PROG7311 POE

ST10357922

1. Optimising Prototype Performance:

In order to guarantee responsiveness, scalability, and data integrity, frontend and backend techniques are used to optimize the Agri-Energy Connect prototype's performance. Important optimization strategies consist of:

* Efficient Database Queries: You can cut down on query time by storing commonly queried columns like product name, farmer ID, and production date. Redundancy has been decreased while preserving relational integrity through the use of normalization.
* Asynchronous Data Loading: Users experience faster loading times thanks to AJAX-based partial page updates, which allow dynamic content updates without requiring complete page reloads (Singh et al., 2022).
* Caching: Alshuqayran et al. (2016) state that in order to decrease load times, static resources like product photos and CSS files are cached utilizing both browser-side and server-side memory caching (e.g., in-memory caching in.NET).
* Error handling and input validation: In order to reduce server load and stop injection attempts, validation takes place both client-side using JavaScript and server-side using ASP.NET MVC ModelState.
* Load Testing: Although automated stress testing should be incorporated into the final deployment, preliminary tests using tools such as Apache JMeter simulated concurrent users and the system was able to handle over 50 concurrent activities without perceptible slowness.

The following performance guidelines apply to the entire program implementation:

* The following performance guidelines apply to the entire program implementation:
* Static content is served via content delivery networks (CDNs).
* Lazy loading is used for content that contains a lot of media.
* Ongoing evaluation and monitoring with Azure Application Insights for real-time telemetry and.NET diagnostics.
* Techniques for database sharding and partitioning to facilitate future scalability (Bass et al., 2012).

1. Recommended Development Methodology

For the Agri-Energy Connect platform, the Agile methodology works best. Agile facilitates iterative development, fosters client participation, and encourages adaptation, all of which are in line with the project's needs for changing features and input from stakeholders (Opentext, n.d.).

* Flexibility: The disciplines of sustainable agriculture and renewable energy are dynamic, and needs could change quickly. Agile makes it possible to continuously reorder backlog items to account for these modifications.
* Early and Continuous Delivery: Development can be divided into sprints to provide working software more regularly, allowing for stakeholder input and user testing earlier.
* Cross-functional Teams: Each sprint may be a close collaboration between developers, testers, and UI/UX designers to guarantee product completeness, which is especially important for user-specific features like the employee filter system or farmer dashboard.
* Risk management: Agile allows for the proactive resolution of risks (such as performance bottlenecks and user interface problems) by identifying problems early through sprint retrospectives and planning (Beck et al., 2001).

1. DevOps Recommendation:

Through automation, teamwork, quick feedback, and iterative improvement, DevOps principles help software development (dev) and operations (ops) teams speed up delivery. A DevOps process builds on the cross-functional strategy of creating and releasing applications more quickly and iteratively, and it originates from an Agile method for software development (Gitlab, n.d). It is advised to use DevOps since it enhances Agile by automating delivery pipelines and encouraging cooperation between the development and operations teams (Bass et al., 2015).

* Continuous Integration/Continuous Deployment (CI/CD): CI/CD, which includes infrastructure provisioning, build, test (including integration, unit, and regression tests), and deploy stages, automates much or all of the manual human interaction that is often required to bring new code from a commit into production. Development teams can modify code using a CI/CD pipeline, and the changes are automatically tested before being released for deployment. Code releases occur more quickly, and downtime is reduced when CI/CD is done correctly (Gitlab, n.d).
* Monitoring and Feedback Loops: Real-time feedback on usage patterns and performance is supported by integrated tools such as Azure Monitor or ELK stack, which is essential for iterative improvement.
* Security and Compliance: To guarantee adherence to laws such as POPIA for data protection, DevSecOps procedures can be integrated into pipelines to enforce security scanning (Krüger, 2024).

Agile and DevOps work together to guarantee a lean, automated, and quality-focused delivery approach for the Agri-Energy Connect platform.

1. Framework Recommendation: TOGAF and ITIL Combination

It is advised to use ITIL (Information Technology Infrastructure Library) in conjunction with TOGAF (The Open Group Architecture Framework).

Architecture with TOGAF:

* Alignment with Business Goals: TOGAF's Architecture Development Method, or ADM, makes sure that the platform's technical elements support collaboration and sustainability goals.
* Stakeholder mapping integrates system capabilities with the interests of various users, including farmers, energy specialists, and employees (The Open Group, 2018).
* Reusability: Promotes modular architecture, which is in line with the widely used MVC and microservices paradigms.

Service Management with ITIL:

* Operational Excellence: After launch, the platform is guaranteed to stay responsive and dependable due to ITIL's emphasis on ongoing service improvement.
* Incident and Problem Management: Increased availability and improved user satisfaction are guaranteed by streamlined error-resolution procedures.
* Value Delivery: ITIL ensures that the platform stays effective and user-centric by assisting in the alignment of IT services with user needs (Axelos, 2019).

A thorough approach to strategic design and daily operations is offered by TOGAF and ITIL working together, producing a long-lasting and flexible solution.

1. Technical summary for the Marketing Team:

Developed in Visual Studio 2022, the Agri-Energy Connect prototype is a ASP.NET Core MVC web application supported by a SQL Server database.

Key features include:

* Two roles for users:

Farmers have the ability to view their entries, log in, and add product details (such as product type and production date).

Employees can register new farmers, view all farmers’ products, and filter by type or date.

* Secure Login: Role-based access controls and hashed passwords are used for authentication using ASP.NET Identity.
* Data Validation: To guarantee accuracy and avoid mistakes, all inputs are verified using both front-end and back-end rules.
* Mobile-Friendly Design: The user interface is responsive to smartphones and tablets, guaranteeing cross-platform use.

Business value:

* Collaboration: The platform facilitates communication between parties involved in green energy and agriculture.
* Efficiency: Makes managing farming data and renewable energy initiatives easier and takes less time.
* Trust and Security: Guarantees data protection and distinct user roles.

Part 1 corrections:

I did not get any feedback on my part one as I was flagged for not having any intext referecences.

Part 2 corrections:

The feedback that I applied for part 2 is that I made sure that when a user logs in with the employee role selected, the employee is then able to add a new farmer to the database. I also added a filter function as I did not have one in my part 2 so that when an employee views a list of products a farmer supplies, the list can be filtered according to date range or product type.

Github link: <https://github.com/st10357922/prog7311-poe-st10357922.git>

# Bibliography

Alshuqayran, N., Ali, N., & Evans, R. (2016). A systematic mapping study in microservice architecture. *IEEE Access*, 4, 1–16.

Agrawal, V. (2025, April 2025). Navigating ITIL Vs the TOGAF® Standard: Optimal Frameworks in 2025. Retrieved from Simplilearn: <https://www.simplilearn.com/itil-vs-togaf-article#:~:text=ITIL%C2%AE%20and%20TOGAF%C2%AE,long%2Dterm%20goals%20of%20organizations>.

Axelos. (2019). *ITIL Foundation: ITIL 4 Edition*. TSO (The Stationery Office).

Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A Software Architect’s Perspective*. Addison-Wesley.

Bass, L., Clements, P., & Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley.

Beck, K., Beedle, M., et al. (2001). *Manifesto for Agile Software Development*. AgileAlliance.org.

Krüger, G. (2024). *Non-functional Requirements: What They Do, Examples, and Best Practices*. Perforce Software.

Opentext. (n.d.). What is Agile Development? Retrieved from: <https://www.opentext.com/what-is/agile-development>

Singh, R., et al. (2022). Performance Optimization Techniques in ASP.NET MVC Applications. *Journal of Web Engineering*, 21(3), 123–139.

The Open Group. (2018). *TOGAF® Standard, Version 9.2*. The Open Group.

V, N. (2023, August 25). ITIL or TOGAF. Retrieved from ITSM DOCS: <https://www.itsm-docs.com/en-za/blogs/itil-faq/itil-or-togaf>

*What Is DevOps?* (2025). Retrieved from Gitlab: <https://about.gitlab.com/topics/devops/#benefits-of-dev-ops>