Ontix: Blockchain-Based NFT Decentralized e-Ticketing Development

Elizabeth Nathania Witanto[1], Christopher Andreas[2*], Rudi Limantara[3], Louis Fernando[4], Lie Samuel Miracle Kristanto[5], Richie Reuben Hermanto[6]

[1,2,3,4,5,6]Informatics, School of Information Technology, Universitas Ciputra, Surabaya, East Java, Indonesia
E-mail: [1]elizabeth.nathania@ciputra.ac.id, [2*]christopher.andreas@ciputra.ac.id, [3]rudi.limantara@ciputra.ac.id, [4]lfernando@student.ciputra.ac.id, [5]lsamuel01@student.ciputra.ac.id, [6]rhermanto01@student.ciputra.ac.id

(Received: 5 Dec 2025, revised: 2 Mar 2026, accepted: 3 Mar 2026)

Abstract

Event ticketing systems, such as concerts, festivals, and sports matches, face persistent challenges, including ticket forgery, duplication, resale manipulation, and fraud in secondary markets. Centralized electronic ticketing systems, while digitized, remain vulnerable to identity theft, seller unaccountability, and unfair distribution due to their reliance on intermediaries and a single point of failure. To address these issues, this research introduces Ontix, a decentralized blockchain-based e-ticketing platform utilizing Non-Fungible Tokens (NFTs) compliant with the ERC-721 standard. By leveraging blockchain's immutability, transparency, and decentralization, Ontix ensures verifiable ownership, tamper-proof ticket issuance, and automated transactions through smart contracts. The system enforces anti-scalping measures, including resale time and price limits, while enabling real-time QR-based validation directly linked to smart contracts. Ontix integrates Layer-2 Optimism Sepolia for scalability and lower gas fees, and employs the InterPlanetary File System (IPFS) via Pinata for decentralized metadata storage, alongside Cloudinary for media management. This hybrid architecture guarantees transparency, security, and operational efficiency. By eliminating intermediaries and automating ticket lifecycle management, Ontix provides an accountable, tamper-resistant, and low-cost e-ticketing ecosystem, as well as a user-centric ticketing ecosystem, representing a significant advancement toward the future of decentralized event management.

Keywords: Blockchain, Decentralized, ERC-721, E-ticketing, NFT, Smart Contract

I. INTRODUCTION

Various event ticketing industries, including concerts, festivals, and sports matches, continue to face significant challenges that negatively impact both organizers and consumers. Traditional ticketing platforms remain vulnerable to forgery, duplication, and resale manipulation, while the proliferation of unauthorized secondary markets enables scalping, fraud, and inflated pricing, eroding trust among both buyers and organizers. Moving from traditional ticketing to electronic ticketing also inherits similar issues. One of the main issues is the widespread problem of ticket forgery, where counterfeit tickets are sold at prices that are lower than the official ones or way too high. This situation often results in financial losses for buyers [1]. Centralized ticketing platforms are especially vulnerable to manipulation and often serve as a single point of failure, creating an unfair distribution environment [2], [3]. Additional issues, such as identity fraud during entry verification and seller unaccountability, further expose weakness in current event ticketing systems, underscoring the urgent need for a transparent and tamper-proof alternative solution [4].

To overcome these limitations, blockchain technology has emerged as a promising solution due to its decentralized, transparent, and immutable nature. These characteristics allow blockchain systems to prevent single points of failure and promote fairness in ticket distribution [5]. Blockchain enables permanent, tamper-proof record-keeping of transactions, helping mitigate fraud and ensuring accountability in the ticketing process [6]. Within the context of digital ticketing, each ticket can be represented as a unique digital asset through Non-Fungible Tokens (NFTs) built on the ERC-721 standard, ensuring authenticity and verifiable ownership [7]. By leveraging blockchain's immutability, NFT-based tickets prevent duplication and forgery and ensure a transparent ownership history that can be publicly verified [8]. Additionally, NFT tickets enable real-time validation via smart contract mechanisms, eliminating intermediaries in ticket sales and transfers [9]. This transformation enhances both security and transparency in the ticketing process, providing equitable

DOI: 10.34148/teknika.v15i1.1432

access for consumers and ensuring fair compensation for organizers.

The integration of smart contracts in blockchain-based ticketing systems automates every stage of the ticket lifecycle, including creation, purchase, validation, and resale [9]. Each transaction is permanently recorded on the blockchain ledger, ensuring data integrity and auditability [10]. Smart contracts can also enforce specific conditions, such as resale price caps and time limits, to prevent ticket scalping and speculative trading [11]. These automated enforcement mechanisms contribute to a more ethical and efficient secondary market for event tickets [4]. Furthermore, NFTs add collectible and engagement value to digital tickets. They can serve not only as proof of entry but also as long-term digital memorabilia that enhances fan interaction and loyalty. NFT-based ticketing systems can also distribute royalties to event organizers or artists automatically during resale, creating a sustainable revenue model across the ticket's lifecycle [12].

In terms of efficiency, blockchain ticketing platforms built on Layer-2 networks such as Optimism or Avalanche have demonstrated lower gas fees and faster transaction confirmation times than the Ethereum mainnet, without compromising transparency. This scalability improvement enables broader adoption across high-volume industries, from entertainment to transportation [9]. However, several barriers remain in widespread implementation, such as user onboarding complexity, interoperability between NFT standards, and the absence of universal frameworks for multi-event ticketing [13]. To address these challenges, this research introduces Ontix, a fully decentralized NFT-based e-ticketing platform built on the Ethereum blockchain. Ontix utilizes ERC-721 tokens for each ticket, ensuring that ownership is unique, verifiable, and transferable in a secure and transparent manner [14]. All business logic—including event creation, purchase, and resale—is governed by immutable smart contracts, eliminating reliance on third-party intermediaries. The system enforces anti-scalping measures by imposing resale price limits and time restrictions on resales. Ticket validation is conducted in real time using QR code verification linked directly to blockchain smart contracts, ensuring instant and tamper-proof authentication [13], [15]. The novelty of Ontix lies in its efficient and scalable system design, achieved through the integration of the Optimism Sepolia Layer-2 Ethereum network, which significantly lowers transaction costs (gas fees) compared to the Ethereum mainnet. Furthermore, Ontix incorporates the InterPlanetary File System (IPFS) via Pinata for decentralized NFT metadata storage and employs Cloudinary for media management. This hybrid architecture guarantees data integrity, decentralization, and resilience against central server failures. In doing so, Ontix represents a next-generation blockchain-based ticketing solution that effectively addresses fraud, scalability, and transparency issues, setting a new standard for the future of decentralized event management.

II. RESEARCH METHODOLOGY

A. Blockchain

Blockchain is a decentralized ledger that stores transaction records in interconnected blocks, secured by cryptography. Each block contains transaction data, a timestamp, and a hash of the previous block, forming a chain of data that cannot be changed without the network's approval [16]. Blockchain has some characteristics that make it trustworthy. First, decentralized, with no central authority that controls the data. All participants in the blockchain network have the same copy of the data, making it hard to manipulate. Second, transparent and accountable. All participants in the blockchain network can track and verify data publicly without revealing personal identities. Lastly, security and immutability. Data in the blockchain is secured by leveraging hash algorithms and consensus mechanisms (i.e., Proof of Work, Proof of Stake), which makes changing data nearly impossible without gaining control of a large number of the network [17]. With those factors combined, blockchain creates a trust system without the need for a third party.

B. Non-Fungible Token (NFT)

Non-Fungible Token (NFT) is a unique digital asset that is stored on the blockchain and represents ownership of a digital of physical assets [18]. Unlike crypto assets like Bitcoin, which are fungible (can be exchanged for one another), each NFT has a unique identity and value and cannot be replaced. NFTs work with smart contract technology that guarantees the authenticity, ownership, and transparency of every transaction made on the blockchain [19]. Since NFTs are built on top of blockchain, they inherit its advantages, such as transparency, trustworthiness, and reliability.

C. Smart Contracts and ERC-721

A smart contract is a program stored in the blockchain and executed automatically when predetermined conditions are met, the results are deterministic [20]. Smart contracts typically run on Ethereum, one of the blockchain platforms. It enables transactions and agreements to occur directly between two parties without the intervention of an intermediary. Every instruction in a smart contract is transparent, immutable once published to the blockchain network, and will be executed according to the code written, thus ensuring the reliability of transactions [21]. For example, in a blockchain-based payment system, a smart contract can regulate the transfer of funds only after the recipient confirms receipt of the goods. No single party can manipulate the outcome of the transaction because the entire process is executed on the blockchain. Smart contracts are crucial to blockchain systems, serving as the foundation for various decentralized applications (dApps), decentralized finance (DeFi), and asset tokenization, such as NFTs. By utilizing smart contracts, blockchain can function beyond simply recording transactions to also automate complex business processes securely and transparently.

One important implementation of smart contracts is ERC-721 (Ethereum Request for Comments 721), a token standard procedure on the Ethereum network used to create and manage

DOI: 10.34148/teknika.v15i1.1432

NFTs, which provides an API for tokens within smart contracts [22]. Unlike ERC-20, which is used for fungible tokens such as cryptocurrencies, each ERC-721 token is unique and has distinct metadata, making it suitable for representing digital assets such as artwork, collectibles, access keys, or digital tickets [22], [23]. ERC-721 uses smart contracts to automatically and transparently manage token minting, transfers, and token ownership validation, as shown in Algorithm 1.

Algorithm 1 Methods in ERC-721 [22]

```
1. function balanceOf(address _owner)
   external view returns (uint256);
2. function ownerOf(uint256 _tokenId)
   external view returns (address);
3. function safeTransferFrom(address
   _from, address _to, uint256
   _tokenId, bytes data) external
   payable;
4. function safeTransferFrom(address
   _from, address _to, uint256
   _tokenId) external payable;
5. function transferFrom(address _from,
   address _to, uint256 _tokenId)
   external payable;
6. function approve(address _approved,
   uint256 _tokenId) external payable;
7. function setApprovalForAll(address
   _operator, bool _approved) external;
8. function getApproved(uint256
   _tokenId) external view returns
   (address);
9. function isApprovedForAll(address
   _owner, address _operator) external
   view returns (bool);
```

D. Proposed System Architecture

Ontix was developed in a full-stack environment that combines web technologies and blockchain developer tools. The frontend user interface was built using React (JavaScript) and Tailwind CSS to provide an interactive system for users, as depicted on Figure *1*, while EtherJS is used to communicate with the blockchain layer via Hardhat. The backend server was implemented using NodeJS with the Express framework to handle routing from the frontend to the backend layer, including blockchain middleware logic. Smart contracts were written in Solidity and compiled and tested using the Hardhat environment on a local blockchain network. These contracts defined the core NFT ticketing logic based on the ERC-721 token standard, including, but not limited to, event creation, ticket transactions and token transfers, ticket validation, and ticket transfers. Furthermore, off-chain data, including users' data for creator/event organizers and buyers, was stored in MongoDB. These data are used for the user's authentication, so they will not be stored on-chain.

For the wallet connectivity and transaction signing, Ontix used Reown as the primary wallet provider. While NFT metadata and associated media files, such as QR-based NFT tickets, were uploaded to Pinata, which served as the IPFS-based decentralized storage layer. In addition, Cloudinary was integrated to support optimized cloud-based media management for assets.

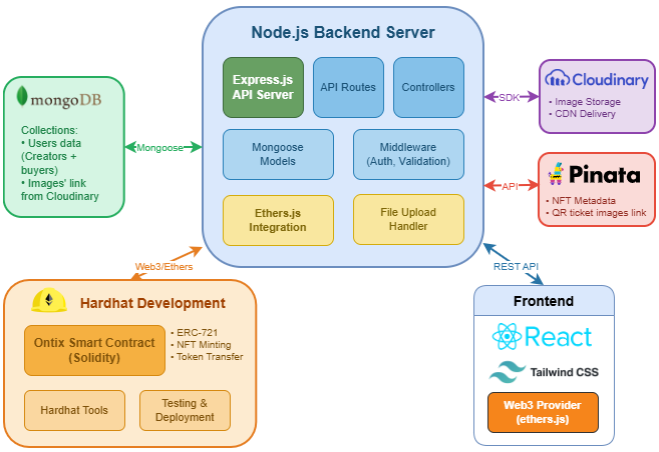

Figure 1. System Architecture

III. RESULTS AND DISCUSSION

A. Results

Ontix offers a decentralized NFT-based ticketing system that is built on top of the Ethereum network. All development activities, including frontend testing, blockchain network testing, and API development, were performed on an Apple M4 Pro workstation equipped with a 14-core CPU, a 20-core GPU, 24 GB of RAM, and a 1 TB SSD, running macOS Sequoia (version 15.3.1) as the host operating system. There are seven features that accommodate user interactions. Each feature contains a modifier to enforce a set of rules. The detailed features for each user's role are written as follows:

a. Event Organizers (EOs)
- Create Event
  - Event organizers use this feature to create a new event and mint the tickets as NFTs. They need to input the event's details, such as time, price, maximum number of tickets, maximum resale time, and maximum resale price, as shown in Figure 2. To enforce anti-scalping measures, EO must set a maximum resale price in advance, so that if users want to resell their ticket, the price cannot exceed that maximum. In addition, EO must define the maximum resale period to prevent resale ticket fraud outside the event window.
- Close Event
  - When the event ended, EO used this feature to withdraw the money from the smart contract to their wallet.

DOI: 10.34148/teknika.v15i1.1432

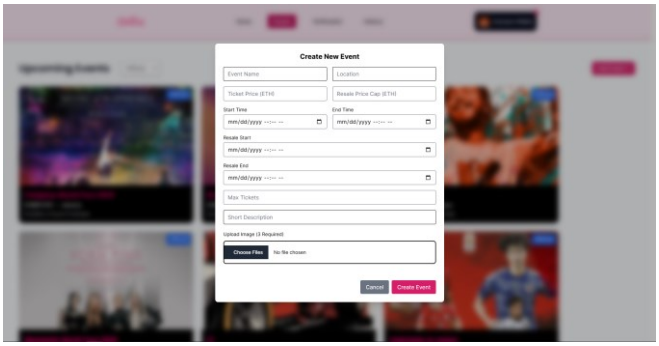

Figure 2. Ontix Create Event by Event Organizer

- Validate Ticket
  - This feature is used to verify the authenticity of the ticket presented by the buyer at the time of the event for registration. It ensures the ticket cannot be reused, preventing duplicate entries and providing real-time ticket validation. The ticket validation is done through the QR-based NFT ticket, as shown in Figure 3 and the user's detailed information is shown in Figure 4.

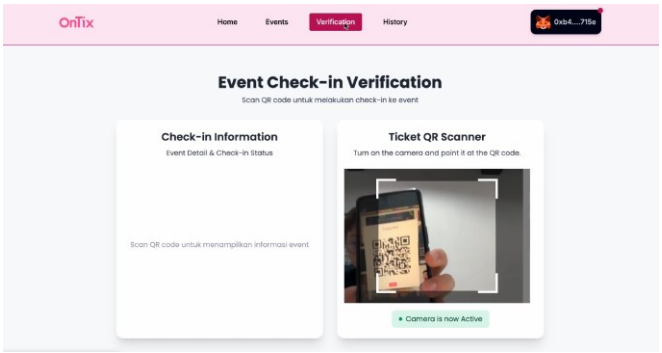

Figure 3. QR-based User Event Check-in Verification

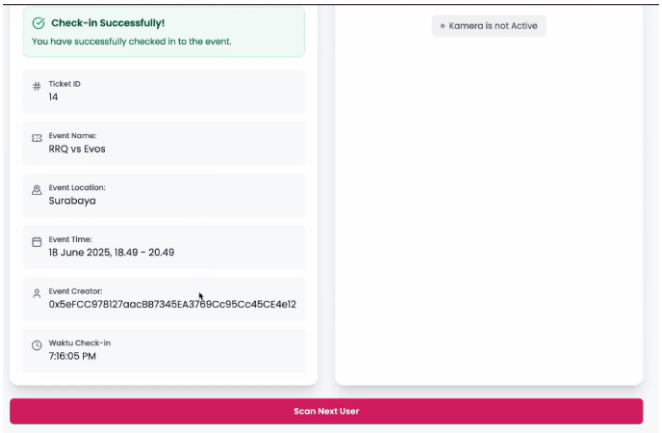

Figure 4. User Check-in Detail

b. Buyers
- Buy Ticket
  - Buyers use this feature to buy a ticket from an event created by the EO within the time range of the event, and depending on the ticket's availability. This function will revert if buyers purchase a ticket after the event sale period, if the ticket is already sold out, or if the buyer's payment amount does not match. The details are shown in Algorithm 2.

Algorithm 2 Buy Ticket

```
1. procedure buyTickets(eventId,
   quantity):
2.     evt ← events[eventId]
3.     revert if current_time >
       evt.resaleEnd
4.     revert if evt.ticketsSold +
       quantity > evt.maxTickets
5.     revert if payment_received ≠
       evt.ticketPrice * quantity
6.     ticketsAssigned ← 0
7.     FOR ticketId FROM 0 TO
       nextTicketId - 1:
8.         IF ticketsAssigned = quantity
           THEN
9.             BREAK
10.        END IF
11.        IF ownerOf(ticketId) =
           CONTRACT_ADDRESS AND
           hash(ticketMetadata[ticketId]
           .eventId) = hash(eventId)
           THEN
12.            TRANSFER(CONTRACT_ADDRESS,
               buyer_address, ticketId)
13.            EMIT
               TicketPurchased(ticketId,
               buyer_address)
14.            ticketsAssigned ←
               ticketsAssigned + 1
15.        END IF
16.    END FOR
17.    revert if ticketsAssigned ≠
       quantity
18.    evt.ticketsSold ← evt.ticketsSold
       + quantity
19.    eventProceeds[eventId] ←
       eventProceeds[eventId] +
       payment_received
```

- Resale Ticket
  - This feature allows buyers to resell their tickets to the system within the range that has been set by the EO. It ensures that the ticket has never been resold before and is sold within the resale time set by the EO. Furthermore, the resale price cannot exceed the maximum resale price set to prevent scalping. The details are shown in Algorithm 3.

Algorithm 3 Resale Ticket

```
1. procedure listForResale(ticketId,
   price):
2.     revert if caller ≠
       ownerOf(ticketId)
```

DOI: 10.34148/teknika.v15i1.1432

```
3.     revert if
       ticketMetadata[ticketId].isResol
       d = TRUE
4.     eventId ←
       ticketMetadata[ticketId].eventId
5.     revert if current_time >
       events[eventId].resaleEnd
6.     revert if price >
       events[eventId].maxResalePrice
7.     resalePrice[ticketId] ← price
8.     resaleSeller[ticketId] ← caller
9.   EMIT_EVENT("TicketListedForResale",
       ticketId, price)
```

- Buy Resale Ticket
  - This feature allows buyers to buy resale tickets from the system. Once a ticket is purchased through this feature, it will be marked as "resold". The details are shown in Algorithm 4.

Algorithm 4 Buy Resale Ticket

```
1. procedure
   buyResaleTickets(ticketIds):
2.     totalPrice ← 0
3.     FOR i FROM 0 TO LENGTH(ticketIds)
       - 1 DO:
4.         ticketId ← ticketIds[i]
5.         eventId ←
           ticketMetadata[ticketId].even
           tId
6.         revert if current_time >
           events[eventId].resaleEnd
7.         revert if
          ticketMetadata[ticketId].isRes
          old = TRUE
8.          seller ←
             resaleSeller[ticketId]
9.         revert if seller =
           NULL_ADDRESS
10.         totalPrice ← totalPrice +
           resalePrice[ticketId]
11.     END FOR
12.     revert if amount_paid ≠
        totalPrice
13.     FOR i FROM 0 TO
        LENGTH(ticketIds) - 1 DO:
14.      ticketId ← ticketIds[i]
15.      seller ← resaleSeller[ticketId]
16.      price ← resalePrice[ticketId]
17.      TRANSFER_TICKET(from = seller,
          to = buyer, ticketId)
18.      SEND_FUNDS(to = seller, amount
          = price)
19.    ticketMetadata[ticketId].isResold
       ← TRUE
20.      resalePrice[ticketId] ← 0
21.      resaleSeller[ticketId] ←
            NULL_ADDRESS
22.      EMIT_EVENT("TicketResold",
        ticketId, seller, buyer)
23.     END FOR
```

- Transfer Ticket
  - Buyers are eligible to transfer their ticket to others directly outside the market. This feature can only be used once; after the ticket is transferred to the new owner, it will be marked as "resold". Before the transaction, the function will check that the ticket has never been resold and that the transaction occurs within the resale period. The details are shown in Algorithm 5.

Algorithm 5 Transfer Ticket

```
1. procedure transferTickets(to,
   ticketIds):
2.     FOR i FROM 0 TO LENGTH(ticketIds)
       - 1 DO:
3.         ticketId ← ticketIds[i]
4.         revert if ownerOf(ticketId) ≠
           caller
5.         revert if
           ticketMetadata[ticketId].isRe
           sold = TRUE
6.         eventId ←
           ticketMetadata[ticketId].even
           tId
7.         revert if current_time >
           events[eventId].resaleEnd
8.         TRANSFER_TICKET(from =
           caller, to = to, ticketId)
9.     ticketMetadata[ticketId].isResold
       ← TRUE
10.      EMIT_EVENT("TicketTransferred",
          ticketId, caller, to)
11.     END FOR
```

B. Discussion

1) Gas Cost Analysis

We deploy the smart contracts on the local blockchain network using Hardhat and run testing code to ensure each function in our smart contract runs correctly without errors. We have successfully run a total of 40 test cases through Hardhat, as shown in Figure 5.

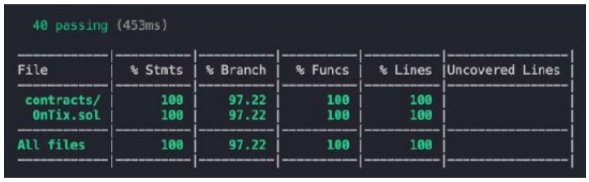

Figure 5. Ontix Smart Contract Testing Result

The list of tested test cases is shown in Table 1. For each function, we run around 4-5 test cases. We leverage Solidity's

DOI: 10.34148/teknika.v15i1.1432

built-in modifiers and require statements to enforce rules and conditions for each function.

Table 1. Test Case Status Run on Local Blockchain Network (Hardhat)

Function	Test Case	Status
`CreateEvent()`	Should allow anyone to create an event successfully, Should emit EventCreated success event, Should revert if event start time is more than event end time, Should revert if resale start time is more than resale end time, Should revert if price cap is lower than original ticket price, Should revert if token URIs do not match max ticket count, Should revert if resale doesn't end before event starts.	Success
`BuyTickets()`	Should allow buyer to buy tickets successfully, Should emit TicketPurchased success event, Should revert if ticket sales period ended, Should revert if not enough tickets available, Should revert if not enough ETH sent.	Success
`ValidateTicket()`	Should allow ticket owner to validate ticket successfully, Should emit TicketValidated success event, Should revert if sender is not the ticket owner, Should revert if ticket already used, Should revert if ticket expired.	Success
`TransferTickets()`	Should allow ticket owner to transfer a ticket successfully, Should emit TicketTransferred success event, Should revert if sender is not the ticket owner, Should revert if ticket has already been resold, Should revert if ticket expired.	Success
`ListForResale()`	Should allow ticket owner to list for resale successfully, Should emit TicketListedForResale success event, Should revert if sender is not the ticket owner, Should revert if ticket is already resold, Should revert if outside of resale period, Should revert if resale price exceeds cap.	Success
`BuyResaleTickets()`	Should allow buyer to purchase resale ticket successfully, Should emit TicketResold success event, Should revert if ticket expired, Should revert if ticket is not listed, Should revert if ticket already resold, Should revert if msg.value is not equal to total resale price.	Success
`WithdrawEventProceeds()`	Should allow event creator to withdraw proceeds successfully, Should emit EventProceedsWithdrawn after successful withdrawal, Should revert if non-creator tries to withdraw, Should revert if no funds to withdraw.	Success

Furthermore, we deploy the Ontix smart contract on the Ethereum test networks, Layer 1 and Layer 2, namely the Sepolia testnet and the Sepolia Optimism testnet, respectively. Then we run seven functions on both layers and compare the gas costs in Gwei. The objective of this comparative deployment was to measure the gas consumption for each function and assess the performance impact of migrating from Layer 1 to a Layer 2 scaling solution.

As shown on Table 2, the data demonstrate a substantial reduction in gas cost across all functions when executed on Layer 2. The Layer 2 deployment achieved an average 99.99% decrease in gas fees compared to the Layer 1 deployment.

The visualization depicted in Figure 6 also shows the most computationally intensive function, BuyResaleTickets(), consumed approximately 1.500001216 Gwei on Layer 1 but only 0.000970252 Gwei on Layer 2. It represents a reduction factor of roughly 1,545x. Lightweight functions such as ListForResale() and TransferTickets() also showed significant efficiency improvements, reducing from around 1.5 Gwei on Layer 1 to approximately 0.0001 Gwei on Layer 2. The remaining functions also exhibited similar performance improvements.

Table 2. Gas Cost Comparison in Gwei for Each Function on Layer 1 and Layer 2

Function	Gas Cost (Gwei)	
	Layer 1	Layer 2
`CreateEvent()`	1.500000362	0.000100253
`BuyTickets()`	1.500000330	0.000194251
`ValidateTicket()`	1.500000432	0.000196192
`TransferTickets()`	1.500000373	0.000100254
`ListForResale()`	1.500000438	0.000100251
`BuyResaleTickets()`	1.500001216	0.000970252
`WithdrawEventProceeds()`	1.500000366	0.000100252

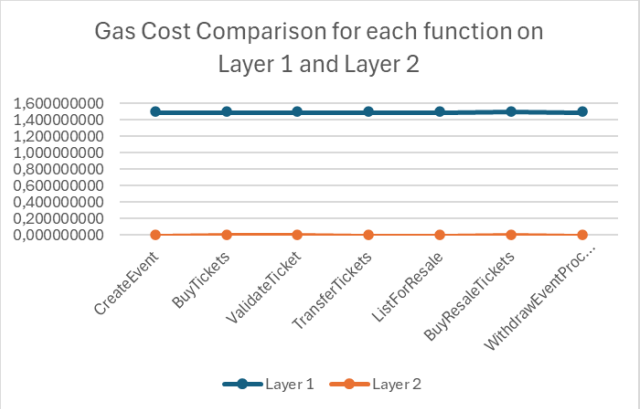

Figure 6. Gas Cost Comparison in Gwei for Each Function

DOI: 10.34148/teknika.v15i1.1432

The experiment confirms that Layer 2 deployment is vastly superior for large-scale ticketing operations with frequent micro-transactions. By reducing transaction fees from the Gwei range to fractions of a micro-Gwei, Ontix demonstrates improved performance in integrating Layer 2 scaling, directly supporting scalability and accessibility for the e-ticketing decentralized application.

2) Comparative Analysis

To evaluate the novelty and effectiveness of the proposed Ontix decentralized NFT-based ticketing platform, a comparative analysis was conducted against the existing related work on blockchain ticketing models. The comparison focuses on seven key functional and technical properties that define robustness, accountability, and scalability of blockchain-based ticketing systems.

As shown on Table 3, most existing work [1], [12], [13], [24], and [25] successfully provide four of seven properties: verifiable ownership, tamper-proof ticket issuance, transparency, and a decentralized system.

Verifiable ownership is supported by the smart contract and the implementation of the ERC-721 standard, which makes NFTs possible. Authors [1], [24], [25] stated that they utilized the ERC-721 protocol to mint tickets as unique NFTs. In the Ontix smart contract, we can accurately verify the ticket owner. Since the transaction is recorded on the blockchain, anyone on the same network can verify the ticket. Furthermore, in the Ethereum ecosystem, Etherscan allows everyone to track every transaction. Ontix smart contracts were deployed to Sepolia Testnet and Sepolia Optimism Testnet.

Tamper-proof ticket issuance. E-tickets issued by Ontix are tamper-proof because they are generated as NFTs only from Ontix smart contracts. Each NFT ticket is unique and is written on the Ethereum network. The data written on blockchain is immutable. If adversaries want to tamper with the blockchain's data, they need to control more than 50% of the blockchain's resources, which is impossible. Therefore, Ontix and five other related works support tamper-proof ticket issuance.

Transparency. Ontix and five other work enable users to view ticket ownership and track transaction history. Data written on the blockchain smart contract is immutable. No adversaries could tamper with the data. Also, users publicly monitor the data.

Decentralized to prevent a single point of failure. Often, when users buy tickets for a high-demand event, huge numbers of requests flood the ticket website, which users call "ticket wars". A centralized system like this is prone to a single point of failure. The result is that most users cannot access the website or buy tickets because the system is inaccessible. Using blockchain as a decentralized technology, Ontix and five other related works offered a decentralized application to prevent a single point of failure. Thus, users/buyers can access the website on time and have a fair chance of buying a ticket.

However, none of the compared studies address three critical challenges in practical large-scale adoption of event management, namely anti-scalping ecosystems, real-time entry verification, and transaction scalability.

Anti-scalping ecosystem. Authors [25] and Ontix introduces smart contract-based price caps and resale time restrictions to prevent speculative reselling and ticket hoarding. E-tickets issued by Ontix can only be sold from the Ontix official website. We ensure that the ticket cannot be sold outside of our system through smart contracts. This functionality enforces fairness within secondary markets, ensuring that resale transactions comply with ethical pricing limits and event-specific policies.

QR-based NFT entry verification. Ontix supports system automation that uniquely integrates QR code validation directly with users' on-chain NFT tickets. At the start of the event, each ticket owner will be verified by showing the NFT ticket as a QR code. Our system will validate the ticket by checking the data against smart contracts. One ticket can only be used once at the time of the event. The Ontix smart contract will prevent ticket fraud or double-entry from a single ticket. This process will be handled by the system; no manual steps are required, nor are there centralized database checks that could be compromised or delayed.

For testing the smart contract, the related work compared focuses on prototypes using free testnets and local simulators, where transaction costs are non-existent or simulated. Authors [25] deployed their prototype on the Ropsten test network and utilized Ganache for local testing. While authors [13] conducted their tests within a local ecosystem on Ganache because performing them on the real Ethereum network would be prohibitively expensive. However, the proposed system, Ontix smart contract, is not only deployed on the local blockchain network but also on the Layer 2 Ethereum network and tested there. Unlike other models, Ontix leverages the Sepolia Optimism network. Supported by the data on Table 2, it shows that Layer 2 significantly reduces gas costs compared to Layer 1. This enhancement makes large-scale ticket issuance viable for high-demand events such as concerts and sports matches.

Table 3. Properties Requirement Comparison

Properties	[1]	[12]	[13]	[24]	[25]	Ours
Verifiable ownership	✓	✓	✓	✓	✓	✓
Tamper-proof ticket issuance	✓	✓	✓	✓	✓	✓
Anti-scalping ecosystem	×	×	×	×	✓	✓
Transparency	✓	✓	✓	✓	✓	✓
Decentralized	✓	✓	✓	✓	✓	✓
QR-based NFT entry verification	×	×	×	×	×	✓
Layer-2 scalability low gas fees	×	×	×	×	×	✓

DOI: 10.34148/teknika.v15i1.1432

IV. CONCLUSION

This paper proposed Ontix, a blockchain-based NFT decentralized e-ticketing system to address ticketing issues such as ticket tampering/fraud, scalping, and inaccessible websites due to a single point of failure. By leveraging blockchain, smart contracts, and ERC-721-based NFTs, Ontix establishes a trusted, verifiable framework for e-ticket issuance, resale, and validation.

The experimental deployment across Layer 1 on the Sepolia Testnet and Layer 2 on the Sepolia Optimism Testnet demonstrated a significant reduction in gas consumption. Layer 2 operations achieved up to a 99.99% decrease in gas consumption, with average costs reduced from approximately 1.5 Gwei on Layer 1 to below 0.0001 Gwei on Layer 2. These results confirm that Layer 2 rollup scaling significantly enhances efficiency and affordability. It enhances the suitability of the system adoption for large-scale events.

The comparative analysis further validated Ontix's advantages over existing related work. While previous studies supported basic blockchain properties such as verifiable ownership, tamper-proof issuance, decentralized, and transparency, they lacked advanced mechanisms for anti-scalping, QR-based NFT validation, and Layer-2 scalability. Therefore, Ontix successfully integrates all seven properties to deliver an accountable, tamper-resistant, and low-cost e-ticketing ecosystem.

REFERENCES

[1] A. A. L. P. Nugraha, N. W. E. R. Dewi, and F. Purnama, "Implementation of Blockchain Smart Contract for Online Concert Ticket Transactions Based on NFTs," *J. Appl. Inform. Comput.*, vol. 9, no. 4, pp. 1115–1126, 2025.

[2] N. Pirpattipanad and P. Ratanaworachan, "User Experiences on a Blockchain-Based Ticket Sales Platform," in *2024 28th International Computer Science and Engineering Conference (ICSEC)*, IEEE, 2024, pp. 1–6.

[3] A. N. Rizki, A. P. P. Prabowo, E. Budi, and C. Tasran, "Comparison of The Role of Digital E- Ticketing in Improving the Quality of Service and Customer Satisfaction," 2021.

[4] P. Cholke, Y. Munde, M. Sayyed, A. Vidhale, and N. Zanwar, "Secure Event Ticketing System Using NFT ERC721 Tokens," in *2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS)*, IEEE, 2024, pp. 1401–1407.

[5] S. Sriram, P. Tharaniesh, P. Saraf, N. Vijayaraj, and T. Murugan, "Enhancing Digital Identity and Access Control in Event Management Systems Using Sui Blockchain," *IEEE Access*, 2025.

[6] Y. E. Oktian, "Design and Implementation of Blockchain-Based Office Attendance System," *Teknika*, vol. 13, no. 1, pp. 137–144, Mar. 2024.

[7] A. Arora, Kanisk, and S. Kumar, "Smart contracts and NFTs: non-fungible tokens as a core component of blockchain to be used as collectibles," in *Cyber security and digital forensics: Proceedings of ICCSDF 2021*, Springer, 2021, pp. 401–422.

[8] J. M. Pajalla, S. D. M. Lu, A. P. Ojenar, R. G. Ado, and J.-A. V. Magsumbol, "PUP UniChain: A Blockchain-Based University Events Access Management System for Polytechnic University of the Philippines," in *TENCON 2024-2024 IEEE Region 10 Conference (TENCON)*, IEEE, 2024, pp. 1817–1820.

[9] T. Le, Y. Kim, and J.-Y. Jo, "Implementation of a blockchain-based event reselling system," in *2019 6th international conference on computational science/intelligence and applied informatics (CSII)*, IEEE, 2019, pp. 50–55.

[10] E. N. Witanto, B. Stanley, and S.-G. Lee, "Distributed data integrity verification scheme in multi-cloud environment," *Sensors*, vol. 23, no. 3, p. 1623, 2023.

[11] D. C. Calderone, "Event management evolution through non-fungible tokens," in *2023 IEEE international workshop on sport, technology and research (STAR)*, IEEE, 2023, pp. 85–89.

[12] A. Aldweesh, "BlockTicket: A framework for electronic tickets based on smart contract," *Plos One*, vol. 18, no. 4, p. e0284166, 2023.

[13] C. Silva, M. Costa, and A. Fonte, "Blockchain-based Smart Contracts Platform for Transparent and Efficient E-Ticketing," *Int. J. Comput. Appl.*, vol. 186.

[14] B. Gómez, K. Sánchez, and C. Salas, "SafeTicket: Tickets Sale Safely Supported by Blockchain and NFTs," in *World Conference on Information Systems for Business Management*, Springer, 2023, pp. 423–431.

[15] Tommy Kuncara, Arman Syah Putra, Nurul Aisyah, and Vh. Valentino, "Effectiveness of the E-Ticket System Using QR Codes For Smart Transportation Systems," *Int. J. Sci. Technol. Manag.*, vol. 2, no. 3, pp. 900–907, May 2021, doi: 10.46729/ijstm.v2i3.236.

[16] F. Duran and others, "Implementation of Blockchain Technology for Image Plagiarism Detection Using DCT, AES128, and SHA-1 Algorithms," *Teknika*, vol. 14, no. 1, pp. 124–134, 2025.

[17] H. Xiong, M. Chen, C. Wu, Y. Zhao, and W. Yi, "Research on progress of blockchain consensus algorithm: A review on recent progress of blockchain consensus algorithms," *Future Internet*, vol. 14, no. 2, p. 47, 2022.

[18] K. Tariq, A. Siddique, A. Rahim, A. Anjum, J. Jamil, and M. Sarwar, "A Non-Fungible Tokens with Solidity Blockchain," *J. Softw. Eng.*, vol. 1, no. 2, pp. 31–48, 2023.

[19] S. Bradić, D. Delija, G. Sirovatka, and M. Žagar, "Creating own NFT token using erc721 standard and solidity programming language," in *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*, IEEE, 2022, pp. 1053–1056.

[20] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends,"

Peer--Peer Netw. Appl., vol. 14, no. 5, pp. 2901–2925, 2021.

[21] "Introduction to smart contracts," Introduction to smart contracts. Accessed: Nov. 30, 2025. [Online]. Available: https://ethereum.org/developers/docs/smart-contracts/

[22] "ERC-721 Non-Fungible Token Standard," ERC-721 Non-Fungible Token Standard. Accessed: Nov. 30, 2025. [Online]. Available: https://ethereum.org/developers/docs/standards/tokens/erc-721/

[23] OpenZeppelin Docs, "ERC20," ERC20. Accessed: Nov. 30, 2025. [Online]. Available: https://docs.openzeppelin.com/contracts/4.x/erc20

[24] K. Okokpujie, O. Owivri, O. Olusanya, S. Daramola, and M. E. Awomoyi, "A single-user electronic ticketing system using ERC-721 protocol for smart contracts," *Bull. Electr. Eng. Inform.*, vol. 14, no. 4, pp. 3110–3120, Aug. 2025, doi: 10.11591/eei.v14i4.8806.

[25] F. Regner, A. Schweizer, and N. Urbach, "NFTs in Practice – Non-Fungible Tokens as Core Component of a Blockchain-based Event Ticketing Application," *ICIS 2019*, 2019.