integration with Agile ………….Page 7
5. Recommended Architecture Frameworks ……….. Page 8
   * 5.1 TOGAF Framework ………….Page 8
   * 5.2 Rationale for TOGAF …………Page 8
6. Technical Solution Description for Marketing Team ……….Page 9
   * 6.1 Overview of the Prototype ……….Page 9
   * 6.2 Technical Decisions and Business Value ………..Page 9
7. Conclusion …………Page 10
8. References ………… Page 11

**1. Introduction**

The Agri-Energy Connect Platform is a transformative digital ecosystem designed to unite South African farmers and green energy experts in pursuit of sustainable agricultural practices and renewable energy integration. This report addresses the third part of the Portfolio of Evidence (PoE) for the PROG7311 module, building on the previously submitted Part 1 (Requirements and Design Patterns) and Part 2 (Prototype Web Application). The report responds to the marketing team’s request for a comprehensive analysis to strengthen the proposal for the Agri-Energy Connect Platform. It covers five key areas: optimizing the prototype’s performance, selecting a software development methodology, evaluating DevOps implementation, choosing an architecture framework, and describing the prototype’s technical solution for marketing purposes. The tone is formal yet accessible, balancing technical precision with business-oriented clarity to appeal to the bid committee and marketing team. Diagrams are included to enhance understanding, and the structure aligns with Part 1 to ensure a cohesive proposal.

**2. Performance Optimization of the Prototype**

**2.1 Strategies for Optimizing Prototype Performance**

The Agri-Energy Connect prototype, developed using ASP.NET Core 8 and SQL Server, demonstrates core functionalities such as farmer and product management, secure authentication, and a responsive UI. However, to ensure scalability and responsiveness, several optimization strategies can be applied:



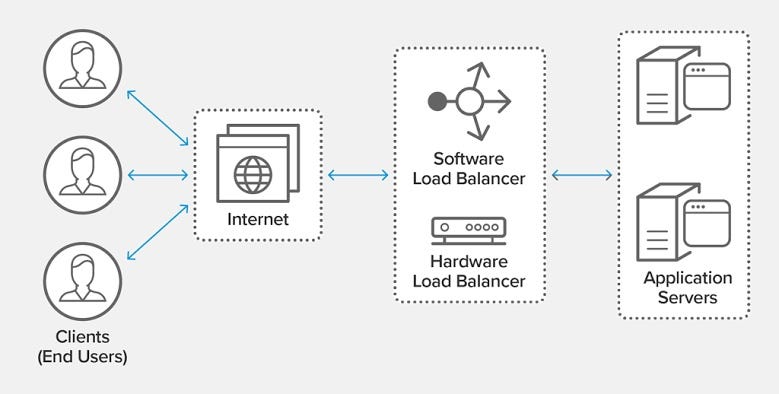
The image showcases the topics related to Performance Optimization (Author’s own work, 2025) as shown in Fig. 1 (Corporation, 2025).

* **Database Optimization**: The prototype uses Entity Framework Core to interact with the SQL Server database. To enhance query performance, database indexing was implemented on frequently queried fields (e.g., product category and production date). However, further optimization can be achieved by:
  1. Using stored procedures for complex queries, such as filtering products by date range and category, to reduce execution time.
  2. Implementing database partitioning to manage large datasets, ensuring faster retrieval as the number of farmers and products grows (Anghel, 2019).
* **Caching**: The prototype currently retrieves data directly from the database for each request. Introducing in-memory caching (e.g., using Redis) for frequently accessed data, such as farmer profiles or product listings, can reduce database load and improve response times. For example, caching the list of farmers on the employee dashboard can minimize redundant queries (Anghel, 2019).
* **Asynchronous Programming**: The prototype leverages asynchronous methods in ASP.NET Core for database operations (e.g., async/await in product filtering). To further optimize, all I/O-bound operations, such as API calls to external services (e.g., weather data integration), should be made asynchronous to prevent thread blocking and improve scalability under high user traffic (Anghel, 2019).
* **Frontend Optimization**: The UI uses Bootstrap 5 and jQuery for responsiveness and interactivity. To reduce page load times:
  1. Minify CSS and JavaScript files to decrease file sizes.
  2. Implement lazy loading for images (e.g., placeholders on the homepage) to improve initial load speed, especially on mobile devices.
  3. Use a Content Delivery Network (CDN) for static assets like Font Awesome icons to reduce latency (Anghel, 2019).
* **Load Balancing**: The prototype runs on a single server instance. To handle increased traffic, deploying the application on a cloud platform like Azure with load balancing can distribute requests across multiple servers, ensuring consistent performance during peak usage, such as when farmers access training resources during planting seasons (Anghel, 2019).

**2.2 Guidelines for Final Software Performance**

To ensure acceptable performance in the final Agri-Energy Connect Platform, the following guidelines should be followed:

* **Performance Benchmarking**: Establish performance benchmarks for key operations, such as loading the product listing page in under 2 seconds or processing a farmer profile creation in under 1 second. Use tools like Azure Application Insights to monitor real-time performance metrics and identify bottlenecks (Rajeshvelmani, 2023).
* **Scalability Planning**: Design the system to handle at least 10,000 concurrent users, anticipating growth in the agricultural sector. Adopt a microservices architecture (as proposed in Part 1) to allow independent scaling of components like the Green Energy Marketplace and Sustainable Farming Hub (Rajeshvelmani, 2023).
* **Regular Load Testing**: Conduct load testing using tools like JMeter to simulate high-traffic scenarios, such as simultaneous product filtering by employees or marketplace transactions. Address any identified weaknesses, such as slow database queries or server overload (Rajeshvelmani, 2023).
* **Optimized Resource Usage**: Minimize server resource consumption by:
  1. Using connection pooling for database connections to reduce overhead.
  2. Implementing efficient algorithms for data processing, such as optimized sorting for product filtering.
  3. Compressing data (e.g., Gzip compression for HTTP responses) to reduce bandwidth usage (Rajeshvelmani, 2023).
* **Continuous Monitoring**: Integrate monitoring tools to track CPU usage, memory consumption, and response times. Set up alerts for anomalies, such as sudden spikes in latency, to enable proactive resolution and maintain a seamless user experience.



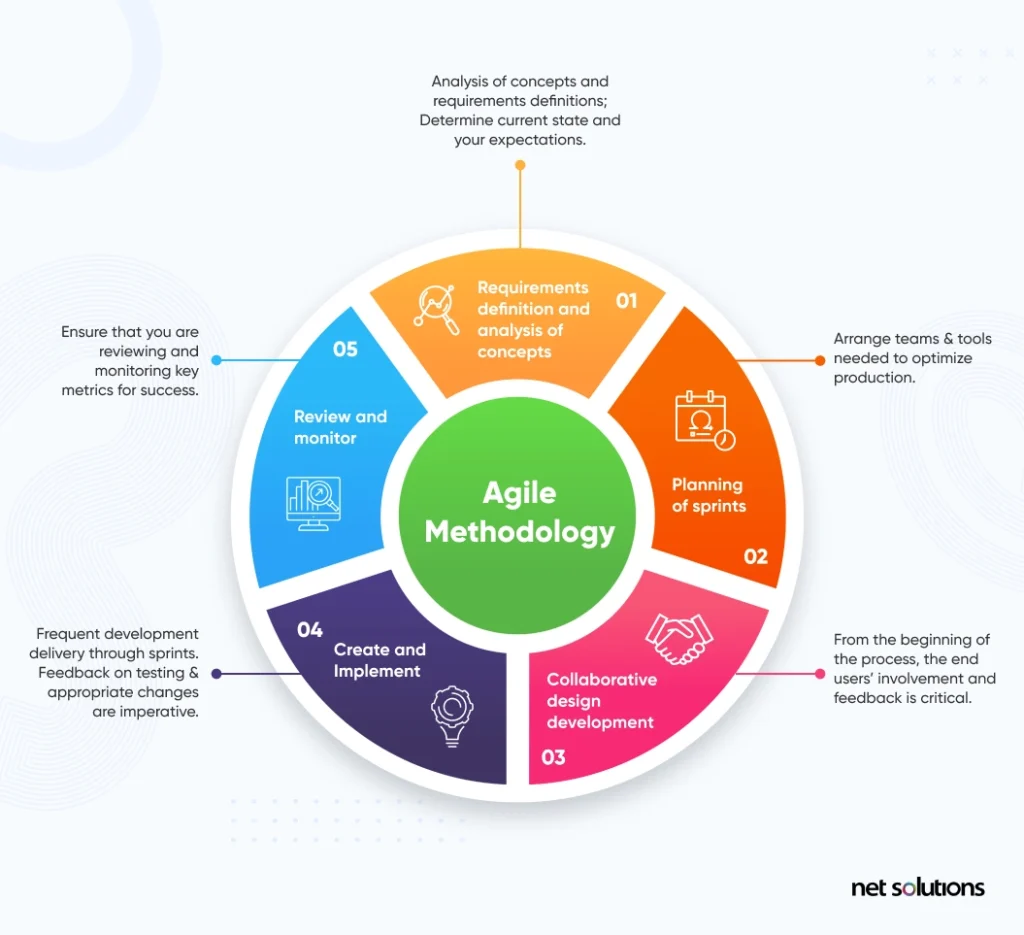
The image showcases how load balancing, caching, and asynchronous processing work together (Medium, 2023) for the Agri-Energy Connect Platform, as shown in Fig. 2

This diagram illustrates how load balancing, caching, and asynchronous processing work together to optimize performance, ensuring fast and reliable access to the platform’s features (Rajeshvelmani, 2023).

**3. Recommended Software Development Methodology**

**3.1 Agile Methodology**

For the Agri-Energy Connect Platform, the Agile methodology, specifically Scrum, is recommended. Agile emphasizes iterative development, collaboration, and adaptability, aligning with the platform’s need for continuous feedback and evolving requirements.



The image showcases Agile Methodology and the various aspects of it Is explained and further explained below,(Net Solutions, 2022) as shown in Fig. 3.

**3.2 Rationale for Agile**

Agile is well-suited to the Agri-Energy Connect project for the following reasons:

* **Flexibility for Evolving Requirements**: The platform’s requirements, such as integrating new green energy solutions or expanding educational resources, may change based on user feedback or market trends. Agile’s iterative sprints (e.g., 2-week cycles) allow the team to prioritize and incorporate new features without disrupting the development process (Milne, 2022).
* **User-Centric Development**: The scenario emphasizes a user-friendly platform for diverse users (farmers and energy experts). Agile’s focus on user stories and regular feedback loops ensures that features like the Sustainable Farming Hub or Green Energy Marketplace are tailored to user needs. For example, usability testing during each sprint can refine the UI based on farmer input (Milne, 2022).
* **Collaboration with Stakeholders**: Agile promotes close collaboration with stakeholders, such as the bid committee and marketing team. Regular sprint reviews provide opportunities to demonstrate progress (e.g., a working prototype of the marketplace) and adjust priorities based on stakeholder input, ensuring the platform aligns with business goals (Milne, 2022).
* **Risk Mitigation**: By delivering working increments at the end of each sprint, Agile reduces the risk of major failures. For instance, if the interactive forums encounter performance issues, they can be addressed early rather than at the project’s end, saving time and resources (Milne, 2022).
* **Alignment with Client Needs**: The client’s request for a comprehensive yet flexible proposal suggests a need for adaptability. Agile’s ability to deliver a minimum viable product (MVP)—such as a functional prototype with core features like product management—followed by iterative enhancements, meets this need effectively (Milne, 2022).

Agile’s strengths map directly to the client’s need for a scalable, user-friendly platform that evolves with the agricultural sector’s demands, making it the ideal choice over rigid methodologies like Waterfall, which are less adaptable to change.

**4. Recommendation for DevOps Implementation**

**4.1 Benefits of DevOps**

DevOps, a set of practices combining development and operations, is strongly recommended for the Agri-Energy Connect Platform. Its benefits include:

* **Faster Delivery**: DevOps enables continuous integration and continuous deployment (CI/CD), allowing frequent updates to the platform. For example, new training resources or marketplace features can be deployed rapidly without downtime (BrowserStack, 2024).
* **Improved Quality**: Automated testing and monitoring in DevOps ensure high-quality releases. Unit tests for product filtering or integration tests for farmer profile creation can catch bugs early, enhancing reliability (BrowserStack, 2024).
* **Enhanced Collaboration**: DevOps fosters collaboration between development and operations teams, aligning with the platform’s goal of uniting diverse stakeholders. Shared tools like Azure DevOps can streamline communication, ensuring smooth deployment of features like the Sustainable Farming Hub (BrowserStack, 2024).
* **Scalability and Reliability**: DevOps practices, such as infrastructure as code (IaC) and containerization (e.g., Docker), enable scalable and reliable deployments. This ensures the platform can handle increased traffic during peak usage, such as when farmers access water conservation methods (BrowserStack, 2024).

**4.2 Integration with Agile**

DevOps complements Agile by enhancing its iterative and collaborative nature:

* **CI/CD in Sprints**: Agile sprints produce incremental features, which DevOps pipelines can automatically build, test, and deploy. For example, a sprint delivering the Green Energy Marketplace can use a CI/CD pipeline to deploy the feature to a staging environment for user testing (BrowserStack, 2024).
* **Feedback Loops**: DevOps monitoring tools provide real-time performance data, which can inform Agile sprint planning. If the interactive forums show high latency, the next sprint can prioritize performance optimization (BrowserStack, 2024).
* **Automation for Agility**: DevOps automates repetitive tasks like testing and deployment, allowing Agile teams to focus on feature development. Automated tests for role-based access control can ensure security without slowing down sprints (BrowserStack, 2024).
* **Cultural Alignment**: Both Agile and DevOps emphasize collaboration and continuous improvement. DevOps tools like GitHub Actions can support Agile’s daily stand-ups by providing visibility into build statuses, fostering a unified team culture (BrowserStack, 2024).

**5. Recommended Architecture Frameworks**

**5.1 TOGAF Framework**

The Open Group Architecture Framework (TOGAF) is recommended for the Agri-Energy Connect Platform. TOGAF provides a structured approach to enterprise architecture, aligning IT solutions with business goals.

**5.2 Rationale for TOGAF**

TOGAF is the most suitable framework compared to ITIL or Zachman for the following reasons:

* **Business-IT Alignment**: TOGAF’s Architecture Development Method (ADM) ensures that the platform’s technical architecture supports business objectives, such as fostering sustainable agriculture. For example, the ADM’s business architecture phase can define how the Green Energy Marketplace drives revenue for farmers, guiding technical decisions like microservices adoption (White, 2022).
* **Comprehensive Guidance**: TOGAF provides a step-by-step process for designing, planning, and implementing the platform’s architecture. Its phases (e.g., data architecture, application architecture) ensure that components like the Sustainable Farming Hub and interactive forums are integrated seamlessly, addressing the scenario’s need for robust backend support (White, 2022).
* **Scalability and Flexibility**: TOGAF supports scalable architectures, such as the microservices pattern proposed in Part 1. Its technology architecture phase can guide the selection of cloud platforms like Azure, ensuring the platform handles large data volumes and user interactions (White, 2022).
* **Stakeholder Engagement**: TOGAF emphasizes stakeholder involvement, aligning with the bid committee’s need for a convincing proposal. Its governance phase ensures that the platform meets user expectations, such as usability for farmers with varying technical expertise (White, 2022).
* **Comparison with Alternatives**:
  1. **ITIL**: Focuses on IT service management, which is less relevant for designing the platform’s architecture. ITIL is better suited for post-deployment service operations, such as managing platform support tickets (Jucan, 2008).
  2. **Zachman**: Provides a taxonomy for architecture but lacks TOGAF’s actionable process. Zachman is too abstract for the platform’s need for a practical, step-by-step implementation plan (Jucan, 2008).

TOGAF’s structured yet flexible approach adds value by ensuring the platform’s architecture is aligned with business goals, scalable, and adaptable to future needs, such as integrating new green energy solutions.

**6. Technical Solution Description for Marketing Team**

**6.1 Overview of the Prototype**

The Agri-Energy Connect prototype is a web application built using ASP.NET Core 8, SQL Server, and Bootstrap 5, designed to demonstrate the platform’s core functionalities. It enables farmers to manage agricultural products and employees to oversee farmer profiles, promoting sustainable farming through a user-friendly, secure, and responsive interface.

**6.2 Technical Decisions and Business Value**

The prototype’s technical decisions were made to deliver business value, as outlined below:

* **ASP.NET Core 8 and Razor Pages**: The backend uses ASP.NET Core 8 for its performance and cross-platform capabilities. Razor Pages simplify the development of dynamic web pages, such as the product listing page. **Business Value**: This ensures a fast, reliable platform that farmers can use to manage products efficiently, driving adoption and engagement (Prabhu, 2019).
* **SQL Server and Entity Framework Core**: A relational database stores farmer and product data, with Entity Framework Core enabling efficient data access. The database includes sample data (e.g., farmers like “Alice” and products like “Organic Tomatoes”) to simulate real-world usage. **Business Value**: Accurate data management supports farmers in showcasing their products and employees in monitoring agricultural trends, fostering collaboration (Prabhu, 2019).
* **Role-Based Authentication**: ASP.NET Core Identity provides secure login and role-based access control, restricting farmers to product management and employees to farmer profile management. Anti-forgery tokens prevent unauthorized actions. **Business Value**: Security builds trust, encouraging users to share proprietary farming techniques or engage in marketplace transactions without fear of data breaches (Prabhu, 2019).
* **Responsive UI with Bootstrap 5**: The UI features a teal-to-cyan gradient color scheme, card-based layouts, and fade-in animations, ensuring accessibility on desktops, tablets, and smartphones. **Business Value**: A user-friendly interface attracts diverse users, from tech-savvy energy experts to farmers with limited digital literacy, expanding the platform’s reach (Prabhu, 2019).
* **Filtering and Validation**: Employees can filter products by category and date range, with client-side (jQuery) and server-side validation ensuring data accuracy. **Business Value**: Efficient filtering helps employees identify market trends (e.g., demand for organic products), supporting data-driven decisions for sustainable agriculture (Prabhu, 2019).
* **Microservices Readiness**: While the prototype is monolithic, its modular design (e.g., separate controllers for farmers and products) supports future migration to a microservices architecture. **Business Value**: This ensures scalability, allowing the platform to grow with the agricultural sector’s needs, such as adding new marketplace features (Prabhu, 2019).

**7. Conclusion**

The Agri-Energy Connect Platform’s success relies on a robust technical foundation and strategic development practices. Optimizing the prototype’s performance through caching, database indexing, and load balancing ensures a fast and reliable user experience. The Agile methodology, complemented by DevOps, provides flexibility and collaboration, aligning with the platform’s evolving requirements. TOGAF’s structured approach ensures the architecture supports business goals, such as scalability and user engagement. The prototype’s technical solution, built with ASP.NET Core, SQL Server, and a responsive UI, delivers a secure and intuitive platform that drives sustainable agriculture and green energy collaboration. By addressing these areas, the proposal demonstrates a comprehensive plan to deliver a transformative digital ecosystem for South Africa’s agricultural sector.

# Bibliography

Rajeshvelmani. (2023, June 21). *Medium*. Retrieved from Medium: https://medium.com/@rajeshvelmani/how-load-balancers-work-and-improve-system-performance-33f4c8c90b16

Milne, A. (2022, June 29). *net solutions*. Retrieved from net solutions: https://www.netsolutions.com/insights/what-is-scrum-development-agile-scrum-methodology/

Anghel, D.-C. (2019, January 22). *ResearchGate*. Retrieved from ResearchGate: https://www.researchgate.net/publication/327976347\_Optimized\_Prototyping\_for\_the\_Increasing\_of\_Performances\_of\_a\_Design\_Process

BrowserStack. (2024, November 21). *Top 15 Core Benefits of DevOps*. Retrieved from BrowserStack: https://www.browserstack.com/guide/benefits-of-devops

White, S. K. (2022, May 30). *What is TOGAF? What the enterprise architecture methodology means for your business*. Retrieved from CIO: https://www.cio.com/article/228328/what-is-togaf-an-enterprise-architecture-methodology-for-business.html

Jucan, G. (2008, September 8). *ITIL and Zachman based methodoogy*. Retrieved from projectmanagement.com: https://www.projectmanagement.com/discussion-topic/12624/itil-and-zachman-based-methodoogy

Prabhu, J. (2019, July 18). *MVC Architecture & Its Benefits in Web Application Development*. Retrieved from techaffinity: http://techaffinity.com/blog/mvc-architecture-benefits-of-mvc/

Corporation, M. (2025, June 22). *Microsoft PowerPoint*. Retrieved from Microsoft PowerPoint: https://powerpoint.cloud.microsoft/