**Ediz Alperen Amin Yurdakul  
ST10242585  
PROG7311 POE PART 3**

**Table Of Contents**

[**Introduction 1**](#_b5doxm6bahe5)

[**Optimising Prototype Performance: 2**](#_x0zksev3b6po)

[**Software Development Methodology: 3**](#_k7kzbwd604w)

[**Implementing DevOps: 4**](#_kaeqpi11j7iy)

[**Framework Recommendations: 5**](#_curj6jfc3x0v)

[**Description of the Technical Solution: 6**](#_61i4v0va3hes)

[**Conclusion: 7**](#_dhweydmgfegy)

[**References 8**](#_32dnbclcmqwp)

# 

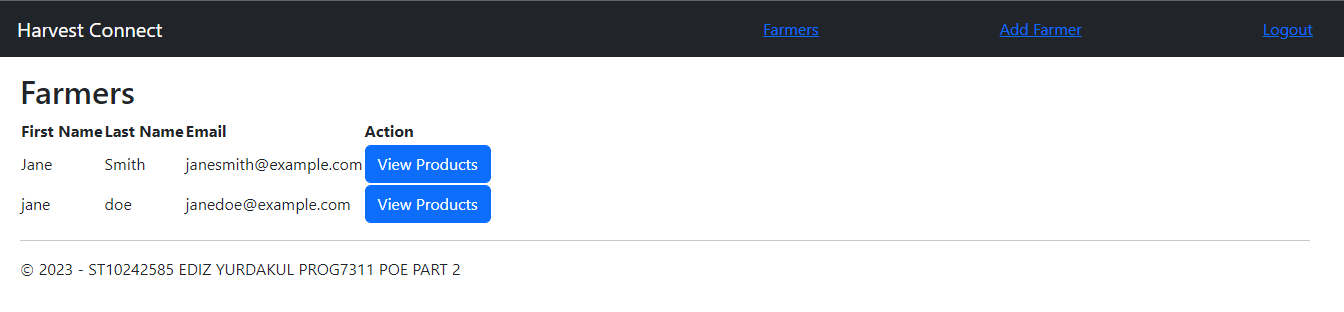
# **Introduction**

This report addresses additional areas the marketing team highlights to enhance the proposal. The sections cover topics such as performance optimization, software development methodology recommendation, implementing DevOps, choosing frameworks, and a technical solution.

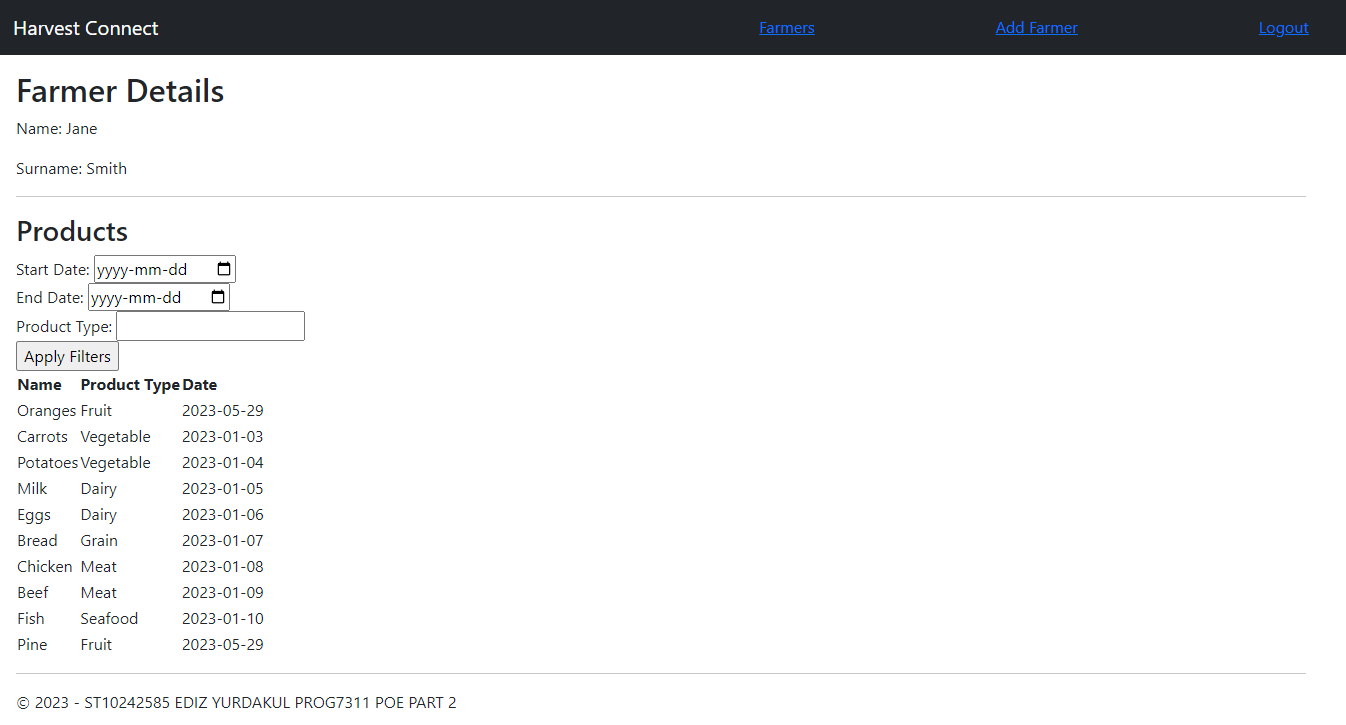
# 

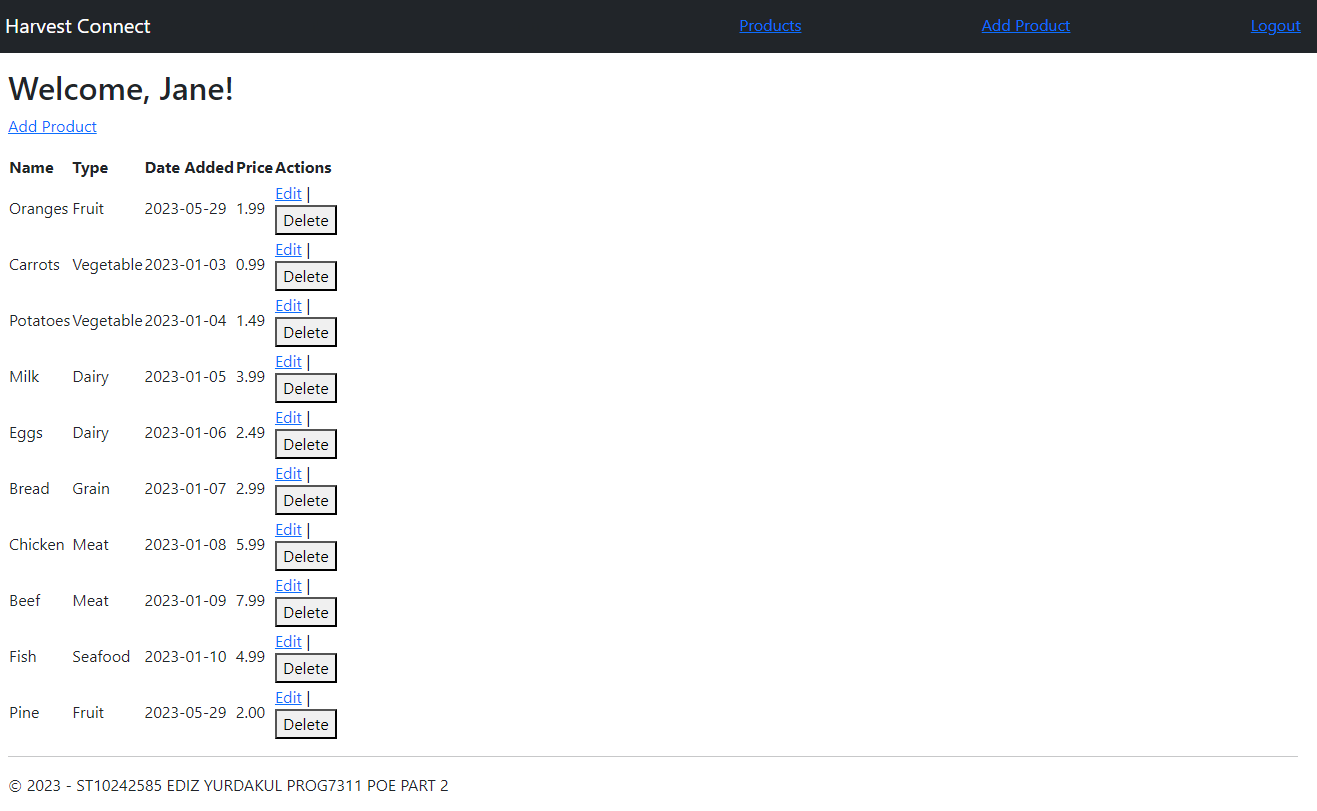
# Optimising Prototype Performance:

The previous prototype enables users to log in, and their dashboard will be based on their assigned role, implementing role-based authentication. Upon successful login, employees gain access to various functionalities, including viewing and adding farmers' information. Additionally, they have the option to log out when needed.



Employees have the ability to access detailed information about individual farmers, including their respective products. Additionally, they can apply filters based on product type and specific date ranges to refine their search results.



When logging in as a farmer, a dashboard displaying their stock list. This allows farmers to manage their inventory by editing, deleting, and adding new products.  


For optimization it is crucial to follow specific guidelines during development. These guidelines focus on enhancing efficiency and delivering a seamless user experience. What could have been improved on the prototype:

* **Efficient Code:** Gray (2007) suggests reviewing and optimising the codebase to eliminate unnecessary code, reduce complexity, and utilise appropriate algorithms and data structures. This optimization ensures that the software executes efficiently and minimises resource usage.
* **Performance Testing:** Gray (2007) argues for conducting performance tests to identify and address bottlenecks, optimise database queries, reduce network latency, and fine-tune data caching strategies. By simulating real-world scenarios, performance testing allows developers to identify potential issues and optimise the system accordingly. No performance testing was done in the prototype phase which could reveal many performance issues that were not addressed.  
  .
* **Scalability and Load Balancing:** Design the software to be scalable, implementing load balancing techniques and distributed architectures to handle increased user load efficiently. According to Microsoft Learn (2022) by distributing the workload across multiple servers, load balancing ensures optimal resource utilisation and improves fault tolerance and security. If one server fails, the load can be automatically transferred to other servers.
* **Caching and Optimization:** Implement caching mechanisms at various levels to minimise response times and improve performance (Microsoft Learn 2021-a). Caching frequently accessed data reduces the need for repeated computations or database queries, resulting in better performance (Microsoft Learn 2021-a), and faster response times and enhanced user experience. No caching was taking place in the prototype which if were to be deployed as a production it would result in high-cost and high latency to the users on high traffic, as many redundant requests would be not cached and fetched from the server.
* **Efficient Resource Management:** Allocate and utilise resources effectively, ensuring proper memory management, connection pooling. Microsoft Learn (2021-b) explains connection pooling is a store of a pool of open connections. When a new connection is needed, the pool is checked to see if there is an available connection. If there is, the connection is reused. If there is not, a new connection is opened which minimises the cost of repeatedly opening and closing connections. Optimising resource usage helps avoid bottlenecks and improves the overall performance of the software. The prototype had not implemented proper user authentication which means no pooling, if we were to use ADO.NET which automatically manages connection pooling according to Microsoft Learn (2021-b).

# 

# Software Development Methodology:

Agile methodology for Farm Central due to its flexibility and iterative approach. Agile enables incremental development, frequent feedback, and continuous improvement. Critical reasons for choosing Agile include:

* **Iterative Development:** Focus on incremental development and regular feedback loops to adapt to changing requirements and deliver valuable features early. By breaking down the development process into smaller iterations, Agile allows flexibility and agility in responding to evolving customer needs (Cohn, 2009).
* **Customer Collaboration:** Active involvement of customers throughout the development process to ensure the software meets their evolving needs. Regular communication and feedback sessions with customers enable better alignment of the software with their expectations and requirements (Leffingwell, 2011).
* **Flexibility and Adaptability:** Ability to respond quickly to changing requirements and market demands, reprioritize tasks, and adjust the product roadmap as needed. Agile methodologies, such as Scrum or Kanban, provide the necessary frameworks and practices to manage to change priorities effectively (Reinertson, 2010).
* **Continuous Improvement:** Emphasis on retrospectives to learn from each iteration and refine processes for enhanced teamwork and optimised software development practices. Continuous improvement ensures that the development process becomes more efficient and effective (Leffingwell, 2011; Reinertson, 2010).

# 

# Implementing DevOps:

DevOps can bring significant benefits when implemented alongside the chosen Agile methodology. DevOps promotes collaboration, automation, and continuous delivery. Reasons for recommending DevOps include:

* **Collaboration and Communication:** Facilitates collaboration between development, operations, and other teams, ensuring shared goals and transparent communication. Kavyashree (2022) eludes by fostering a collaboration and cross-functional cooperation culture, DevOps breaks down silos and promotes better understanding and alignment among team members.
* **Automation and Continuous Delivery:** Enables automation of build, testing, deployment, and infrastructure provisioning processes, reducing manual errors and accelerating release cycles. Kavyashree (2022) explains using automation tools and practices, DevOps enables frequent and reliable software releases, ensuring faster time-to-market and improved customer satisfaction.
* **Infrastructure as Code:** Encourages the use of infrastructure as code (IaC) for consistent and repeatable configuration and provisioning of environments. Kavyashree (2022) argues infrastructure automation using tools like Terraform or Ansible allows for version-controlled and reproducible environments, ensuring consistency between development, testing, and production environments.
* **Monitoring and Feedback:** Implements monitoring and feedback mechanisms to gain insights into software performance, user behaviour, and operational metrics. By leveraging monitoring tools, DevOps teams can identify issues, track system health, and make data-driven decisions for continuous improvement.

# 

# Framework Recommendations:

A combination of ITIL and TOGAF frameworks allows for effective management and architectural guidance. Each framework serves a specific purpose and provides valuable insights into software development and management aspects. Reasons for choosing these frameworks include:

* **ITIL (Information Technology Infrastructure Library):** ITIL offers best practices for IT service management, focusing on delivering value to customers and ensuring efficient service operations. Organisations can optimise service delivery, incident management, change management, and other crucial IT operations by implementing ITIL processes and guidelines. Sarah, White, and Greiner (2022) explains "ITIL 4 contains seven guiding principles that were adopted from the most recent ITIL Practitioner Exam, which covers organizational change management, communication, and measurement and metrics."
* **TOGAF (The Open Group Architecture Framework):** The Open Group explains as TOGAF provides a comprehensive approach to enterprise architecture development and management. It offers a structured methodology, framework, and tools for creating and maintaining architecture artefacts, ensuring alignment with business goals, and driving effective decision-making.
* **Combining ITIL and TOGAF:** ITIL and TOGAF can be combined to complement each other's strengths. ITIL focuses on IT service management, while TOGAF provides a broader perspective on enterprise architecture. They enable organisations to align IT services with business objectives and establish a robust architectural framework for efficient software development and management.

# 

# Description of the Technical Solution:

The technical solution implemented in the prototype for Farm Central is a modern and scalable web application built using a microservices architecture. This architecture comprises loosely coupled services communicating through lightweight protocols, facilitating independent development, deployment, and scaling. The critical components of the technical solution are:

* **Microservices:** Richardson explains microservices divides the application into small, autonomous services, each responsible for a specific functionality or business domain. This modular approach enables flexibility, scalability, and easier maintenance.
* **Containerization:** Containerized services using Docker, Docker (2023) explains containers providing isolation, portability, and consistent environments across different stages of the software development lifecycle.
* **API Gateway:** According to NGINX (2023) An API gateway is employed to centralise request handling, authentication, and data transformation. It acts as a single entry point for client applications, enhancing security and simplifying communication between services.
* **Event-Driven Architecture:** Amazon Web Services describes event-driven architecture as services which communicate asynchronously through events, enabling loose coupling and scalability.
* **Relational Database Management System (RDBMS):** Manage Data storage using a reliable RDBMS, ensuring data consistency and integrity across different services.
* **Caching:** Redis is implemented as a caching mechanism to improve performance by reducing the need for repetitive computations or database queries.

# 

# Conclusion:

In conclusion, this comprehensive report addresses the additional areas requested by the marketing team for the proposal. It provides detailed insights into optimising prototype performance, recommending the Agile methodology, implementing DevOps practices, and choosing the ITIL and TOGAF frameworks. The technical solution description showcases the modern and scalable nature of the implemented prototype.

# 

# **References**

* Microsoft Learn. 2023. Caching in .NET Framework Applications, *Microsoft Learn*, 15 September 2021-a. [Online]. Available At: <https://learn.microsoft.com/en-us/dotnet/framework/performance/caching-in-net-framework-applications> [Accessed 22 June 2023]
* Microsoft Learn. 2023. Load balancing, *Microsoft Learn*, 21 March 2022. [Online]. Available At: <https://learn.microsoft.com/en-us/dotnet/orleans/implementation/load-balancing>[Accessed 22 June 2023]
* Gray, J. 2007. Writing Faster Managed Code: Know What Things Cost, *Microsoft Learn*, 30 April 2007. [Online]. Available At: <https://learn.microsoft.com/en-us/previous-versions/dotnet/articles/ms973852(v=msdn.10)> [Accessed 22 June 2023]
* Microsoft Learn. 2023. Connection Pooling, *Microsoft Learn*, 15 September 2021-b. [Online]. Available At: <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/connection-pooling>[Accessed 22 June 2023]
* Cohn, M. 2009. *Succeeding with agile: Software development using Scrum*. Boston: Addison-Wesley.
* Leffingwell, D. 2011. *Agile software development: A practical guide*. Boston: Addison-Wesley.
* Reinertson, M. 2010. *Agile development with Scrum: A practical guide*. Sebastopol, CA: O'Reilly Media.
* Kavyashree, K. 2023. Why DevOps is Important? A Comprehensive Picture, CloudThat, 16 May 2022. [Blog]. Available At: <https://www.cloudthat.com/resources/blog/why-devops-is-important> [Accessed 22 June 2023]
* Sarah, K., White. and Greiner, L. 2023. What is ITIL? Your guide to the IT Infrastructure Library, *Cio*, 16 May 2022. [Blog]. Available At: <https://www.cio.com/article/272361/infrastructure-it-infrastructure-library-itil-definition-and-solutions.html> [Accessed 22 June 2023]
* The Open Group. 2022. TOGAF® Standard — Introduction - Core Concepts. *The Open Group Architecture Framework*. [Article]. Available At: <https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html> [Accessed 22 June 2023]
* Richardson, C. 2023. What are microservices?. *Microservice Architecture*. [Blog]. Available At: <https://microservices.io/> [Accessed 22 June 2023]
* Docker. 2023. Docker overview. 21 June 2023. [Online]. Available At: <https://docs.docker.com/get-started/overview/> [Accessed 22 June 2023]
* NGINX. 2023. API Gateway, 28 March 2023. [Blog]. Available at: <https://www.nginx.com/learn/api-gateway/>[Accessed: 22 June 2023]
* Amazon Web Services. 2023. What is an Event-Driven Architecture?. [Online]. Available at: <https://aws.amazon.com/event-driven-architecture/> [Accessed: 22 June 2023]