Smart Agriculture: Software Platform for Telematics Monitoring in Farm Machinery

De Jong Yeong
STEM, IMaR Research Centre
Munster Technological University
(MTU), Co. Kerry
Tralee, Ireland
dejong.yeong@mtu.ie

Krishna Panduru
STEM, IMaR Research Centre
Munster Technological University
(MTU), Co. Kerry
Tralee, Ireland
krishna.panduru@mtu.ie

Joseph Walsh
STEM, IMaR Research Centre
Munster Technological University
(MTU), Co. Kerry
Tralee, Ireland
joseph.walsh@mtu.ie

***Abstract*—Agriculture 4.0, also known as Smart Farming or AgriTech 4.0, signifies a paradigm shift in agribusinesses. It represents the fourth agricultural revolution that uses state-of-the-art and innovative technologies, such as Internet of Things (IoT) and data analysis, to cultivate a sustainable, resilient, and more efficient agricultural practices that can address the ever-growing challenges of the 21st century. This paper proposes the development and implementation of a software system designed to gather, store, and monitor telematics data in farm machinery. By reviewing various software components (front-end and back-end), we aim to provide insights into the key components of the software system and outline a scalable software architecture for monitoring machine telematics in smart farm machinery. Using the proposed software system (web application), users can view historical machine telematics data and monitor the operational performances to optimize farming practices. Furthermore, our system allows real-time alerts to ensure timely intervention and offers potential for integrating predictive analytics to anticipate maintenance needs and improve operational efficiencies.**

Keywords—agriculture, smart farming, telematics monitoring, software platform, front-end, back-end, progressive web app

I. Introduction

Agriculture 4.0 has been identified as a top priority in the strategic technology landscape for the global economy in the next decade, as highlighted in the World Economic Forum's report titled "Markets of Tomorrow 23" [1]. This recognition underscores the pivotal role of innovative and transformative trends in agriculture and such developments are imperative to move towards a smarter, more efficient, and environmentally sustainable agricultural sector, which are essential to address the ever-growing challenges, such as population growths and climate change [2]. Agriculture 4.0 paradigm builds upon the previous agricultural revolutions and leverages digitalization, automation, and data-driven decision-making to optimize and improve various aspects of the agricultural practices.

At its core, Agriculture 4.0 paradigm consists of four core technologies: (a) IoT devices; (b) cloud computing to collect and process data; (c) data analytics; and (d) decision-making support application for user interaction and data visualization. *IoT devices* can be considered as a network of interconnected devices that are embedded with sensors and software systems to collect and share information across various platforms [3]. From livestock and crop management to machine, waste, and nutrient management, IoT devices are essential in measuring the environmental and telematics parameters, e.g., pH levels, temperature, and vacuum pump speed, to provide insight into its operational effectiveness and environmental repercussions [4]. This data is then transmitted wirelessly to a central hub or cloud-based platform for processing and analysis – a process commonly referred to as *cloud computing* that utilizes remote servers hosted on internet to store, manage, and analyse data. It provides inexpensive data storage services, intelligent large scale computing systems to transform raw data into insightful information, and a secure platform to develop agriculture IoT applications [5]. However, cloud computing is susceptible to network delays and would struggle to manage the continuous flow of data between IoT-devices and the cloud, especially in rural areas with limited internet access.

As IoT devices continue to generate vast amounts of data, big *data analytics* emerges as a vital component in extracting valuable and actionable insights from multi-sources real-time and historical agricultural datasets. In this process, it involves aggregating and analysing the dataset to identify the patterns, trends, and correlations using advanced analytical algorithms and computational methods, such as predictive modelling and machine learning. The resulting insights can then be employed to refine agricultural practices and enact data-driven solutions aimed at enhancing efficiency, productivity, and sustainability [3]. Besides, *decision-making support applications* are critical to provide user interaction and data (insights) visualization. It serves as a tool for end-users to interact with analysed data or visualize machinery performance metrics and make informed decisions through interactive dashboards (charts or graphs) in mobile applications or web-based platforms [6]. For instance, xFarm is a multi-platform (mobile and web-based) application that offers comprehensive, end-to-end solutions for farmers to enhance sustainability, efficiency, and productivity, including livestock monitoring, fleet or asset management, environment monitoring, and agricultural sustainability management [7].

This study centres on the technological aspects involved in prototyping a decision-making support system and exploring cloud computing, including cloud-based databases. It aims to outline a scalable software architecture of the system, present an Entity Relationship (ER) diagram, and propose a software system (a progressive web application) that uses cutting-edge software frameworks and cloud-based databases to gather and monitor farm machinery telematics data. Section II presents a summary of the prior art. Section III explores the cutting-edge software technologies or frameworks to develop a web-based platform. Section IV outlines the architecture of the proposed software system and an ER diagram for database designs, and discusses the adopted frameworks to develop the application. A detailed discussion of the underlying hardware architecture

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094_P2 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero – the Science Foundation Ireland Research Centre for Software (www.lero.ie)

2024 35th Irish Signals and Systems Conference (ISSC) | 979-8-3503-5298-6/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISSC61953.2024.10602840

and the selection of sensors to develop smart farm machinery is out of scope in this study. Section V summarizes the current research and outlines and outlook for future work.

II. Related Work

Prior literature on decision-making support applications in agriculture has thoroughly investigated various facets of farm management, ranging from crops management applications to fleet or asset management systems. These studies have aimed to develop innovative solutions that utilize both hardware and software technologies to optimize decision-making processes based on analysed data. In this review, our main focus lies on the software technology, i.e., software frameworks and cloud platforms, employed in the development of a software system (user interface) for user interaction and visualizing telematics data. For example, Negru, C. et al. introduced CLUeFARM – a cloud-based web platform that monitors and manages smart farm activities or workflows. It was built based on the end-to-end microservice architectural approach, which highlights its scalability, flexibility, and resilience, and comprises two core applications that maintain constant communication with each other via Representational State Transfer (REST), mainly [8]:

- A *back-end application* built using Spring Framework (Java) and MongoDB, cloud database. It is responsible for managing user authentication and authorization, as well as data operations, such as CRUD (Create, Read, Update, and Delete) methods, processing, and storage. It also integrates with other services, including e-mail service to send emails and a weather service to collect weather data.
- A dynamic *front-end application* that offers numerous features that facilitates informed decision-making and optimize workflows. These includes task management tools, greenhouse geospatial coordination capabilities, and an interactive data visualization dashboard. It was built employing the well-known AngularJS JavaScript Framework, integrated with Google Maps to visualize geospatial coordinates, and styled with Bootstrap 3 [8].

Reference [9] proposed a cloud-based data-driven system designed for smart farming applications. It supports real-time or near real-time livestock remote monitoring system in farms and a scalable intelligent real-time analytic system to analyse the data using cloud computing technologies. The application uses some of the serverless cloud-based services in Microsoft Azure to (a) establish a reliable bi-directional communication between on-premise IoT devices and the cloud; (b) gather and process large volumes of fast streaming agricultural data with minimal latency from multiple sources; and (c) predict future milk quantities using machine learning model – regression. In addition, the application integrates dynamic Power BI reports and dashboards to visualize real-time data and historical data, as well as reported insights.

Reference [10] presented an end-to-end AI-powered IoT-based low-cost platform to monitor environment parameters, such as air temperature and humidity, soil moisture, and solar radiation, for smart farming. It comprises five main layers: (a) *perception layer* – sense and gather environmental parameters using a suite of heterogeneous sensors; (b) *transmission layer* – establish real-time and near-real-time data communications between network sensors and the base station (in local server) using transmission protocols e.g., Message-Queue Telemetry Transport (MQTT) and Hypertext Transfer Protocol (HTTP); (c) *middleware layer* – serve as the base station or central hub of the platform that is responsible for storing, processing, and analysing the sensor data. It uses various software like Node-Red, a visual programming tool designed for developing IoT systems, InfluxDB open-source database for persisting time-series data, OpenVPN for establishing a secure virtual private network (VPN) connection, and Apache for deploying a local web server to the cloud; (d) *presentation layer* – an interactive graphical user interface that monitors the evolution of critical parameters, creates alerts and notifies users through different channels, as well as visualizes real-time data and insights; (e) *management layer* – constantly monitor health statuses of the base station and provide real-time alarm of different network elements. In summary, its hybrid and layered architecture that combines centralized and distributed techniques enhances the overall system scalability, portability, and flexibility, making it suitable for various agricultural needs.

Our approaches to designing and delivering a scalable and adaptable software application for telematics data monitoring are to investigate the current front-end and back-end software technologies and identify the appropriate frameworks or tools that meet the specific user requirements of our application: (a) telematics monitoring; (b) near-real-time notification; and (c) fleet management.

III. Software Technologies

In modern software development, the integration of state-of-the-art, innovative technologies has revolutionized the way applications are designed, developed, deployed, and accessed. In software development, front-end technologies manage the design, functionality, and user interface of the applications. It dictates the visual elements, user interactions, and the overall user experience. In contrast, back-end technologies are tasked with data management i.e., processing, storage, and retrieval, and managing communication between the database or server and the front-end (client). It ensures seamless interaction and data flow across the system. Moreover, back-end systems can adopt cloud-based serverless services to reduce infrastructure management and enhance flexibility, scalability, security, and cost-efficiency [5].

In this section, we examine some of the technologies, key concepts, and software frameworks that can be utilized in the development of both the front-end and back-end components of the proposed application.

A. Front-end Technologies

In recent years, various front-end development techniques and strategies have been introduced to enhance cross-platform accessibility and the selection of these methods plays a crucial role in defining the overall user experiences and performance of the software systems. Among these, the concepts of native mobile apps, hybrid apps, and Progressive Web Apps (PWA) have gained significant attention due to the increasing demand for a dynamic and engaging cross-platform digital experience [11].

1) Native mobile apps: It refers to applications that were developed specifically for a particular operating system (OS) or platform, such as Apple iOS or Android. Such applications are built using platform-specific languages and frameworks, e.g. Swift or Objective-C for iOS devices and Java or Kotlin for Android devices. These applications do not rely on third-party plugins (or middleware) of the device's built-in feature and functionality, thereby enhancing security, reliability, and

optimized performance. However, native mobile apps require distinct codebase for different platforms, which can be time-consuming and resource-intensive [12].

2) Hybrid Apps: It refers to applications built utilizing an amalgamation of native apps elements and web technologies e.g., Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript, packaged within a native container that allows the applications to function across multiple platforms – an approach commonly known as cross-platform [13]. In other words, it comprises a single, reusable codebase that can be deployed across various OS and devices, thereby streamlining the development process and reducing time-to-market and resource expenditure. However, hybrid apps have limited access to native built-in features and may experience performance degradation due to their reliance on processing memory and power issues. In addition, it poses a maintenance challenge to guarantee consistent performance across various platforms and the reliance of third-party plugins/frameworks may pose security challenges [14]. Example frameworks that can be adopted to develop hybrid apps are: React Native [15] and Flutter [16].

3) Progressive Web Apps: It is a revolutionary approach to develop app-like applications. In 2015, Google introduced PWA [17], which aims to bridge the gap between traditional web and mobile experiences by leveraging the capabilities of native mobile apps and modern web technologies and allows developers to implement responsive, reliable, and interactive applications that can access through web browsers on various devices. It also offers push notifications, offline capabilities, web bluetooth technologies, and eliminates the need for app store distribution and manual updates [18]. Nonetheless, it is crucial to acknowledge that PWA lacks full access to device-specific features and components, such as iOS Bluetooth and iOS Face ID. Besides, PWAs are incompatible with obsolete browsers and old devices. For instance, push notifications in PWA is incompatible with Safari in Apple iOS devices [19]. Example frameworks that can be adopted to build PWAs are Next.js [20], Vue.js [21], and AngularJS [22].

Table I below summarizes the pros and cons of the front-end concepts: Native Apps, PWAs, and Hybrid Apps, based on various parameters.

TABLE I. AN OVERVIEW OF THE FRONT-END CONCEPTS [12].

Parameters	***Native Apps***	***PWAs***	***Hybrid Apps***
Performance	Fast	Fast	Slow
Offline Support	Yes	Yes	Yes
Security	High	Moderate	Low
Complexity	High	Low	Moderate
Codebase	Multiple	Single	Single
Maintenance	High	Low	Moderate
Installable	Yes	Yes	Yes
Cross-Platform	No	Independent via browser	Yes
R&D Cost	High	Low	High
Code Reusability	No	Yes	Yes
Native APIs	Yes	Very Limited	Limited
Technology Used	Platform-specific languages	Web technologies	Web technologies and native elements

B. Back-end Technologies

Back-end technologies comprise a broad range of services and platforms that empower the functionalities and operations of digital systems. Such technologies serve as the backbone of software applications, facilitating essential tasks, such as user authentication and authorization, data processing, storage, and management. Their multifaceted capabilities are instrumental in facilitating seamless communication between the front-end components and the underlying infrastructures [23]. In recent years, the landscape of back-end technologies have witnessed a significant transformation, largely driven by the widespread adoption of cloud computing and the advent of various "as-a-service" models. Each model offers a level of abstraction that minimizes the efforts required to maintain the infrastructures. These models are [24], [25], [26]:

- *Infrastructure-as-a-Service* (IaaS): IaaS revolutionizes traditional on-premise IT infrastructure by virtualizing computing resources such as servers, networking, and storage. IaaS offers on-demand access to cloud-hosted resources and abstracts complex tasks associated with managing and maintaining physical infrastructure and data centre. Nevertheless, developers are still required to configure security settings and manage applications and OS. In addition, IaaS environments are commonly shared among multiple users, which can pose potential security breaches and performance degradation during peak usage times [27]. Examples of IaaS providers are Google Compute Engine, Azure Virtual Machine, and Amazon EC2.
- *Platform-as-a-Service* (PaaS): It is a cloud computing model that offers end-to-end solutions for developing, deploying, and administering of applications via third-party service provider. Unlike IaaS, PaaS abstracts the underlying infrastructure complexity and offers a pre-configured, comprehensive platform environment e.g. software tools, frameworks, and runtime environment in the cloud. In other words, PaaS allows developers to focus on building and optimizing business applications without managing and maintaining the infrastructures. However, PaaS applications pose potential challenges to customize highly specialized business functionality or integrate with other third-party services – otherwise referred to as a vendor lock-in [28]. Examples of PaaS provides include AWS Lambda, Vercel, and Heroku.
- *Software-as-a-Service* (SaaS): It is a software delivery model wherein software applications are managed and hosted by the service providers and made accessible to users over the internet. In this model, service provider is responsible to manage all aspects of the application, including the underlying infrastructures, maintenance, updates, and security. Users, in turn, gain access to the applications via a subscription-based or pay-as-you-go model, eliminating the need for upfront investments in software licenses or infrastructure. However, the cons of SaaS applications are limited customization options to handle complex business requirements. In addition, reliable internet connection is required to access to the application effectively; and software maintenance and

technical glitches may cause service interruptions [29]. Examples of SaaS solutions include Resend to deliver emails and Slack for teams collaboration.

- *Database-as-a-Service* (DBaaS): DBaaS is a managed or hosted cloud service model that enables developers to provision and access database instances on-demand through Application Programming Interfaces (API) or a web-based interface. Thus, it eliminates the need for physical hardware installation, configuration, updates, and maintenance. In addition, DBaaS offers automatic backups, data replication, and ensures scalability, data durability, high availability, and security measures like encryption and access control. It also supports various types of databases including relational databases, time series databases, and NoSQL databases [30]. However, DBaaS applications pose interoperability challenges to integrate with on-premises systems or other providers and lack of direct control over data [31]. Examples of DBaaS applications include InfluxDB, MongoDB, and Amazon RDS.

Other models may include *Backend-as-a-Service* (BaaS) – a cloud computing solution that abstracts the underlying back-end infrastructure and offers features like user authentication, data storage, push notifications, and business logics. It allows developers to focus on building the front-end components and accelerates time-to-market [32]. Examples of BaaS providers include Supabase and Convex; Content-as-a-Service (CaaS) – a web-based content management infrastructure that provides flexible content management and distribution to the front-end system [33]; etc [34]. A comparative overview of IaaS, PaaS, and SaaS, along with a detailed review of the appropriate use cases for utilizing these "as-a-service" models are available in [24] and [26], respectively.

IV. Findings / Results

Section IV outlines the underlying architecture that forms the proposed telematics monitoring application and discusses the employed front-end and back-end technologies to develop the proposed telematics monitoring application in accordance with pre-defined requirement specifications and development resource constraints. However, it is important to acknowledge that a detailed review of the underlying hardware architecture in the farm machines is out of scope in this study.

Fig. 1. depicts the underlying architecture of the proposed web-based telematics monitoring application. It encompasses three main components: (a) client application; (b) cloud-based user authentication and databases with BaaS and DBaaS; and (c) machines hardware. In the *client application*, **Next.js** [20] is utilized to implement the web-based application and styled utilizing **shadcn/ui** [35]. Next.js is an open-source JavaScript web development framework that enables developers to build dynamic and responsive web applications. It supports server-side rendering – a method that significantly enhances loading times, built-in optimization, and seamless integration of third-party services or API routes for serverless functions. Thus, it optimizes user experience and performance [36]. In addition, **Ant Design Charts** [37], a chart components library with 27 chart types, is utilized to visualize time-series data as it offers high-interactive charts and enhances the effectiveness of data representations. **Resend** [38] – a third-party SaaS platform is adopted to create impactful email communications. It enables developers to build, test, and send transaction emails at scale, such as invitation emails, email or password reset emails, etc. Moreover, PWA features were integrated into the application to provide offline capabilities and mobile app-like experience employing a third-party plugin [39]. Finally, the application is hosted on **Vercel** [40] platform, which provides fast and easy deployment and detailed debugging logs.

BaaS and DBaaS models were used to build the *back-end components* (authentication and data storage) of the proposed application. **Supabase** [41] was selected to develop the back-end infrastructure of the proposed application as it provides a wide range of services, such as authentication service, hosted Postgres relational database to store entity (user and machine) data, and real-time capabilities. **Drizzle ORM** [42], an open-source, lightweight object relational mapping (ORM) library, was used to simplify the handling of database operations, i.e., queries and mutations, when interfacing with the database. In addition, the client application leverages Supabase's real-time functionality to promptly generates notifications when a new

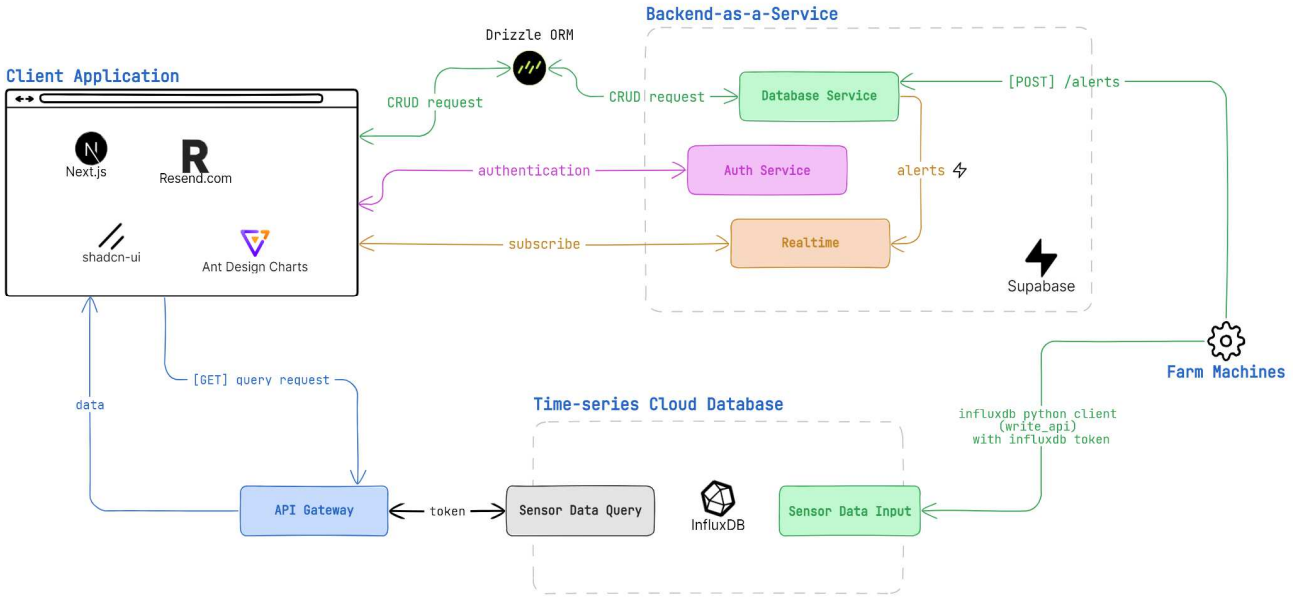

Fig. 1. An overview illustrating the underlying front-end and back-end architecture of the proposed web-based telematics monitoring application.

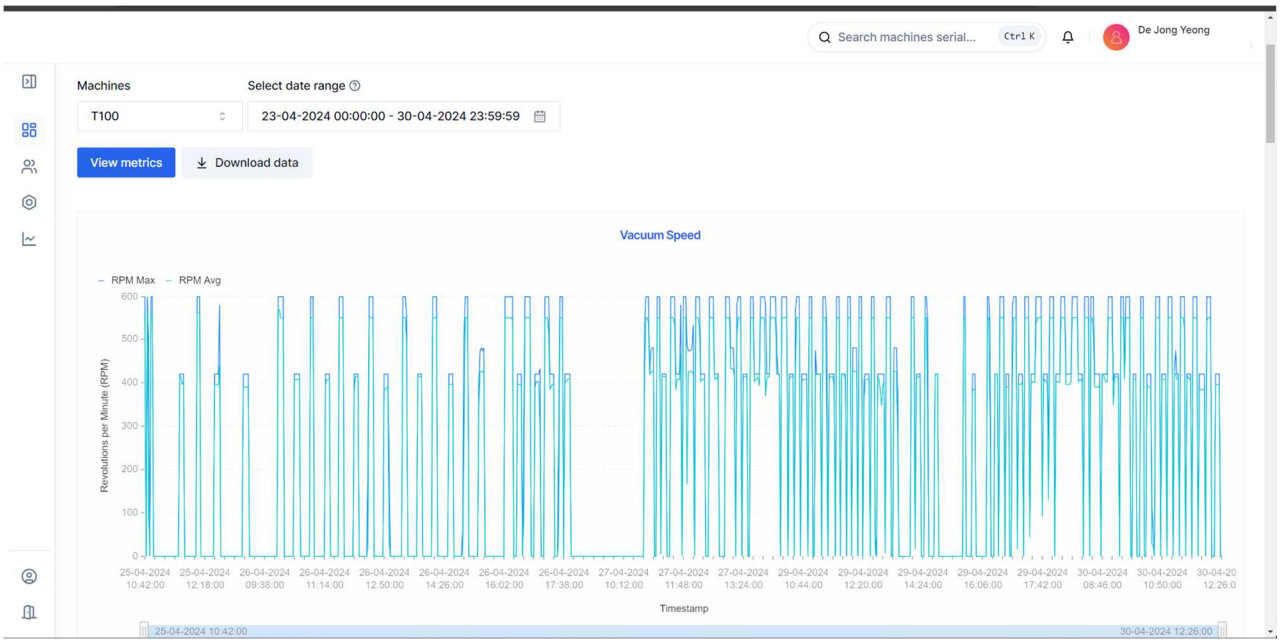

Fig. 2. Screenshot of the client front-end application for user interaction and data visualization.

alert is inserted into the database. These alerts were sent from the machines to the database via a secure REST API.

InfluxDB [43], a time-series DBaaS platform is also used to persist and analyse time-stamped telematics data. A hosted cloud service is adopted to eliminate the need for self-hosting and physical hardware maintenance. In addition, a secure API Gateway is developed adhering to REST protocols to establish telematics data communication between the client application and InfluxDB. InfluxDB is known for its scalability and high performance when querying large volume of time-series data. Nonetheless, InfluxDB is subject to limitations due to its data retention policies which restricts data retention to a maximum of 30 days under the free plan, posing potential challenges for users requiring longer-term data storage and analysis.

In Agriculture 4.0, a potential application of the proposed web-based telematics monitoring application may involve the continuous monitoring and analysis of heavy Agri-machinery performances, including aspects such as the rated speed of the vacuum pump to prevent overheating, tyre pressure to ensure safe and efficient operation, or the oil quality to reduce pump malfunctioning. By leveraging of both the front-end and back-end technologies, users are able to visualize the data based on the selected date range, while the robust cloud-based back-end infrastructure ensures reliable data storage, retrieval, and data analysis, as illustrated in Fig. 2. In conclusion, this section has deliberated the front-end and back-end technologies that were used to design and prototype the proposed telematics system, including: (a) front-end: Next.js, shadcn/ui, Ant Design Chart, and Vercel; (b) back-end: Supabase and InfluxDB. However, the selection of strategies and approaches to develop the back-end and front-end architectures must be carefully considered based on multiple factors, such as target audience, scalability, accessibility, performance requirements, maintainability, and cross-platform compatibility.

V. Conclusion and Future Work

In the current work, a scalable and cloud-hosted front-end and back-end architecture was outlined to design and develop an app-like (web-based) telematics monitoring application. It allows stakeholders perform operations such as user and fleet management, farm machine telematics monitoring, and make informed decisions based on analysed data and near-real-time or real-time notifications. Moreover, the proposed techniques and approaches: PWA and Cloud Computing are secure, cost effective, flexible, and ensure high availability and reliability with built-in redundancy and failover mechanisms. However, our approaches are dependent on third-party frameworks and services may pose potential vendor lock-in, dependency risks, and restrict portability. Further development may incorporate real-time (timestamp) data streaming, querying, and analytics using Apache Kafka [44]; and a machine learning-based farm machines health report generation to predict potential failures and provide recommendations for preventive maintenance or machine operations optimization [9].

Acknowledgment

The successful completion of this paper and the associated research owes much gratitude to the invaluable assistance and enlightening guidance provided by the IMaR team at Munster Technological University (MTU), Co. Kerry. Their guidance has been instrumental in ensuring the smooth implementation of this project.

References

[1] A. Collins, P. Grosskurth and S. Zahidi, "Markets of Tomorrow Report 2023: Turning Technologies into New Sources of Global Growth," World Economic Forum, Switzerland, 2023.

[2] U. Pati, D. S. Usha and N. J. Banu, "Advanced Agriculture Field Monitoring and Controlling System," International Research Journal of Engineering and Technology (IRJET), vol. VII, no. 5, pp. 7943-7948, 2020.

[3] R. Chataut, A. Phoummalayvane and R. Akl, "Unleashing the Power of IoT: A Comprehensive Review of IoT Applications and Future Prospects in Healthcare, Agriculture, Smart Homes, Smart Cities, and Industry 4.0," Sensors, vol. XXIII, no. 16, p. 7194, 2023.

[4] A. Ali, T. Hussain, N. Tantashutikun, N. Hussain and G. Cocetta, "Application of Smart Techniques, Internet of Things and Data Mining for Resource Use Efficient and Sustainable Crop Production," Agriculture, vol. XIII, no. 2, p. 397, 2023.

[5] X. Shi, X. An, Q. Zhao, H. Liu, L. Xia, X. Sun and Y. Guo, "State-of-the-Art Internet of Things in Protected Agriculture," Sensors, vol. XIX, no. 8, p. 1833, 2019.

[6] S. O. Araújo, R. S. Peres, J. Barata, F. Lidon and J. C. Ramalho, "Characterising the Agriculture 4.0 Landscape—Emerging Trends, Challenges and Opportunities," Agronomy, vol. XI, no. 4, p. 667, 2021.

[7] xFarm Technologies, "xFarm Technologies - The app for farm management," 2024. [Online]. Available: https://xfarm.ag/en. [Accessed 29 February 2024].

[8] C. Negru, G. Musat, M. Colezea, C. Anghel, A. Dumitrascu, F. Pop, C. D. Maio and A. Castiglione, "Dependable workflow management system for smart farms," Connection Science, vol. XXXIV, no. 1, pp. 1833-1854, 2022.

[9] K. Dineva and T. Atanasova, "Cloud Data-Driven Intelligent Monitoring System for Interactive Smart Farming," Sensors, vol. XXII, no. 17, p. 6566, 2022.

[10] A. Faid, M. Sadik and E. Sabir, "An Agile AI and IoT-Augmented Smart Farming: A Cost-Effective Cognitive Weather Station," Agriculture, vol. XII, no. 1, p. 35, 2022.

[11] D. Palumbo, "The Flutter Framework: Analysis in a Mobile Enterprise Framework," Politecnico di Torino, Turin, Italy, 2021.

[12] A. K. Singh and R. Narayan, "Progressive Web Apps: A Smart Way to Build Mobile-Web Apps for Academic Libraries," in Applying Mobile Technologies in Transformation of Library Services, India, Social Development Federation, 2021, pp. 91-117.

[13] W. Berggren, "An analysis and comparison of the Native mobile application versus the Progressive web application," Mid Sweden University, Östersund, Sweden, 2023.

[14] L. Simoliunaite, "Mobile app replacing loyalty cards," Häme University of Applied Sciences, Hämeenlinna, Finland, 2023.

[15] Meta Platforms, Inc., "React Native," Meta, [Online]. Available: https://reactnative.dev/. [Accessed 2024].

[16] Google, "Flutter," Google, [Online]. Available: https://flutter.dev/.

[17] J. Spero, "Why a progressive web app might be right for you," July 2017. [Online]. Available: https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/progressive-web-apps-benefit-brands/. [Accessed 2 March 2024].

[18] U. A. Abdurakhimovich, "The Future of JavaScript: Emerging Trends and Technologies," in Formation of Psychology and Pedagogy as Interdisciplinary Sciences, Italy, 2023.

[19] K. I. Roumeliotis and N. D. Tselikas, "Evaluating Progressive Web App Accessibility for People with Disabilities," Network, vol. II, no. 2, pp. 350-369, 2022.

[20] Vercel, "Next.js," Vercel, 2024. [Online]. Available: https://nextjs.org/. [Accessed 3 March 2024].

[21] E. You, "Vue.js," 2014-2024. [Online]. Available: https://vuejs.org/. [Accessed 3 March 2024].

[22] Google, "Angular," 2024. [Online]. Available: https://angular.io/. [Accessed 3 March 2024].

[23] Y. Guo, W. Zhang, Q. Qin, K. Chen and Y. Wei, "ntelligent manufacturing management system based on data mining in artificial intelligence energy-saving resources," Soft Computing, vol. XXVII, no. 2023, pp. 4061-4076, 2022.

[24] A. Babaei, P. M. Kebria, M. M. Dalvand and S. Nahavandi, A Review of Machine Learning-based Security in Cloud Computing, arXiv, 2023.

[25] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy and B. Luo, "SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review," Big Data Cogn. Comput., vol. VII, no. 2, p. 97, 2023.

[26] M. Kavis, "Cloud Service Models," in Architecting The Cloud, Hoboken, New Jersey, John Wiley & Sons, Inc., 2014, pp. 13-22.

[27] T. Sale, "Exploring the Advantages and Disadvantages of Cloud Computing Services for Businesses," 28 February 2023. [Online]. Available: https://www.linkedin.com/pulse/exploring-advantages-disadvantages-cloud-computing-services-tony-sale/?trk=article-ssr-frontend-pulse_more-articles_related-content-card. [Accessed 3 March 2024].

[28] N. Sakovich, "IaaS vs. PaaS vs. SaaS: What's the Difference?," 2 February 2024. [Online]. Available: https://www.sam-solutions.com/blog/iaas-vs-paas-vs-saas-whats-the-difference/. [Accessed 3 March 2024].

[29] M. Rahman, "Disadvantages of SaaS," 27 August 2023. [Online]. Available: https://www.linkedin.com/pulse/disadvantage-saas-mizanur-rahman. [Accessed 3 March 2024].

[30] S. K. Akinade, "Database as a service: Security and privacy issues, and appropriate controls," April 2020. [Online]. Available: https://era.library.ualberta.ca/items/2e0befbc-8065-4b24-a0a8-1b02ffe2bf06/view/445fad25-14fe-41b7-8519-83c79afb3e7e/Akinade_2020_Spring_MISSM.pdf. [Accessed 4 March 2024].

[31] M. Al-Refai, A. Haya, H. Fawareh and H. H. Khafajeh, "Database as a Service (DBaaS) Challenges and Solutions," in 2021 22nd International Arab Conference on Information Technology (ACIT), Muscat, Oman, 2021.

[32] O. Glib, "Backend as a Service (BaaS): What is it & Key Benefits," 24 January 2024. [Online]. Available: https://acropolium.com/blog/first-look-at-backend-as-a-service. [Accessed 3 March 2024].

[33] A. Sivakrishnan, "What is Content-as-a-Service? Definition, Examples and Top 15 CaaS Vendors in 2020," 28 May 2020. [Online]. Available: https://www.spiceworks.com/tech/it-strategy/articles/what-is-content-as-a-service/. [Accessed 3 March 2024].

[34] H. Hourani and M. Abdallah, "Cloud Computing: Legal and Security Issues," in 2018 8th International Conference on Computer Science and Information Technology (CSIT), Amman, Jordan, 2018.

[35] shadcn, "shadcn/ui," shadcn, 2023. [Online]. Available: https://ui.shadcn.com. [Accessed 23 September 2023].

[36] J. Dakowicz, "Pros and Cons of Next JS - 2024 Updated Version," 5 January 2024. [Online]. Available: https://pagepro.co/blog/pros-and-cons-of-nextjs/#cons-of-using-nextjs. [Accessed 3 March 2024].

[37] AntV, "AntV React," AntV, 2024. [Online]. Available: https://ant-design-charts.antgroup.com. [Accessed 22 April 2023].

[38] Z. Rocha, "Resend," Resend, 2024. [Online]. Available: https://resend.com. [Accessed 23 November 2023].

[39] N. Đ. Anh, "DuCanhGH/next-pwa: Zero-config PWA plugin for Next.js," GitHub, 2024. [Online]. Available: https://github.com/DuCanhGH/next-pwa. [Accessed 5 February 2024].

[40] Vercel, "Vercel," Vercel, 2024. [Online]. Available: https://vercel.com. [Accessed 3 March 2024].

[41] Supabase, "Supabase | The Open Source Firebase Alternative," Supabase, 2024. [Online]. Available: https://supabase.com. [Accessed 22 June 2023].

[42] D. ORM, "Drizzle ORM," Drizzle ORM, 2024. [Online]. Available: https://orm.drizzle.team. [Accessed 2 March 2024].

[43] InfluxData, "InfluxDB Time Series Platform | InfluxData," InfluxData, 2024. [Online]. Available: https://www.influxdata.com. [Accessed 14 May 2023].

[44] Community, "Getting Started with Apache Kafka and InfluxDB," InfluxData, 29 September 2022. [Online]. Available: https://www.influxdata.com/blog/getting-started-apache-kafka-influxdb. [Accessed 4 March 2024].