Sensors and Materials, Vol. 35, No. 7 (2023) 2603–2618
MYU Tokyo

S & M 3343

A Novel Framework for Integrating Heterogeneous Data Sources through Data Exchange

Yin-Ting Cheng* and Ming-Chih Chen

Department of Electronic Engineering, National Kaohsiung University of Science and Technology (First Campus), No. 1, University Rd., Yanchao Dist., Kaohsiung City 82445, Taiwan

(Received December 31, 2022; accepted July 11, 2023)

Keywords: data integration, modularized agent, interface, artificial intelligence

We present a design framework for an electronic system that facilitates the integration of heterogeneous data sources. To meet the specific requirements of data manipulation, different database systems need to exchange their data content to fulfill the needs of their respective systems. To address this challenge, a data exchange mechanism is proposed to enable the integration of external data from multiple sources and facilitate data exchange among multiple systems. Within this framework, a data exchange agent (DE-Agent) is designed to handle the integration and distribution of multiple external data sources into the internal system. The DE-Agent acts as an intermediary, effectively transmitting external data to the system's database. Experimental results demonstrate that the designed DE-Agent is capable of supporting stimulus operations from over forty inspectors while maintaining the correct and efficient functioning of the system. This research contributes to the development of an electronic system that effectively integrates heterogeneous data sources. The proposed design framework and DE-Agent provide a reliable mechanism for data exchange, enabling seamless integration between different database systems and ensuring the proper functioning of the overall system.

1. Introduction

When an enterprise decides to improve its internal operational processes through information technology, it becomes necessary to digitize and electronicize existing operations while integrating them with the enterprise's existing software systems for collaborative work. This includes software for financial accounting, enterprise resource planning, human resources management, and other programs. As the enterprise grows, the number and variety of enterprise collaboration software also increase. However, each software is typically independently developed by different manufacturers, which makes communication compatibility and scalability challenging, leading to integration difficulties between information systems. To address these challenges, an intermediary software or program is needed to receive and integrate data from different sources, converting it into the required data format for transmission to the electronic system. This intermediary software plays a crucial role in ensuring compatibility and

*Corresponding author: e-mail: inten789@gmail.com
https://doi.org/10.18494/SAM4299

ISSN 0914-4935
https://myukk.org/

scalability between the different information systems. The conversion of the data to the electronic system's required format enables efficient operation and utilization of the integrated data.

We propose a framework design for integrating heterogeneous data sources, utilizing a data exchange mechanism to facilitate the integration and application of data from different database systems. The primary objective is to integrate diverse data sources, including databases of various types or from various manufacturers, and convert them into the required data format for seamless integration with the electronic system. The framework design incorporates data agent technology to implement the data exchange and integration mechanism within the electronic system.

Furthermore, we suggest combining the equipment inspection workflow with the proposed data exchange mechanism framework as part of the electronic system design. By digitizing equipment inspection processes, enterprises can effectively reduce costs and streamline the integration of different data sources, resulting in a more functional and efficient system application. Overall, this research aims to provide a comprehensive framework and mechanism for integrating heterogeneous data sources within an electronic system, enabling seamless collaboration and efficient utilization of integrated data.

2. Literature Review

ASP.NET is a development platform provided by Microsoft as part of the .NET Framework.[1] It is the successor to ASP technology and is specifically designed for creating web applications with a rich set of libraries. It is important to note that ASP.NET itself is not a programming language, but rather a development platform that can be used with any .NET language, such as C#, VB.NET, and C++.NET, to develop web pages or web services. ASP.NET provides three frameworks for building web applications: Web Forms, ASP.NET MVC, and ASP.NET Web Pages. These frameworks offer different approaches and features for developing web applications based on specific needs and preferences.

The author utilizes ASP.NET to develop a B/S system that supports network information sharing for a dairy cow breeding system.[2] The system architecture comprises three layers, with the client being the first layer responsible for receiving user messages and data feedback. At this layer, the client needs to install a service engine capable of browsing the website. The second layer is the Web Server, responsible for the logical processing of the program and submitting the client's request to the next layer for processing. The final layer is the Database Server, which receives structured query language (SQL) statements provided by the second layer, queries the database, and responds with the results to the user.

In the era of mobile device dominance, the application programming interface (API) has become increasingly important. A software service can be accessed by people on multiple devices, including mobile phones, browsers, and wearable devices.[3] These devices need to interact with the same back-end database to display consistent content. The client, which can be a browser, a mobile phone, a wearable device, or other device, is the party using the API. The client and the server communicate through the hypertext transfer protocol (HTTP) communication protocol, making requests and receiving responses.

The representational state transfer (REST) was proposed by Dr. Roy Fielding in his doctoral dissertation in 2000.[4] REST is an architectural style for decentralized hypermedia systems, and a Web API that adheres to the REST architecture is referred to as a RESTful Web API. At runtime, a RESTful Web API can be decomposed into multiple RESTful Web services. In other words, each web page can be considered as a resource that users can access using a uniform resource locator (URL) address and retrieve for use in a browser. Most RESTful Web APIs follow a set of design principles, including the use of standard HTTP methods. In this article, the API for the raw material item allows users to directly access the data of the raw material item through HTTP. The API structure follows REST principles, requiring the use of a unified URL format.[5] Additionally, specifying the type in advance helps reduce the number of round trips needed to use the interface.

The advanced encryption standard (AES) is a modern encryption method that has been adopted by the US government as a replacement for the data encryption standard (DES).[6] It is widely used in various encryption standards today. DES was replaced primarily because it was found to be an insecure encryption method. It had limitations such as being able to encrypt only 64 bits at a time, which was time-consuming, and having a key length of only 56 bits, which was deemed insecure. In this article, the cloud system is divided into three layers. The first layer consists of the client, where customer data can be stored in the cloud in any form after undergoing identity verification using AES encryption technology.[7] The second layer comprises application servers that include a cloud storage business logic layer and a database. The cloud storage business logic layer stores data directly in the database through encryption. The third layer is handled by a third-party storage server, which typically contains data accessible to third-party agents of the cloud system.

The design of the JSON Web Token (JWT) is well-suited to the stateless principle of RESTful APIs.[8] In a RESTful API, each client request sent to the server is independent, and user authentication occurs only once. After authentication, the server stores the authentication status without needing to reverify it. As a result, each time the client sends a request to the server, it includes a JWT string to indicate its identity. In 2016, Laksono *et al.* developed an application called SIKASIR, which enhances the management performance of Indonesian small or medium size enterprises (SMEs).[9] To apply this management platform, it needs to be connected to the RESTful web of the terminal. This connection requires a stateless identity verification method, making JWT a suitable option.

In 1990, Sheth *et al.* defined a heterogeneous database system (HDBS) as an integrated system comprising computational models and software implementations.[10] HDBS exhibits the following characteristics.

(1) Heterogeneity: Databases in HDBS employ different data models, query languages, and method structures.
(2) Independent control: Each database has the authority to control permissions on its own data within HDBS.
(3) Vertical integration: HDBS integrates previous data or information to achieve the advantages of interoperability, where vertical integration necessitates a unified connection standard within HDBS.

In 1995, Karp proposed a query-based interoperability framework that includes the concept of relationship linking among different biological databases.[11] This framework established a unified management system for integrating diverse biological data and enabled data exchange at various levels. In 2001, Sujansky utilized the model of heterogeneous database systems to provide query functions, resolving the integration of different types of medical data in centralized and distributed databases.[12] In 2012, Kumar *et al.* introduced a standardized framework for integrating heterogeneous sources and applied it to patient record queries across multiple hospitals, receiving significant positive feedback compared with the original systems.[13] In 2017, Koçer and Biroğul developed an asset management system for maintenance and repairs, designed using a web-based architecture.[14] This system facilitates the retrieval of device-related information, data storage, and analysis, thereby reducing the time and cost associated with maintenance and downtime, as judged from the analysis results.

3. Systematic Procedures

3.1 System overview

To address the shortcomings of the traditional filling of the equipment checklist and record retrieval process, we propose the digitization of the process. The shortcomings of the original process are improved through the utilization of the data exchange agent framework and API technology, as designed in this study. This approach aims to facilitate equipment checklists, ensure data storage security, and provide real-time record retrieval. The process is illustrated in Fig. 1.

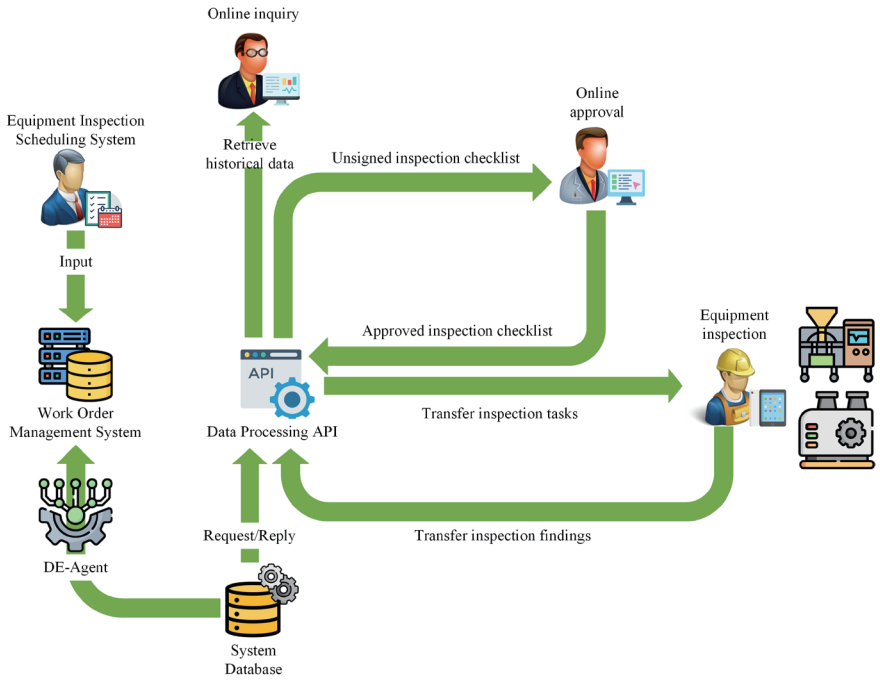

Fig. 1. (Color online) The process of filling in equipment inspection and checking records is electronic.

3.2 Framework design for data exchange mechanism

To address the challenges posed by different formats and inconsistent types of multiple data sources, we face the challenge of the development of a data integration system that necessitates the creation of communication programs for each data source protocol. In light of this, we propose a framework called the data exchange agent (DE-Agent), which integrates an agent database with a data exchange mechanism. The overall scenario is depicted in Fig. 2, which illustrates the concept of the proposed framework.

The DE-Agent architecture consists of two main components: the data exchange mechanism and the agent database. The data exchange mechanism is composed of two modules: the data-out module (DOM) and the data-in module (DIM). On the other component is an agent database that includes two types of table: one for storing synchronization lists and the other for storing synchronizing data. The synchronizing data is referred to as datasets, as there may be multiple data tables to be synchronized. Figure 3 illustrates this architecture.

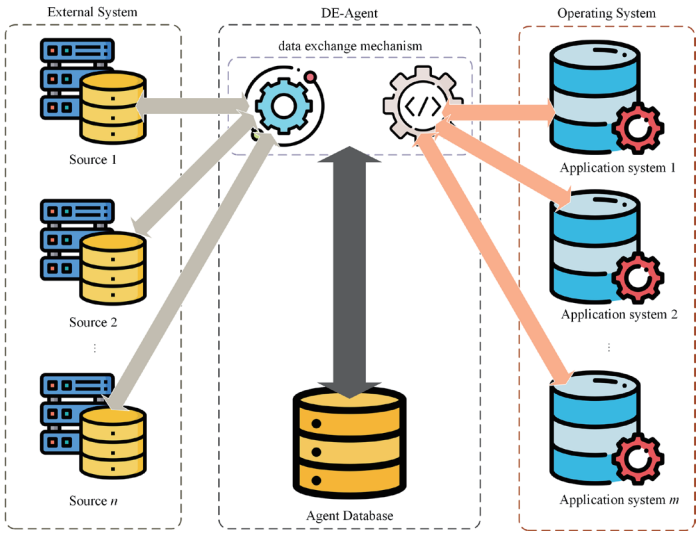

Fig. 2. (Color online) Scenario of DE-Agent.

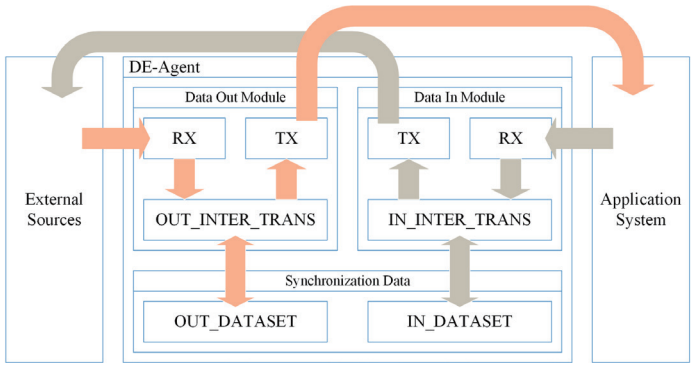

Fig. 3. (Color online) Architecture of DE-Agent.

In the data exchange mechanism, both modules have transmission (TX) and reception (RX) functions, as well as synchronization lists. The DOM is responsible for transferring data from external data sources to the application system, while the DIM handles the flow of data from the application system to external data sources. The definitions of data fields for the synchronized data tables are presented in Table 1.

3.3 Data processing API

Data requests and responses are facilitated through the data processing API, enabling on-site inspectors or engineers to download scheduled equipment checklist items or transmit completed results to the application system database using their mobile devices, such as smartphones or tablets. The synchronization of data flow is depicted in Fig. 4.

Users can access the system interface and interact with the relevant functional buttons. For instance, if they want to download a checklist form, the system will generate the corresponding functionality and provide a URL for requesting the necessary resources. When the API receives the URL, it establishes a connection with the application system database to retrieve the checklist items. Subsequently, the checklist items are returned to the mobile database for storage. The mobile application subsystem interface extracts the data from the mobile database and consolidates it into the required format for display.

After on-site inspectors or engineers complete the equipment checklists, supervisors and relevant personnel can access the API through an internet connection to query or approve the checklist results. The API can be designed with specific URL names and required request parameters for each functionality. By integrating the data into the application system database through the DE-Agent data integration, the time required for searching traditional paper documents and obtaining cross-departmental data is significantly reduced.

4. System Architecture

In this study, the DE-Agent and API technologies are implemented to enhance the original process and integrate multiple external data sources, resulting in the establishment of an equipment maintenance mobility platform system (EMMPS).

Table 1
Definitions of data fields for the synchronized data table.

Field name	Field format	Field description
TABLENAME	varchar	Corresponding synchronized table name
APISEQ	bigint	Corresponding synchronized data content number
TIMESTAMP	datetime	Data import time
FLAG	varchar	Data exchange flag

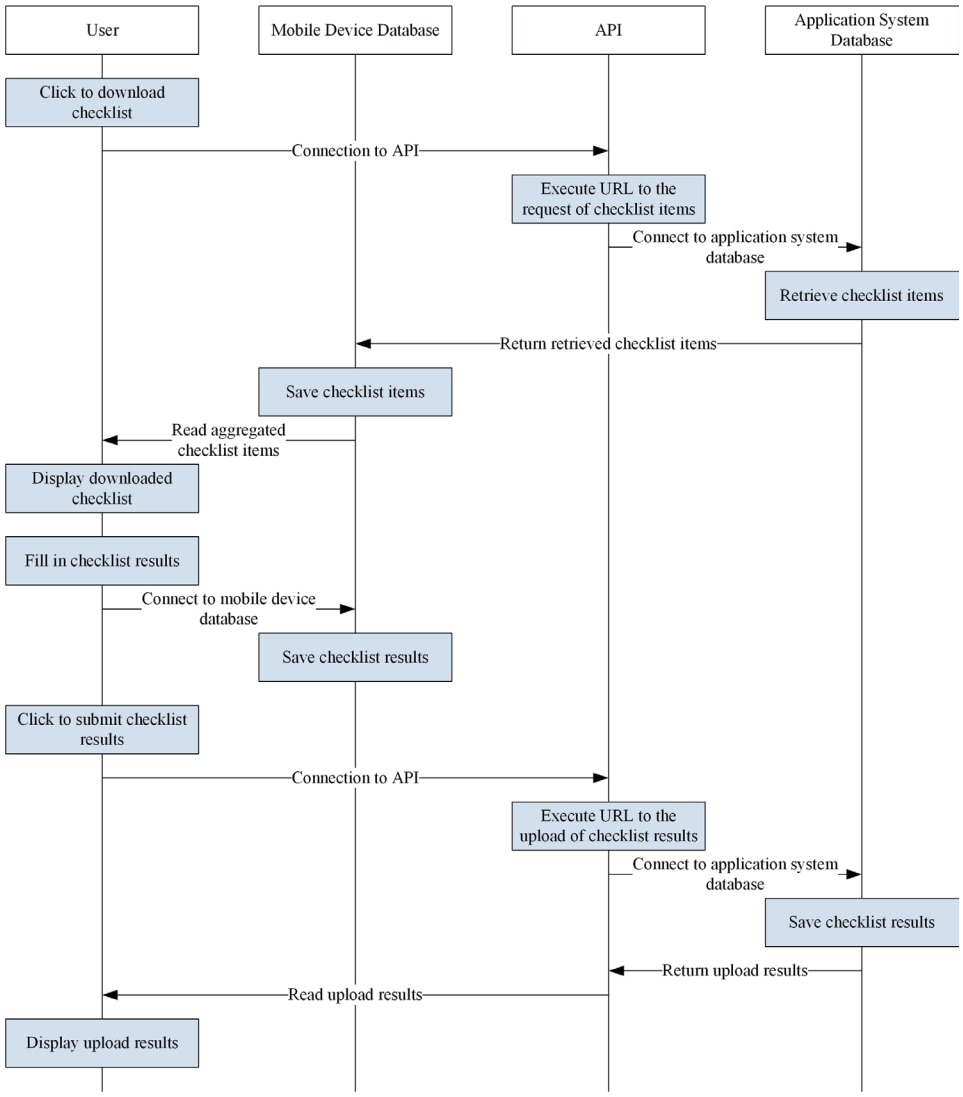

Fig. 4. (Color online) Data flow for downloading equipment inspection items and uploading completed results.

4.1 Architecture of data communication

The designed EMMPS enhances the traditional paper-based process of equipment checklist filling and record retrieval. It primarily relies on the application system database and the data exchange mechanism framework to establish an electronic system. The API is employed to enable multiplatform usability. The data communication architecture of EMMPS is depicted in Fig. 5.

One of the external data sources, the work order system, establishes a direct connection with the application system database of EMMPS and DE-Agent using transmission control protocol/Internet protocol (TCP/IP). Another external data source is the personnel system, which

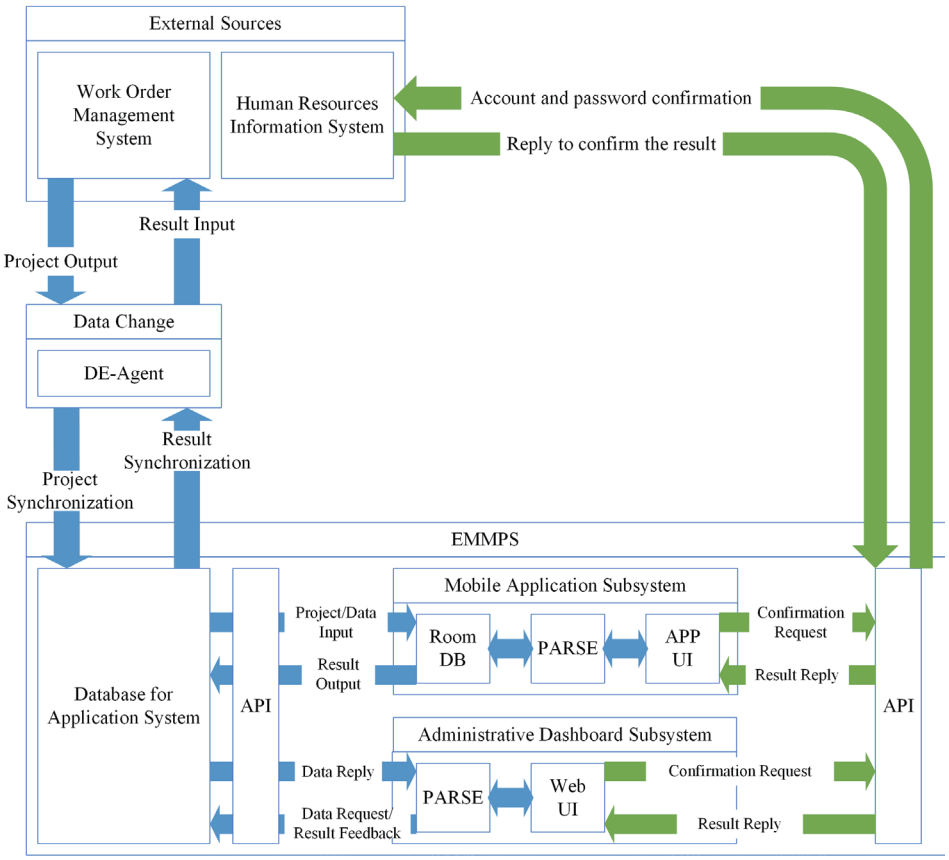

Fig. 5. (Color online) Architecture of EMMPS data communication.

primarily manages the verification and identification of user account credentials. Since users can log in through both an administrative dashboard subsystem and the mobile application subsystem, the API is utilized for requesting and responding to ensure the protection of sensitive device data. The application system database on EMMPS also employs the API for data requests and responses, as it needs to be accessed by users through the user interface (UI).

4.2 Architecture of system functionalities

The functional architecture of EMMPS, depicted in Fig. 6, encompasses various external data sources, such as the work order system and the active directory (AD) server. The work order system is responsible for scheduling regular inspection items and facilitates data synchronization with the EMMPS application system database through the DE-Agent. The synchronized content can consist of checklist items or inspection deadlines. On the other hand, the AD server primarily manages user accounts and password comparisons.

Within the EMMPS application system, there exist two subsystems: the administrative dashboard subsystem and the mobile application subsystem. The mobile application subsystem is utilized by equipment owners or engineers on their mobile devices. They use this subsystem to download checklist forms for equipment inspection and input the inspection results. Once the

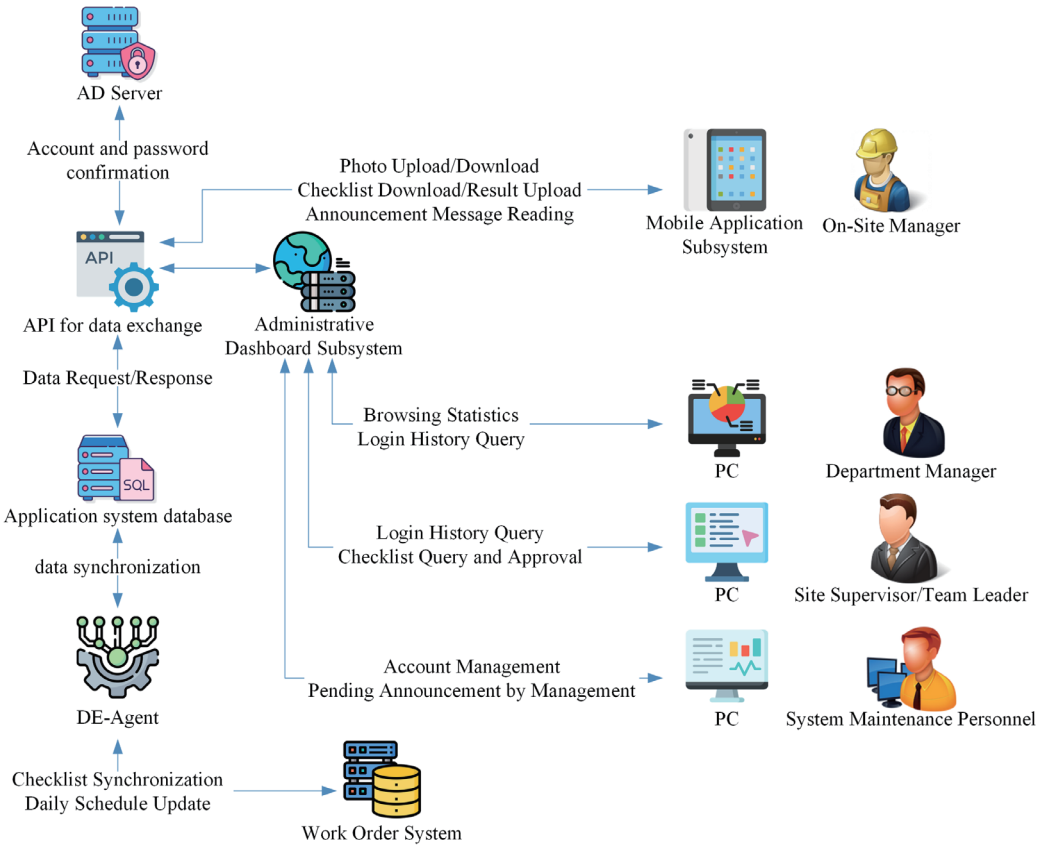

Fig. 6. (Color online) EMMPS usage scenarios.

form is completed, the mobile application subsystem sends the equipment inspection results back to the EMMPS application system's database for storage via the network. These results await supervisor approval.

The administrative dashboard subsystem provides an online approval function. Team leaders or relevant supervisors can promptly access the completed equipment inspection results and verify if any abnormalities exist. If no abnormalities are found, they can proceed with the approval process and finalize the regular equipment inspection. Additionally, when a supervisor needs to retrieve the history of a specific equipment inspection, they can utilize this subsystem for online querying. This eliminates the need for manual retrieval and enables direct access to the inspection records using a computer in an internet-connected environment.

4.3 Architecture of system network

The main equipment of EMMPS is deployed within the internal network, which consists of application system (System AP) servers, System API programs, and a system database (System DB). The network architecture of EMMPS is depicted in Fig. 7.

Consequently, authorized personnel within the internal network (Internet Zone) can log in and access EMMPS by providing their username and password. However, owing to strict

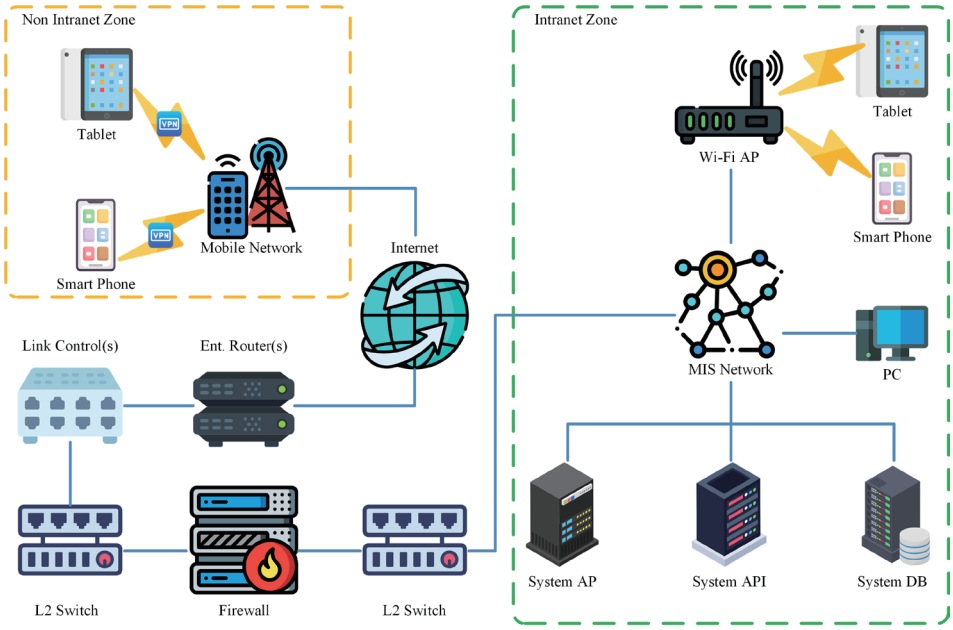

Fig. 7. (Color online) Cross-system network architecture.

security measures imposed by the company on external networks (Non Intranet Zone), personnel outside the internal network must establish a secure connection using technologies such as a virtual private network (VPN). This ensures the authentication of connecting personnel and the security of the connection before they can access the resources of the digital system.

5. Experiments and Results

5.1 System construction environment

DE-Agent and API technology are utilized to develop the EMMPS, which consists of two main subsystems: the administrative dashboard subsystem and the mobile application subsystem. The administrative dashboard subsystem enables remote operations by accessing the system through a browser using a specified input URL. On the other hand, the mobile application subsystem operates through a dedicated mobile app interface. When users require access to data, it is transmitted through the API. The construction environment parameters for the EMMPS system are detailed in Table 2.

5.2 System load verify

5.2.1 Simultaneous online testing

In terms of the number of concurrent users, 200 and 500 people simultaneously use EMMPS for testing. The usage frequency and the time to reach the maximum number of concurrent users are fixed at 10 times and 240 s, respectively.

Table 2
Construction environment parameters of EMMPS.

	Administrative dashboard subsystem	Mobile application subsystem	DE-Agent	API server
Host type	VM	Physical tablet	VM	VM
OS	Windows Server 2019 Standard Edition	Android 13.0	Windows Server 2019 Standard Edition	Windows Server 2019 Standard Edition
Memory	16 GB	16 GB	16 GB	16 GB
CPU	2 core	ARM	2 core	2 core
Storage	HHD 150 GB	HHD 150 GB	HHD 150 GB	HHD 150 GB
Application server	IIS 10.0.19014	None	None	IIS 10.0.19014
Database	SQL Server 2019 Standard	Room DB 2.4.2	DB2 11.1.4.5	None
Network type	Wired, fixed IP	Wireless, DHCP	Wired, fixed IP	Wired, fixed IP

(1) Figure 8 shows the response time when 200 people go online simultaneously. The average response time is 1374.68 ms with a minimum value of 238 ms and a maximum value of 13052 ms. The high value observed at 18 00:00 may be attributed to network delays.
(2) Figure 9 displays the response time when 500 people go online simultaneously. The average response time is 1400.22 ms with a minimum of 127 ms and a maximum of 5256 ms. During the time period from 18 07:49 to 18 07:51, there are three data points that appear to be unusually low or high, which could be attributed to network delays.

When 200 people and 500 people go online simultaneously, it is possible for the minimum response time of the 500 people case to be smaller than that of the 200 people case. This could be due to the relationship between execution time and the number of users. Since the total execution time for 200 people is higher than that for 500 people, there is a possibility that the 500 people group may experience shorter response times. It is important to note that in the test, the time parameters are fixed at 240 s to reach the maximum number of concurrent users, which may impact the response times observed.

5.2.2 Read count test

In this study, the same number of online users is used to compare different times of reading data. Specifically, the data are read 10 times and 50 times simultaneously, respectively.

(1) Figure 10 illustrates the response time for each person to read the data 10 times. The average response time is 1438.68 ms with a minimum of 247 ms and a maximum of 3648 ms. However, there is a data point at 18 16:38 that exceeds the average value. This could be attributed to the fact that the test network is a public network, and its true unpredictability is relatively high, leading to occasional variations in response times.
(2) Figure 11 depicts the response time for each person to read the data 50 times. The average response time is 13764.30 ms with a minimum of 191 ms and a maximum of 55545 ms. It is observed that the response time exhibited a tendency to increase before 18 22:31 but then experienced a sharp drop afterward. This change in response time can be attributed to the difference in bandwidth, indicating that the network conditions improved, resulting in faster response times.

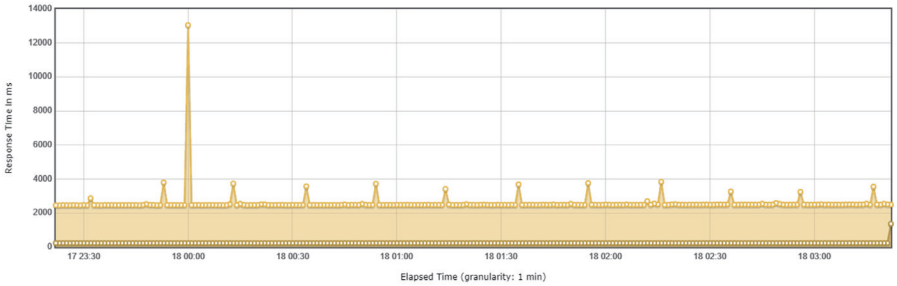

Fig. 8. (Color online) Response time for 200 people online at the same time.

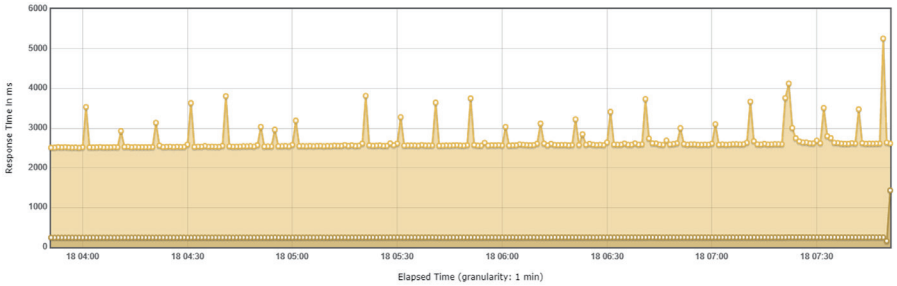

Fig. 9. (Color online) Response time for 500 people online at the same time.

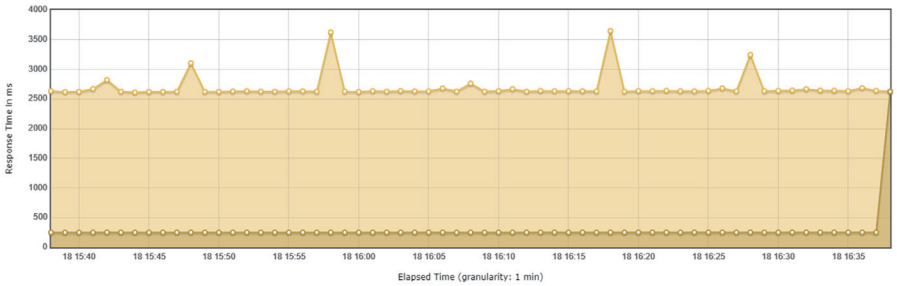

Fig. 10. (Color online) Response time for each person to read data 10 times.

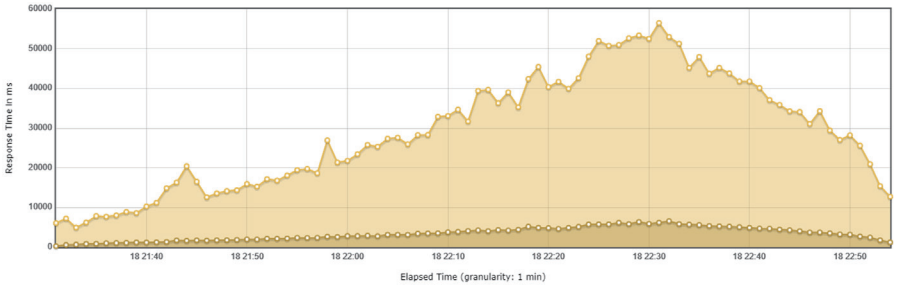

Fig. 11. (Color online) Response time for each person to read data 50 times.

When comparing different numbers of readings per person, it is observed that the average response time is longer for reading 50 times than for reading 10 times. However, it is interesting to note that the minimum response time remains the same as that observed during the simultaneous online test. The occurrence of relatively small minimum values at the same time indicates that there might be certain factors or optimizations in the system that lead to consistent and improved performance, even when handling higher reading loads per person.

5.2.3 Time test

When the number of people and the number of readings are the same, with the number of people being fixed at 100 people and the number of readings at 10 times, the time to reach the maximum number of people within a certain period of time varies. Specifically, the time periods considered are 60, 120, 240, and 480 s.

(1) Figure 12 presents the response time to reach the maximum number of people within a 60 s timeframe. The average response time is 400.56 ms with a minimum of 121 ms and a maximum of 1375 ms.
(2) Figure 13 displays the response time to reach the maximum number of people within a 120 s timeframe. The average response time is 441.02 ms with a minimum of 127 ms and a maximum of 1060 ms.
(3) Figure 14 shows the response time to reach the maximum number of people within a 240 s timeframe. The average response time is 458.52 ms with a minimum of 124 ms and a maximum of 856 ms.

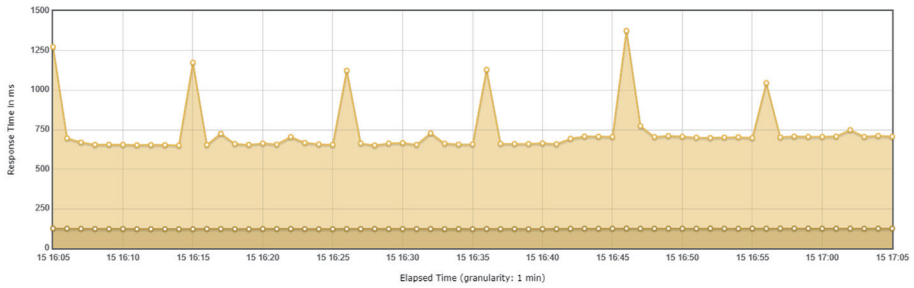

Fig. 12. (Color online) Response time to reach the maximum number of people in 60 s.

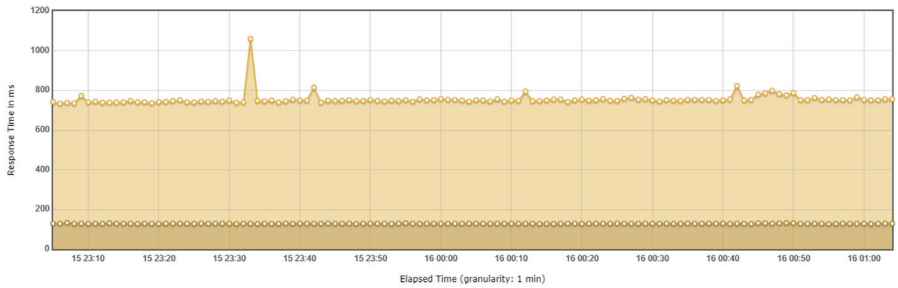

Fig. 13. (Color online) Response time to reach the maximum number of people in 120 s.

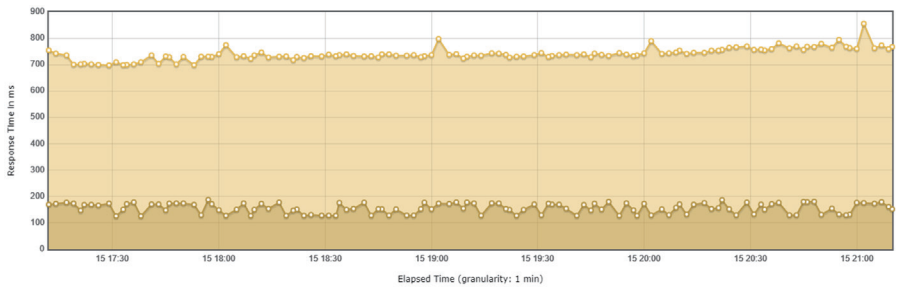

Fig. 14. (Color online) Response time to reach the maximum number of people in 240 s.

(4) Figure 15 presents the response time to reach the maximum number of people within a 480 s timeframe. The average response time is 463.06 ms with a minimum of 113 ms and a maximum of 812 ms.

Among the response times to reach the maximum number of people at different times, it is observed that there is not a significant difference between the minimum values. However, the average response time tends to increase with time, while the maximum value becomes smaller with time. This can be attributed to the fact that although the number of people is set to be the same, when a longer time duration is set, it allows for a more effective utilization of the entire network bandwidth. As a result, the average response time becomes shorter as the system processes more requests over time. However, the maximum value decreases as the system becomes more efficient in handling the workload within the given time frame.

5.3 Related system comparison

Since integrating heterogeneous data often involves converting different formats, it is typically integrated into a single standard, making it challenging to integrate data from multiple sources with different formats. Therefore, we utilize the DE-Agent to classify and cleanse data in various formats to meet the format requirements of the application system. To assess the EMMPS system, in this study, we chose two comparable applications for evaluation. The first comparison involves the interoperable electronic health record (EHR) implementation using the GIE System (referred to as HER-GIE). The second comparison involves the mobile and web-based system for maintenance, repair, and asset management (referred to as MRAM). The findings of these comparisons are summarized in Table 3.

From the comparison of the similar system features of EMMPS, HER-GIE, and MRAM, we can summarize the following points.

(1) Data management: All three systems involve data management, but with different focuses. EMMPS is centered around equipment checklist filling and record retrieval. HER-GIE deals with electronic health records and interoperability within healthcare systems. The MRAM system focuses on maintenance, repair, and asset management data.

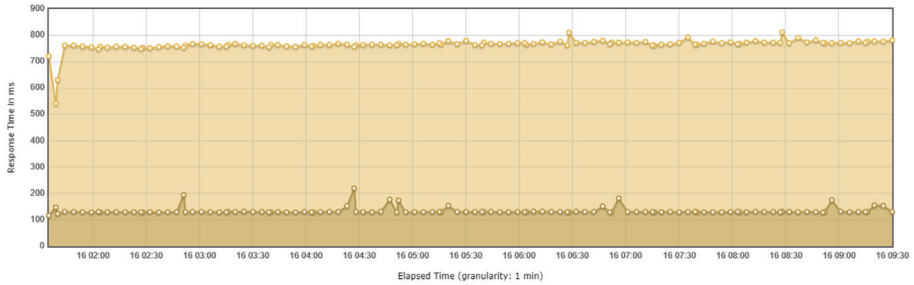

Fig. 15. (Color online) Response time to reach the maximum number of people in 480 s.

Table 3
Features of EMMPS, HER-GIE, and MRAM.

	EMMPS	HER-GIE	MRAM
Compatibility	High	Low	Middle
Convenience	High	Low	High
Data analysis	Simple	Difficulty	Simple
Data security	High	Low	Low

(2) Mobile and web access: EMMPS and MRAM systems are designed to be accessible through mobile and web platforms, enabling users to conveniently interact with the systems using various devices. HER-GIE may have mobile and web-access capabilities depending on its specific implementation.
(3) Integration and compatibility: EMMPS is developed on a platform, enhancing its compatibility with other systems and devices. HER-GIE may have interoperability features to facilitate integration with other healthcare systems. The MRAM system may offer integration capabilities for asset management and maintenance processes.
(4) Data security: EMMPS prioritizes data security by utilizing API-based data reading and storage methods. It is expected that HER-GIE and MRAM systems also have measures in place to ensure data security, considering the sensitivity of healthcare and asset management data.
(5) Specific functionality: Each system has unique functionalities tailored to its specific domain. EMMPS focuses on equipment checklist filling and record retrieval, HER-GIE on electronic health records and interoperability, and the MRAM system on maintenance, repair, and asset management tasks.

It is important to note that the actual features and capabilities of EMMPS, HER-GIE, and MRAM systems may vary in accordance with their specific implementations and requirements. Further detailed information is necessary for a comprehensive comparison.

6. Conclusions

We proposed a framework for integrating heterogeneous data sources through a data exchange mechanism. The system implementation involves transmitting different types of heterogeneous data sources to designated application systems using the DE-Agent proposed in

this research. The effectiveness of the designed data exchange mechanism for integrating these data sources was verified through the implementation of the EMMPS system, specifically focusing on the equipment inspection filling and record query process. The integration was facilitated using both the DE-Agent and API technologies.

To ensure the security of data transmission and protect the system from external attacks, both the administrative dashboard subsystem and the mobile application subsystem employ HTTPS and API for secure data transmission. The administrative dashboard subsystem adopts web technology to develop its UI for convenient usage on the management side. Meanwhile, the mobile application subsystem utilizes app technology to develop its user interface, making it more convenient for mobile users.

The EMMPS system demonstrates normal operation under various conditions such as different online user numbers, different reading times, and different maximum number of users reached. Although there may be some abnormal data owing to network bandwidth constraints, it does not affect the system's normal operation. Overall, the proposed framework, DE-Agent, and API technology effectively enable the integration of heterogeneous data sources in the EMMPS system. The system demonstrates secure data transmission, reliable functionality, and successful data exchange between the application subsystems.

References

1 Releasing the Source Code for the NET Framework Libraries. https://web.archive.org/web/20100907233621/http://weblogs.asp.net/scottgu/archive/2007/10/03/releasing-the-source-code-for-the-net-framework-libraries.aspx (accessed October 2021).
2 F. A. Radwan and T. W. Martin: Int. Conf. Industrial Technology (IEEE, 2003) 387.
3 W. Zhang and C. Chen: Int. Conf. Safety Produce Informatization (IEEE, 2019) 500.
4 S. Kaplan, R. Callaway, L. Ngan, and K. Passow: Conf. Photovoltaic Specialists (IEEE, 2018) 1204.
5 R. T. Fielding: Architectural Styles and the Design of Network-based Software Architectures (University of California, Irvine, 2000) Chap. 5.
6 S. P. Ong, S. Cholia, A. Jain, M. Brafman, D. Gunter, G. Ceder, and K. A. Persson: Comput. Mater. Sci. **97** (2014) 209. https://doi.org/10.1016/j.commatsci.2014.10.037
7 J. Daemen and V. Rijmen: AES Proposal: Rijndael (1999).
8 G. Raj, R. C. Kesireddi, and S. Gupta: Int. Conf. Next Generation Computing Technologies (IEEE, 2015) 374.
9 M. Haekal and Eliyani: Int. Conf. Informatics and Computing (IEEE, 2016) 175.
10 A. Sheth and J. Larson: ACM Comput. Surv. **22** (1990) 183. https://doi.org/10.1145/96602.96604
11 P. Karp: J. Comput. Biol. **2** (1995) 573. https://doi.org/10.1089/cmb.1995.2.573
12 W. Sujansky: J. Biomed. Inf. **34** (2001) 285. https://doi.org/10.1006/jbin.2001.1024
13 C. Kumar, C. Rao, and A. Govardhan: Am. J. Database Theory Appli. 1 (2012) 1. https://doi.org/10.5923/j.database.20120101.01
14 K. Koçer and S. Biroğul: Am. J. Software Eng. 5 (2017) 1. https://doi.org/10.12691/ajse-5-1-1