JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

SMART CONTRACT-DRIVEN QUEUE MANAGEMENT FOR EFFICIENT ONLINE TICKET PURCHASING ON BLOCKCHAIN

Mery Oktaviyanti Puspitaningtyas*[1)], Happid Ridwan Ilmi[2)], Yulita Ayu Wardani[3)], Irwansyah Saputra[4)]

1. Computer Science, Faculty of Information Technology, Nusa Mandiri University, Jakarta, Indonesia
2. Computer Science, Faculty of Information Technology, Nusa Mandiri University, Jakarta, Indonesia
3. Computer Science, Faculty of Information Technology, Nusa Mandiri University, Jakarta, Indonesia
4. Computer Science, Faculty of Information Technology, Nusa Mandiri University, Jakarta, Indonesia

Article Info

Keywords: Blockchain; FIFO; Online Ticket Purchasing; Smart Contract

Article history:
Received 14 January 2025
Revised 16 February 2025
Accepted 3 March 2025
Available online 2 December 2025

DOI :
https://doi.org/10.29100/jipi.v10i4.7367

* Corresponding author.
Mery Oktaviyanti Puspitaningtyas
E-mail address:
14230004@nusamandiri.ac.id

ABSTRACT

This study investigates how smart contract-driven queue management can be utilized to increase online ticket purchasing efficiency via blockchain technology. The system is designed to manage ticket purchase queues transparently and securely, using smart contracts written in the Solidity programming language and the Ionic UI framework. In addition, the system is connected with MetaMask as a transaction wallet, allowing users to purchase tickets directly and securely. Ganache serves as a testing environment for replenishing wallet balances without involving real transactions. The First In First Out (FIFO) approach is used to manage the transaction queue, with the first purchased ticket being processed first by the administrator. The administrator accepts each transaction, which is then confirmed by MetaMask. When the transaction is confirmed, the system automatically updates the ticket status. The implementation results show that this system effectively optimizes ticket transaction management transparently and securely. This work also makes a significant contribution to the application of blockchain technology for better management of online ticket purchasing systems, as well as minimizing the possibility of transaction errors and fraud.

.

I. INTRODUCTION

The e-commerce service provider matured to sell and distribute digital content as well, like e-books, e-music, e-tickets, etc. This two-sided marketplace offers a platform for content producers to market their goods online. Traditional e-ticketing models depend on a centralized body. Furthermore, they struggle with efficiency and scalability [1].

Blockchain-based applications called ‘smart contracts’ contain the terms that must be fulfilled in a contract between two parties. Once the terms are satisfied, the contract is automatically executed. The contract can be deployed on the platform and is written in Solidity. Instead, it should operate more like an independent agent that resides within the execution environment. It has direct control over its own balance and key/value storage, and it always executes some code once some msg or transaction is sent; thus it can monitor persistent variables [2].

Moreover, blockchain technology enables the formation of smart contracts among users lacking mutual trust, ensuring transaction confidentiality [3].

Smart contracts use blockchain technology to enable and enforce agreements between untrustworthy parties without the need for a trusted third party. Smart contracts enabled network automation and the conversion of paper contracts to digital contracts. Smart contracts automate transactions without the need for a central authority, allowing users to document their agreements and build confidence. Smart contracts are copied to all blockchain nodes to prevent tampering. Blockchain technologies can eliminate human error and help avoid contract disputes by automating procedures [4].

Blockchain technology enables smart contracts. Smart contracts are embedded programming contracts that can be integrated into blockchain data, transactions, or assets to create systems, markets, or assets controlled by the program [5].

Blockchain is derived from the white paper released by Nakamoto Satoshi in 2008. We have a blockchain, or distributed ledger, which stores a chain of consecutive blocks linked with each other by the hash value of the previous block header. Besides the unavoidable cryptographic hash in a transaction's data (which the block contains), a timestamp, nonce, and transaction data are aggregated in a block. The block timestamp itself is also

only valid if its value exceeds the network-adjusted time plus two hours and is larger than the median timestamp of the previous eleven blocks, therefore potentially preventing a malicious actor from manipulating the blockchain. Please take into account that network-adjusted time means the median of the fully connected nodes' timestamps. It ensures that not just one or several nodes maintain the smooth execution of the blockchain, but rather, every node in the blockchain network must adhere to a common consensus protocol to create and verify new blocks. The consensus protocol is the base of blockchain where all the principles of operation and legitimate actions will be regulated [6].

Blockchain technology is one of the most significant innovations of the century. This technology improves operational and regulatory verification, as well as transparency and traceability in various sectors' supply chains. DLT, often known as 'blockchain,' has gained popularity among financial organizations. A blockchain is a decentralized list of documents, known as 'blocks,' connected by mining. This approach transforms pending transactions into mathematical puzzles. Miners use computer systems to solve puzzles and generate unique hashes (letters and numbers) for each block [7].

Blockchain is a decentralized node network that holds data. This technique effectively safeguards confidential data in the system. This technology enables secure and secret communication of crucial data. This tool is ideal for securely storing all associated papers. Blockchain also speeds up searches for applicants who meet certain trial criteria by combining a single patient database. Blockchain is a decentralized network of nodes that store and record transaction data [8].

Blockchain technology allows for the simultaneous recording of operation records across several devices. This system stores digital data in interconnected blocks, including transactions, contracts, and contact databases. Inadequate financial regulations might lead to errors and misinterpretations. Blockchain technology is gaining popularity. A small group of individuals is exploring ways to implement and utilize the benefits of this technology in their businesses [9].

Blockchain can aggregate and manage data to provide the highest quality of service. Combining blockchain and big data enables the development of a smart city by combining the benefits of key data innovations. A classic transaction system requires centralized authorization or third-party verification for transactions. Untrustworthy parties can jeopardize the transaction's security. Blockchain transactions are decentralized and employ public and private keys, which reduces fraud [10].

Blockchain stores transactions enforces rules, and transmits data. Smart contracts facilitate transactions and operations. Smart contracts are scripts or programs that execute automatically during transactions, as seen through the lens of a programmer. Blockchains can hold both data and executable code, which serves as the programming logic for updating data and activating external activities. Smart contracts have great promise in the legal field, which is rapidly expanding [11].

Smart contracts, which may digitally mimic real-world contracts, are an important component of blockchain technology. Smart contracts include code for executing agreements between parties, monitoring terms, and performing embedded operations. Smart contracts replace traditional legal third-party contracts by network consensus. They can improve efficiency and save transaction costs since they run automatically when specific criteria are met and preserve a digital record of the rules and business logic [12].

Blockchain's smart contract feature allows for non-subjective computer code to define how events are managed and what steps should be taken when they occur, in addition to providing a distributed and unchangeable record of past events. Ethereum's smart contracts aim to overcome Bitcoin's constraints. Smart contracts are computer code that responds to major events. The smart contract does not need to involve many parties or be legally binding [13].

Smart contracts cannot activate themselves. It only comes alive when one of its functions is expressly used in a transaction. Smart contracts cannot function independently and must be triggered by an external event, such as a transaction. Smart contract use cases have challenges in processing non-blockchain information and events. A smart contract can react to blockchain transactions 'on-chain,' but it cannot receive information from off-chain sources, such as a person leaving a hotel room. Although the blockchain offers a secure environment, participants in a smart contract may not trust the information communicated to it [14].

A smart contract is a contractual arrangement that is incorporated in a self-enforcing code. The participants in the agreement agree to interact based on preset limitations. When a condition is satisfied, the predefined procedures are executed automatically. Smart contracts offer greater transparency and eliminate the need for trusted third parties [15].

Ethereum supports smart contract development using high-level programming languages like Solidity, Obsidian, and Vyper. To interact with the smart contract, the code must be compiled to bytecode (e.g., using the solc compiler for Solidity or the vyper compiler for Vyper) and a corresponding application binary interface (ABI) file. When deployed, the bytecode is part of the payload of an Ethereum transaction. Once the bytecode is published, it is permanently stored on the blockchain in the form of a smart contract. Like externally held accounts used by

businesses outside of the distributed ledger, each Ethereum smart contract has its account with a unique address. Smart contracts can store and transfer funds and can interact with other accounts. The smart contract read function will only need you to call the contract address and the signature of the contract function on the data field [16].

Define "smart legal contracts" (or "Ricardian contracts"), which strive to capture the defining aspects of a legal agreement in a format that can be written and performed in software code. Many smart contracts published in the literature do not focus as much on formal legal considerations as the Ricardian approach. Their "smart" nature is connected to their ability to self-enforce utilizing a certain set of rules whenever predefined criteria are met. When applied on a blockchain, smart contracts may automatically reach and execute agreements, resulting in faster processing and lower costs. This is especially useful for recurrent trust-free agreements/transactions of little financial value, such as half-hourly peer-to-peer energy trading. Smart contracts use "if-then" logic to program desired outcomes and conditions. A smart contract can trigger an action, such as discharging a battery when the electricity export price exceeds a threshold or funds are successfully transferred from a buyer [17].

Ethereum smart contracts have three main partners. Functions, events, and state variables are some of the building blocks of Ethereum smart contracts, and they are all Solidity-written. Solidity is a Turing-complete programming language optimized for this purpose, as with smart contracts. The EVM bytecode of the smart contract code would be created and placed on the Ethereum blockchain using the contract formation transaction. The Ethereum address will be computed as a consequence of the effective creation of a contract [18].

The accompanying data are stored in a public ledger that contains every transaction ever completed via the blockchain. The blockchain has the advantage of being a decentralized system that eliminates the need for a third-party organization to act as a middleman. Blockchain transactions are more transparent than centralized third-party transactions due to information sharing across all nodes. Additionally, anonymizing nodes in the blockchain improves transaction confirmation security. End-to-end encryption in blockchains requires two keys: public and private. In public-key cryptography, each public key corresponds to just one private key. These two keys encrypt and decrypt essential messages. Encoding a message with a public key allows only the recipient to decode it with their corresponding private key. Public keys are similar to business addresses in that they are publicly accessible and can be shared. These encrypt messages before they are sent to the recipient. This public key is associated with a unique private key, similar to a key to the front door. Only the person with the private key can unlock the door or decipher the message. These keys safeguard the exchanged data [19].

The smart contract, like traditional contracts, is a set of organizational rules and conditions that govern the trust between the parties participating in the contract. The only distinction is that a smart contract is created using a programming language. The rules, terms, and conditions are applied via regulated coding, which accurately reflects the agreement approved by all parties. The goal of a smart contract was to incorporate contractual stipulations within a combination of hardware and software, making violating it difficult and prohibitively expensive, eventually boosting contract security and minimizing the risk of an attack. The Ethereum blockchain popularized the concept of smart contracts and their practical application in 2016 [20].

Blockchains have evolved from distributed digital ledgers to distributed computing platforms that include immutable data, logical and behavioral information, and automated stakeholder relationships (since smart contracts). Smart contracts can add functionality to blockchain data, allowing for extra services. Contracts aggregate data and initiate action based on specific conditions. The data used in contract logic is primarily received from the blockchain when it is deployed [21].

FIFO scheduling, also known as first-come-first-served (FCFS), orders jobs based on their arrival times, regardless of priority or urgency. FIFO policies fall short of fixed-priority (FP) or earliest-deadline-first (EDF) policies in terms of hard real-time scheduling. FIFO scheduling is commonly used in low-cost embedded systems, model-based design, the Linux kernel, and real-time packet processing. The primary reason to use FIFO is its simplicity, which presents itself in a variety of ways. FIFO scheduling is easy to implement as it only requires managing a list of jobs. Because jobs are done in arrival order, there is no need for preemption. This reduces bookkeeping code, context-switch overheads, cache-related delays, and the system's overall state space. Moreover, FIFO has incredibly minimal scheduling overhead, as a scheduling decision occurs only once per job, and the only activity performed is removing the head element from a FIFO queue, which requires negligible time. Furthermore, the scheduler can avoid loops, resulting in a constant time. EDF scheduling does not provide a constant-time scheduler, whereas FP scheduling involves restricting the number of priorities to a constant number. The intrinsic simplicity of FIFO makes it appropriate for hardware implementation. FIFO does not encounter scheduling problems related to execution times on either uni- or multiprocessors. Furthermore, it is free from starvation, indicating that all tasks are finished unless deliberately abandoned, leading to limited lateness [22].

Preece et al. [23], this study introduces ‘System for Ticketing Ubiquity with Blockchain’ (STUB), an ‘onto chains’ based transport ticketing solution, which is a combination of ontology and blockchain. The goal of STUB is to address the weaknesses of traditional systems such as ticket fraud, lack of interoperability, and inability to

adapt to changing transport networks. The Transport Ontology (TOnNes) simulates the complex relationships that exist between system components, and the blockchain ensures security and transparency by recording changes to the TOnNes. Merkle proofs, which effectively and securely connect on-chain and off-chain data, are used for data validation. Research shows that STUB can make ticketing systems easier to use and easier to deploy. This solution offers many opportunities to transform transport ticketing systems to be more sustainable, efficient, and transparent.

Leinweber et al. [24], this study looks at how a decentralized ticketing system, 'Mobility-as-a-Service' (MaaS), uses distributed account technology (DLT). By using a unified interface and customer approaches such as 'pay-as-you-go' fares, MaaS combines various transport services. However, the conventional, centrally managed MaaS ticketing system may lead to dependence on the main provider and discrepancies between service providers. This study develops an initial model of a decentralized ticketing system with token-based distribution (DLT), maps technical issues such as confidentiality, cost, latency, and maintenance, and assesses solutions to these issues. To improve system performance and confidentiality, trusted execution environments (TEEs) were considered as potential. This study describes how DLT and TEEs can be applied in MaaS ticketing systems to create a more flexible and inclusive mobility ecosystem.

Grønli et al. [25], developed a new Be-in and Be-out (BIBO) automated ticketing system for public transport, called BlocKoo: a novel high-throughput automated ticketing system combining mobile sensors and blockchain technology. Current BIBO systems with Bluetooth and GPS may struggle against certain types of fraud (e.g., users shutting off Bluetooth will allow them to exploit fare calculation) or problems due to a lost mobile connection. Therefore, to remove the problems, the proposed system, the Blockchain Mobile Ticketing System (BMTS-BIBO), is based on socket programming and blocks, which do not rely on Bluetooth and GPS. A DCNN is used for training ticketing models, which are embedded in the application. The results of the simulation indicate that the BMTS-BIBO algorithm leads to 98% accuracy in ticketing without GPS and Bluetooth relying on any sort of input, making it a non-intrusive and reliable solution for public transport automation.

Rafati et al. [26], in this paper, we introduce DeTi, a decentralized ticketing platform to counter the issues with electronic ticketing, such as counterfeiting, speculative resale, and scalping. DeTi leverages Ethereum-based smart contracts to control the distribution of tickets and oversee aftermarket sales. It can also allow users to verify the validity of tickets for certain events, as well as detect fraudulent ones. DeTi utilizes the trustless and decentralized nature of blockchain technology to provide secure and efficient validation processes to leave ticket forgery, replication, and scalping to the judiciary. Furthermore, the platform institutes resale conditions by the original price defined by event organizers to promote a healthy ticketing ecosystem. Results of the evaluation show that DeTi validates and invalidates tickets well, making it an effective solution to manage event tickets and overcome major weaknesses of conventional systems.

Feulner et al. [27], in this study, we show how SSI can be applied in the event ticketing domain to alleviate pervasive issues of ticket counterfeiting and scalping, thus leading to low costs and trust in secondary markets. While blockchain-based ticketing systems add a greater layer of digital trust and validate purchased tickets, they can lead to privacy issues and an inefficient identification process when it comes to visitors entering an attraction. Through a design science research paradigm, the authors design and evaluate an SSI–based framework for event ticketing. This allows efficient secondary market control through centralized exchange models with generic support to validate ticket ownership with SSI without compromising user privacy. Key findings stress the importance of revocation registries, then derive design principles for secure and efficient identity and ticket verification, demonstrating that SSI has the potential to radically transform digital ticketing ecosystems to be more trustworthy and give end-users higher control over their data.

Saputra et al. [28], this study proposes AniraBlock, a dynamic smart contract system that helps solve challenges in agricultural supply chains, specifically the coffee and fish sectors. Static smart contracts (SSCs) provide a structured path for execution but no flexibility and adaptability to the dynamic changes of the supply chain. In terms of addressing these constraints, dynamic smart contracts (DSCs) with a key-value format-based framework have been proposed in this paper. They make for better adaptability and scalability, responsiveness, better data management, and visibility in the supply chain. Two types of data, qualitative and quantitative, were collected to prove the effectiveness of AniraBlock using a mixed-method approach. The preliminary results have shown improvements across the partner organizations in terms of data integrity, operational efficiency, and the ability to create transformational impact for agricultural supply chains through improved transparency, traceability, and flexibility.

Saputra et al. [29], in this paper, we propose a blockchain-based system with DSCI for managing NOAA weather and environmental data. Using the decentralization of blockchain ensures the authenticity, integrity, and secure access to the data, whereas DSCI allows dynamic configuration and real-time updates of the system. The solution tackles struggles in the fisheries sector by ensuring secure NOAA data ingestion into the blockchain to avoid data leakage and data tampering and giving fishermen access to trustworthy data for decision-making. The main

contributions are encryption algorithms, gradual integration methods, and quantitative results that indicate the efficiency and robustness of the presented system. Extensive testing from unit, integration, performance, and security tests proves considerable advantages of improved data management and better security, transparency, and operational efficiency over current solutions.

Viskars [30], these include transparency, ensuring ticket authenticity, preventing bots and ticket hoarding, and pricing. It explores a blockchain solution in the form of a decentralized app (dApp) on top of Polygon's blockchain while Ethereum can't scale. According to the research, blockchain could enhance transparency, reduce fraud, and create new revenue models for all stakeholders, including ticketing firms and musicians. Today, the bot angst remains, but nearly all the other ticketing annoyances have been addressed. The results further emphasize the confidential aspect of NFTs in customer interactions, such as valuable ads (DeFi for ahead of working capital, automated tax payments)—one of the challenges contemplated: legislative and cultural pushback and a need to educate consumers.

To explore the use of smart contracts for queue management systems in online ticketing, we performed a specific implementation and evaluation study, to showcase the potential and limitations of such blockchain-based systems. The study addresses smart contracts both in respect of their technical functionality, for instance, regarding transaction logic codification, security, and performance, as well as their practical aspects, for instance, their correlating with existing applications such as online ticketing systems. This study makes the following major contributions:

1. Smart Contract-Based Queue Management System Design and Implementation: Proposed a system and implemented it to address the queue to purchase tickets using Solidity for smart contracts to guarantee secure, clear, and rapid processing of tickets. Key technologies used in the system are Ganache (testing), MetaMask (payment handling), and Ionic (UI development) integrated into one seamless service.
2. First-In-First-Out (FIFO) Method applied: To complete the transaction, the system follows the FIFO procedure queue entries are processed in a systematic manner.
3. Simulation of Real-world Use Cases With Blockchain Tools: The research shows how tools such as Ganache and MetaMask can be integrated and help simulate real-world scenarios and enable easy payment processes.
4. User and Admin Interaction Analysis: Explaining the user and admin functionality in detail, the system can efficiently manage the ticket purchase process, transaction approval, and updating the ticket queue.

Legacy ticketing providers rely on centralized bodies, which can be difficult to scrutinize, susceptible to fraud, and require interference from other parties, resulting in inefficiencies, delays, and issues like ticket scalping. Conversely, they always use decentralized networks and smart contract technologies to empower secure, transparent, and automated exchanges of tickets without involving intermediaries. The immutable ledger means fewer instances of fraud and more trust and enables users to have full control over their transactions. While being more complicated to set up (and pricier), blockchain systems are addressing ingrained problems of fraud, inefficiency, and centralization, making them the ultimate answer to ticketing in this day and age.

The utilization of blockchain technology is not a newer thing only for e-ticketing; you can say the same thing for the other sectors mentioned above and therefore justify its sight on e-ticketing. As an example, validation by blockchain and validation in the banking industry facilitate secure and efficient cross-border payments, reduce transaction time and costs, and guarantee that transactions are transparent due to the public ledger. One of them is Ripple and Stellar (active parallel, which allows real-time settlement in international financial transactions). Moreover, the logistics industry also makes extensive use of it in supply chain management, which allows it to track goods in real time and also guarantees the authenticity of the goods at each stage. For instance, IBM and Maersk use blockchain solutions that increase the transparency of international commerce and reduce opportunities for fraud and loss. In short, insurance automates claim processing, which is conducted by smart contracts, eliminating human error risk or any other kind of error. Etherisc is one example of an insurtech startup that uses blockchain to provide insurance for flight delays. Thus, the examples illustrate the strong potential of blockchain technology and the value it can bring to various sectors. This will help readers understand the benefits of blockchain in e-ticketing of increased transparency, reduced fraud, and less convoluted operations, and thereby know how it is applicable here in all/any of these sectors. So if you want to learn something that will be fundamental and base for a lot of the technology not only for this era but for the upcoming era as well, blockchain is the right thing to learn, as it could also be customized for different purposes.

II. Method

By using blockchain technology to build an online ticket-purchasing system with the advantages of being transparent and secure. This research blockchain system consists of several main components, namely smart contracts, a frontend framework, a MetaMask wallet, and Ganache, aimed as a testing environment. Smart contracts

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

are built with the Solidity programming language, which contains functions for managing business logic, including transaction queue management and data validation. The Ionic framework is utilized to build a highly responsive and engaging user interface website, while MetaMask handles all blockchain transactions and payment confirmations. Ganache functions as a blockchain network emulator, allowing you to test the system without using a public network.

This research utilizes the FIFO algorithm that mandates the first in users' orders being fulfilled, followed by the next in the queue. This is done by first creating a queue on the smart contracts and then later appending individual users' transactions on an array. The transaction is confirmed by the admin of the system, and the status of the queue is instantaneously updated as soon as the transaction is confirmed in MetaMask. There are many stages in this implementation phase. The Ionic serves as the client side and works together with MetaMask to confirm the transactions, while the two interact with each other through a common interface. The system is tested extensively to confirm that all systems are functioning properly, including the smart contract, user interface, and its interactions with the blockchain.

In developing the SolidEye system, various tools and technologies were integrated: Solidity, which is a programming language used to develop smart contracts; Ionic, which is used for a user interface; MetaMask, which acts as a transaction manager; Ganache, a blockchain network simulation; and Visual Studio Code, a development platform. Shallow analysis is also given concerning the efficiency of the transaction time performance, effects on the transparency of the blockchain logs, and security considerations against attacks and fake transactions. Considering the system developed from this approach and its implementation, the conducted study efficiently controls ticket purchase transactions openly and dependably, thus being a significant step towards the deployment of distributed ledger technology in the ticketing industry. The following is figure 1, which is a flowchart diagram of this research.

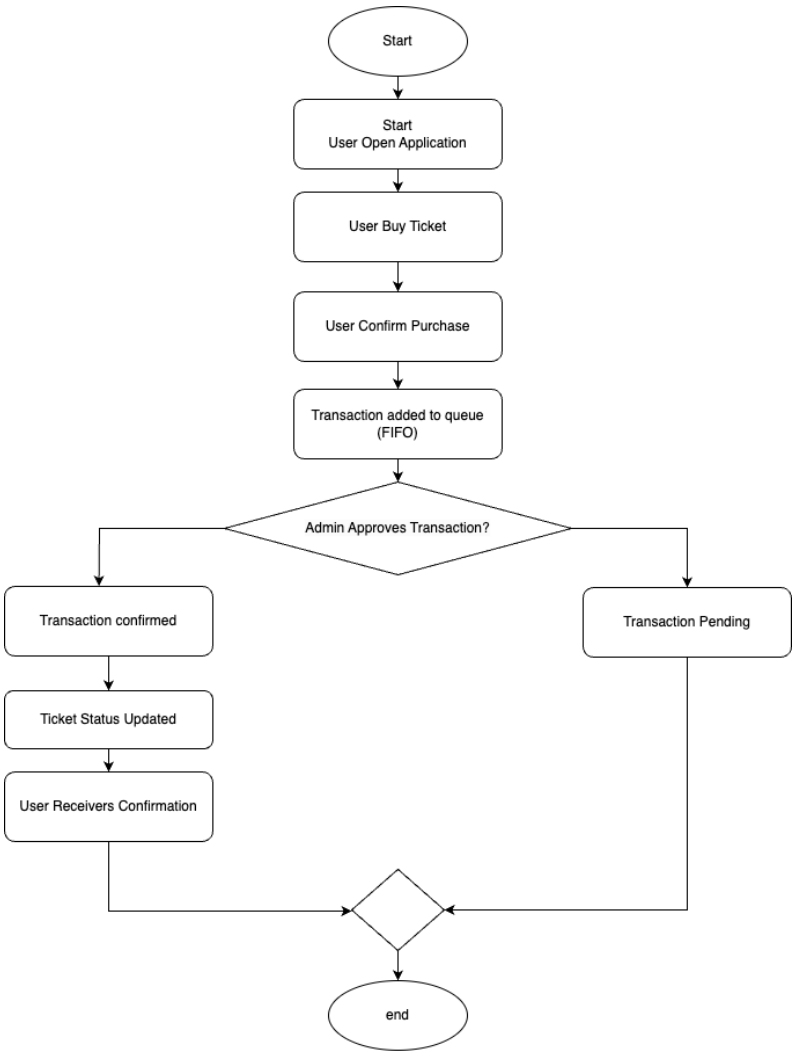

Fig. 1. Flowchart Diagram

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

In the figure 1, the diagram of the online ticket purchase system that uses blockchain technology explains how the entire system works, which starts with the user to the ticket purchasing confirmation. The process begins when a user opens the app and attempts to access the ticket-purchasing service. Then a user steps into an interoperable framework-developed ticket-buying interface and picks the ticket to purchase. Afterward, you complete the ticket purchase using Blockchain MetaMask and NFT. You will have to queue settled transactions on the smart contract level in a first-in, first-out sequence so that these tokens can be executed according to when they arrived. These gray transactions then have to be placed in the administrator queue to be approved by an administrator, which is attained using Metamask. Once the ticket has been purchased, the ticket's status will automatically be adjusted in the system to reflect that the purchase of the ticket has been completed. Finally, the app will confirm the purchase and send the ticket to the user. All the specified procedures are meant to utilize the features that the blockchain provides in terms of online ticket purchase transaction processing, which are security, efficiency, and transparency. The flowchart captures the basic process of system implementation, which combines several elements such as the use of smart contracts, MetaMask, and queuing for better usability. The following is figure 2, which is a use case diagram of this research.

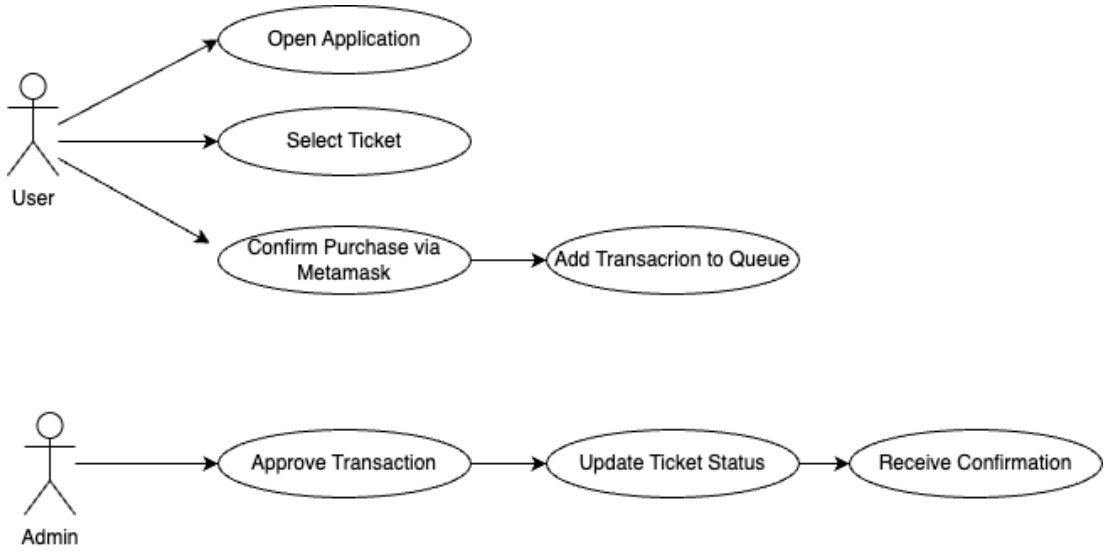

Fig. 2. Use Case Diagram

In the figure 2, the use case diagrams for the blockchain-powered online ticketing system illustrate the interaction among participants and key functionalities. It illustrates the main participants, the user and the administrator, along with their interactions with the different functionalities of the system. The user starts the process by launching the app, choosing the ticket they wish to buy, and finalizing the transaction through MetaMask, a digital wallet built into the system. After confirmation, the user's transaction will be added to the smart contract's FIFO queue for processing. The request sent through MetaMask is currently being processed to authorize the transaction that requires confirmation from the admin. The system then automatically updates the ticket status once the transaction is approved, and the user receives a ticket purchase confirmation. This diagram depicts the complete procedure of the system, highlighting clarity and effectiveness in managing ticket sales. This seeks to ensure that each use case can efficiently utilize blockchain technology to support a fundamental objective that ultimately enhances the primary aim of the system—secure transactions.

III. RESULTS AND DISCUSSION

We will discuss implementation details and the system architecture along with how we fit all of these different technologies (Ionic, Solidity, Ganache, and MetaMask) into one seamless application. This chapter describes setting up a smart contract supporting ticketing transactions, connecting the smart contract to the MetaMask wallet, and the process of purchasing tickets by users requiring administrator access. Furthermore, it helps explain the processes of transactions in the system, where blockchain technologies not only help safeguard purchases of tickets but also ensure transparency and accountability in buying from the system. The following is figure 3, which is a user interface before transaction.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

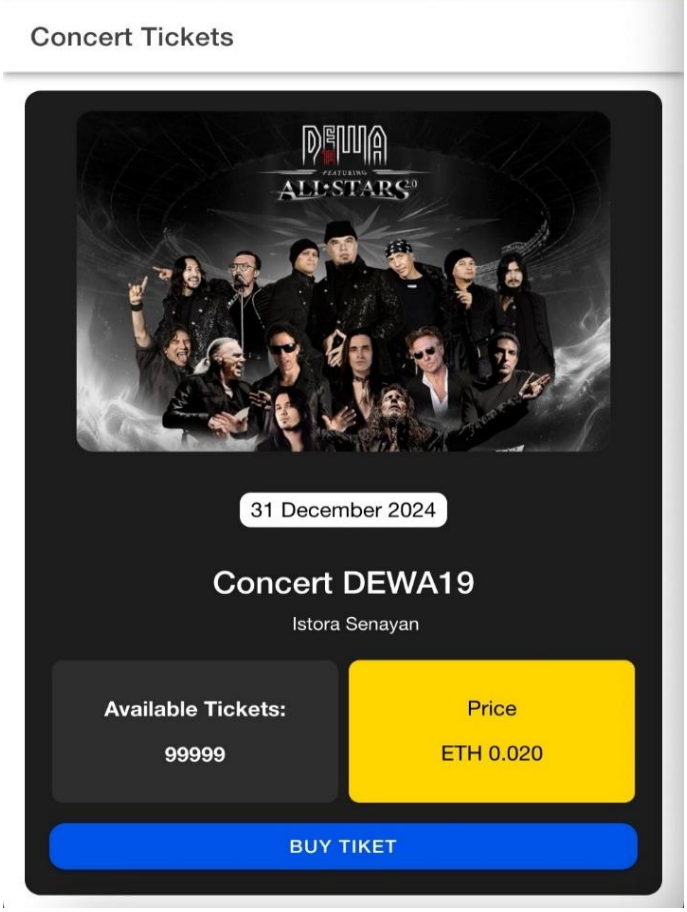

Fig. 3. User Interface Before Transaction

In the figure 3, a simple interface design is performed, and the user page is developed with the help of the Ionic framework. In the center of the page, there is a summary of the available tickets, explaining which can be purchased and their prices, and a button that states 'Buy Ticket,' which (when pressed) will navigate the user to various ticket options. There are buttons under each ticket choice, labeled 'Buy Ticket,' which the user uses to select their ticket type. This has enhanced the user-friendly and informative console graphical user interface, enabling users to see the tickets with their associated prices to be purchased via the gateway. The following is figure 4, which is a ganache to populate the MetaMask wallet.

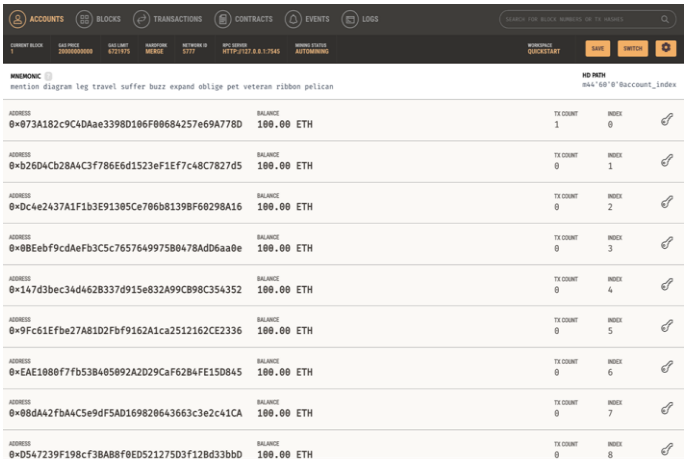

Fig. 4. Ganache to Populate The MetaMask Wallet

In the figure 4, shows the Ganache interface using fake balances in the MetaMask wallet that are used to execute transactions in the Blockchain Ticketing system. Ganache: A blockchain development tool that helps users build a private Ethereum network for testing purposes, thus saving them the trouble of using the public one and not

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

incurring real transaction fees.

The Ganache interface shows several wallet addresses that have been automatically created, each with an initial balance. Every one of them possesses a distinct address and balance that allows you to experiment with various transactions. These wallets are linked via MetaMask, which functions as a browser extension. Next, to refill the balance in the MetaMask wallet, the user just copies a wallet from MetaMask into Ganache and transfers the balance from any Ganache account to this address. The balance sent is a test balance—it's part of simulated/replay transactions, with no real money involved. Developers have the opportunity to conduct hands-on testing in a secure (controlled) environment, enabling the transaction process to seamlessly operate within the system queues before being deployed on the real blockchain.

By utilizing a Ganache view that displays the address and current balance, users can conveniently check and control the MetaMask wallet balance used for buying tickets in the application. The following is figure 5, which is a private key ID used to fill the wallet with MetaMask.

Fig. 5. Private Key ID

In figure 5, this is the section that contains the account details, that is, the account address and the private key. Account Address: The public address of your wallet used with MetaMask to send and receive transactions on the blockchain. This is the address of the wallet; like the user account in the blockchain network, it is unique to each wallet. This view also shows you the private key. Private key: This is essentially the secret key giving access to the wallet and allows the wallet to sign transactions, corresponding to the blockchain. Each wallet has an associated private key that must be kept secret because if anyone has that, they can use (access & spend) that wallet.

We now have the private key ID displayed here to fill in a MetaMask wallet at this stage. Doing so, users will launch MetaMask, select Import Wallet Using Private Key, and insert the given Private Key ID. After that, the import process ends, connecting the MetaMask wallet with the particular account and allowing its owner to execute operations such as purchasing tickets with their MetaMask wallet balance. More interestingly, thanks to this interface, users can understand and read smart contracts from their wallets along with secure top-ups and transactions by a private key in MetaMask. The following is figure 6, which is a added the private key ID from Ganache to MetaMask so that the balance is populated.

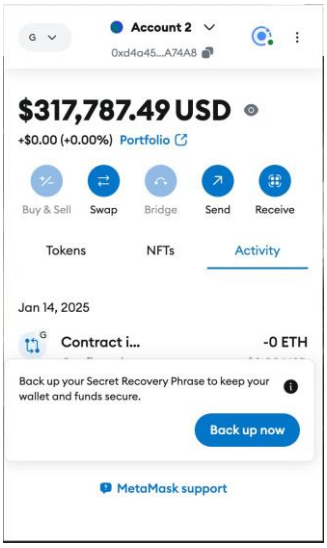

Fig. 6. Private Key ID from Ganache to MetaMask

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

In the figure 6, image demonstrates the procedure—the actions performed by an administrator to fund the MetaMask wallet (with Ganache acting as the local blockchain environment) by transferring balances and utilizing a private key from Ganache to populate the MetaMask wallet's balances. Subsequently, the administrator extracts the private key from the current account in Ganache, using the corresponding registered wallet address. The admin subsequently launches the MetaMask app chooses the 'Import Wallet' option, and imports a wallet by using the private key. Subsequently, the administrator copies the private key generated in Ganache and pastes it into MetaMask, confirming the action; now the Ganache account will appear in MetaMask. Currently, the balance visible in Ganache will appear in the MetaMask wallet that the admin uses to verify and approve user ticket transactions. This procedure guarantees that the administrators can execute transactions requiring testing with simulated balances from Ganache, ensuring that all transaction processes are accurate before being deployed on the actual blockchain network. The following is figure 7, which is a purchase tickets; confirm MetaMask for ticket purchase.

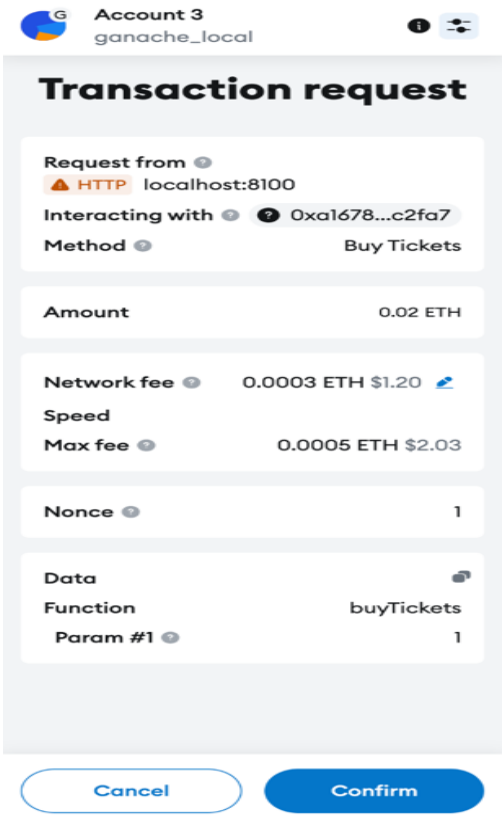

Fig. 7. Confirm MetaMask for Ticket Purchase

In the figure 7, illustration is from the ticket buying and verification process through MetaMask. Users are required to select the ticket type they want to purchase on the app interface, enter the amount of tickets, and then click the 'Buy Ticket' button to proceed. Then you will see MetaMask with a request to confirm the transaction. The user reviews (1) the purchase information (such as the ticket cost and transaction fees) and continues (2) by confirming in the MetaMask, which initiates the payment and processes the transaction on the chain. After a transaction is completed upon authorization, MetaMask serves as an additional confirmation window that indicates that the respective payment—and hence, the previously defined amount in profile settings—has been deducted from the user's wallet. This user now gets a queue number as well as a successful transaction status. This is how the process works—it also means that all ticket-buying transactions happen via a DApp wallet like MetaMask, which ensures fast, secure, and transparent transactions between the user and the venue where the user purchases his/her ticket, confirming the entire process. The following is figure 8, which is a user interface for the admin page: to view ongoing transactions, the admin can approve tickets that have entered the queue. The process that takes place is using the first-in, first-out method.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

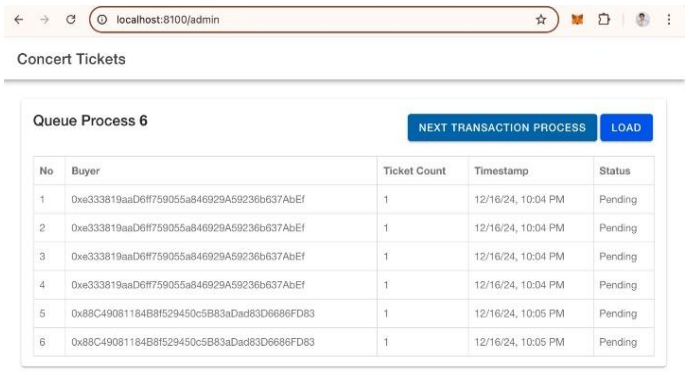

Fig. 8. User Interface for The Admin Page

In the figure 8, admin will be able to view currently active transactions and approve tickets in the queue using the user interface provided on the admin page. This blade will provide the admin with a quick way to see the transaction status for each ticket purchase. Therefore, this system works with FIFO, which is First In First Out, meaning that the earlier ticket purchased will be attempted first, and thus the next transactions will be displayed accordingly. Whenever a transaction is made for any of the stadiums, a list will show a queue number, buyer information, number of tickets purchased, time of booking, and the status of the transaction, among other details. The administrator can choose each transaction to view more detailed information, and if everything is in order, the administrator can approve the transaction. Once approved successfully, the process will start and the buyer will receive a confirmation. This helped the administrator to keep an eye on the tickets while allowing them to handle transactions fairly in a **First In First Out** manner and also allowed the administrator to effectively monitor and possibly timestamp the transactions on a blockchain. The following is figure 9, which is a display when the transaction is approved, then there is a confirmation from MetaMask.

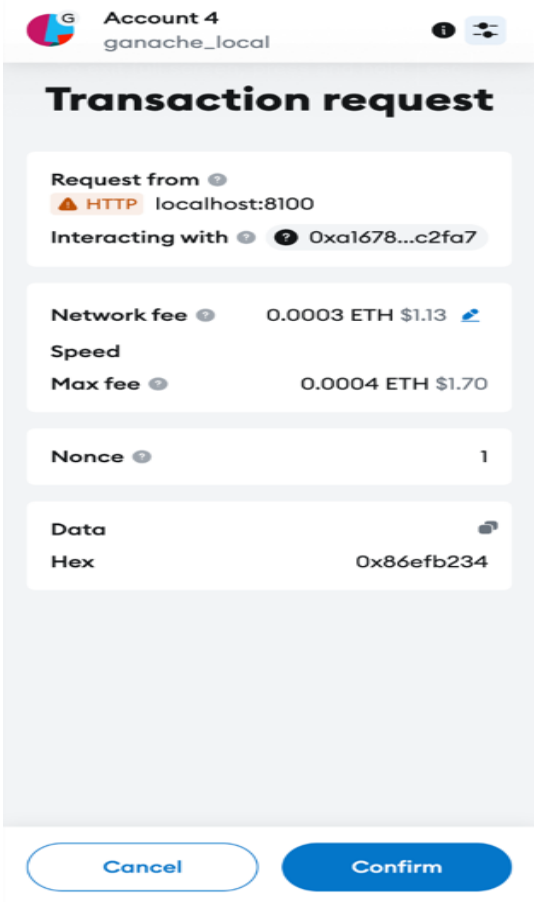

Fig. 9. Confirmation from Metamask when The Transaction is Approved

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

In the figure 9, displays the process on the screen after the admin approves the transaction and after the confirmation from MetaMask. After an admin approves a ticket transaction from the queue, MetaMask will ask the admin to approve it for the final time. It confirms the details of the transaction that are to be recorded on the blockchain, like the number of tickets purchased, fees paid, etc. Admin pop-up window for confirming the details before submitting The administrator should press the button “Confirm” in MetaMask to permit the action. After confirmation of the transaction is established, MetaMask will give you a notification showing the transaction status being successful. After the ticket is purchased, the buyer receives a notification of confirmation, and the admin's wallet is reduced in balance by the ticket purchase amount. This also means that, as a response to the second question—transaction approval and confirmation—is a secure, transparent, and compliant procedure corresponding to the regulations for the blockchain protocol. The following is figure 10, which is a Display when the transaction is successful.

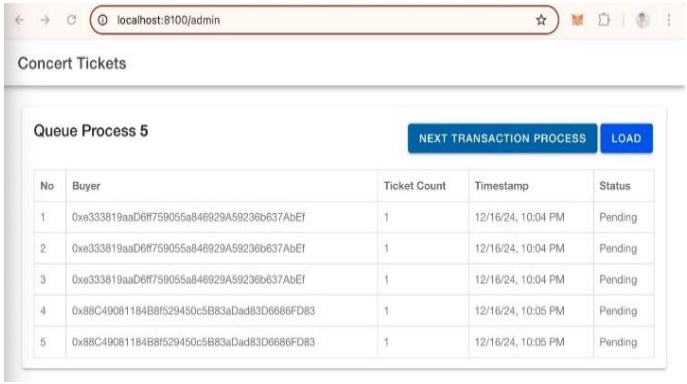

Fig. 10. Display when The Transaction is Successful on The Admin Page

In the figure 10, after getting approved via the admin approval flow, a successful transaction on the admin page After the admin approves the transaction from the queue in the admin panel and the transaction is processed via MetaMask, the successful transaction details will get refreshed in the success section in the admin page. This is a window that shows numerous columns, like No., a serial number that designates the transaction; Buyer, the name of the person who had booked the ticket; Ticket Count, which is the total tickets purchased; and Timestamp, which indicates the date and time of the ticket. Status shows the transaction, successful or not, so for the successful transactions, it shows success. Transactions are executed on a first-come, first-served (FIFO) basis. Utilizes verified data structured into successful transactions called Views to guarantee that each purchased ticket, along with any relevant details of that purchase, has been recorded. The following is figure 11, which is a user display when the transaction is successful, then the available tickets will be reduced according to the number of successful transactions.

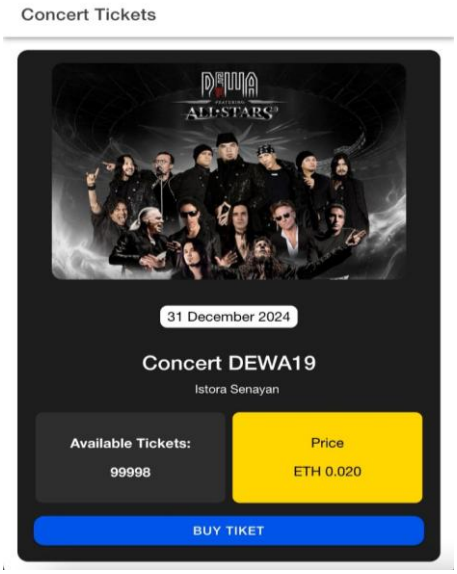

Fig. 11. User Interface After Transaction

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

In the figure 11, what the user interface will look like upon the successful completion of the transaction. If successful, the available tickets on the user page will now go down by the number of tickets purchased. This is apparent in the available ticket field, which is used to list the available ticket count. If you successfully process the ticket purchase and the status is approved, this action automatically reduces the number of tickets available, which means the ticket has been sold. This ensures that users are updated on still-available ticket supplies data after every transaction: moreover, the system also keeps transparency; the quantity of tickets booked is updated in real-time, thus enhancing user experience, and the booking of tickets should never exceed the capacity. Following figure 12, the smart contract code buy ticket.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract MyContract {
    struct Transaction {
        address buyer;
        uint256 ticketCount;
        uint256 timestamp;
        bool processed;
    }

    uint256 public availableTickets = 99999;
    uint256 public ticketPrice = 0.02 ether;
    uint256 public maxTicketsPerTransaction = 100;

    Transaction[] public transactionQueue;

    event TicketQueued(address buyer, uint256 ticketCount);
    event TicketProcessed(address buyer, uint256 ticketCount);


    function buyTickets(uint256 ticketCount) public payable {
        require(ticketCount > 0, "Ticket count must be greater than 0");
        require(ticketCount <= maxTicketsPerTransaction, "Cannot buy more than 100 tickets");
        require(msg.value >= ticketCount * ticketPrice, "Insufficient funds");
        require(availableTickets >= ticketCount, "Not enough tickets available");


        transactionQueue.push(Transaction({
            buyer: msg.sender,
            ticketCount: ticketCount,
            timestamp: block.timestamp,
            processed: false
        }));

        emit TicketQueued(msg.sender, ticketCount);
    }
```

Fig. 12. Smart Contract Code Buy Ticket

In the figure 12, buyTickets(uint256 ticketCount) function to buy tickets and also validate some conditions: the number of tickets must be greater than 0, the number of tickets cannot exceed 100 tickets per transaction, the funds sent are sufficient to purchase the tickets, the available tickets are sufficient, the transaction is added to the queue (transactionQueue), and the TicketQueued event is fired to record the purchase. Following figure 13, the smart contract code next transaction.

```
    function processNextTransaction() public {
        require(transactionQueue.length > 0, "No transactions in queue");


        Transaction storage nextTransaction = transactionQueue[0];
        require(!nextTransaction.processed, "Transaction already processed");


        require(availableTickets >= nextTransaction.ticketCount, "Not enough tickets available");
        availableTickets -= nextTransaction.ticketCount;
        nextTransaction.processed = true;

        emit TicketProcessed(nextTransaction.buyer, nextTransaction.ticketCount);
```

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 10, No. 4, December 2025, Pp. 4201-4215

Fig. 13. Smart Contract Code Next Transaction

In the figure 13, processNextTransaction(): processes the first transaction in the queue, ensures there is a transaction in the queue, ensures the transaction has not been processed before, decreases the number of available tickets, marks the transaction as processed, raises the TicketProcessed event, and removes the transaction from the queue. Following figure 14, the smart contract code remove transaction.

```
    function removeTransaction(uint256 index) internal {
        require(index < transactionQueue.length, "Index out of bounds");

        for (uint256 i = index; i < transactionQueue.length - 1; i++) {
            transactionQueue[i] = transactionQueue[i + 1];
        }
        transactionQueue.pop();
    }


    function getQueueLength() public view returns (uint256) {
        return transactionQueue.length;
    }


    function getQueue() public view returns (Transaction[] memory) {
        return transactionQueue;
    }


    function getTransaction(uint256 index) public view returns (address, uint256, uint256, bool) {
        require(index < transactionQueue.length, "Index out of bounds");

        Transaction memory txn = transactionQueue[index];
        return (txn.buyer, txn.ticketCount, txn.timestamp, txn.processed);
    }
}
```

Fig. 14. Smart Contract Code Remove Transaction

In the figure 14, removeTransaction(uint256 index) (internal): removes the transaction from the queue at the specified index, moving elements behind that index and reducing the length of the array. Then, getQueueLength(): returns the total number of transactions in the queue. After that, getQueue(): returns the entire array of transactions in the queue. Next, getTransaction(uint256 index): returns the transaction details at the specified index and includes buyer information, ticket amount, timestamp, and process status.

IV. CONCLUSION

The research analyzed in this paper contributes to finding more efficient and safe ways to administer online ticketing transactions by proposing a blockchain-based online ticketing system. The system was developed using the tools Solidity, Ionic, Ganache, and Metamask, and it digitalizes the whole process of buying a ticket, from waiting in the queue to making a purchase. Every proposal has to be accepted in the precise sequence in which it was made thanks to the FIFO process. The findings of the experiment demonstrated that the system can minimize the chances of errors and misconduct while at the same time enhancing the ease of use for both customers and administrators. In contrast to the other papers, this research is distinctive in that it focuses on the seamless integration of smart contracts and blockchain-based ticketing systems with modern web application cashier interfaces based on the Ionic framework. Also, the combination of the FIFO mechanism and instant approval of purchases via MetaMask ensures a level uplift that is more systematized for the selling of purchase tickets online.

Where many older tokenized systems (Aventus, GET Protocol) were focused on solving crime (show-related fraud, scalpers, etc.) with tokenization and proof of circulation, our solution took the dynamics of queueing and token redemption into consideration using a FIFO methodology. Unlike other systems that focus on resale restrictions and dynamic pricing, our solution utilizes smart contracts to optimize transaction speed and certainty of access to tickets. Also, by deploying the system with Ganache as a testing environment and using MetaMask for secure transactions, we can get more users. This also reduces the case for making more calls for the security to work, which is desirable for a high-volume event with a low tolerance for delay/error.

Our work also extends earlier efforts that examine blockchain applications in ticketing, for instance, those that emphasize fraud mitigation, open ticket information, and secure transfers. While previous methods have focused on ticket authenticity and problems with the secondary market, our work adds a smart contract-driven queue

management system to increase transaction throughput and fairness. Using the FIFO method, as well as MetaMask and Ganache, our system generates instant and secure ticket allocation. Overcoming traditional problems such as fraud or other application-related concerns is manageable in this approach and results in lesser queue times in high-demand situations, thus greatly assisting blockchain applications in ticketing systems with optimized queue management.

REFERENCES

[1] A. Aldweesh, "BlockTicket: A framework for electronic tickets based on smart contract," *PLoS One*, vol. 18, no. 4 April, pp. 1–20, 2023, doi: 10.1371/journal.pone.0284166.

[2] J. G. Song, E. S. Kang, H. W. Shin, and J. W. Jang, "A smart contract-based p2p energy trading system with dynamic pricing on ethereum blockchain," *Sensors*, vol. 21, no. 6, pp. 1–27, 2021, doi: 10.3390/s21061985.

[3] M. An *et al.*, "Blockchain Technology Research and Application: A Literature Review and Future Trends," *J. Data Sci. Intell. Syst.*, no. 2020, pp. 1–16, 2023, doi: 10.47852/bonviewjdsis32021403.

[4] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 2901–2925, 2021, doi: 10.1007/s12083-021-01127-0.

[5] S. Y. Lin, L. Zhang, J. Li, L. li Ji, and Y. Sun, *A survey of application research based on blockchain smart contract*, vol. 28, no. 2. 2022. doi: 10.1007/s11276-021-02874-x.

[6] T. Huynh-The *et al.*, "Blockchain for the metaverse: A Review," *Futur. Gener. Comput. Syst.*, vol. 143, pp. 401–419, 2023, doi: 10.1016/j.future.2023.02.008.

[7] M. Attaran, "Blockchain technology in healthcare: Challenges and opportunities," *Int. J. Healthc. Manag.*, vol. 15, no. 1, pp. 70–83, 2022, doi: 10.1080/20479700.2020.1843887.

[8] A. Haleem, M. Javaid, R. P. Singh, R. Suman, and S. Rab, "Blockchain technology applications in healthcare: An overview," *Int. J. Intell. Networks*, vol. 2, no. May, pp. 130–139, 2021, doi: 10.1016/j.ijin.2021.09.005.

[9] M. Javaid, A. Haleem, R. P. Singh, R. Suman, and S. Khan, "A review of Blockchain Technology applications for financial services," *BenchCouncil Trans. Benchmarks, Stand. Eval.*, vol. 2, no. 3, p. 100073, 2022, doi: 10.1016/j.tbench.2022.100073.

[10] F. A. Sunny *et al.*, "A Systematic Review of Blockchain Applications," *IEEE Access*, vol. 10, pp. 59155–59177, 2022, doi: 10.1109/ACCESS.2022.3179690.

[11] S. U. Ahmed, A. Danish, N. Ahmad, and T. Ahmad, "Smart Contract Generation through NLP and Blockchain for Legal Documents," *Procedia Comput. Sci.*, vol. 235, pp. 2529–2537, 2024, doi: 10.1016/j.procs.2024.04.238.

[12] K. Bułkowska, M. Zielińska, and M. Bułkowski, "Implementation of Blockchain Technology in Waste Management," *Energies*, vol. 16, no. 23, 2023, doi: 10.3390/en16237742.

[13] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain Res. Appl.*, vol. 3, no. 2, p. 100067, 2022, doi: 10.1016/j.bcra.2022.100067.

[14] L. Ante, "Smart contracts on the blockchain – A bibliometric analysis and review," *Telemat. Informatics*, vol. 57, no. 10, pp. 1–48, 2021, doi: 10.1016/j.tele.2020.101519.

[15] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H. N. Lee, "Ethereum Smart Contract Analysis Tools: A Systematic Review," *IEEE Access*, vol. 10, pp. 57037–57062, 2022, doi: 10.1109/ACCESS.2022.3169902.

[16] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, "Software security patch management - A systematic literature review of challenges, approaches, tools and practices," *Inf. Softw. Technol.*, vol. 144, 2022, doi: 10.1016/j.infsof.2021.106771.

[17] D. Kirli *et al.*, "Smart contracts in energy systems: A systematic review of fundamental approaches and implementations," *Renew. Sustain. Energy Rev.*, vol. 158, no. January, p. 112013, 2022, doi: 10.1016/j.rser.2021.112013.

[18] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H. N. Lee, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," *IEEE Access*, vol. 10, pp. 6605–6621, 2022, doi: 10.1109/ACCESS.2021.3140091.

[19] F. Ullah and F. Al-Turjman, "A conceptual framework for blockchain smart contract adoption to manage real estate deals in smart cities," *Neural Comput. Appl.*, vol. 35, no. 7, pp. 5033–5054, 2023, doi: 10.1007/s00521-021-05800-6.

[20] T. H. Pranto, A. A. Noman, A. Mahmud, and A. B. Haque, "Blockchain and smart contract for IoT enabled smart agriculture," *PeerJ Comput. Sci.*, vol. 7, pp. 1–29, 2021, doi: 10.7717/PEERJ-CS.407.

[21] S. Dustdar, P. Fernandez, J. M. Garcia, and A. Ruiz-Cortes, "Elastic Smart Contracts in Blockchains," *IEEE/CAA J. Autom. Sin.*, vol. 8, no. 12, pp. 1901–1912, 2021, doi: 10.1109/JAS.2021.1004222.

[22] K. Bedarkar, M. Vardishvili, S. Bozhko, M. Maida, and B. B. Brandenburg, "From Intuition to Coq: A Case Study in Verified Response-Time Analysis 1 of FIFO Scheduling," *Proc. - Real-Time Syst. Symp.*, vol. 2022-Decem, pp. 197–210, 2022, doi: 10.1109/RTSS55097.2022.00026.

[23] J. D. Preece, C. R. Morris, and J. M. Easton, "Leveraging ontochains for distributed public transit ticketing: An investigation with the system for ticketing ubiquity with blockchains," *IET Blockchain*, no. June, 2024, doi: 10.1049/blc2.12071.

[24] M. T. Systems, M. Leinweber, N. Kannengießer, H. Hartenstein, and A. Sunyaev, *Towards the New Normal in Mobility*, no. July. Springer Fachmedien Wiesbaden, 2023. doi: 10.1007/978-3-658-39438-7.

[25] T. M. Grønli, A. Lakhan, and S. Memon, "A Novel BIBO Automated Ticketing System Based on Blockchain Mobile Sensors for Public Transport Modes," *IEEE Veh. Technol. Conf.*, no. September, pp. 1–5, 2024, doi: 10.1109/VTC2024-Spring62846.2024.10683170.

[26] S. Rafati Niya, S. Bachmann, C. Brasser, M. Bucher, N. Spielmann, and B. Stiller, *DeTi: A Decentralized Ticketing Management Platform*, vol. 30, no. 4. Springer US, 2022. doi: 10.1007/s10922-022-09675-3.

[27] S. Feulner, J. Sedlmeir, V. Schlatt, and N. Urbach, "Exploring the use of self-sovereign identity for event ticketing systems," *Electron. Mark.*, vol. 32, no. 3, pp. 1759–1777, 2022, doi: 10.1007/s12525-022-00573-9.

[28] I. Saputra, Y. Arkeman, I. Jaya, I. Hermadi, N. A. Akbar, and I. Sutedja, "AniraBlock: A leap towards dynamic smart contracts in agriculture using blockchain based key-value format framework," *Commun. Sci. Technol.*, vol. 8, no. 2, pp. 154–163, 2023, doi: 10.21924/cst.8.2.2023.1240.

[29] I. Saputra *et al.*, "Enhancing Environmental Data Management through Dynamic Smart Contract Interaction on Blockchain," *Proceeding - 2024 Int. Conf. Inf. Technol. Res. Innov. ICITRI 2024*, pp. 341–346, 2024, doi: 10.1109/ICITRI62858.2024.10698936.

[30] G. Viskars Fabrizio Traversa, "Blockchain technology as an innovative system to tackle major issues in concert ticketing Leisure Management Management and Organisation Project," Universitaria Professionale della Svizzera Italiana (SUPSI), 2023.