2024

PROG7311 POE

ST10083252
AVARN SEWLAL

CONTENTS PAGE:

Optimizing of prototype performance:

Optimizing the Prototype:

1. **Optimize Database Design**:
    - Ensure proper indexing for frequently queried columns.
    - Normalize the database to reduce redundancy while avoiding excessive joins.
    - Use optimized queries, avoiding SELECT * and fetching only necessary columns.
2. **Efficient Data Access**:
    - Use asynchronous programming for database operations (e.g., async/await in Entity Framework Core).
    - Implement pagination for large datasets to avoid loading too much data at once.
3. **Caching**:
    - Cache frequently accessed data to reduce database queries.
    - Use in-memory caching (e.g., IMemoryCache) for temporary storage of frequently used data.
4. **Reduce Payload Size**:
    - Minimize the size of data sent to the client by using compression (e.g., Gzip).
    - Return only the necessary data in API responses or views.
5. **Optimize the User Interface**:
    - Use lazy loading for images and large components.
    - Minimize JavaScript and CSS file sizes using bundling and minification.
6. **Profile and Benchmark**:
    - Use profiling tools like Visual Studio's Diagnostic Tools or JetBrains' dotTrace to identify bottlenecks.
    - Benchmark critical operations to evaluate their performance impact.
7. **Scalable Architecture**:
    - Design the prototype with scalability in mind, even if it's not immediately needed (e.g., microservices, cloud deployment).

      (Simon, 2018)

Guidelines for Final Software Development:

1. **Adopt a Performance-Driven Development Approach**:
    - Set measurable performance goals during the requirements phase (e.g., response time < 200 ms).
    - Regularly test performance throughout development using automated tools.

2. **Follow Clean Code Practices**:
    - Write modular and reusable code.
    - Avoid redundant computations or unnecessary method calls.
3. **Load Testing and Stress Testing**:
    - Perform load testing to simulate concurrent users and measure response times (tools: JMeter, k6).
    - Conduct stress testing to find breaking points and understand system limits.
4. **Optimize Resource Usage**:
    - Avoid memory leaks by properly disposing of unmanaged resources (e.g., database connections, file streams).
    - Use dependency injection to manage object lifetimes efficiently.
5. **Database Optimization**:
    - Use stored procedures for complex queries where applicable.
    - Regularly monitor query execution plans and optimize slow queries.
6. **Use CDN for Static Assets**:
    - Offload static content (e.g., images, CSS, JavaScript) to a Content Delivery Network (CDN).
7. **Security Without Performance Sacrifice**:
    - Use lightweight encryption algorithms for data in transit (e.g., HTTPS).
    - Optimize authentication mechanisms (e.g., token expiration policies).
8. **Monitoring and Logging**:
    - Implement application monitoring using tools like Application Insights or ELK Stack.
    - Log errors and performance metrics to analyze trends and prevent issues.
9. **Responsive and Adaptive Design**:
    - Ensure the UI performs well across different devices and screen sizes.
    - Optimize frontend frameworks (e.g., React, Angular) for fast rendering.
10. **Test in Realistic Environments**:
    - Deploy the application in staging environments that mimic production settings.
    - Test with production-like data and user scenarios.

    (McGuire, 2024)

Performance Checklist for Deployment:

- **Minimize Latency**:
    - Deploy close to the majority of users (geographically).
- **Horizontal Scaling**:
    - Use load balancers to distribute traffic across multiple servers.
- **Optimize Build Pipeline**:
    - Automate builds, tests, and deployments for consistent and optimized releases.

(Dwyer, 2023)

Recommended Software Methodology:

The Recommended Software Development Methodology that I suggest : Agile Development Methodology

Why Agile?

The Agile methodology is highly recommended for the development of this project because it aligns with the project's requirements and context. Here's a detailed breakdown of why Agile is the most suitable approach:

1. Emphasis on Prototypes and Iterative Development:

- **Project Requirement**: The task explicitly calls for a prototype as part of the development process.
- **Agile Strength**:
    - Agile promotes **iterative development**, where a functional product is built in stages.
    - Prototypes can be developed as **increments** or **iterations**, with continuous feedback incorporated into subsequent stages.

2. Flexibility to Handle Changing Requirements:

- **Project Context**:
    - The project may evolve based on stakeholder feedback, particularly as it moves from the prototype phase to the final software.
- **Agile Strength**:
    - Agile welcomes changing requirements, even late in the development process, by focusing on **adaptive planning**.
    - This is critical for projects with evolving goals, like ensuring usability or performance improvements based on feedback

3. Collaboration and Communication:

- **Project Context**:
    - Collaboration between developers, stakeholders, and potentially farmers or employees will be essential to ensure the software meets user needs.
- **Agile Strength**:
    - Agile emphasizes **continuous communication** through daily stand-ups, sprint reviews, and planning sessions.
    - It fosters close collaboration with stakeholders, ensuring the product aligns with real-world requirements.

4. Focus on Usability and Performance:

- **Project Requirement**:
    - The project requires a user-friendly interface and acceptable performance.
- **Agile Strength**:

- Usability and performance can be improved incrementally in Agile by testing small, functional increments and gathering real user feedback.
- Early and frequent testing reduces the risk of usability or performance issues being discovered late.

5. Risk Reduction:

- **Project Context**:
  - Developing a prototype involves technical, design, and performance risks.
- **Agile Strength**:
  - Agile mitigates risks by delivering working increments early. Issues can be detected and resolved at each stage rather than at the end of the project.
  - Regular feedback ensures that the prototype evolves in the right direction.

6. Better Alignment with Final Software Development

- **Project Requirement**:
  - While the project begins with a prototype, the final software needs to meet higher scalability, performance, and functionality standards.
- **Agile Strength**:
  - Agile allows for seamless scaling from the prototype phase to the final product by using sprint planning and a backlog system to manage enhancements and additional features.

    (Laoyan, 2024)

How Agile Could Work for This Project:

Stages and Iterations

1. **Sprint 1**:
   - Basic user authentication and database setup (MVP for farmers and employees).
   - Deliverable: Functional prototype with login and basic data management.
2. **Sprint 2**:
   - Adding and viewing products for farmers.
   - Deliverable: Farmers can add and view products; employees can see a list.
3. **Sprint 3**:
   - Filters for employees, usability improvements.
   - Deliverable: Filtering and a polished interface.
4. **Sprint 4**
   - Performance tuning, responsive design, and validation testing.
   - Deliverable: A near-production-ready prototype.

Agile Roles:

- **Scrum Master**: Ensures team productivity and manages impediments.
- **Product Owner**: Represents stakeholders, prioritizing features in the backlog.
- **Development Team**: Builds features iteratively.

Comparison to Other Methodologies:

Waterfall

- Sequential and rigid; not ideal for iterative feedback or changing requirements.
- Delivers the product only at the end, increasing risk.

Spiral

- Focuses on risk analysis and prototyping but can be too heavy and complex for smaller projects.

Lean

- Works well for highly streamlined projects but may lack structure for prototypes requiring stakeholder involvement and feedback.

  (Anantatmula & Anantatmula, 2008)

Conclusion:

The Agile methodology is ideal for this development effort because:

1. It supports iterative prototyping and stakeholder feedback.
2. It ensures flexibility for changing requirements.
3. It focuses on delivering a user-cantered product incrementally, with performance and usability at its core.

Should DevOps be implemented:

Recommendation: Implement DevOps

Yes, I highly recommend implementing **DevOps** alongside the Agile methodology for this project. DevOps complements Agile by providing tools, practices, and automation that enhance the efficiency, reliability, and speed of the software development and deployment lifecycle.

Why DevOps?

1. Continuous Delivery of Value

- **Agile Focus**: Agile prioritizes delivering small, incremental features and updates regularly.
- **How DevOps Fits**: DevOps enables **Continuous Integration (CI)** and **Continuous Deployment (CD)**, ensuring that these updates can be tested, integrated, and deployed seamlessly without manual intervention.

2. Improved Collaboration

- **Agile Focus**: Agile promotes collaboration between development teams and stakeholders.
- **How DevOps Fits**: DevOps extends collaboration to include operations teams, breaking down silos between developers and IT professionals. This unified approach ensures faster and smoother deployments.

3. Enhanced Feedback Loops

- **Agile Focus**: Agile relies on stakeholder feedback to shape future increments.
- **How DevOps Fits**: DevOps automates feedback collection through monitoring and logging tools, allowing for faster detection of bugs, performance issues, and user experience concerns.

4. Faster Development and Deployment

- **Agile Focus**: Agile delivers working software frequently.
- **How DevOps Fits**: DevOps automation (e.g., CI/CD pipelines) reduces the time required for testing, building, and deploying software, ensuring that updates reach end-users quickly.

5. Reliability and Quality

- **Agile Focus**: Agile emphasizes frequent testing and continuous improvement.
- **How DevOps Fits**: DevOps integrates automated testing and monitoring tools to catch issues early and ensure deployments are stable and reliable.

(Bigelow, et al., 2024)

How DevOps Fits with Agile

Aspect	Agile	DevOps
Core Goal	Deliver software incrementally.	Streamline software delivery and operations.
Process	Iterative development.	Continuous integration, testing, and deployment.
Collaboration Focus	Development team and stakeholders.	Development and operations teams (end-to-end).
Feedback Mechanisms	Stakeholder/user feedback.	Automated monitoring and logging for real-time feedback.
Adaptability	Responds to changes quickly.	Ensures changes are deployed reliably and quickly.

Together, Agile and DevOps create a seamless workflow from development to production, ensuring fast delivery, continuous improvement, and operational stability.

(Bigelow, et al., 2024)

How to Implement DevOps in This Project:

1. **Set Up a CI/CD Pipeline**
    - Use tools like **GitHub Actions**, **Azure DevOps**, or **Jenkins** to automate the building, testing, and deployment of the application.
    - Example Workflow:
        1. Push code to a repository.
        2. Trigger automated build and testing.
        3. Deploy successful builds to a staging environment for stakeholder review.
2. **Version Control and Collaboration**
    - Use **Git** to manage source code with branching and pull requests.
    - Implement **code reviews** and **merge checks** to ensure code quality.
3. **Automated Testing**
    - Integrate automated unit and integration tests into the CI pipeline.
    - Tools: **xUnit**, **Selenium** for UI testing, or **Postman** for API tests.
4. **Infrastructure as Code (IaC)**
    - Use tools like **Terraform** or **Azure Resource Manager** to define and manage infrastructure declaratively.
5. **Monitoring and Logging**

- Integrate monitoring tools like **Azure Application Insights**, **ELK Stack**, or **Prometheus** to capture real-time performance and usage metrics.
- Use automated alerts for performance bottlenecks or failures.

6. **Containerization**
    - Use Docker to containerize the application, ensuring consistency across development, testing, and production environments.
7. **Cloud Deployment**
    - Host the application on a scalable cloud platform like **Azure**, **AWS**, or **Google Cloud**.
    - Use services like Azure App Service or Kubernetes for scaling.

    (Marusiak, 2021)

Benefits of DevOps in This Project:

- **Faster Releases**: Automates deployment and testing for quicker delivery cycles.
- **Improved Quality**: Integrates testing and monitoring into the pipeline.
- **Scalability**: Ensures the application can scale as it transitions from prototype to production.
- **Reliability**: Monitors and resolves issues proactively, reducing downtime.

(Bigelow, et al., 2024)

Conclusion:

DevOps is an excellent complement to Agile for this project. While Agile ensures iterative development and collaboration, DevOps enables automated, efficient, and reliable delivery. Together, they form a robust framework for developing high-quality software with acceptable performance and scalability.

If you'd like detailed steps on setting up a CI/CD pipeline or integrating monitoring tools.

(Bigelow, et al., 2024)

Framework Recommendation

(ITLL, Zachman,TOGAF)

I recommend the use of TOGAF with ITIL

For this project, I recommend a combination of **TOGAF (The Open Group Architecture Framework)** and **ITIL (Information Technology Infrastructure Library)**. These two frameworks complement each other well for developing, deploying, and managing a software application like this one. (Hajela, 2024)

Here's why TOGAF and ITIL are ideal and how they align with your project:

Why TOGAF?

TOGAF is an enterprise architecture framework designed to create and maintain IT architectures that align with business goals.

Advantages for This Project:

1. **Structured Architectural Development**:
    - TOGAF provides a **methodical approach to designing the architecture**, ensuring the system aligns with the overall business strategy (e.g., scalability, usability).
    - The **Architecture Development Method (ADM)** guides the project through key phases: vision, design, implementation, and governance.
2. **Focus on Scalability and Integration**:
    - The project involves creating a prototype that will eventually scale into full production software.
    - TOGAF ensures that design decisions (e.g., database structure, application layers) support future scalability and integration with other systems.
3. **Alignment with Stakeholder Goals**:
    - TOGAF prioritizes stakeholder input, ensuring that the final architecture meets the needs of farmers, employees, and other potential users.
4. **Risk Management**:
    - The framework includes tools for **gap analysis**, ensuring that key architectural risks (e.g., performance issues, integration problems) are identified and addressed early.

      (Ardoq, 2024)

Why ITIL?

ITIL is a best-practice framework for IT service management (ITSM), focusing on delivering value to customers through efficient IT processes.

Advantages for This Project:

1. **Focus on Operational Excellence**:
    - ITIL ensures the application operates reliably after deployment, covering incident management, performance monitoring, and support.
2. **Lifecycle Approach**:
    - ITIL's **Service Lifecycle** aligns with the ongoing evolution of the project, covering stages like:
        - **Service Strategy**: Planning to align the system with business goals.
        - **Service Design**: Ensuring usability, reliability, and scalability.
        - **Service Transition**: Deploying the application from prototype to production.
        - **Service Operation**: Managing the live system to ensure smooth performance.
        - **Continual Service Improvement (CSI)**: Making incremental enhancements.
3. **User-Centric Focus**:
    - ITIL emphasizes delivering value to end-users (e.g., farmers and employees), ensuring that the application meets their operational needs.
4. **Performance Monitoring and Incident Management**:
    - The framework provides guidelines for detecting, resolving, and preventing issues, ensuring smooth operation after deployment.

      (White & Greiner, 2022)

Why Not Zachman Framework Alone?

The **Zachman Framework** focuses on organizing and modeling enterprise information systems but lacks detailed implementation guidance. It works well for high-level planning but doesn't offer practical tools for system design, deployment, or operations.

Limitations for This Project:

- **Rigid Structure**: Primarily a taxonomy, the Zachman Framework is less flexible for iterative development processes like Agile.
- **No Process Guidance**: It does not provide specific processes for creating and managing systems.

While the Zachman Framework can complement TOGAF for high-level conceptual design, it is not sufficient on its own for a project requiring detailed implementation and operational management. (Hajela, 2024)

How TOGAF and ITIL Work Together

Aspect	TOGAF	ITIL
Scope	Focuses on enterprise architecture and system design.	Focuses on IT service delivery and operational excellence.
Phases	Guides system development lifecycle through ADM.	Manages system operation and support through the ITSM lifecycle.
Focus	Long-term system scalability and alignment with business goals.	Day-to-day reliability, usability, and performance.
Development Approach	Defines technical architecture and governance.	Defines operational processes and service improvements.
Complementary Value	Ensures the prototype evolves into a scalable, integrated solution.	Ensures the live system runs smoothly and delivers value.

Implementation Example for This Project:

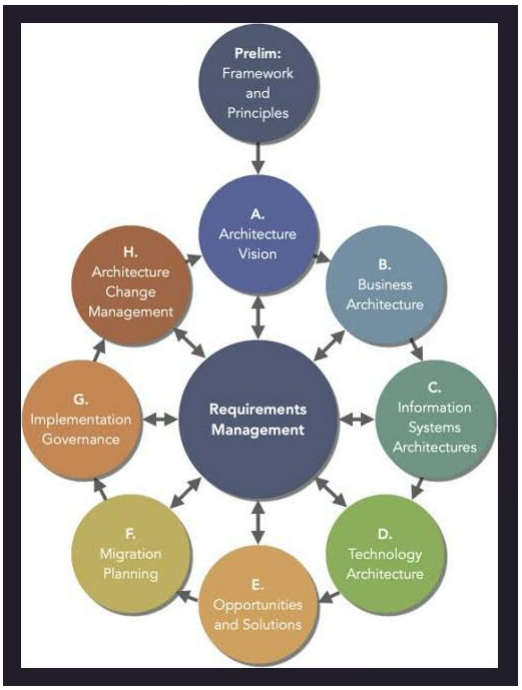
1. **TOGAF for Development:**
    - **Architecture Vision**: Define key requirements, such as scalability, performance, and usability.
    - **Business Architecture**: Identify processes (e.g., farmer product addition, employee filtering) and align them with stakeholder needs.
    - **Technology Architecture**: Define tools (e.g., ASP.NET Core, Entity Framework, SQL Server) and ensure they fit together seamlessly.
    - **Transition Planning**: Develop a roadmap for scaling the prototype into a full production system.

      (Ardoq, 2024)
2. **ITIL for Operations:**
    - **Service Transition**: Use ITIL's guidance to move the application from prototype to production.
    - **Service Operation**:
        - Implement incident management to quickly resolve user issues.
        - Use monitoring tools to track system performance (e.g., Azure Application Insights).
    - **Continual Service Improvement**: Gather user feedback and monitor KPIs (e.g., response times, uptime) to iteratively improve the application.

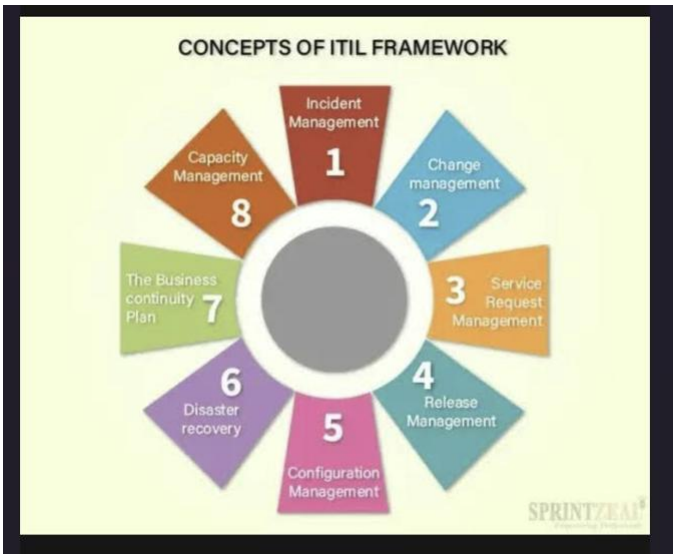
      (White & Greiner, 2022)

Conclusion:

- **TOGAF** ensures the application is well-designed, scalable, and aligned with business goals.
- **ITIL** ensures the application operates efficiently and delivers value post-deployment.
- Together, these frameworks cover the full lifecycle: from architecture and development (TOGAF) to reliable operations and service management (ITIL).



TOGAF Framework



ITLL framework

Technical Solution Description for Agri-Energy Connect Prototype:

Introduction

In part 2 I had created a prototype using a Windows Form Application, through feedback and more extensive research an ASP.NET MVC (model,view,controller) application was created because through research this application type fully compliments all technical and non-technical requirements needed for the application therefore:

The **Agri-Energy Connect Platform** is a prototype web application designed to support farmers and employees in managing agricultural data efficiently. The system offers a user-friendly interface, secure data handling, and future-ready architecture.

This solution is built with modern web technologies to demonstrate the core functionalities and potential scalability of the platform. The prototype emphasizes simplicity, accessibility, and reliability, making it a robust tool for managing farmer profiles and their product listings.

(Simon, 2018)**,** (Geeksforgeeks, 2023)

Core Features and Their Benefits:

To make the system easy to use and effective for its target audience, the platform focuses on these core features:

1. Role-Based Access Control

- **What It Is**: The system has two types of users—farmers and employees. Each user type has access to specific features tailored to their needs.
- **How It Works**: Farmers can only manage their own products, while employees can view data from all farmers and use advanced tools like filtering.
- **Business Benefit**:
    - Keeps data secure by ensuring users only access what they are authorized to.
    - Simplifies the experience for each role by showing only relevant tools and data.

2. Product Management for Farmers

- **What It Is**: Farmers can add, view, and manage their products. For each product, they can input details like:
    - Product Name (e.g., "Organic Wheat").
    - Category (e.g., "Grain").
    - Production Date (to track when it was produced).
- **How It Works**: Farmers log into their accounts, click on the "Add Product" button, and fill out a simple form.
- **Business Benefit**:
    - Reduces manual paperwork, saving farmers time.

- Provides a digital record of products, helping farmers keep track of their inventory.

3. Advanced Product Filtering for Employees

- **What It Is**: Employees can view all farmer products and use filters to narrow their search based on specific criteria such as:
  - Product Category (e.g., "Vegetables").
  - Production Date (e.g., "Products produced in the last 30 days").
- **How It Works**: Employees access a search panel, enter their filter preferences, and see the refined results instantly.
- **Business Benefit**:
  - Helps employees quickly find the data they need.
  - Improves productivity by reducing time spent searching through irrelevant information.

4. Responsive Design

- **What It Is**: The platform adapts to the size of the user's screen, whether it's a computer, tablet, or smartphone.
- **How It Works**: Built using responsive web design principles, the interface adjusts automatically to look good and function well on any device.
- **Business Benefit**:
  - Farmers and employees can use the system anywhere, even in the field, with a smartphone or tablet.
  - Enhances user satisfaction by providing a seamless experience on all devices.

5. Data Validation and Error Handling

- **What It Is**: The system checks the data entered by users to ensure it's correct and complete. It also prevents crashes by handling unexpected errors gracefully.
- **How It Works**: For example:
  - If a farmer forgets to enter a product name, the system will display a message prompting them to fill it in.
  - If a system error occurs, the user sees a friendly error message, and the issue is logged for troubleshooting.
- **Business Benefit**:
  - Ensures accurate data, which is critical for making decisions.
  - Prevents frustration for users by guiding them to fix errors easily.

(Geeksforgeeks, 2023)

Technical Overview

To ensure the system is both reliable and scalable, it was developed using cutting-edge tools and frameworks.

Key Technologies

1. **ASP.NET Core**:
    - A powerful framework for building secure and scalable web applications.
    - Provides fast performance, making the platform responsive and reliable.
2. **SQL Server**:
    - A robust database system used to store information about farmers and products.
    - Ensures that data is organized and accessible.
3. **Entity Framework Core**:
    - Simplifies the interaction between the application and the database, making development faster and less error-prone.
4. **Bootstrap**:
    - A front-end framework used to create a responsive and user-friendly interface.

How It All Works Together

1. Farmers and employees interact with the system through a **web browser**.
2. The browser sends requests to the **web application**, hosted on a secure server.
3. The application processes these requests and retrieves or updates data stored in the **SQL Server database**.
4. The results are displayed back to the user in an intuitive interface.

System Architecture Diagram

(A diagram can show the interaction between users, the web application, and the database. It would highlight the user roles, requests sent to the server, and how data is retrieved or updated.)

Scalability and Future-Readiness

This prototype is designed with the future in mind. While it demonstrates essential features today, the architecture supports easy expansion.

Future Enhancements

1. **Real-Time Analytics**:
    - Add dashboards to provide insights such as sales trends or product availability.
    - Helps farmers and employees make data-driven decisions.

2. **Mobile Application**:
    - Develop dedicated Android and iOS apps for offline functionality and better usability.
3. **Integration with IoT Sensors**:
    - Collect real-time data from sensors in the field (e.g., soil moisture levels, weather conditions).
4. **Third-Party Integrations**:
    - Connect with supply chain management tools or e-commerce platforms to enable seamless sales.

(Geeksforgeeks, 2023)

Development and Operations:

Development Methodology

The prototype was developed using the **Agile methodology**:

- Features were developed incrementally, allowing frequent testing and feedback.
- Prioritized user needs to ensure the final product meets business goals.

Operational Excellence

The platform incorporates **DevOps practices**:

- **Continuous Integration**: Automated testing ensures that new features do not break existing functionality.
- **Continuous Deployment**: Frequent updates allow for rapid improvement based on user feedback.

Business Value:

The Agri-Energy Connect Platform directly supports the goals of modernizing agricultural operations and improving efficiency.

(Geeksforgeeks, 2023)

Key Benefits:

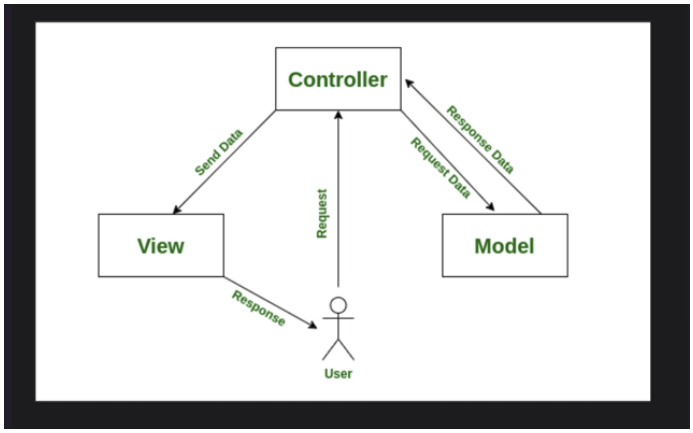
- **Efficiency**: Farmers save time by managing products digitally, and employees can access relevant data quickly.
- **Scalability**: The system can grow to support more users, features, and integrations.
- **Accessibility**: Works on any device, making it easy for users to adopt.
- **Data Accuracy**: Validation ensures reliable information for better decision-making.

(Geeksforgeeks, 2023)

Conclusion

The Agri-Energy Connect Platform is a modern, user-friendly solution for managing agricultural data. By using secure, scalable technologies and focusing on user needs, it offers significant benefits to farmers and employees alike. This prototype lays the foundation for a future-ready system capable of transforming agricultural operations.

(Geeksforgeeks, 2023)



MVC Diagram.

Reference list:

Anantatmula, V. S. & Anantatmula, M., 2008. *use of agile methodology for IT consulting projects*. [Online]
Available at: https://www.pmi.org/learning/library/use-agile-methodology-consulting-projects-7113#:~:text=Agile%20project%20management%20methodology%20is,the%20life%20of%20the%20project.
[Accessed 20 November 2024].

Ardoq, 2024. *what is TOGAF? definition and uses of this EA framework*. [Online]
Available at: https://www.google.com/amp/s/www.ardoq.com/knowledge-hub/togaf%3fhs_amp=true
[Accessed 20 November 2024].

Bigelow, S. J., Courtemanche, M. & Gillis, A. S., 2024. *What is DevOps? Meaning, methodology and guide*. [Online]
Available at: https://www.techtarget.com/searchitoperations/definition/DevOps
[Accessed 20 November 2024].

Dwyer, J., 2023. *5-step deployment checklist & comprehensive SDLC guide*. [Online]
Available at: https://zeet.co/blog/software-deployment-checklist
[Accessed 20 November 2024].

Geeksforgeeks, 2023. *what is MVC*. [Online]
Available at: https://www.geeksforgeeks.org/benefit-of-using-mvc/
[Accessed 20 November 2024].

Hajela, S., 2024. *TOGAF vs Zachman: which enterprise architecture framework should you choose*. [Online]
Available at: https://cioindex.com/reference/togaf-vs-zachman/
[Accessed 20 November 2024].

Laoyan, S., 2024. *What is Agile methodology?(A beginners guide)*. [Online]
Available at: https://asana.com/reasources/agile-methodology
[Accessed 20 November 2024].

Marusiak, W., 2021. *How to do DevOps*. [Online]
Available at: https://www.atlassian.com/devops/what-is-devops/how-to-start-devops
[Accessed 20 November 2024].

McGuire, J., 2024. *Software Development Standards: Best practices and strategic considerations in development standards*. [Online]
Available at: https://www.pulsion.co.uk/blog/software-development-standards/

S., 2018. *15 simple ASP.NET performance tuning tips*. [Online]
Available at: https://stackify.com/asp-net-performance-tuning/

White, S. K. & Greiner, L., 2022. *what is ITLL? your guide to the IT infrastructure*. [Online]
Available at: https://www.cio.com/article/272361/infrastructure-it-infrastructure-library-itil-definition-and-solutions.html?amp=1
[Accessed 20 November 2024].