Article

PBEMS: A Permissioned Blockchain-based Equipment Maintenance System in Smart Manufacturing

Mohammad Iqbal Saryuddin Assaqty * , Ying Gao * , Ali Alfatemi , Ashraf Osman Ibrahim , Anas W. Abulfaraj , Faisal Binzagr , Fezan Nabawi , Mohammad Reza Fahlevi

Posted Date: 28 October 2023

doi: 10.20944/preprints202310.0437.v2

Keywords: blockchain application; permissioned blockchain; equipment maintenance; smart manufacturing

Article

PBEMS: A Permissioned Blockchain-Based Equipment Maintenance System in Smart Manufacturing

Mohammad Iqbal Saryuddin Assaqty [1,6,*], Ying Gao [1], Ali Alfatemi [2], Ashraf Osman Ibrahim [3], Anas W. Abulfaraj [4], Faisal Binzagr [5], Fezan Nabawi [6] and Mohammad Reza Fahlevi [6]

1 School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China; iqbal@unusia.ac.id (M.I.S.A.); yinggao@scut.edu.cn (Y.G.)
2 Computer and Information Science Department, Fordham University, Bronx, NY 10458, USA; aalfatemi@fordham.edu (A.A.)
3 Creative Advanced Machine Intelligence Research Centre, Faculty of Computing and Informatics, Universiti Malaysia Sabah, 88400 Kota Kinabalu, Sabah, Malaysia; ashrafosman@ums.edu.my (A.O.I.)
4 Department of Information Systems, King Abdulaziz University, P.O. Box 344, Rabigh; 21911, Saudi Arabia; awabulfaraj@kau.edu.sa (A.W.A.)
5 Department of Computer Science, King Abdulaziz University, P.O. Box 344, Rabigh 21911, Saudi Arabia; fbinzagr@kau.edu.sa (F.B.)
6 Faculty of Engineering and Computer Science, Universitas Nahdlatul Ulama Indonesia, Jakarta 10320, Indonesia; iqbal@unusia.ac.id (M.I.S.A.); fezan@unusia.ac.id (F.N.); rezafah@unusia.ac.id (M.R.F.)
* Correspondence: iqbal@unusia.ac.id (M.I.S.A.)

Abstract: In the realm of smart manufacturing, equipment plays a pivotal role, impacting the efficiency and quality of production. Ensuring equipment availability and optimal performance is crucial, necessitating effective maintenance strategies. This paper addresses the complexities of smart device maintenance, which often involves collaboration between multiple entities like device owners, manufacturers, suppliers, and external maintenance services. To enhance this cooperative maintenance, we propose a novel maintenance system based on permissioned blockchain technology, offering superior security and reliability compared to conventional cloud-based platforms. Our contributions include developing an integrated maintenance system framework for smart manufacturing, which facilitates interactions among diverse stakeholders. Additionally, we have developed a prototype of this maintenance system, serving as a practical model for real-world application. Our research primarily demonstrates the practicality and advantages of using permissioned blockchain for maintenance systems in the manufacturing sector, paving the way for more secure, efficient, and collaborative maintenance practices.

Keywords: blockchain application; permissioned blockchain; equipment maintenance; smart manufacturing

1. Introduction

The implementation of smart manufacturing systems has revolutionized industrial production processes, paving the way for Industry 4.0, the current trend of automation, and data exchange in manufacturing technologies [1]. Key components of smart manufacturing systems include cyber-physical systems, the Internet of Things (IoT), and cloud computing. Amidst these innovative advancements, the role of effective maintenance systems becomes increasingly significant for the efficient functioning and longevity of smart manufacturing devices.

Smart manufacturing relies heavily on interconnected machinery and devices that operate seamlessly within an ecosystem powered by digital data. Consequently, maintenance systems, especially those equipped with advanced predictive and preventive capabilities, have become indispensable to manage the complexity of these devices.

The maintenance system's core function is to monitor the health of devices, identify potential failures, and initiate remedial actions before those failures cause significant operational disruption. Maintenance systems for smart manufacturing devices are critical in ensuring the continued efficient functioning of manufacturing processes and minimizing downtime [2].

Maintenance system can benefit smart manufacturing with improved efficiency, cost saving, enhanced quality control, and safety. Advanced maintenance systems can continuously monitor machine health, allowing for real-time diagnosis and troubleshooting. This early fault detection leads to a reduction in machine downtime and boosts overall operational efficiency [3]. The predictive capability of modern maintenance systems helps avoid catastrophic machine failures, significantly reducing repair and replacement costs. Furthermore, they help streamline maintenance tasks, resulting in further financial savings [4]. By maintaining machine performance at an optimal level, advanced maintenance systems indirectly ensure the consistency of product quality, which is crucial in a competitive market [5]. Regular and predictive maintenance can lead to safer operating conditions by identifying potential issues before they pose a risk to the operators or the overall production process [6].

The widely implemented maintenance systems are cloud based which have emerged as a promising approach in the field of smart manufacturing, offering a myriad of benefits including cost savings, scalability, and increased efficiency [7]. However, their deployment is not without associated risks in security, data integrity, privacy, and system compatibility issues.

A significant concern in cloud-based maintenance systems is security. The integration of these systems into the manufacturing ecosystem exposes operational data and intellectual property to potential threats. Cyberattacks could lead to data breaches, sabotage, and interruption of service, potentially causing considerable economic and reputational damage. Companies using cloud-based systems must have robust cybersecurity measures in place to mitigate these risks [8,9].

Data integrity and privacy are other pressing concerns in cloud-based systems. The remote storage and management of data introduce risks associated with data corruption, loss, and unauthorized access. Data privacy is particularly critical given the sensitivity of the data involved, as it often includes proprietary machine specifications and production details [10]. A thorough understanding and application of encryption and privacy-enhancing technologies are paramount to ensure data security and maintain trust with stakeholders [11].

The heterogeneity of machines, software, and protocols in manufacturing makes system compatibility a complex issue [12]. For a seamless integration of a cloud-based maintenance system, manufacturers and their suppliers need to consider the compatibility of different systems, both in terms of hardware and software [13,14]. This is particularly challenging in legacy manufacturing systems, where upgrading machinery and software to cloud-compatible versions can be cost-prohibitive and operationally disruptive.

Dependence on service providers is another significant risk. The availability and performance of a cloud-based maintenance system are tied to the service provider's reliability. If the provider experiences outages, the entire manufacturing operation can be affected. This underlines the need for a well-drafted Service Level Agreement (SLA) that explicitly outlines responsibilities and contingencies [15].

To address those risks, we propose a permissioned blockchain-based maintenance system. Blockchain, a decentralized, distributed ledger technology, presents substantial benefits for smart manufacturing [16]. It ensures data integrity, transparency, and security by logging every transaction in a tamper-proof record across multiple nodes. However, traditional, public blockchains present limitations like scalability issues and slower transaction speeds. This is where permissioned blockchains come into play. Permissioned blockchain, unlike its public counterpart, restricts access and operational rights to approved participants only [17]. This feature brings in enhanced scalability, faster transaction speeds, and more control over the data while maintaining the core advantages of blockchain technology.

Permissioned blockchain can be utilized for maintenance system in smart manufacturing with the following benefits:

- **Security:** Cloud-based maintenance systems are vulnerable to cyber threats that could disrupt the manufacturing process or compromise data privacy. Permissioned blockchain could enhance security by providing an immutable and tamper-resistant platform, ensuring that only authorized users can access the data [18].
- **Data Integrity:** Permissioned blockchain ensures that every transaction or change in the system is recorded in a distributed ledger, which is virtually impossible to modify or delete. This provides an automatic safeguard for data integrity [19].
- **Transparency and Traceability:** Blockchain technology creates a transparent system where every transaction can be tracked and audited. This promotes trust among participants and improves the traceability of actions, which is crucial for effective maintenance and process [16].
- **Streamlined Processes:** Permissioned blockchains can automate manual processes, which not only saves time but also reduces the likelihood of human error. Through the use of smart contracts, routine maintenance tasks can be automatically triggered when certain conditions are met. This streamlining effect can improve the efficiency and effectiveness of maintenance systems in smart manufacturing [20].
- **Interoperability:** Permissioned blockchains can facilitate interoperability between different systems and stakeholders. In a smart manufacturing environment, where numerous entities are interacting, it is important to have a system in place that can communicate with various parties [21].

Regarding our proposal of a blockchain-based maintenance system for smart manufacturing in this paper, we make the following key contributions:

- We design an integrated framework for maintenance systems in smart manufacturing that defines the interactions of various stakeholders within it.
- We create a maintenance system prototype that can be used as a reference for real-world implementation.
- We demonstrate the feasibility of permissioned blockchain technology for the maintenance system in the manufacturing industry.

The remaining paper is organized as follows: Section 2 offers a comprehensive review of current maintenance systems, highlighting blockchain technologies in use. Section 3 presents material and method, while Section 4 discusses the results derived from the prototype's development, performance, and security analysis. Lastly, Section 5 concludes with a discussion on future research prospects.

2. Related Works

Currently, published research on the implementation of blockchain-based maintenance systems in the smart manufacturing industry is still limited. Similar studies have been published in other industries such as in the aviation and military industries.

Regarding smart manufacturing industry, Stodt et al. created an excellent formal description for the maintenance process in industry 4.0 using blockchain which includes various stakeholders both internal and external. The authors also provided integration of smart contracts and the respective use cases, but did not provide experiment and analysis on their proposals [22]. Welte et al. provided a more detailed use case on shop floor maintenance by creating an integrated blockchain-based architecture on the top of Hyperledger platform and providing three solutions to the issues of data corruption, data protection and input falsification. However, the authors did not provide more detailed experiments to support their proposal [23].

Bai et al. introduced BPIIoT, a ligh-weight platform for the Industrial Internet of Things (IIoT). This blockchain-based platform provided an application case for predictive maintenance by involving maintenance service providers. The authors also provided a smart contract algorithm on top of Etherium for this maintenance service [[24].

Chen et al. presented an intelligent maintenance system for complex equipment using blockchain technology by creating a five-dimensional model of digital twins and defining a maintenance process based on the digital twins, but without defining the various parties involved in it [25]. Tran et al. introduced Machine-as-a-Service (MaaS), a blockchain-based system to manage and maintain industrial appliances by integrating Ethereum and InterPlanetary File System (IPFS). The Etherium selection aims to open up a choice of public modes of operation in the future. The author also explains the implementation details by defining the various components in the system [26].

An interesting proposal for a decentralized marketplace was made by Miehle et al. for self-maintaining machines. By utilizing blockchain technology which has smart contracts in it, this marketplace can improve the procurement process. The author has also prototyped it by implementing various payment options [27]. However, by using a public blockchain, this proposal did not explain in detail how membership is managed in it.

Also related to the maintenance process, a blockchain-based solution was proposed by Hasan et al. for the traceability of spare parts in manufacturing that is integrated with the IPFS. The authors also described the implementation details and cost analysis [28]. However, this solution was not intended to be a solution for a whole maintenance system.

Apart from the smart manufacturing industry, Aleshi et al. proposed a blockchain-based model to increase the security of maintenance records in the aviation industry. The authors defined the various actors involved on the top of Hyperledger Fabric platform. To support their proposal, the authors included simulation results for both the frontend and backend scenarios [29]. There is also Airchain, a proposal of a blockchain-based maintenance record system made by Jensen et al. that records data on the Cabin Log Book (CLB) and Technical Log Book (TLB) on the top of Hyperledger Besu platform [30,31]. The authors also presented the frontend prototype to support the proposal. However, the authors did not include external suppliers as active parties involved in this system. A blockchain-based aviation maintenance recording solution was also proposed by Andrei et al. Furthermore, the author defined the various parties involved and the process of interaction within them and presented the complete experimental results [32].

Ahmad et al. explained in part of their paper regarding Maintenance, Repair and Overhaul (MRO) operations in the aviation industry. The authors defined the various data that must be managed, and also created a flow of smart contracts involving various parties ranging from aircraft technicians, airlines, component manufacturers, to MRO service providers [33]. A similar proposal was also presented by Schyga et al. by defining more parties including regulator, and in more detail explaining its implementation using the Hyperledger Fabric platform [34].

A blockchain-based maintenance framework proposed by Mohril et al. in the military industry with various scenarios ranging from field, workshop, to headquarter. This framework runs on a permissioned blockchain with Proof of Elapsed Time (PoET) consensus [35]. However, this framework does not involve external suppliers to actively participate in performing transactions on the system, it only provides read access for maintenance analysis needs instead.

3. Materials and Methods

3.1. Groundwork

We outline the foundational work and preliminary findings for the current research endeavor in this sub section.

3.1.1. Equipment Maintenance

The maintenance performed on an equipment will depend on various factors such as the type of equipment, its usage, and maintenance history. It's important to develop a comprehensive maintenance plan to ensure that machines are regularly maintained and that maintenance is performed in an effective and efficient manner.

There are several types of maintenance performed on equipment in smart manufacturing [36]:

- **Preventive Maintenance:** Regularly scheduled maintenance performed to prevent machine breakdowns and minimize downtime.
- **Predictive Maintenance:** Maintenance performed based on the analysis of machine data and performance indicators, to identify potential issues before they cause breakdowns.
- **Corrective Maintenance:** Maintenance performed to fix a machine that has already broken down.
- **Emergency Maintenance:** Maintenance performed in response to an unexpected machine breakdown.
- **Condition-Based Maintenance:** Maintenance performed based on the actual condition of the machine, rather than a predetermined schedule.
- **Scheduled Maintenance:** Regularly scheduled maintenance, often performed during scheduled downtime.
- **Unscheduled Maintenance:** Maintenance performed outside of regular scheduled downtime, often in response to a machine breakdown.
- **Predetermined Maintenance:** Maintenance performed at predetermined intervals, regardless of the machine's actual condition.

There are many types of equipment used in the manufacturing industry depending on the product being produced. Here is a list of examples of equipment commonly used in smart manufacturing:

- **CNC Machines:** Computer Numerical Control (CNC) machines are a staple in smart manufacturing. They use computer-aided design (CAD) and computer-aided manufacturing (CAM) to control and automate the machining process.
- **Industrial Robots:** Industrial robots, including collaborative robots (cobots), are used to automate tasks such as assembly, welding, material handling, and packaging.
- **3D Printers:** Additive manufacturing or 3D printing is used for creating prototypes, tools, and even end-use products. In smart manufacturing, 3D printers can be connected to networks and are often automated.
- **Sensors and IoT Devices:** These devices are used to collect data from the manufacturing process. This includes data about machine performance, environmental conditions, and product quality.
- **Smart Conveyors and AGVs:** Automated guided vehicles (AGVs) and smart conveyors are used for the transportation of materials and products within the facility. They can adapt to the manufacturing process dynamically.
- **Vision Systems:** Cameras and vision systems are used for quality control, inspection, and guiding robots. They can detect defects and ensure the precision of the manufacturing process.
- **PLCs and PACs:** Programmable Logic Controllers (PLCs) and Programmable Automation Controllers (PACs) are used to control machinery and processes in response to the data received from sensors and other input devices.
- **Add-on Technologies:** This includes equipment that can be added to existing machinery to make them smarter, like retrofit kits with sensors or small processors.
- **Process Control Equipment:** This includes various controllers and instruments that monitor and control process variables such as temperature, pressure, flow, etc., to ensure that the process operates within desired parameters.
- **Augmented Reality (AR) Equipment:** AR glasses and devices can help workers by providing real-time information, instructions, and insights during assembly, maintenance, and training processes.

3.1.2. Permissioned Blockchain

A digital ledger that is decentralized, distributed, and records transactions across numerous computers is referred to as a blockchain. It allows for secure, transparent, and tamper-proof storage and transfer of information and data, making it popular for applications such as cryptocurrencies and supply chain management. It uses cryptographic algorithms to secure transactions, create a decentralized network, and without the need for intermediaries.

The basic components of a blockchain system include blocks and nodes. Blocks are fundamental components of a blockchain that contain details pertaining to transactions, which may include the time, date, and transaction amount. To initiate a transaction, it is first transmitted to the network. The nodes on the network authenticate the transaction to confirm its authenticity. Following validation, the transaction is compiled with other transactions to construct a block. The process of consensus is then executed to include the block in the blockchain, in which nodes on the network agree on the blockchain's state. After the block has been appended to the chain, the transaction is finalized [37].

In blockchain networks, a consensus algorithm is employed to establish agreement among nodes regarding the current state of the blockchain. The primary objective of this algorithm is to guarantee that all nodes within the network possess identical information, and that every new block appended to the blockchain is legitimate. Multiple consensus algorithms, such as Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), and others, are utilized by different blockchain networks. Each algorithm has its own unique regulations, benefits, and drawbacks, and the decision of which algorithm to use is often based on the specific requirements and objectives of the blockchain network [38].

One specific consensus algorithm that has been proposed for the maintenance of machines in the manufacturing industry due to its potential to enhance transparency, security, and efficiency in maintenance processes, is Practical Byzantine Fault Tolerance (PBFT). PBFT is a consensus algorithm that ensures agreement among a group of nodes even in the presence of Byzantine faults, where nodes may be malicious or fail arbitrarily. In a PBFT-based system, a set of nodes, known as a replica group, work together to validate maintenance requests submitted by users. The maintenance request is broadcast to all replicas in the group, and a leader node is selected to coordinate the validation process. The leader node proposes a block of validated maintenance requests, which is then verified and agreed upon by the other replicas in the group. Once consensus is reached, the block is added to the blockchain.

The use of PBFT in blockchain-based maintenance systems for equipment in the smart manufacturing can provide several benefits. Firstly, PBFT can ensure that only validated maintenance requests are added to the blockchain, thereby enhancing the integrity and reliability of maintenance records. Secondly, the use of PBFT can enable faster consensus compared to other consensus algorithms, which can improve the efficiency of maintenance processes. Lastly, the decentralized and transparent nature of the blockchain can increase accountability and reduce the risk of fraudulent activities.

Contracts that are self-executing and have their terms of agreement explicitly encoded into lines of code are known as smart contracts. They are implemented on blockchain technology and are automatically executed when certain predetermined conditions are met. Smart contracts have the potential to greatly increase efficiency and reduce the costs of traditional contractual processes [39]. Smart contracts are typically written in programming languages specifically designed for blockchain such as Chaincode (for Hyperledger Fabric), Solidity (for Ethereum), Simplicity (for Blockstack), or RIDE (for Waves).

For membership, the emergence of Bitcoin as a pioneer of the blockchain network was made open for anyone to join, which was then called a public or permissionless blockchain. To meet the needs of various industries that want to limit membership in the network, many have implemented permissioned blockchains. Unlike public blockchains that allow anyone to participate in verifying transactions, permissioned blockchains only permit authorized nodes to take part in the consensus process. The entities controlling these nodes are often pre-selected by the organization that oversees the blockchain network.

One of the primary benefits of permissioned blockchains is improved privacy. As only selected nodes can verify transactions, confidential data can be shared securely within a defined group. This feature is beneficial for industries such as finance, healthcare, or governmental organizations, where sensitive data handling is the norm. Another critical advantage of permissioned blockchains is scalability. Due to their restricted nature, these networks can process transactions more quickly than their public counterparts. As the number of nodes involved in the consensus process is fewer, transactions can be validated faster, enhancing the overall efficiency of the network.

The maintenance records of each equipment would be stored as blocks on the permissioned blockchain, allowing for a tamper-proof and transparent history of maintenance activities. Each equipment would be assigned a unique identifier to ensure that maintenance records are accurately linked to the correct equipment. Access to the system could be restricted to authorized personnel only, ensuring the security and confidentiality of the maintenance information. The blockchain in the supply chain can enable the tracing of goods and materials, thereby ensuring transparency and accountability for all parties engaged in the process [40]. The utilization of blockchain technology can facilitate the implementation of smart contracts, which can automate various processes, minimize the requirement for manual intervention, increase efficiency, and decrease the possibility of errors. Blockchain can be used to store real-time machine data and use predictive analytics algorithms to analyze this data, allowing for more accurate predictions about machine health and reducing the risk of machine breakdowns. Blockchain can be used to provide decentralized identity management, allowing for secure and reliable identity verification in Industry 4.0. Blockchain's decentralized architecture and cryptographic algorithms make it an effective tool for securing data and transactions in Industry 4.0, reducing the risk of cyber-attacks and data breaches [41].

3.1.3. Hyperledger Fabric

Hyperledger Fabric is a permissioned blockchain framework developed by the Linux Foundation to provide a platform for creating enterprise-grade blockchain-based applications [42]. Fabric is a modular, flexible, and scalable framework that allows developers to build custom blockchain solutions tailored to specific business needs. Unlike public blockchains like Bitcoin and Ethereum, Hyperledger Fabric is a private and permissioned network, meaning that access to the blockchain and its data is restricted to authorized participants only.

One of the key features of Hyperledger Fabric is its support for smart contracts, which are self-executing programs that can automate the execution of complex business logic. Smart contracts are written in general-purpose programming languages such as Go, Java, and Node.js, making it easy for developers to create and deploy their own custom smart contracts.

Hyperledger Fabric also incorporates a modular architecture that allows for the use of pluggable consensus algorithms, key management systems, and identity services. This allows developers to choose the components that best fit their specific use case, making the framework highly customizable. Fabric also offers support for private data collections, allowing businesses to store sensitive data off-chain and securely share it with authorized parties. This feature is particularly useful for applications in industries such as healthcare and finance, where privacy and data confidentiality are of utmost importance.

Another key aspect of Hyperledger Fabric is its focus on interoperability and collaboration. Fabric can integrate with other Hyperledger projects, as well as with external blockchains and legacy systems, making it a versatile and adaptable solution for a wide range of use cases, including equipment maintenance system in smart manufacturing [17,43,44].

3.1.4. Hyperledger Caliper

As organizations increasingly adopt blockchain platforms, it becomes crucial to evaluate and compare their performance to ensure optimal deployment. In this regard, Hyperledger Caliper emerges as a robust benchmarking framework designed specifically for evaluating the performance of blockchain platforms [45].

Hyperledger Caliper, developed under the Hyperledger Project, is an open-source blockchain benchmarking tool that enables performance evaluation of different blockchain frameworks. It provides a standardized and extensible platform for conducting performance tests and comparing the throughput, latency, and scalability of various blockchain solutions. Hyperledger Caliper aims to assist developers, researchers, and enterprises in making informed decisions when selecting and optimizing blockchain platforms.

Hyperledger Caliper follows a modular architecture that supports pluggable blockchain adapters, allowing users to benchmark a wide range of blockchain frameworks. It supports major blockchain platforms, including Hyperledger Fabric, Ethereum, Corda, and others, making it a versatile tool for performance analysis.

Hyperledger Caliper is a powerful benchmarking framework that plays a vital role in evaluating and comparing the performance of different blockchain platforms. Its modular architecture, scalability, and comprehensive metrics make it a versatile tool for developers, enterprises, and researchers. By leveraging Hyperledger Caliper, organizations can make informed decisions when selecting blockchain platforms, optimize their blockchain networks, and contribute to the ongoing advancements in blockchain technology [46,47].

To evaluate the performance of permissioned blockchain-based equipment maintenance system in smart manufacturing, Hyperledger Caliper can be employed. By defining benchmarking scenarios, including network size, workload, and transaction types, Caliper allows us to simulate real-world conditions and measure the system's performance under different scenarios. The collected data provides valuable insights into the system's scalability, throughput, and responsiveness, enabling manufacturers to optimize the equipment maintenance system.

3.2. System Architecture

As seen in Figure 1, the PBEMS architecture is built by involving various roles both internal and external to the organization. All parties involved can interact on a permissioned blockchain network that stores maintenance records and equipment logs in the shared ledger.

The following are the roles involved in PBEMS:

1. **Internal Stakeholders (Smart Manufacturing Company):**

    - **Operator.** The role of an operator is crucial for ensuring the efficient operation and maintenance of the equipment. Operators play a key role in managing and monitoring the equipment, facilitating the maintenance process, and ensuring the integrity and security of the blockchain network. Operators also act as intermediaries between the equipment and the blockchain network. They are responsible for recording maintenance-related data, such as maintenance schedules, tasks performed, spare parts used, and any other relevant information, onto the blockchain.
    - **Finance.** The finance division is important in the maintenance system since smart contracts can be used for automating payments and other transactions with external stakeholders. The finance division should manage these smart contracts to ensure that payment terms, conditions, and timelines are met as per agreements.
    - **Smart Manufacturing Equipment.** Equipment in smart manufacturing that does not have networking features must go through an operator to be able to convey its data to the blockchain network. However, networkable smart devices can automatically send the

required data to the blockchain by themselves. Operators only need to configure what data types and schedules when data should be sent.

2. **External Stakeholders (Vendors):**
   - **Maintenance Service Provider.** A service provider plays a key role in ensuring that manufacturing equipment is properly maintained and that all maintenance activities are recorded securely and transparently on the blockchain. This can lead to improved operational efficiency, reduced downtime, and enhanced compliance with regulations and standards.
   - **Equipment Manufacturer.** An equipment manufacturer typically refers to a company or entity that designs and produces the physical equipment being used in the manufacturing process. The equipment manufacturer can interact with the blockchain to provide, monitor, and enhance the maintenance and performance of the equipment they produce.

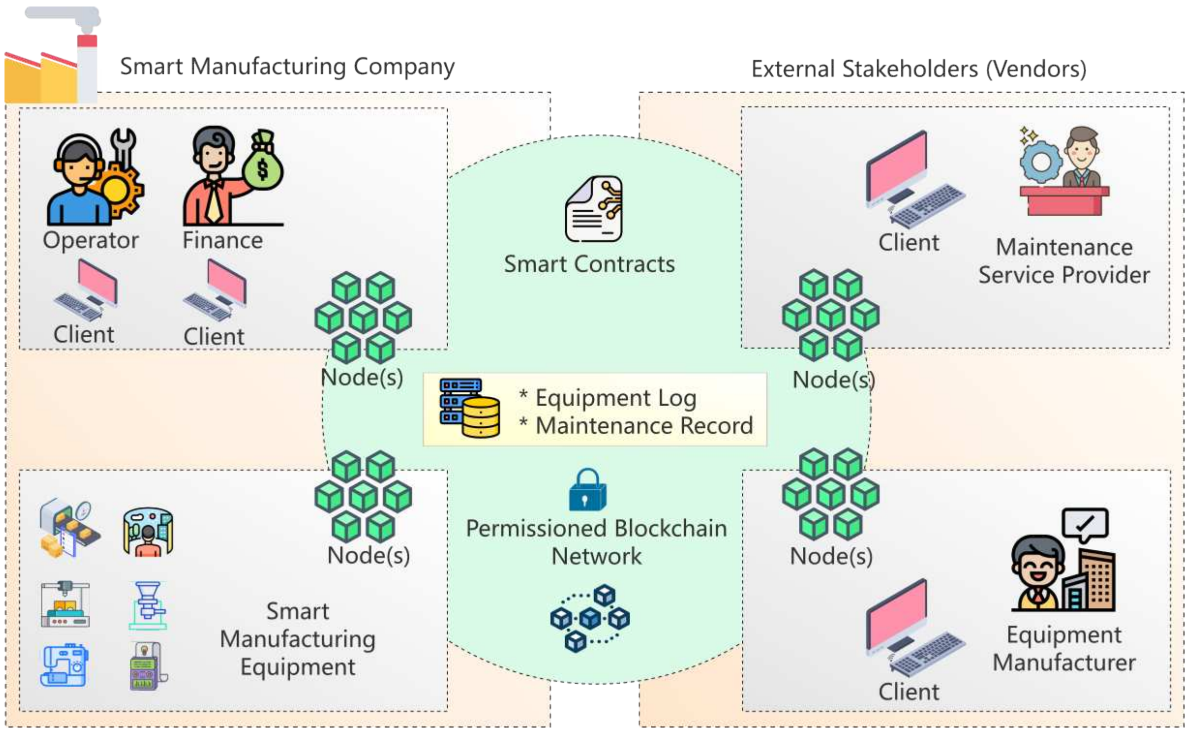
Smart Manufacturing Company
External Stakeholders (Vendors)
Operator
Finance
Client
Client
Node(s)
Smart Contracts
Client
Maintenance Service Provider
Node(s)
* Equipment Log
* Maintenance Record
Node(s)
Smart Manufacturing Equipment
Permissioned Blockchain Network
Node(s)
Equipment Manufacturer
Client

Figure 1. PBEMS Architecture

Each role must deploy one or more blockchain nodes to be connected to the network. After the node is deployed, they can access data in the blockchain according to their access rights through the blockchain client connected to the node. Table 1 describes an example of the stages of requesting maintenance services into a blockchain network following the PBFT consensus.

In Table 1, a client is sending a maintenance request to a primary node, which broadcasts the request to the other nodes in the network. Each node validates the request and responds with a pre-prepare message, indicating the acceptance of the request. The primary node collects the pre-prepare messages and sends out a prepare message, which includes a digest of the request and the responses from the pre-prepare messages. Each node validates the prepare message and responds with a commit message, indicating the agreement with the request and its order in the ledger.

Once the primary node collects enough commit messages, it sends out a reply to the client indicating the request has been approved by the network. The approved request is then added to the blockchain, and the maintenance is carried out by a designated maintainer. If a node fails to respond at any step of the process, the other nodes can detect the failure and initiate a view-change protocol, which elects a new primary node and resumes the consensus process. By using PBFT, the maintenance system can achieve fast consensus and high fault tolerance, ensuring the validity and order of maintenance requests. The system can also benefit from the immutability and transparency of the blockchain, enabling secure and auditable tracking of maintenance records for machines in the manufacturing industry.

Pre-Prepare(n, h(m))

Each node i verifies the request and sends a prepare message to all other nodes in the network for the same request with sequence number n, and the hash value of the request $h(m)$.

Prepare(n, h(m))

Where; n represents the sequence number for the maintenance request, $h(m)$ represents the hash value of the maintenance request, f represents the number of faulty nodes that the algorithm can tolerate, m represents the latest stable sequence number, v represents the current view number, and p represents the new primary node. A node i sends a pre-prepare message to all other nodes in the network for a proposed maintenance request with a sequence number n, and the hash value of the request $h(m)$.

Table 1. Algorithm for PBFT consensus.

Step	**Description**
Initialize the system	1. Create a network of nodes that will participate in the consensus process. 2. Establish communication channels between the nodes. 3. Choose a leader node that will initiate the consensus process.
Create a maintenance request	1. When an equipment requires maintenance, a request is created and sent to the network. 2. The request contains the equipment's ID, description of the problem, and other relevant details.
Pre-prepare phase	1. The leader node broadcasts the request to all other nodes. 2. Each node verifies the request's authenticity and checks if it meets the required criteria.
Prepare phase	1. Nodes that validate the request send a "prepare" message to all other nodes. 2. Each node waits for a sufficient number of prepare messages before proceeding to the next phase.
Commit phase	1. Nodes that received enough prepare messages send a "commit" message to all other nodes. 2. Each node waits for a sufficient number of commit messages before proceeding to the next phase.
Response phase	1. The leader node broadcasts the response to all other nodes. 2. Each node verifies the response's authenticity and checks if it meets the required criteria.
Finalize phase	1. Nodes that validate the response send a "finalize" message to all other nodes. 2. Each node waits for a sufficient number of finalize messages before the request is considered complete.
Update the ledger	1. Once the request is complete, the system updates the blockchain ledger with the maintenance details. 2. The ledger is immutable and tamper-proof, ensuring the request's authenticity and accuracy. 3. Repeat. 4. The system repeats this process for each maintenance request, ensuring that all nodes agree on the validity of each request.

After receiving $2f + 1$ prepare messages, each node i sends a commit message to all other nodes in the network for the same request with sequence number n, and the hash value of the request $h(m)$.

Commit(n, h(m))

When a node i receives $2f + 1$ commit messages, it sends a checkpoint message to all other nodes in the network with the latest stable sequence number m.

Checkpoint(m)

If a node i detects that the current primary node is faulty, it sends a view change message to all other nodes in the network with the current view number $v + 1$ and the checkpoint sequence number m.

ViewChange(v+1, m)

Upon receiving $2f + 1$ view change messages, a node j broadcasts a new view message with the latest checkpoint sequence number m' and the new primary node p.

NewView(m', p)

PBFT algorithm ensures that all nodes in the network possess identical information and that every new block appended to the blockchain is legitimate. By using this algorithm, a network can reach a consensus on the state of maintenance data, which can be used to verify the correctness of the data and improve overall maintenance efficiency in the manufacturing industry.

3.3. System Workflow

Every role above has a specific view that they can see when they login into the system. The systems automate the process of calling a technician of service provider, scheduling maintenance, and payments. The scheduled maintenance is done by the system which could be annually, monthly, or quarterly. The technician receives a prompt notification on their view showing the maintenance is to be done on a specific date and time. If a repair of the equipment is needed instantly the repair request is uploaded by the manufacturer on the blockchain system. The technician receives it and does repairs. On completion of work, payment is released by the system. The blockchain is responsible for the safe and secure transmission of payment.

The data and details of all the equipment are stored on the blockchain in cryptographic form. Every equipment has a specific address which shows its unique identification, hence making it easy to trace. The history of all the data is stored in an immutable ledger on the blockchain. When uploading a repair request on the system, the manufacturer has to specify the type of repair needed. A record of daily assessments of the equipment is also stored on the equipment by the operator of the factory. In this way, the technician can view the history and pinpoint the problem in no time.

The automation of tasks is another specialty of this system. Whenever a material or parts run out, it can send an automated notification to the vendor in order to send products. The payment is made on the blockchain. This improves the time efficiency of the factory. The smart contracts are generated for the vendor, manufacturer, payments, and maintenance. When the pre-determined conditions of the contract are met, the contract deploys.

Graphs and analytics showing the health of each and every machine in the industry are displayed on the system as well. Maintenance activities are logged on the blockchain, including the date, time, and nature of the maintenance performed, as well as the maintenance personnel involved.

In Algorithm 1, we define the MaintenanceContract class that extends the Contract class provided by the Fabric SDK. The class defines two methods.

Algorithm 1 Maintenance System Smart Contract

```
procedure INITLEDGER                                                ▷ No operations
end procedure
procedure MACHINEUSAGE($ctx, machine$)
    if CLIENTIDENTITY($ctx$) $\neq$ $machine$ then
        throw “Access denied”
    end if
    $lastMaintenance \leftarrow$ GETSTATE($machine$)
    $maintenanceInterval \leftarrow$ GETSTATE(“maintenanceInterval”)
    if $(current\ date - lastMaintenance) \geq maintenanceInterval$ then
        emit event “MaintenanceNeeded” with $machine$ as data
    end if
end procedure
procedure ASSIGNMAINTENANCE($ctx, machine, technician$)
    if CLIENTIDENTITY’S MSPID($ctx$) $\neq$ “MaintenanceTeamMSP” then
        throw “Access denied”
    end if
    if $technician$ = null or $technician$ = “” then
        throw “Invalid argument: technician”
    end if
    SETSTATE($machine$) to current date
    emit event “MaintenanceAssigned” with payload: {machine, technician}
end procedure
```

The machineUsage method first checks whether the client that invoked the method is the same as the machine ar-gument. If not, an error is thrown. Then it retrieves the last maintenance timestamp for the given machine from the ledger, and checks if the time elapsed since the last maintenance is greater than or equal to the maintenance interval defined in the ledger. If so, it emits a maintenance needed event using the setEvent method provided by the Fabric SDK. The assignMaintenance method checks whether the client invoking the method is a member of the MaintenanceTeamMSP organization. If not, an error is thrown. It also checks that a valid technician argument is provided. Then it updates the last maintenance timestamp for the given machine in the ledger, and emits a maintenance assigned event using the setEvent method. In order to make this chaincode operational on a Hyperledger Fabric network, it must be packaged into a chaincode package and subsequently installed on the peers that will execute it. We would also need to define the required configuration files such as the endorsement policy, chaincode instantiation policy, and channel configuration, depending on the specific network setup.

In Algorithm 2, Init is the initialization function that is called when the chaincode is first instantiated. In this implementation, it does not perform any actions and simply returns nil. RequestPart method allows the inventory manager to request parts from a vendor. It checks if the requester is authorized, if the vendor is valid, and if the inventory quantity is below the threshold. If all conditions are met, the method updates the inventory quantity for the vendor and emits an event called OrderPlaced. GetInventory method retrieves the inventory data for a specific vendor from the world state. It takes the vendor address as input and returns an instance of InventoryVendor which contains the vendor’s address and quantity. SetInventory method sets the inventory data for a specific vendor in the world state. It takes the vendor address and an instance of InventoryVendor as input and stores the data in the world state. The InventoryVendor struct represents a vendor and its corresponding inventory. It has two fields: Address which is the vendor’s address, and Quantity which is the current quantity of inventory for that vendor.

In Algorithm 3, Init function is called when the chaincode is deployed to the network, but it doesn’t have any implementation in this code. RequestPayment function is used by the manufacturer to request a payment for a spare part from a vendor. It takes four arguments: the invoiceID as a unique identifier for the invoice, the vendor address to which the payment is to be made, the amount of the payment, and the paymentMethod used for the payment. It checks if the requester is authorized and creates an invoice object with the provided details. The invoice object is then marshaled into JSON format and stored in the world state. ApprovePayment function is used by the vendor to approve a payment request. It takes the invoiceID as an argument and checks if the requester is authorized. If the invoice exists and has a pending status, its status is updated to approved, and the updated invoice data is stored in the world state. GetInvoice function is used to retrieve an invoice from the world state. It takes the invoiceID as an argument and returns the invoice data as a pointer to the Invoice struct.The Invoice struct contains four fields: invoiceID as the unique identifier for the invoice, vendor as the vendor's address, amount as the payment amount, status as the status of the invoice, which can be "Pending" or "Approved," and paymentMethod as the payment method used for the payment.

Algorithm 2 Chaincode for Inventory Management in Golang

```
1: Input: ctx, vendor, quantity
2: Output: Error or nil
3: procedure INIT return nil
4: end procedure
5: procedure REQUESTPART($ctx, vendor, quantity$)
6:     if $ctx.GetClientIdentity().GetID() \neq$ "inventoryManager" then return
   "only inventoryManager can request a part"
7:     end if
8:     if $vendor =$ "" then return "vendor address is required"
9:     end if
10:    $v, err \leftarrow GetInventory(ctx, vendor)$
11:    if $v.Quantity \geq Threshold()$ then return "vendor inventory already meets the threshold"
12:    end if
13:    $v.Quantity \leftarrow v.Quantity + quantity$
14:    $err \leftarrow SetInventory(ctx, vendor, v)$
15:    $err \leftarrow ctx.GetStub().SetEvent(\text{"OrderPlaced"}, \text{formatted message})$
16:    if $err \neq nil$ then return $err$
17:    end ifreturn nil
18: end procedure
19: procedure GETINVENTORY($ctx, vendor$)
20:    $data, err \leftarrow ctx.GetStub().GetState(vendor)$
21:    if $data = nil$ then return New InventoryVendor with zero quantity
22:    end if
23:    Unmarshal $data$ into $v$ return $v, nil$
24: end procedure
25: procedure SETINVENTORY($ctx, vendor, v$)
26:    Marshal $v$ into $data$
27:    $err \leftarrow ctx.GetStub().PutState(vendor, data)$
28:    if $err \neq nil$ then return $err$
29:    end ifreturn nil
30: end procedure
```

Algorithm 3 Smart Contract for Payment Process in Golang

```
1: Input: ctx, invoiceID, vendor, amount, paymentMethod
2: Output: Error or nil
3: procedure INIT return nil
4: end procedure
5: procedure REQUESTPAYMENT($ctx, invoiceID, vendor, amount, paymentMethod$)
6:     if $ctx.GetClientIdentity().GetID() \neq$ "manufacturer" then return
       “only manufacturer can request a payment”
7:     end if
8:     if $vendor =$ "" then return “vendor address is required”
9:     end if
10:    Create invoice object with provided parameters and status as "Pending"
11:    Marshal invoice into data
12:    $err \leftarrow ctx.GetStub().PutState(invoiceID, data)$
13:    if $err \neq nil$ then return $err$
14:    end if
         return nil
15: end procedure
16: procedure APPROVEPAYMENT($ctx, invoiceID$)
17:    if $ctx.GetClientIdentity().GetID() \neq$ "vendor" then return
       “only vendor can approve a payment”
18:    end if
19:    $data, err \leftarrow ctx.GetStub().GetState(invoiceID)$
20:    if $data = nil$ then return “invoice does not exist”
21:    end if
22:    Unmarshal data into invoice
23:    if $invoice.Status \neq$ "Pending" then return “invoice has already been processed”
24:    end if
25:    $invoice.Status \leftarrow$ "Approved"
26:    Marshal invoice into data
27:    $err \leftarrow ctx.GetStub().PutState(invoiceID, data)$
28:    if $err \neq nil$ then return $err$
29:    end if
         return nil
30: end procedure
```

3.4. Technology Stack

For the implementation of PBEMS, various supporting components are needed. For development of simulation and prototyping, we use the following components:

- Ubuntu 22.04 LTS, an open-source operating system from Canonical Ltd;
- Hyperledger Fabric 2.1.0, a permissioned blockchain platform from The Linux Foundation;
- Docker 20.10.21, a container platform to run nodes of Hyperledger fabric;
- Docker Compose 1.25.0, a tool to manage of Docker multi-container applications;
- CouchDB 2.3, an open-source database in JSON format to store transactions in the blockchain network;
- Node.js 16.13.0, an open-source cross-platform JavaScript runtime environment from The OpenJS Foundation;
- Golang 1.17.3, an open-source programming language supported by Google to run Chaincode smart contracts in Hyperledger Fabric;
- PHP 7.4.3, a general-purpose scripting language for web application to develop blockchain client;
- MetroUI 4.5.1, a frontend toolkit with Fluent design;
- Nginx 1.18.0, an open-source web server to present blockchain client application;
- Hyperledger Caliper 0.4.0, a blockchain network benchmarking tool from The Linux Foundation.

3.5. Development and Simulation Environment

Our development environment for client applications is powered by the Windows 10 Pro 64-bit operating system. The hardware setup includes an Intel® Core i7-7820HQ processor, which, with its quadcore architecture and hyper-threading capabilities, is complemented by 16GB of RAM. For client application development, Visual Studio is the integrated development environment (IDE) of choice.

Our simulation environment is hosted on a cloud server and is built to foster efficient and seamless simulation executions. It runs on Ubuntu 22.4 LTS 64-bit, ensuring stability and support as Ubuntu's Long Term Support versions are known for their prolonged maintenance periods. The server is powered by an Intel® Xeon® Platinum 8249C CPU, clocking at 2.10 GHz, and boasting 8 virtual cores. Complementing the CPU, the environment comes with 16GB of RAM and a 100GB disk space that provides sufficient storage for datasets, simulation outputs, and necessary software tools.

4. Results and Analysis

In this section, we present the results of the experiments that have been carried out. We display a prototype for the client application, and perform a performance and security analysis for PBEMS.

4.1. Client Prototype

We developed a generic client application prototype that can be used by all parties involved to interact with the blockchain network. Each party may make adjustments to the application according to their needs in the future.

This client application has three main menus, namely:

- **Equipment:** This menu is for managing equipment including viewing equipment data, as well as adding, editing, and deleting equipment.
- **Maintenance:** For manufacturing companies, this menu is for submitting maintenance requests to vendors. Meanwhile for vendors, this menu is used to receive and carry out maintenance activities.
- **Payment:** For vendors, this menu is for submitting payment requests. Meanwhile for manufacturing companies, this menu is used to review, accept, or reject payment requests. In case a maintenance service activity is contained in a smart contract, payment can be processed automatically.

4.2. Performance Analysis

The more parties involved, especially if more and more equipment can channel data directly and automatically to the blockchain, the performance of PBEMS is very important. To test the performance of the blockchain network from PBEMS, we made a simulation by increasing the number of transactions in the network, from 10 to 1,000,000. This simulation is to see the throughput which are marked with transactions per second (TPS), in addition to the latency in seconds. We measured performance for both write and read operations.

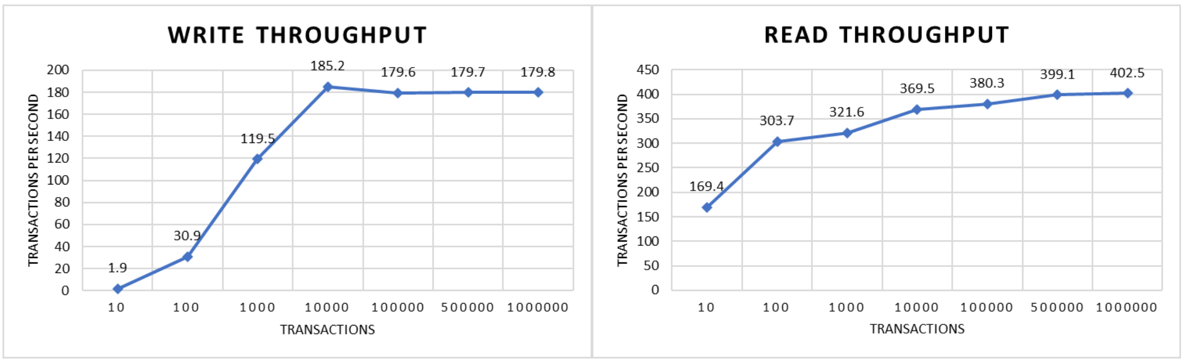

Figure 2. Throughput of PBEMS, including (a) write throughput and (b) read throughput.

As shown in Figure 4 (a), the write throughput of PBEMS is higher when the number of transactions is increased. Throughput is only 1.9 TPS when 10 transactions are sent, and reaches the highest of 185.2 TPS when 10,000 transactions are sent. The next increase in the number of transactions tends to be stable around 179.6 to 179.8 TPS.

The read throughput of PBEMS is higher than the write throughput by approximately two times, as shown in Figure 4 (b). Like the write throughput, the read throughput also has an increasing trend when the number of transactions sent is getting bigger. Starting when 10 transactions were sent, the read throughput that was obtained was 169.4 TPS, and without any decrease in throughput then reached its peak when 1,000,000 transactions were 402.5 TPS.

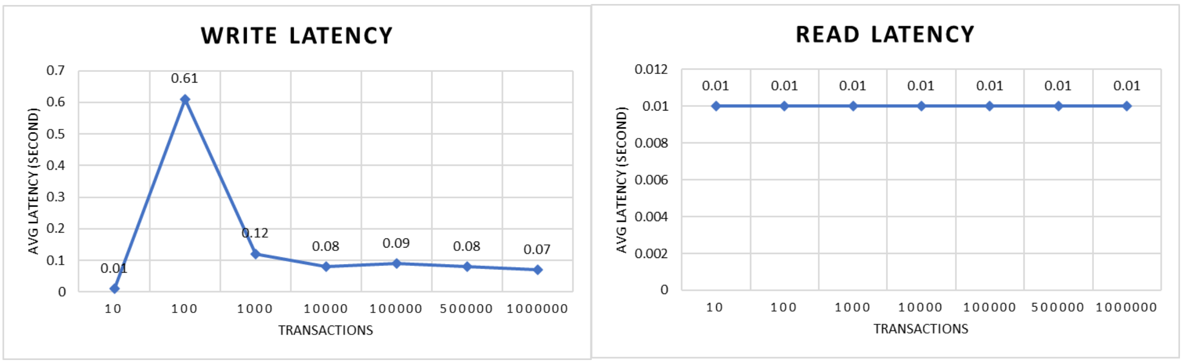

Figure 3. Latency of PBEMS, including (a) write latency and (b) read latency.

As shown in Figure 5 (a), there is a fluctuation in the average write latency of PBEMS when the number of transactions sent is small, namely 0.01 seconds for 10 transactions, then rises drastically to 0.61 seconds for 100 transactions, then drops to 0.12 seconds for 1000 transaction. After that, the average write latency is stable at 0.08 to 0.09 seconds for the addition of the next transaction, up to a minimum of 0.07 seconds in 1000000 transactions. Meanwhile, the average PBEMS read latency is stable at 0.01 seconds regardless of the number of transactions obtained, as shown in Figure 5 (b).

4.3. Security Analysis

Hyperledger Fabric has proven to be a secure permissioned blockchain platform. However, some potential security risks need attention, including:

- **Smart Contract Vulnerabilities:** PBEMS utilizes smart contracts for automating and enforcing business logic. Common smart contract vulnerabilities, such as reentrancy, integer overflow, or unauthorized access, should be addressed during development. Smart contracts should undergo rigorous testing and security auditing to minimize the risk of exploitation.

- **Node and Peer Security:** The security of nodes and peers within the Hyperledger Fabric network is critical. Adequate measures should be implemented to protect against unauthorized access to nodes, compromised peers, or Denial-of-Service (DoS) attacks. Regular monitoring and patching of software vulnerabilities are crucial to maintain a secure network.
- **User Authentication and Authorization:** The security of nodes and peers within the PBEMS relies on proper user authentication and authorization mechanisms to ensure the integrity and confidentiality of data. However, vulnerabilities can emerge if weak or easily guessable passwords are used, if users share their login credentials, or if adequate access controls are not implemented. Insider threats also pose a risk if an authorized user intentionally or unintentionally abuses their privileges or grants access to unauthorized individuals.
- **System Administration:** The security of PBEMS heavily depends on the effective management and administration of the system. Weak system administration practices, such as not promptly revoking access rights for former employees, inadequate monitoring and logging, or insufficient patch management, can introduce vulnerabilities. If these aspects are not addressed, unauthorized access, data leaks, or malicious activities can occur.
- **External Integration Points:** PBEMS may integrate with external systems, such as enterprise resource planning (ERP) software, asset management tools, or supply chain systems. These integrations may introduce security risks if proper security controls are not implemented. Unauthorized access, data leakage, or system compromise can occur if the integrated systems are not adequately protected or if the integration mechanisms themselves are flawed.
- **Human Error and Training:** The effectiveness of PBEMS relies heavily on the competence and awareness of its users. Human error, such as misconfiguration or unintentional exposure of sensitive information, can result in security breaches. Insufficient training on security best practices, inadequate user documentation, or lack of awareness campaigns can amplify this vulnerability. Regular training programs, clear user guidelines, and continuous monitoring of user activities can help reduce the impact of human error.

5. Conclusion

In conclusion, this paper has offered development of PBEMS, a Permissioned Blockchain-based Equipment Maintenance System, tailored for the domain of smart manufacturing. The PBEMS leverages the capabilities of Hyperledger Fabric to ensure a secure, reliable, and efficient system for the management and monitoring of equipment maintenance processes.

Throughout this research, it was evidenced that integrating blockchain technology into the equipment maintenance framework bolsters the trust and transparency among the stakeholders involved. Hyperledger Fabric was selected due to its prominent attributes in terms of privacy, security, scalability, and flexibility, which are essential for the highly-sensitive environment of smart manufacturing. Additionally, a client application prototype was developed as part of this research. The prototype exhibits the practical applicability of PBEMS and demonstrates how various parties can interact with the blockchain network. This prototype is instrumental in substantiating the real-world feasibility of integrating PBEMS into existing manufacturing systems.

Furthermore, the performance of the blockchain network was rigorously evaluated through a set of benchmarks using Hyperledger Caliper. These tests confirmed that the PBEMS is able to handle a considerable number of transactions per second, while maintaining an acceptable latency, thus meeting the performance requirements for a wide range of manufacturing scenarios. However, it is imperative to acknowledge that the implementation of PBEMS in a real-world scenario can be associated with challenges. For instance, the integration of PBEMS with existing legacy systems, as well as addressing potential concerns regarding the adoption and understanding of blockchain technology among the stakeholders, need to be considered.

This research has set the groundwork for further research in this area. Future studies could focus on the development of more advanced features for the client application, fine-tuning of the blockchain network for optimal performance under various conditions, and exploring interoperability with other blockchain platforms or traditional databases. Moreover, pilot implementations in actual manufacturing settings and case studies analyzing the benefits and challenges of adopting PBEMS on a larger scale would provide invaluable insights.

In summation, PBEMS represents a pioneering approach in harnessing the power of blockchain technology to revolutionize equipment maintenance in smart manufacturing. Its development is not only a technical achievement but also a testament to the potential of blockchain in contributing towards the evolution of Industry 4.0. Through continued research, development, and collaboration among stakeholders, PBEMS could play a crucial role in shaping the future landscape of smart manufacturing.

Author Contributions: Conceptualization, M.I.S.A. and Y.G.; methodology, M.I.S.A.; software, M.I.S.A., F.N. and M.R.F.; validation, M.I.S.A., Y.G., A.W.A and F.B; formal analysis, M.I.S.A.; investigation, M.I.S.A.; resources, M.I.S.A.; data curation, M.I.S.A., A.W.A, F.B.; writing—original draft preparation,M.I.S.A.; writing—review and editing, A.A., A.W.A, F.B.; visualization, A.A., A.O.I., F.N. and M.R.F; supervision, Y.G.; project administration, A.O.I. and F.B.; funding acquisition, A.O.I. and A.W.A.; All authors have read and agreed to the published version of the manuscript

Funding: This research has no external funding.

Data Availability Statement: Not Aplicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1

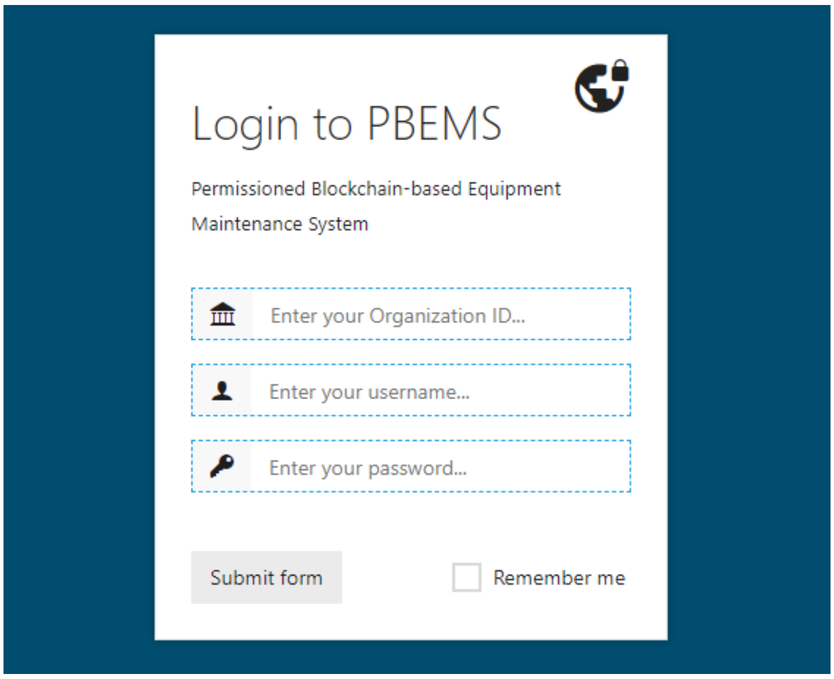
Login to PBEMS
Permissioned Blockchain-based Equipment Maintenance System
Enter your Organization ID...
Enter your username...
Enter your password...
Submit form
Remember me

Figure A1. Application Login Screen.

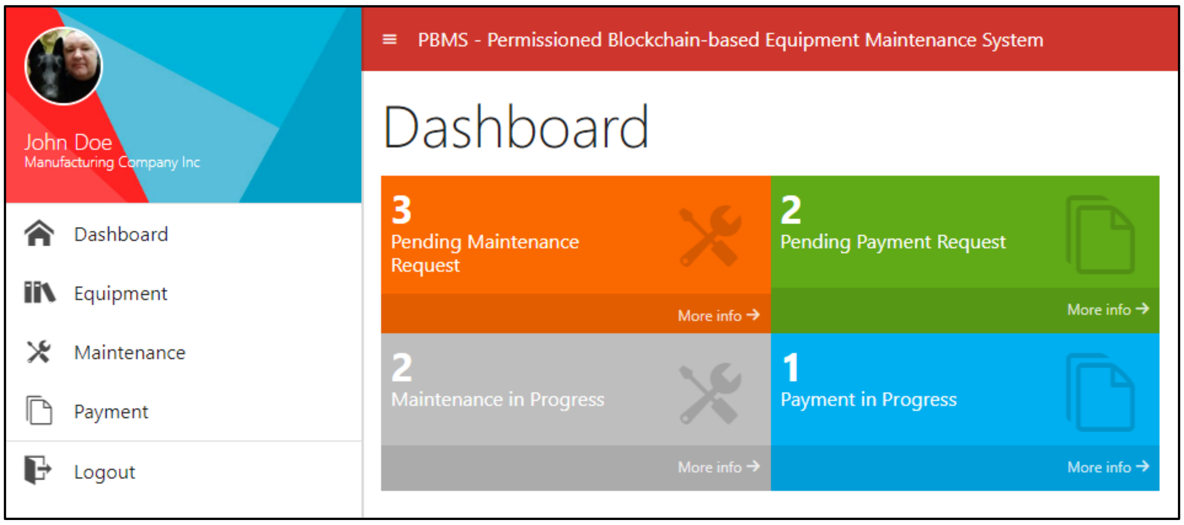

Figure A2. Application Dashboard.

References

1. Lee, J.; Bagheri, B.; Kao, H.A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters* **2015**, *3*, 18–23.
2. Mobley, R.K. *An introduction to predictive maintenance*; Elsevier, 2002.
3. Lee, J.; Wu, F.; Zhao, W.; Ghaffari, M.; Liao, L.; Siegel, D. Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mechanical systems and signal processing* **2014**, *42*, 314–334.
4. Jardine, A.K.; Lin, D.; Banjevic, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing* **2006**, *20*, 1483–1510.
5. Muchiri, P.; Pintelon, L.; Gelders, L.; Martin, H. Development of maintenance function performance measurement framework and indicators. *International Journal of Production Economics* **2011**, *131*, 295–302.
6. Susto, G.A.; Schirru, A.; Pampuri, S.; McLoone, S.; Beghi, A. Machine learning for predictive maintenance: A multiple classifier approach. *IEEE transactions on industrial informatics* **2014**, *11*, 812–820.
7. Lu, Y. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of industrial information integration* **2017**, *6*, 1–10.
8. Rittinghouse, J.W.; Ransome, J.F. *Cloud Computing*; CRC Press, 2017.
9. Osman, A.M.; Dafa-Allah, A.; Elhag, A.A.M. Proposed security model for web based applications and services. In Proceedings of the 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE). IEEE, 2017, pp. 1–6.
10. Mendling, J.; Weber, I.; Aalst, W.V.D.; Brocke, J.V.; Cabanillas, C.; Daniel, F.; Debois, S.; Ciccio, C.D.; Dumas, M.; Dustdar, S.; et al. Blockchains for business process management-challenges and opportunities. *ACM Transactions on Management Information Systems (TMIS)* **2018**, *9*, 1–16.
11. Kieseberg, P.; Hobel, H.; Schrittwieser, S.; Weippl, E.; Holzinger, A. Protecting anonymity in data-driven biomedical science. *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics: State-of-the-Art and Future Challenges* **2014**, pp. 301–316.
12. Wollschlaeger, M.; Sauter, T.; Jasperneite, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE industrial electronics magazine* **2017**, *11*, 17–27.
13. Solyman, A.M.; Ibrahim, O.A.; Elhag, A.A.M. Project management and software quality control method for small and medium enterprise. In Proceedings of the 2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE). IEEE, 2015, pp. 123–128.
14. Mahmoud, Z.; Solyman, A.; Elhag, A.A.M. Harmonized software quality improvement models for Sudanese SME based on CMMI. In Proceedings of the 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE). IEEE, 2019, pp. 1–6.
15. Faniyi, F.; Bahsoon, R. A systematic review of service level management in the cloud. *ACM Computing Surveys (CSUR)* **2015**, *48*, 1–27.

16. Casino, F.; Dasaklis, T.K.; Patsakis, C. A systematic literature review of blockchain-based applications: Current status, classification and open issues. *Telematics and informatics* **2019**, *36*, 55–81.
17. Androulaki, E.; Barger, A.; Bortnikov, V.; Cachin, C.; Christidis, K.; De Caro, A.; Enyeart, D.; Ferris, C.; Laventman, G.; Manevich, Y.; et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the Proceedings of the thirteenth EuroSys conference, 2018, pp. 1–15.
18. Ferrag, M.A.; Maglaras, L.; Ahmim, A. Privacy-preserving schemes for ad hoc social networks: A survey. *IEEE Communications Surveys & Tutorials* **2017**, *19*, 3015–3045.
19. Xu, X.; Weber, I.; Staples, M.; Zhu, L.; Bosch, J.; Bass, L.; Pautasso, C.; Rimba, P. A taxonomy of blockchain-based systems for architecture design. In Proceedings of the 2017 IEEE international conference on software architecture (ICSA). IEEE, 2017, pp. 243–252.
20. Wang, Y.; Singgih, M.; Wang, J.; Rit, M. Making sense of blockchain technology: How will it transform supply chains? *International Journal of Production Economics* **2019**, *211*, 221–236.
21. Hawlitschek, F.; Teubner, T.; Weinhardt, C. Trust in the sharing economy. *Die Unternehmung* **2016**, *70*, 26–44.
22. Stodt, J.; Jastremskoj, E.; Reich, C.; Welte, D.; Sikora, A. Formal description of use cases for industry 4.0 maintenance processes using blockchain technology. In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). IEEE, 2019, Vol. 2, pp. 1136–1141.
23. Welte, D.; Sikora, A.; Schönle, D.; Stodt, J.; Reich, C. Blockchain at the shop floor for maintenance. In Proceedings of the 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). IEEE, 2020, pp. 15–22.
24. Bai, L.; Hu, M.; Liu, M.; Wang, J. BPIIoT: A light-weighted blockchain-based platform for industrial IoT. *IEEE Access* **2019**, *7*, 58381–58393.
25. Chen, Q.; Zhu, Z.; Si, S.; Cai, Z. Intelligent maintenance of complex equipment based on blockchain and digital twin technologies. In Proceedings of the 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2020, pp. 908–912.
26. Tran, V.H.; Lenssens, B.; Kassab, A.; Laks, A.; Rivière, E.; Rosinosky, G.; Sadre, R. Machine-as-a-Service: Blockchain-based management and maintenance of industrial appliances. *Engineering Reports* **2022**, p. e12567.
27. Miehle, D.; Meyer, M.M.; Luckow, A.; Bruegge, B.; Essig, M. Toward a decentralized marketplace for self-maintaining machines. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019, pp. 431–438.
28. Hasan, H.R.; Salah, K.; Jayaraman, R.; Ahmad, R.W.; Yaqoob, I.; Omar, M. Blockchain-based solution for the traceability of spare parts in manufacturing. *IEEE Access* **2020**, *8*, 100308–100322.
29. Aleshi, A.; Seker, R.; Babiceanu, R.F. Blockchain model for enhancing aircraft maintenance records security. In Proceedings of the 2019 IEEE International Symposium on Technologies for Homeland Security (HST). IEEE, 2019, pp. 1–7.
30. Jensen, W.L.; Jessing, S.; Chiu, W.Y.; Meng, W. AirChain-towards blockchain-based aircraft maintenance record system. In Proceedings of the 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2022, pp. 1–3.
31. Jensen, W.L.; Jessing, S.; Chiu, W.Y.; Meng, W. A practical blockchain-based maintenance record system for better aircraft security. In Proceedings of the International Conference on Science of Cyber Security. Springer, 2022, pp. 51–67.
32. Andrei, A.; Balasa, R.; Costea, M.; Semenescu, A. Building a blockchain for aviation maintenance records. In Proceedings of the Journal of Physics: Conference Series. IOP Publishing, 2021, Vol. 1781, p. 012067.
33. Ahmad, R.W.; Salah, K.; Jayaraman, R.; Hasan, H.R.; Yaqoob, I.; Omar, M. The role of blockchain technology in aviation industry. *IEEE Aerospace and Electronic Systems Magazine* **2021**, *36*, 4–15.
34. Schyga, J.; Hinckeldeyn, J.; Kreutzfeldt, J. Prototype for a permissioned blockchain in aircraft MRO. In Proceedings of the Hamburg International Conference of Logistics (HICL) 2019. epubli GmbH, 2019, pp. 469–505.
35. Mohril, R.S.; Solanki, B.S.; Lad, B.K.; Kulkarni, M.S. Blockchain enabled maintenance management framework for military equipment. *IEEE Transactions on Engineering Management* **2021**, *69*, 3938–3951.
36. Mohamed, E.R.; MABROUKI, M. Critical Study of the Different Types of Maintenance Used in Industry. *Research Journal of Applied Sciences, Engineering and Technology* **2018**, *15*, 91–97.

37. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review* **2008**.
38. Yusoff, J.; Mohamad, Z.; Anuar, M. A Review: Consensus Algorithms on Blockchain. *Journal of Computer and Communications* **2022**, *10*, 37–50.
39. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems* **2020**, *105*, 475–491.
40. Assaqty, M.I.S.; Gao, Y.; Hu, X.; Ning, Z.; Leung, V.C.; Wen, Q.; Chen, Y. Private-blockchain-based industrial IoT for material and product tracking in smart manufacturing. *IEEE Network* **2020**, *34*, 91–97.
41. Mohamed, N.; Al-Jaroodi, J. Applying blockchain in industry 4.0 applications. In Proceedings of the 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). IEEE, 2019, pp. 0852–0858.
42. Aggarwal, S.; Kumar, N. Hyperledger. In *Advances in computers*; Elsevier, 2021; Vol. 121, pp. 323–343.
43. Nasir, Q.; Qasse, I.A.; Abu Talib, M.; Nassif, A.B.; et al. Performance analysis of hyperledger fabric platforms. *Security and Communication Networks* **2018**, *2018*.
44. Krstić, M.; Krstić, L. Hyperledger frameworks with a special focus on hyperledger fabric. *Vojnotehnički glasnik/Military Technical Courier* **2020**, *68*, 639–663.
45. Kuzlu, M.; Pipattanasomporn, M.; Gurses, L.; Rahman, S. Performance analysis of a hyperledger fabric blockchain framework: throughput, latency and scalability. In Proceedings of the 2019 IEEE international conference on blockchain (Blockchain). IEEE, 2019, pp. 536–540.
46. Shuaib, M.; Hassan, N.H.; Usman, S.; Alam, S.; Bakar, N.A.A.; Maarop, N. Performance evaluation of DLT systems based on hyper ledger fabric. In Proceedings of the 2022 4th International Conference on Smart Sensors and Application (ICSSA). IEEE, 2022, pp. 70–75.
47. Al-Sumaidaee, G.; Alkhudary, R.; Zilic, Z.; Swidan, A. Performance analysis of a private blockchain network built on Hyperledger Fabric for healthcare. *Information Processing & Management* **2023**, *60*, 103160.