A Project Report on

DECENTRALIZED TICKETING USING ETHEREUM

Submitted in Partial Fulfillment of the Requirement for Degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT) Awarded by Tribhuvan University

Submitted by:

Aakriti Bhusal (25034/076)

Karuna Acharya (25046/076)

Sulav Parajuli (25065/076)

Supervised by:

Er. Santosh Panth
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

MOUNT ANNAPURNA CAMPUS

Pokhara-5, Parshyang

March, 2024

A Project Report on

DECENTRALIZED TICKETING USING ETHEREUM

Submitted by:

Aakriti Bhusal (25034/076)

Karuna Acharya (25046/076)

Sulav Parajuli (25065/076)

Supervised by:

Er. Santosh Panth

Assistant Professor

Submitted to:

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Mount Annapurna Campus

March, 2024

STUDENT'S DECLARATION

We hereby declare that this project work entitled **DECENTRALIZED TICKETING USING ETHEREUM** is based on our original work. All concepts, data, code, and any other work from external sources have been properly cited and referenced in accordance with the guidelines provided by Institute of Science and Technology, Tribhuvan University.

We owe all the liabilities relating to the authenticity and originality of this project work and project report.

1. Aakriti Bhusal (25034/076) ……..
2. Karuna Acharya (25046/076) ……..
3. Sulav Parajuli (25065/076) ……..

Date: …….

SUPERVISOR'S RECOMMENDATION

This is to certify that this project report entitled **DECENTRALIZED TICKETING USING ETHEREUM** prepared and submitted by below listed team of students in partial fulfilment of the requirements of the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University, has been prepared and completed under my supervision.

I hereby recommend the same for acceptance by Tribhuvan University.

1. Aakriti Bhusal (25034/076)
2. Karuna Acharya (25046/076)
3. Sulav Parajuli (25065/076)

……..

Er. Santosh Panth

Department of Computer Science and Information Technology

Mount Annapurna Campus

Date: …….

Department of Computer Science and Information Technology

MOUNT ANNAPURNA CAMPUS

APPROVAL LETTER

The undersigned certify that they have read and recommended to the Institute of Science and Technology, Tribhuvan University for the acceptance, a project report entitled **DECENTRALIZED TICKETING USING ETHEREUM** submitted by **Aakriti Bhusal, Karuna Acharya and Sulav Parajuli** in partial fulfillment for the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT).

……..
Er. Santosh Panth
Supervisor
Department of CSIT
Mount Annapurna Campus
Date: …….

……..
Er. Ranjan Adhikari
Internal Examiner
Department of CSIT
Mount Annapurna Campus
Date: …….

……..
Mr. Prithivi Raj Paneru
External Examiner
Prithvi Narayan Campus
Date: …….

……..
Mr. Surya Bahadur G.C
Director
Mount Annapurna Campus
Date: …….

ACKNOWLEDGMENT

We are deeply grateful to Tribhuvan University for granting us the opportunity to apply our academic knowledge in real-world settings under supervised conditions.

We extend our sincere appreciation to our project supervisor, Er. Santosh Panth, for his invaluable guidance, encouragement, coordination, and support throughout this project.

This project is a culmination of the efforts of many individuals, and we would like to thank the officials and staff members of Mount Annapurna Campus B.Sc. CSIT for their assistance during the project period.

We are also grateful for the cooperation and encouragement provided by our project group members, without whom this project would not have been possible.

Aakriti Bhusal [25034/076]

Karuna Acharya [25046/076]

Sulav Parajuli [25065/076]

ABSTRACT

This project report outlines the development of a decentralized ticketing system that utilizes the Ethereum blockchain, Solidity smart contracts, and IPFS for decentralized storage. The project follows the agile methodology to ensure effective development. The system aims to overcome challenges found in traditional centralized ticketing systems, such as counterfeit tickets, high fees, lack of transparency, and ticket scalping. By leveraging Ethereum's distinct features, including its immutable ledger, strong security, and scalable infrastructure, a user-friendly web application has been developed. This application enables event organizers to efficiently create and manage events and allows attendees to securely purchase and view tickets. The project results in an innovative ticketing solution that significantly improves trust and transparency in the ticketing industry.

Keywords: *Decentralized ticketing, Ethereum blockchain, Smart contracts, Agile methodology, Ticketing industry*

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ABI	Application Binary Interface
BC	Blockchain
CID	Content Identifier
CSS	Cascading Style Sheets
DAI	Dai Stablecoin
DSS	Decentralized Software Services
EE	Electrical Engineering
ETH	Ethereum
EVM	Ethereum Virtual Machine
GB	Gigabyte
GET	Guaranteed Entrance Token
GUTS	Guaranteed Unique Ticketing System
ICCCN	International Conference on Computer Communications and Networks
IDE	Integrated Development Environment
IPFS	Inter Planetary File System
IS	Information Systems
JS	JavaScript

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. Introduction

The project "Decentralized Ticketing Using Ethereum" is in the context of the ongoing transformation of the ticketing industry. For many years, the ticketing industry has relied on centralized systems that facilitate the purchase of tickets for a variety of events, such as live concerts, sporting events, and cultural performances. Despite its longevity, the traditional ticketing system has faced a series of persistent problems.

The challenges include issues like fraudulent tickets, imposition of high and often unclear fee by ticketing platform, the inadequate transparency of ticket verification, and the continued existence of secondary ticket markets and scalping.

In response to these problems, the project represents an evolving step toward revamping the ticketing industry. By utilizing the unique capability of the Ethereum, including transparent digital ledgers and smart contracts, the project seeks to create a decentralized ticketing system called EthTix. The primary goals are to deal with counterfeit tickets, improve transparency, reduce costs, and mitigate the issues associated with ticket scalping. The adoption of the Ethereum blockchain signifies a transformative shift in the ticketing industry process.

1.2. Problem Statement

The conventional centralized ticketing systems employed in the ticketing industry have confronted a range of problems. These issues include rising numbers of fake tickets, the high and opaque fees charged by ticketing companies, the absence of reliable and efficient systems for verifying tickets, and the existence of secondary ticket markets and scalping. These problems have led to financial losses for event organizers, inflated ticket costs, and contributed to a widespread perception of unfairness in ticket distribution. The central concern motivating this initiative is the need to utilize blockchain technology specifically Ethereum to create a decentralized

ticketing system that can effectively enhance security and transparency replacing the traditional ticketing system.

1.3. Objective

The central objective is to improve the ticketing industry, where the key attributes of trust, efficiency, and accessibility will be instilled as its defining characteristics. The main objective of the project is:

- To develop a web application for decentralized ticketing utilizing Ethereum blockchain technology.

1.4. Scope and Limitation

Scope

EthTix includes developing a comprehensive platform to facilitate event creation and sale of tickets for events. The system tries to give attendees an effortless and secure way to search for and buy tickets using digital wallets such as MetaMask and provides an intuitive interface for event organizers to create and manage events.

Key scopes include:

Event Creation: Event organizers can create events including details like the event name, date, venue, timing, and ticket price.

Search Events: Users can look up events, analyze the details, and select events that grab their attention.

Wallet Integration: Attendees can interact with the blockchain using a secure Ethereum wallet like MetaMask.

Purchasing Tickets: Users can buy tickets using a cryptocurrency like Ether through a MetaMask wallet.

Transparency: The approach ensures every activity related to tickets is trackable and verifiable eliminating the potential for fake tickets.

Limitations

Despite various valuable functionalities like event creation, event management, ticket sales, etc. provided by EthTix, it also endures several obstacles. Some major limitations include:

Resistance to Web3 Adoption: The implementation of Web3 is held down by the necessity to learn modern technologies and concepts, which can be challenging for both developers and users. Rather than investing time and energy in putting Web3 concepts into execution, they may prefer to stick with established technologies.

Slow Transactions: Ethereum's existing transaction speed can cause delays while creating and purchasing tickets, particularly during periods of network congestion.

Limited Blockchain Storage: Maintaining all the ticket data on the blockchain can be costly, potentially compromising platform performance.

Scalability Challenges: As the number of events and users in the platform increases, there might be scalability problems related to the capacity of the Ethereum blockchain to handle increased transactions efficiently.

1.5. Development Methodology

The project adopted the Scrum framework, a widely recognized Agile methodology, to guide its development process. Scrum involves breaking down tasks into manageable segments and iterating through them in defined sprints, fostering efficient development. This iterative approach facilitates ongoing enhancements to the ticketing solution. Effective collaboration is central to Scrum, with regular meetings including sprint planning, daily stand-ups, development, testing, and review fostering transparent communication and team collaboration.

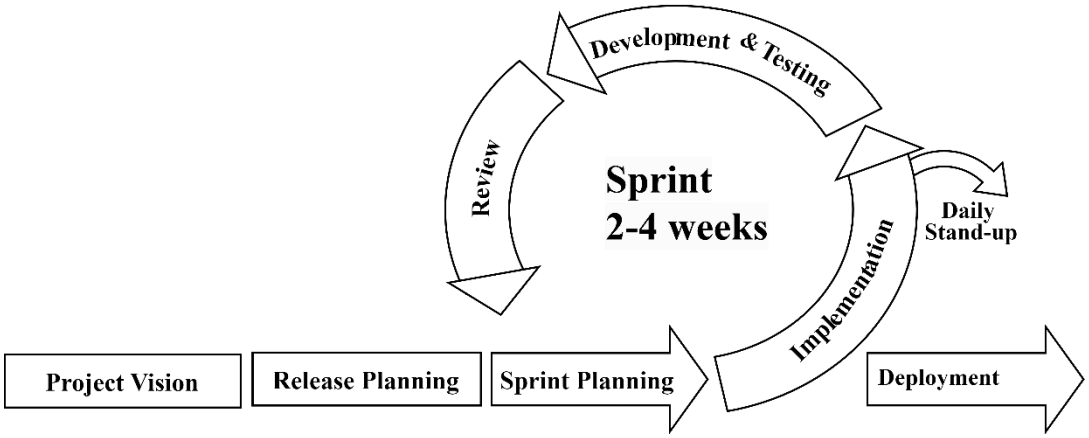


Figure 1.1: Phases of scrum process

Step-by-step sprints of the project are given below:

Sprint 1: Environment Setup and Contract Development

Sprint Planning:

The team outlined key tasks for the project, focusing on setting up the Hardhat environment and integrating Vite for streamlining front-end development using React, according to project requirements.

Daily Stand-ups:

Team members virtually met to discuss progress on Hardhat environment setup, frontend Vite project integration, and the initiation of basic front-end components.

Development and Testing:

The team established the Hardhat environment and integrated the Vite front end. The initial step includes creating a fundamental Ethereum smart contract with basic functions and structures, while also initiating basic front-end components.

Review:

The team demonstrated the progress made in assigned task and feedback was collected on the initial implementation, guiding adjustments for the next sprint.

Sprint 2: Front-end Enhancement and Testing Iterations

Sprint Planning:

The team prioritized enhancing front-end components, focusing on improved User Interfaces and Bootstrap integration. Tasks were defined for front-end enhancement.

Daily Stand-ups:

Team members discussed progress on front-end enhancement, Bootstrap integration, and ongoing testing of front-end components.

Development and Testing:

The assigned member enhanced front-end components, focusing on better User Interfaces and Bootstrap integration.

Review:

The team demonstrated the improved front-end components and showcased the results of testing. Feedback was collected which guided the modification and adjustment in the next sprint.

Sprint 3: Contract Optimization and Improvements

Sprint Planning:

Priorities were set for optimizing smart contracts, ensuring gas efficiency, and finalizing contract functionalities. Tasks included integrating finalized contracts with frontend components and conducting further testing.

Daily Stand-ups:

Team members discussed progress on contract optimization, finalizing contract functionalities, and integrating contracts with front-end components.

Development and Testing:

Smart contracts were optimized for gas efficiency, security, and performance. Finalization of contract functionalities based on project requirements occurred. Contracts were integrated with front-end components, and comprehensive testing was conducted.

Review:

The team demonstrated the optimized smart contracts and showcased their integration with front-end components and collected feedback.

Sprint 4: Integration and Finalization

Sprint Planning:

Tasks were defined for integrating frontend and backend components, conducting final testing, and deploying contracts on the Sepolia testnet of Ethereum.

Daily Stand-ups:

Team members discussed progress on integrating frontend and backend components, final testing, and preparing for contract deployment.

Development and Testing:

Integration of frontend and backend components for end-to-end functionality occurred. The project was executed, and comprehensive final testing was conducted. Contracts were deployed and executed on the Sepolia testnet.

Review:

The team demonstrated the fully integrated project, showcasing end-to-end functionality. Feedback from final testing, deployment, and execution guided bug fixes, issue resolution, and preparation for project finalization.

1.6. Report Organization

The first section comprises the following: the Title Page, Acknowledgment, Abstract, Table of Contents, List of Abbreviations, List of Tables, and List of Figures. The primary report has been split into six chapters, which are as follows:

Chapter 1: Introduction

The chapter introduces decentralized application, outlining its purpose and the need for its development. It describes how the inadequacies of conventional ticketing systems prompted the development of the system, called EthTix. The goals of the system are defined in this chapter, outlining its intended outcomes. It explores the system's scope in more detail, defining its parameters and emphasizing its drawbacks. A review of the development technique used on this project wraps up the chapter.

Chapter 2: Background Study and Literature Review

The chapter offers a background study and literature review concerning decentralized ticketing systems and blockchain technology. It covers essential theories, concepts, and terminologies relevant to the project. The literature review explores existing research, papers, and articles on alternative algorithms and similar works.

Chapter 3: System Methodology

The chapter concentrates on system analysis, outlining both the functional and non-functional requirements of EthTix. It includes a feasibility study to evaluate the practicality of system development, considering technical, operational, economic and scheduling feasibility aspects.

Chapter 4: System Design

The chapter covers the system design, including architectural design, activity diagrams, sequence diagram, class and object diagram, and component diagram. It includes the algorithmic details of the system.

Chapter 5: Implementation and Testing

The system's testing and implementation are addressed in this chapter. It lists the tools and modules used in developing EthTix and describes the testing procedures conducted for validation and verification purposes.

Chapter 6: Conclusion and Future Recommendations

This chapter culminates the project report and offers an overview of the web application's creation process. With the goal to further improve EthTix, the developers have included recommendations for future modifications.

CHAPTER 2
BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

Trust plays a vital role in the prosperity of online marketplaces. Purchasers are cautious about what they view as potential self-interest on the part of intermediary platforms [1]. In the online marketplace, buyers equally prioritize risk perception as a significant factor to consider [2]. Bitcoin was introduced as a system for facilitating transactions without reliance on trust. Nakamoto characterizes the foundational technology of Bitcoin as a network of peer-to-peer nodes that construct a transparent transaction chain using cryptographic hashes that include the prior block in the chain [3]. Bitcoin's emergence has spurred the acceptance of distributed ledgers in numerous applications. Ethereum, conceived as an extension of the Bitcoin model, was designed to promote trust-based transactions among end users. Gavin Wood, the creator of Ethereum and Ethcore, articulated the project's mission as enabling transactions between individuals who lack established trust mechanisms, aiming to establish a system that ensures interactions with unwavering confidence in the results [4].

Pongnumkul and colleagues conducted a comparative study between Ethereum and Hyperledger, an open source blockchain framework hosted by the Linux Foundation. They assessed various performance metrics, including transaction deployment time, transaction completion time, execution time, latency, throughput, and maximum concurrent transactions. Their results revealed that while Hyperledger showed superior throughput, Ethereum showed better capability to handle more concurrent transactions. This positions Ethereum as the more fitting platform for building scalable applications. Moreover, Ethereum's integration of its own cryptocurrency within applications developed on its framework empowers developers to construct e-commerce applications utilizing the Ethereum cryptocurrency, known as ether [5]. Toneli and his team conducted a comprehensive analysis of over 12,000 smart contracts to identify software metrics, revealing that smart contract metrics typically

exhibit narrower ranges compared to their counterparts in traditional software systems. The software metrics they investigated encompassed attributes such as the number of events, mappings, modifiers, contracts, functions, lines of code, comments, bytecode, and cyclomatic complexity. While these metrics are commonly applicable to all software, smart contracts introduce additional unique metrics such as calls to or from other addresses, intra-smart contract calls, gas consumption, cryptocurrency transactions, and bytecode/ABI metrics [6].

At Vanderbilt University, researchers focused on a healthcare-specific application and identified a range of qualitative measures including structural interoperability, support for Turing-complete operations, capabilities for user identification and authentication, cost-effectiveness, and the ability to scale effectively across large populations [7] .Additional recommendations included proposing security patterns for newly created smart contracts, involving comparisons based on measures such as checks-effects-interaction, emergency stop, speed bump rate, limit mutex, and balance limit [8].

2.2. Literature Review

Among the various blockchain-based ticketing management systems, the key players include Origin Protocol, GUTS, Blockparty, Aventus Protocol, and Blocktix. The research indicates that IPFS is the preferred Distributed Storage System (DSS) of the projects we examined. IPFS stands for InterPlanetary File System and is a peer-to-peer (P2P) public DSS used for storing and accessing various data objects, files, websites, and applications. What sets IPFS apart is its use of content addressing, meaning it assigns addresses and retrieves files based on the content's hash, rather than where it is in the distributed storage network. Whereas Ethereum is used by most of the ticketing systems studied as their underlying BC infrastructure [9].

Origin Protocol [10] primarily focuses on decentralized marketplaces and e-commerce, making it less suitable for event ticketing. Their application does not support event management or ticket issuance. However, their architecture offers a decentralized solution for managing digital asset identities.

GUTS [11] and Blockparty [12], despite marketing themselves as decentralized ticketing platforms, do not allow users to manage their private keys. These companies

manage users' private keys, creating a crucial point of failure and putting all the tickets at risk.

In contrast to the Aventus protocol [13], other systems lack a second layer solution. A second layer solution refers to processes managed outside the blockchain to enable off-chain peer-to-peer communications, enhancing transaction throughput. Aventus, built on top of Ethereum, tackles Ethereum's scalability issues by handling selected transactions off-chain. Ethereum's Proof-of-Work version processes only about 15 transactions per second, slower than centralized systems like Visa with over 20,000 transactions per second. Aventus mitigates this by publishing the root of a Merkle tree onto Ethereum for security and validation, making transactions more efficient and batch processing them into Aventus blocks before sending them to the Ethereum blockchain for public access.

Some systems, such as Blocktix, use a native token for payments. This requires event guests to first exchange their general-purpose tokens like Bitcoin or Ethereum for platform-specific tokens, adding complexity. Most users do not own such platform-specific tokens, making common cryptocurrencies or widely adopt stablecoins, like DAI, more user-friendly. However, except for the Origin Protocol, none of the investigated ticketing systems integrate stablecoins for payments. This exposes event hosts and guests to cryptocurrency price fluctuations and may discourage end users from using the platform.

CHAPTER 3
SYSTEM ANALYSIS

3.1. System Analysis

System analysis is a crucial step of the system development lifecycle. Various tasks such as functional and non-functional requirement analysis, feasibility analysis like technical, operational, economic and schedule, etc. are carried out in order to understand the system and its objectives.

3.1.1. Requirement Analysis

Requirement analysis has been performed by analyzing user's functional and non-functional requirements and building a solid foundation for further system development processes.

i Functional Requirement

During the system analysis phase, the following functions are found to be most essential in order to address the user's basic requirements:

Event Creation:

- Allows event organizers to create events with relevant details (event name, date, time, price, ticket limit, and description).
- Allows each ticket to be uniquely identifiable and tied to a specific event.

Ticket Purchase:

- Allows attendees to purchase tickets directly from event organizers.
- Implements a user-friendly interface for ticket selection and purchase.

Wallet Integration:

- Integrates Ethereum wallets like MetaMask for handling ticket transactions.
- Confirm successful transactions and update ticket ownership accordingly.

Use Case Diagrams for Admin, Event Organizer and User

The use cases visually show how the various components interact in the system. It is a graphical representation used in system analysis to gather, clarify, and organize the system requirements for the dApp.

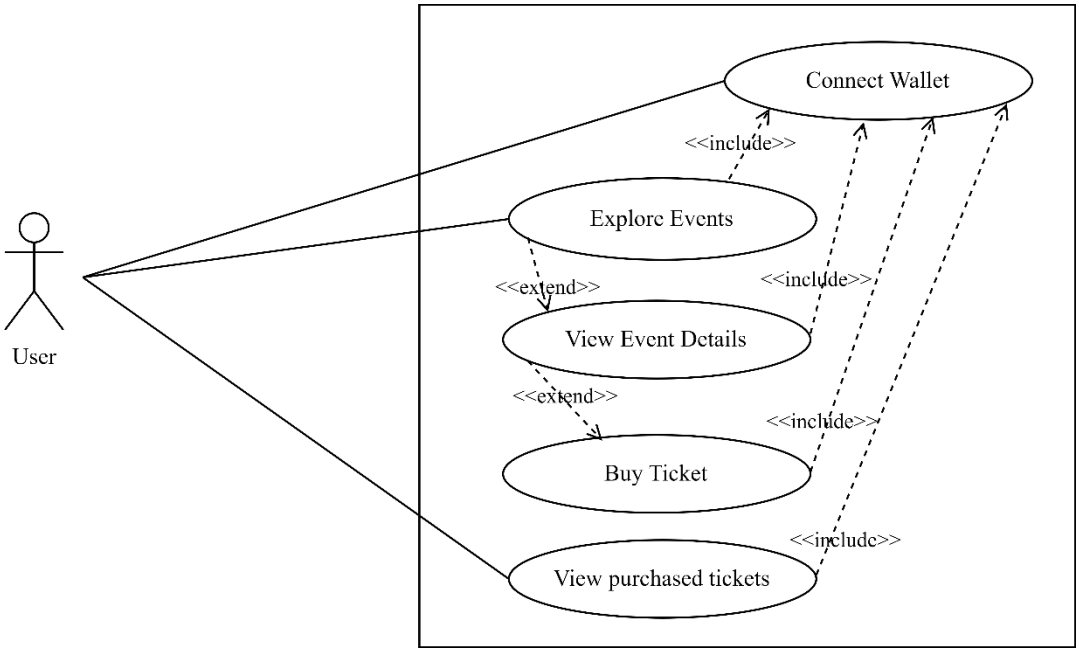


Figure 3.1: Use Case Diagram for User

The user's activity includes connecting wallet, exploring and checking details of events along with the date, time, location, price per ticket, etc., and buying tickets through the MetaMask wallet. They can also review purchased tickets in the "My Tickets" section along with the event name, date, and QR code.

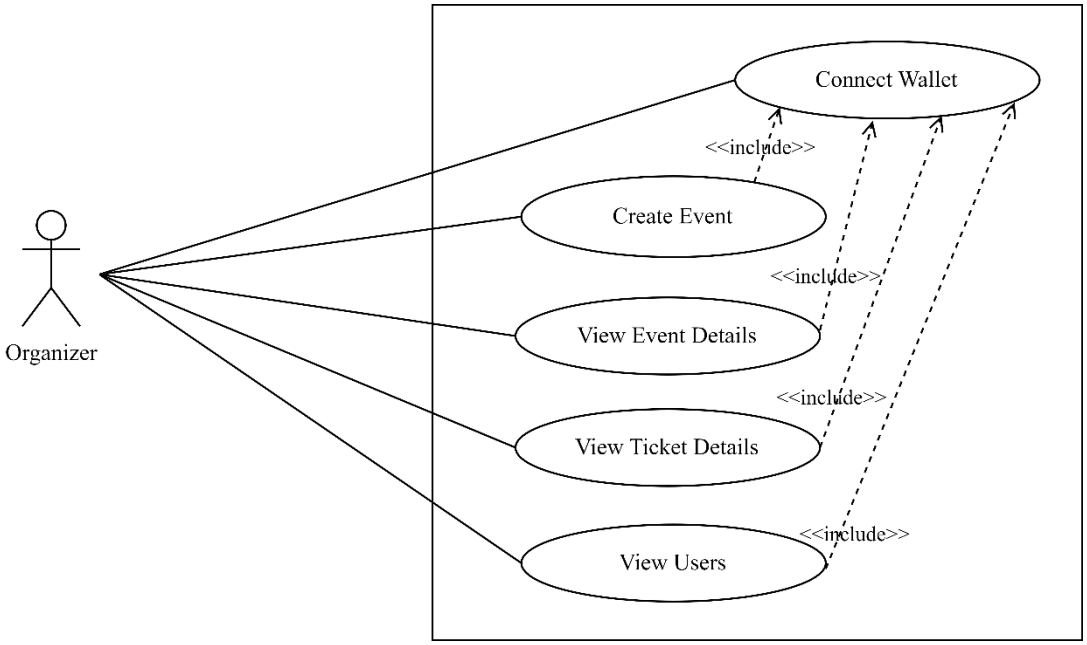


Figure 3.2: Use Case Diagram for Organizer

The event organizer initiates event creation by connecting their wallet, adding organizational details, adding event details, and making payments for each event creation. They can seamlessly view details of their events such as number of tickets sold, remaining tickets, user details, etc.

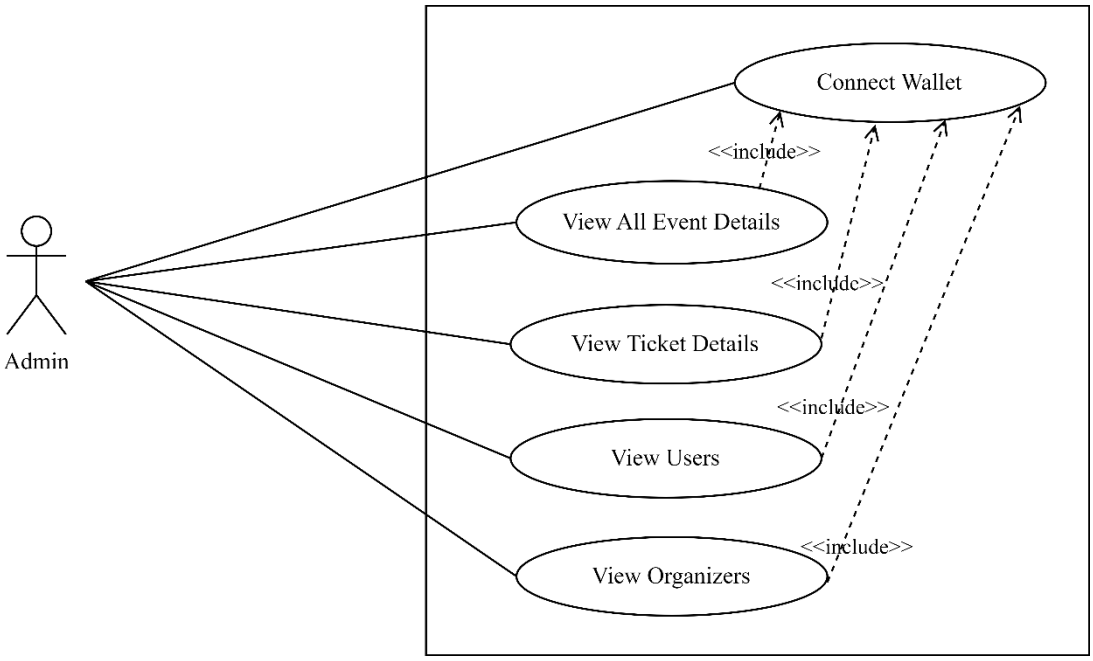


Figure 3.3: Use Case Diagram for Admin

The administrator's responsibilities include reviewing information about all the events created by registered organizers in the dApp along with the event details such as the number of tickets sold, revenue generated, remaining tickets, etc by connecting wallet. They can also view all the users that have purchased tickets from that platform.

ii Non-Functional Requirement

The following attributes describe the quality of the project, and hence results in a positive user experience:

Performance:

The utilization of the Ethereum blockchain ensures faster transactions by eliminating the need for central authority to validate transactions. Additionally, optimized smart contracts further enhance the transaction speed by streamlining the processing logic resulting in increased performance.

Security:

The transparency and immutability of blockchain guarantee that every activity related to a ticket is trackable and verifiable. This eliminates the risk of data tampering and unauthorized access. The smart contract further enhances security by automating processes and reducing the potential for human error or fraud.

Usability:

The ticketing dApp provides an intuitive and user-friendly interface to ensure ease of use and positive user experience across various devices and platforms.

Compatibility:

The platform is compatible with a range of devices, operating systems, and browsers to maximize user accessibility.

3.1.2. Feasibility Analysis

Feasibility analysis is vital in project development, assessing the practicality and viability of the decentralized ticketing platform. Technical, operational, economic, and schedule aspects are evaluated to ensure alignment with project goals. Considering accessibility, user-friendliness, cost-effectiveness, and adherence to a

realistic timeline, this thorough analysis boosts confidence in the project's potential success and alignment with the objectives. Some of the feasibility that are included in this project are discussed below:

i. Technical Feasibility

The project has been built on a computer using Visual Studio Code and the Hardhat development environment with no need for additional hardware, ensuring the project's technical feasibility.

Hardware Requirements

- A standard computer (<i3 5th Gen, 4GB RAM or higher)
- Hard Disk

Software Requirements

- Frontend: HTML, CSS, JS, Bootstrap, React, Ether.js
- Backend: Solidity
- Tools: Visual Studio Code
- Development Environment: Hardhat

ii. Operational Feasibility

The project is built on its user-friendly design, which aims to meet the varied needs and expectations of its target users. Through the incorporation of user experience principles, the platform has been customized to provide easy navigation, clear instructions, and seamless interaction. Rigorous testing has also been conducted to ensure the platform's usability across different devices and user groups. By focusing on user satisfaction and accessibility, the project enhances its operational feasibility, promoting widespread adoption and sustained usage of the decentralized ticketing platform.

iii. Economic Feasibility

The utilization of Sepolia testnet for smart contract application development ensure project is economically viable. Leveraging the Sepolia testnet contributes to the financial sustainability of the project allowing to experiment and test smart contracts without the need for actual financial transactions.

iv. Schedule Feasibility

Given that the project's requirements, functions, and performance specifications have indicated its accomplishment within the allotted time, it has been feasible to undertake this project within the specified period. The estimated working schedule is outlined below:

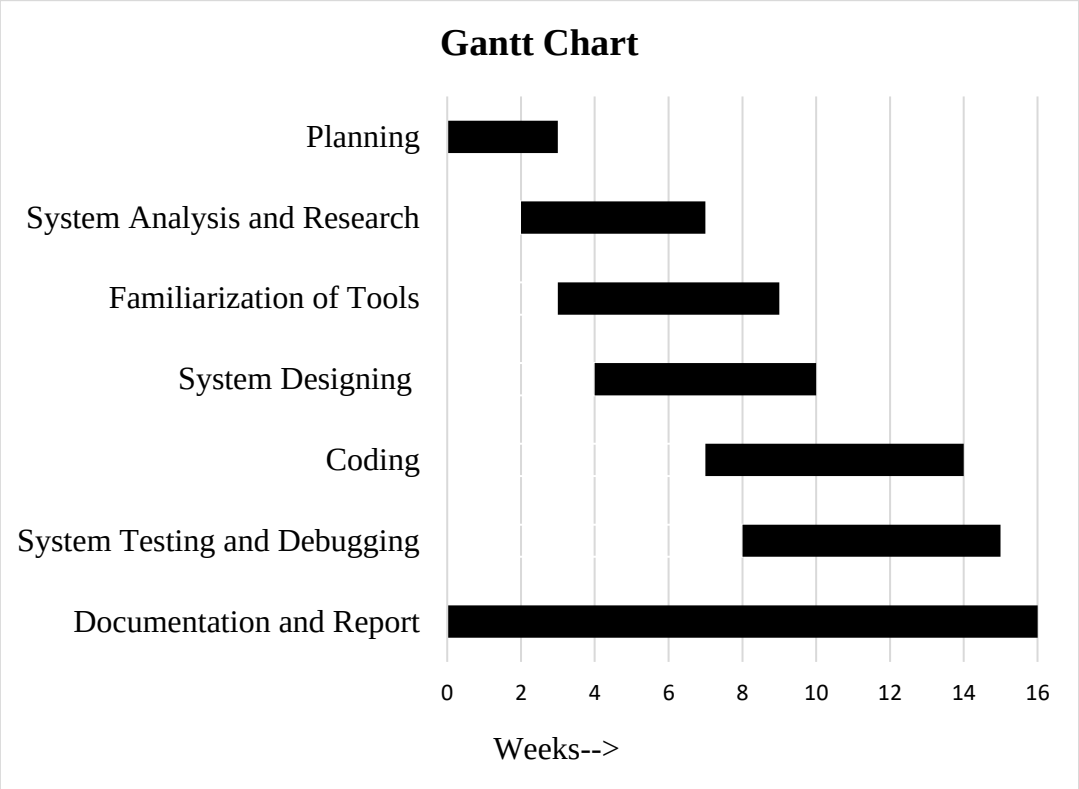


Figure 3.4: Gantt Chart

3.1.3. Analysis

Activity Diagrams

In the project, activity diagrams visualize process flows, decision points, and concurrency. They illustrate actions like event creation, event browsing, ticket purchasing, reviewing purchased tickets, etc. These diagrams help us understand the system's dynamic behavior.

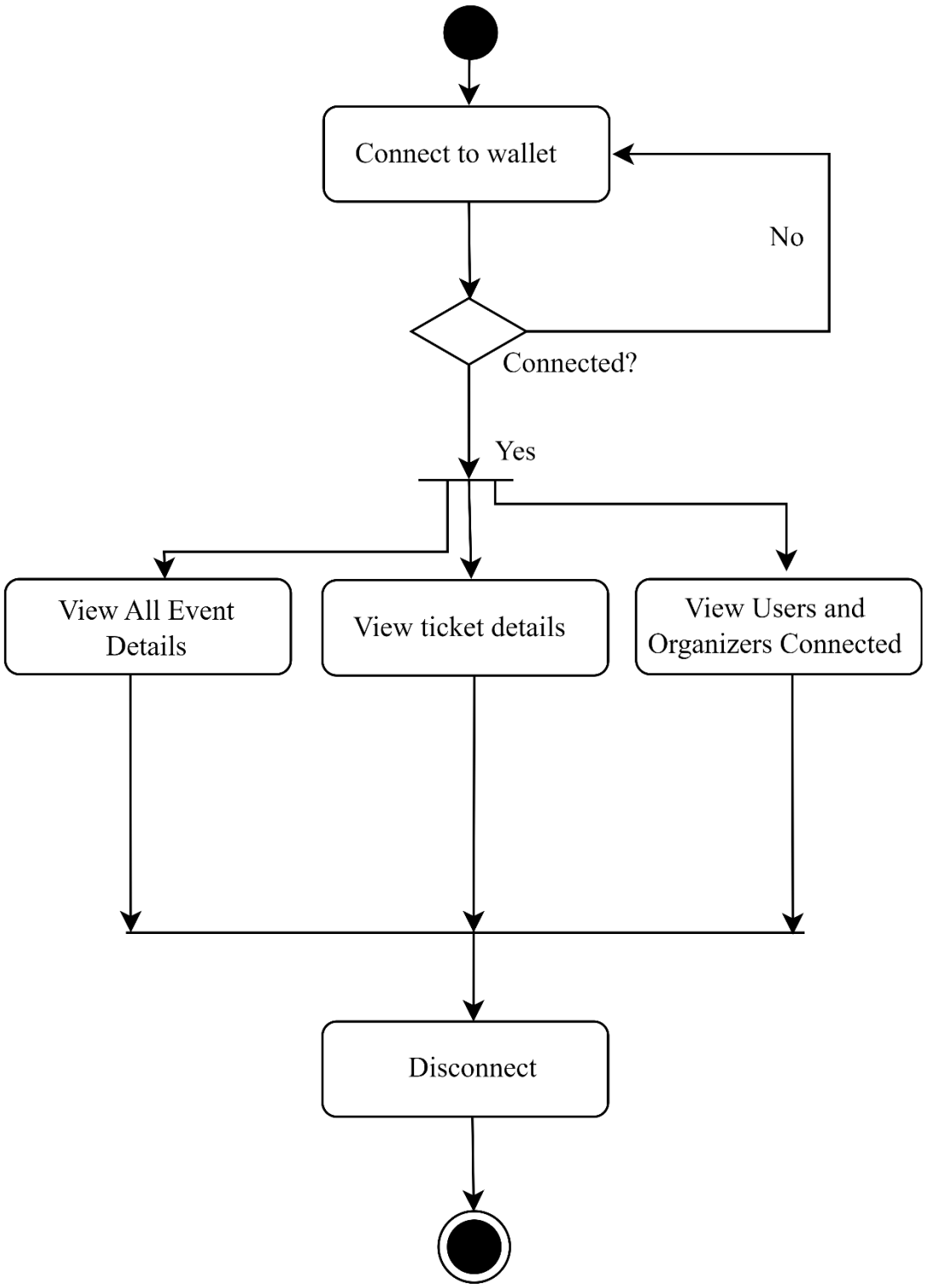


Figure 3.5: Activity Diagram for Admin

The above diagram illustrates the flow of activities for the admin, from accessing the system to interacting with different sections of the admin dashboard as per their needs.

The process begins with the admin accessing the system and deciding whether to connect their wallet. If the admin connects their wallet, they can choose to search for events and view event details or directly go to the admin dashboard. Admin can access many sections, including Users, Events, and Reports, within the admin dashboard. The administrator can see the overall number of tickets sold, events created, organizers registered, and so forth. In the Users section, they can view event organizers and connected users. In the Events section, they can view all the event details. In the Reports section, they can view detailed reports.

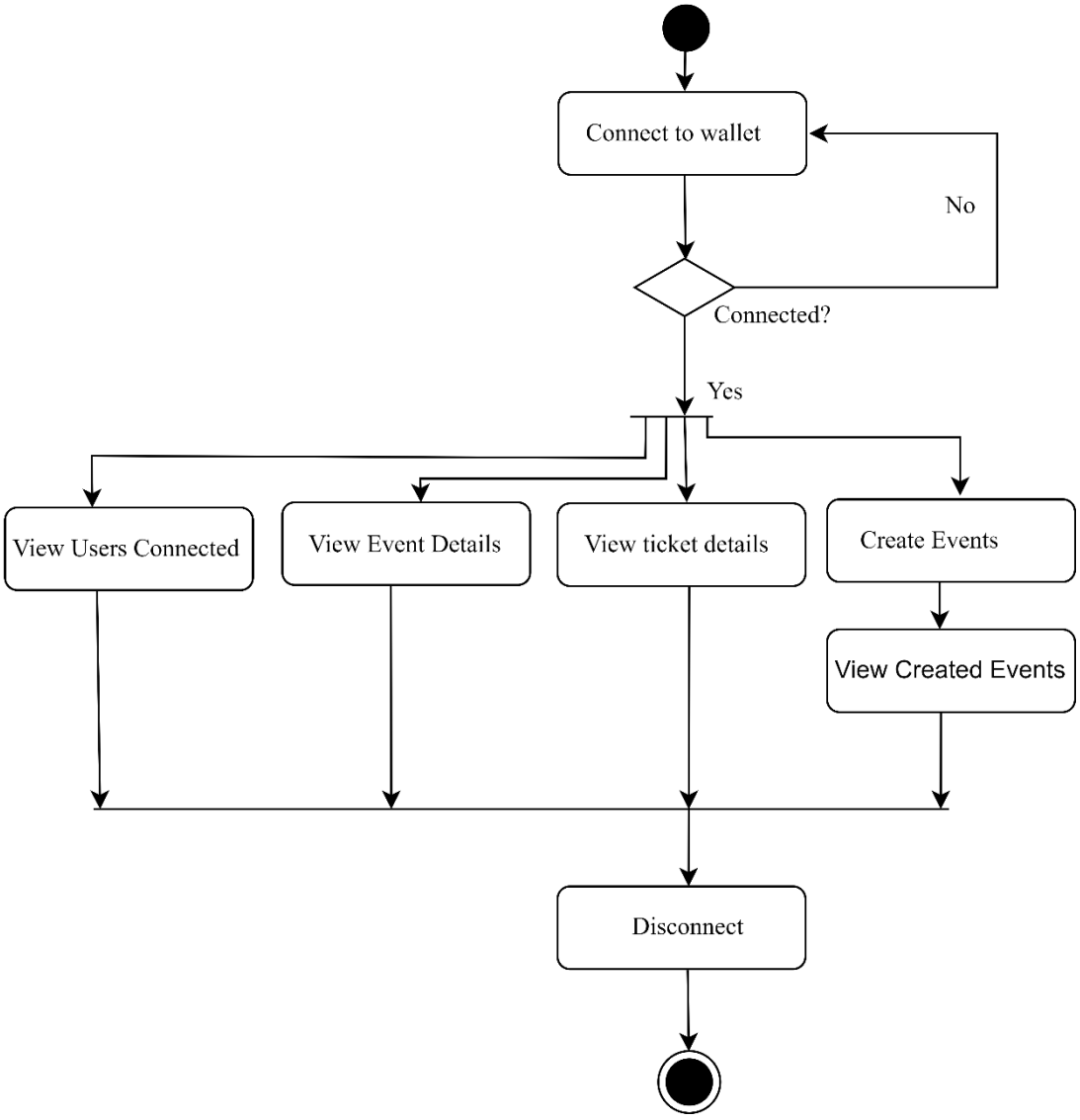


Figure 3.6: Activity Diagram for Organizer

The above diagram illustrates the flow of activities for the event organizer, from accessing the system to interacting with different sections of organizer dashboard as

per their requirements. The organizer's activity diagram starts with accessing the system and deciding whether to connect their wallet. If they connect their wallet, they can then choose to search for events and view events directly or go to organizer dashboard. In the dashboard, organizers can view the events created by them, tickets sold, pending tickets, total tickets, and so on. There are different sections for Create Event, Users, Events, and Reports. The Create Event section allows event organizers to create events, the Users section shows connected users for that organizer's events, the Events section displays the details of that organizer's events, and the Reports section provides detailed reports.

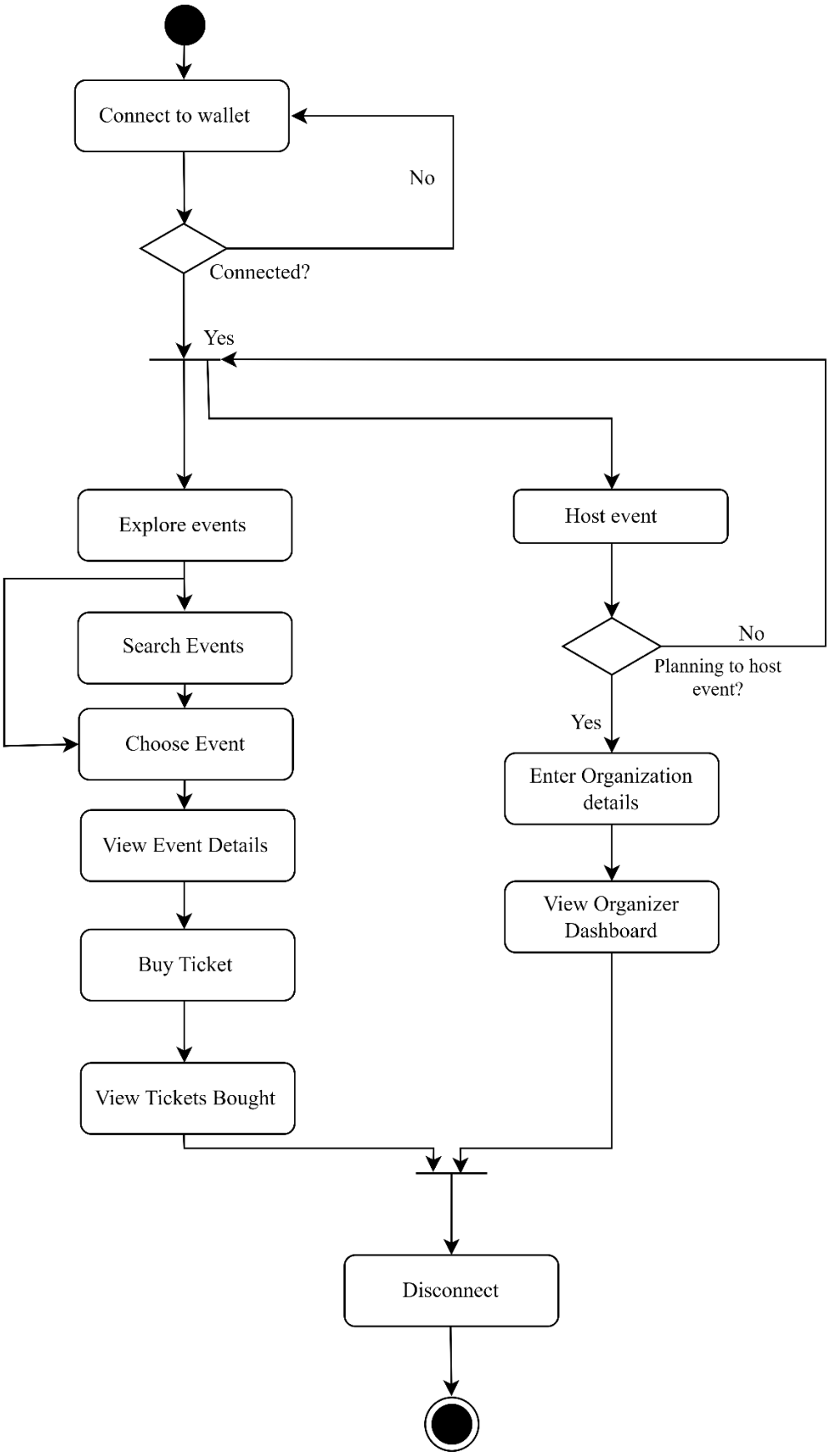


Figure 3.7: Activity Diagram for User

The activity diagram for the web application user begins with accessing the system and considering whether to connect their wallet. If the wallet is connected, the user can either explore events, search for specific events, choose an event, view event details, buy tickets, and view purchased tickets, or they can directly host events by providing organizational details and registering as an event organizer.

CHAPTER 4

SYSTEM DESIGN

4.1. Design

System Design

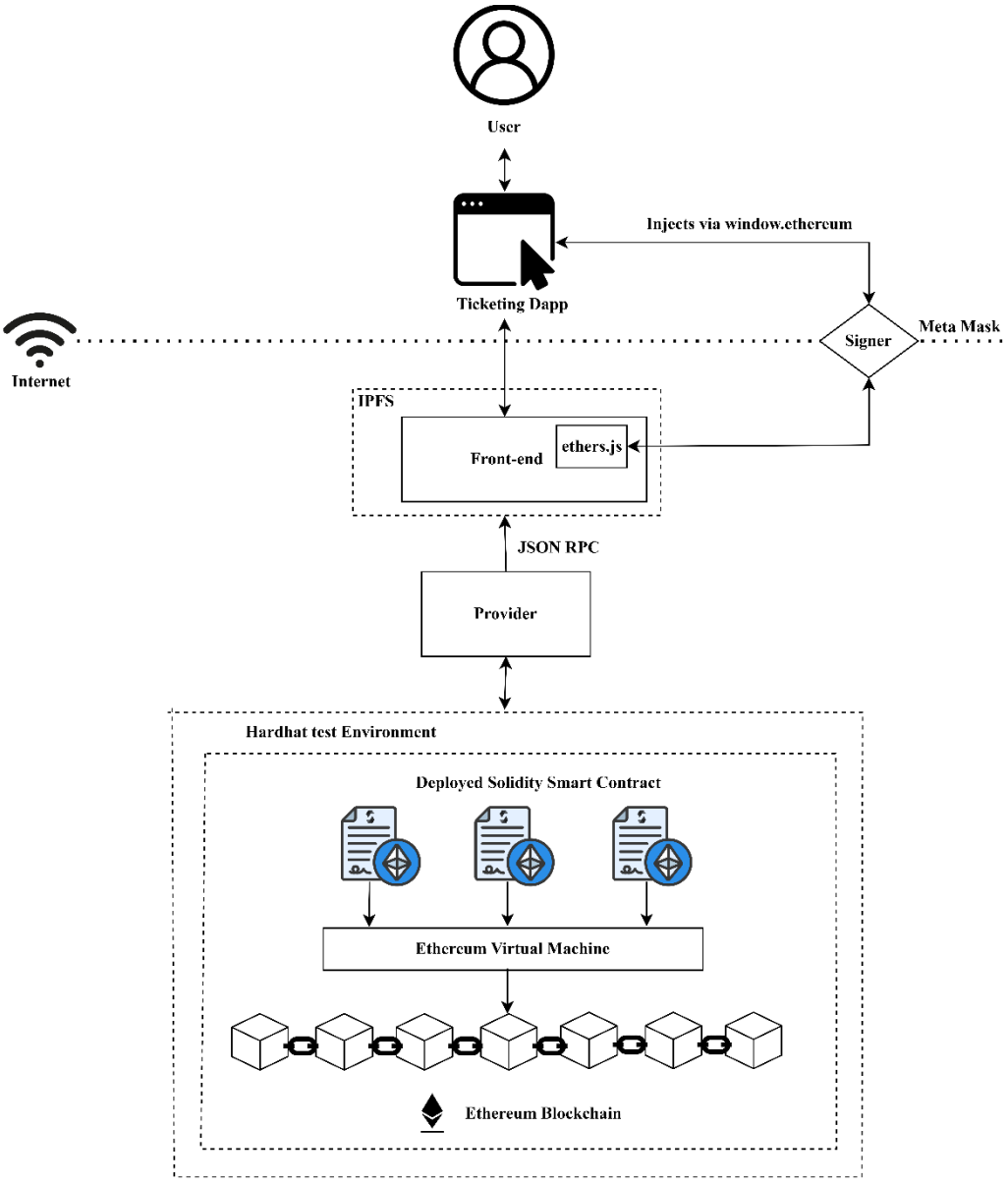


Figure 4.1: Architectural Design of EthTix

The above diagram is an architectural design of EthTix that constitutes of various components and interrelation between them.

User: The user indicates the end user of the system who interacts with ticketing dApp. They explore events, search for specific events, choose events, view event details, buy tickets, and view purchased tickets. They can also host events by registering as an event organizer.

Ticketing dApp: The ticketing dApp is the user-friendly interface accessible over the Internet. It allows us to interact with the functionalities of the ticketing system.

Signer: The MetaMask wallet serves as the signer in the system, securely handling transactions initiated by the user. It verifies and authorizes transactions before they proceed, ensuring security and authenticity in the ticketing system.

Internet: Internet is the network infrastructure that enables communication between components of the system, allowing users to access the ticketing dApp and interact with Ethereum.

Ethereum Blockchain: Ethereum is the decentralized ledger where the Solidity smart contract is deployed, and ticket transactions are recorded. It provides a secure and transparent platform for managing ticket processes.

EVM: EVM executes solidity smart contracts on the Ethereum ensuring its functionality within the network. It processes transactions and ensures that they are executed correctly according to defined rules in the smart contract.

Hardhat: Hardhat test environment is employed for development and testing of Ethereum smart contract in the platform. A suite of tools for compiling, deploying, and testing smart contracts are provided by hardhat, making it easier to ensure functionality and reliability of the code before deploying it to the Ethereum.

Solidity Smart Contracts: Solidity smart contract automates the contract execution based on pre-defined rules, thus eliminating the need for intermediaries. Once it is deployed, it operates independently by managing agreements, verifying them, or enforcing them without human intervention.

Frontend: Frontend is the user interface of the ticketing dApp that users interact with. Ether.js provides the front-end of the dApp with the ability to interact with smart contract and facilitates seamless exchange of data between blockchain and the front end, ensuring smooth communication and transaction processing.

IPFS: IPFS stores and distributes content related to events and tickets in the ticketing system, hence providing a decentralized and censorship-resistant way to store and access files.

Provider: JSON RPC is a provider to facilitate communication between frontend and the Ethereum. It allows the frontend to send requests to the blockchain and receive responses, enabling smooth data exchange and interaction.

Class Diagram

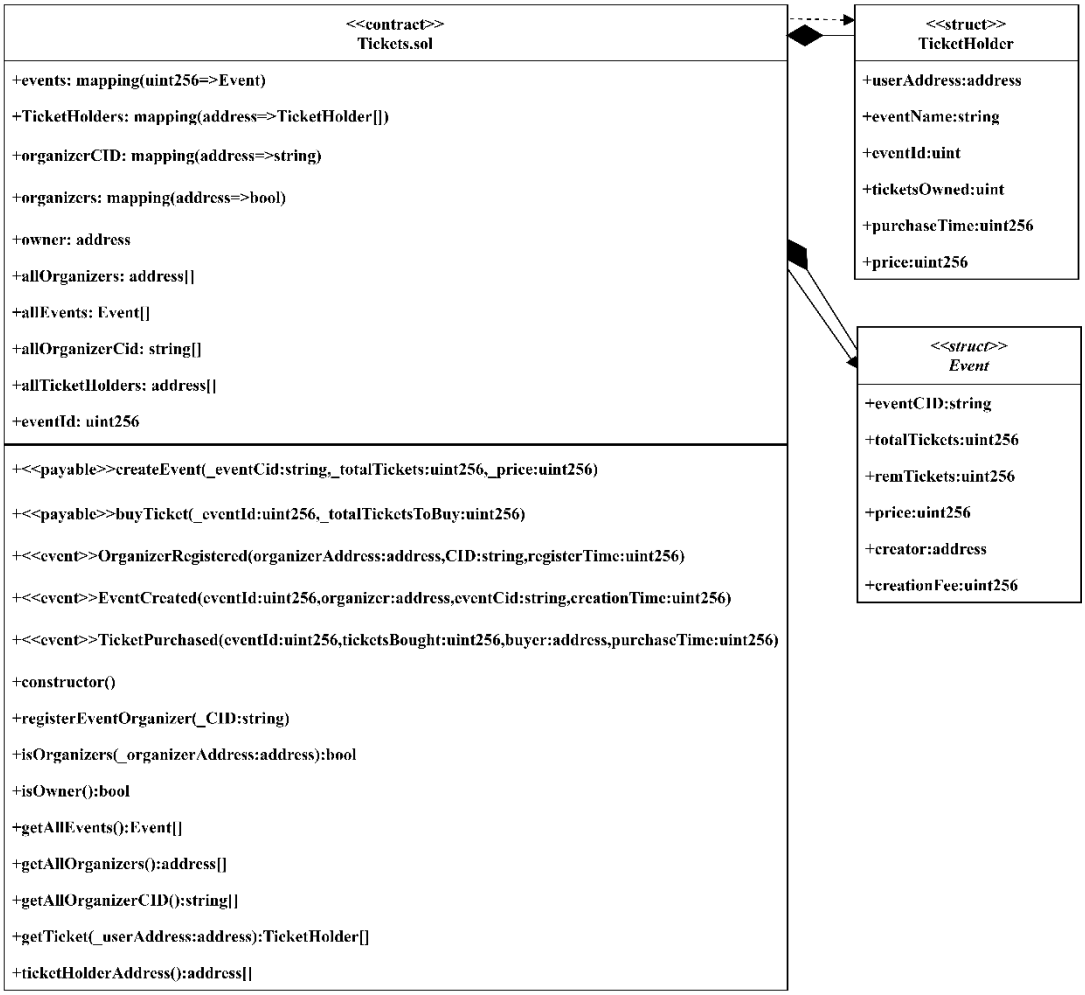


Figure 4.2: Class Diagram

The above diagram depicts a Ticket contract that functions as the core component of a ticketing system. It defines the contract associated with the ticketing system. The Tickets contract contains the Event and TicketHolder struct which represents the different attributes of the Event and ticket holder. This design ensures that ticket

holder and event information is closely tied to individual tickets, simplifying the overall ticketing system design and management.

Sequence Diagram

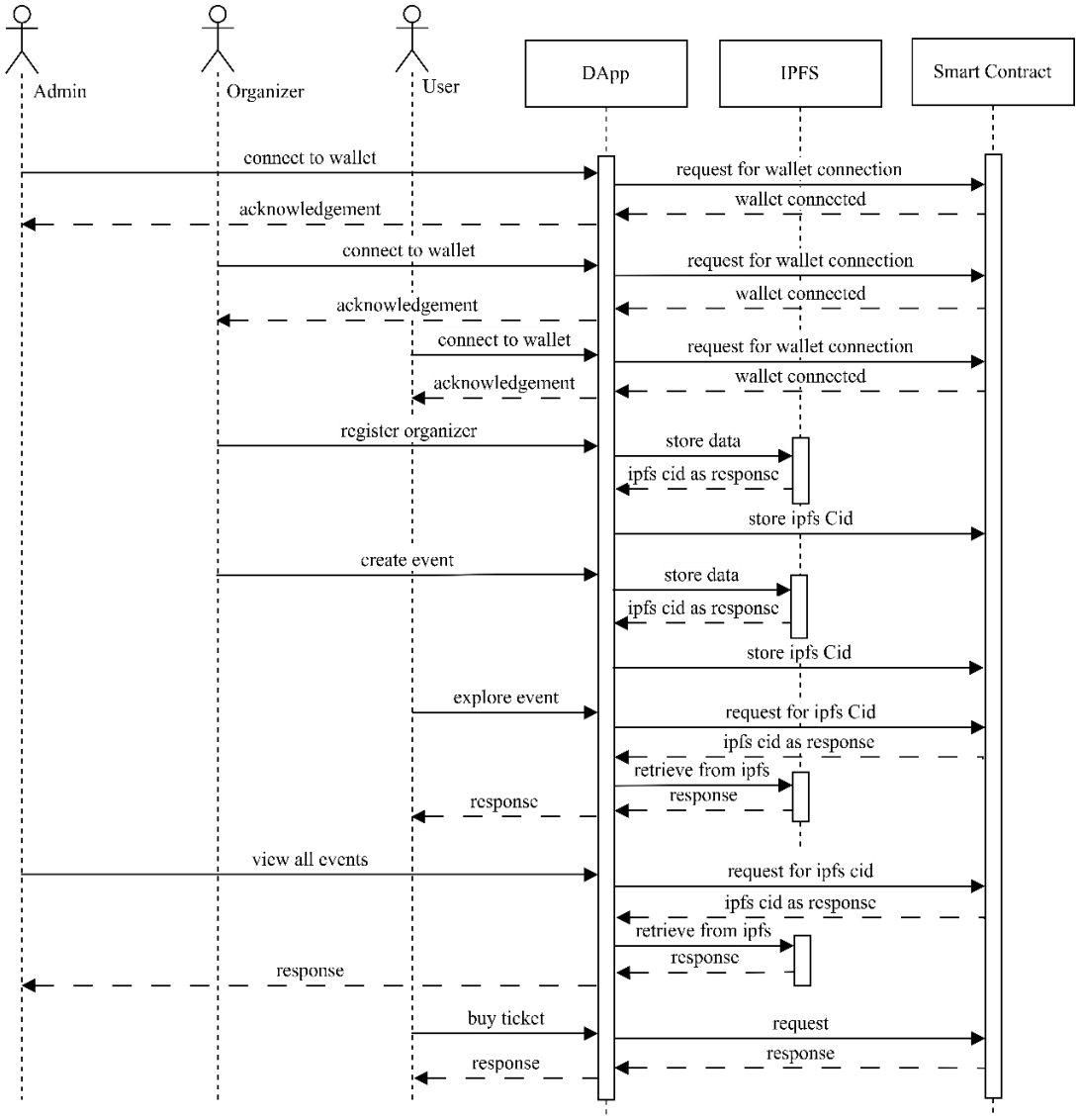


Figure 4.3 Sequence Diagram

The sequence diagram showcases the dynamic interactions between the actors (admin, organizer, and user) and key system components (dApp, IPFS, and smart contract) within the system. Initially, all actors connect their digital wallets to the dApp, which facilitates the connection with the smart contract. Once the wallet connection is established, acknowledgment is provided. For event registration by organizers, data is stored in IPFS, a distributed and secure storage network. The IPFS CID, which serves as a unique identifier for the stored data, is then returned and

stored by the dApp. When organizers create events, a similar process occurs, with the event data and associated IPFS CID stored for future reference. When admin views events or users explore available events, the dApp requests the event data from IPFS using the associated CID. This approach ensures that event data remains securely stored in a decentralized manner, leveraging the benefits of blockchain technology (smart contracts) for managing ticket issuance and ownership.

Component Diagram

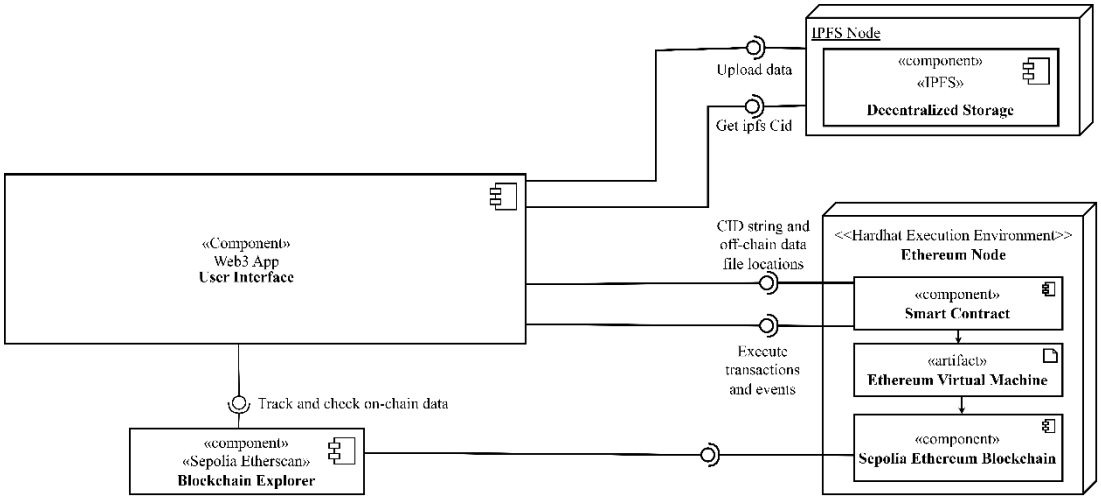


Figure 4.4: Component Diagram

The component diagram illustrates the architecture of the project utilizing IPFS for decentralized storage. The Web3 App serves as the user interface, leveraging the Hardhat Execution Environment to connect to the Ethereum blockchain. This environment simplifies smart contract development and deployment. The Ethereum Node validates transactions and stores the blockchain state, while the IPFS Node provides decentralized storage. Smart Contract defines project rules and data storage on the Ethereum blockchain. Sepolia Ethereum Blockchain serves as a testnet for development, and Sepolia Etherscan acts as a block explorer. This diagram showcases the project's components, interfaces, and dependencies, highlighting its decentralized and transparent nature.

4.2. Algorithm Details

Ticket Purchase Algorithm:

STEP 1: User selects a specific event through the dApp, providing the eventId and a total ticket to purchase.

STEP 2: Invoke smart contract function ` buyTicket(eventId , totalTicketsToBuy)`.

STEP 3: Contract validate event existence and available tickets.

If the event does not exist or has insufficient tickets, contract state are reverted.

STEP 4: Contract check user's balance for sufficient funds.

If the user has an insufficient balance, contract state is reverted.

STEP 5: If validation pass:

Transfer funds to the event creator.

Update the remaining tickets for the event.

Update the ticket holders mapping for the buyer.

STEP 6: Ticket Purchased successfully

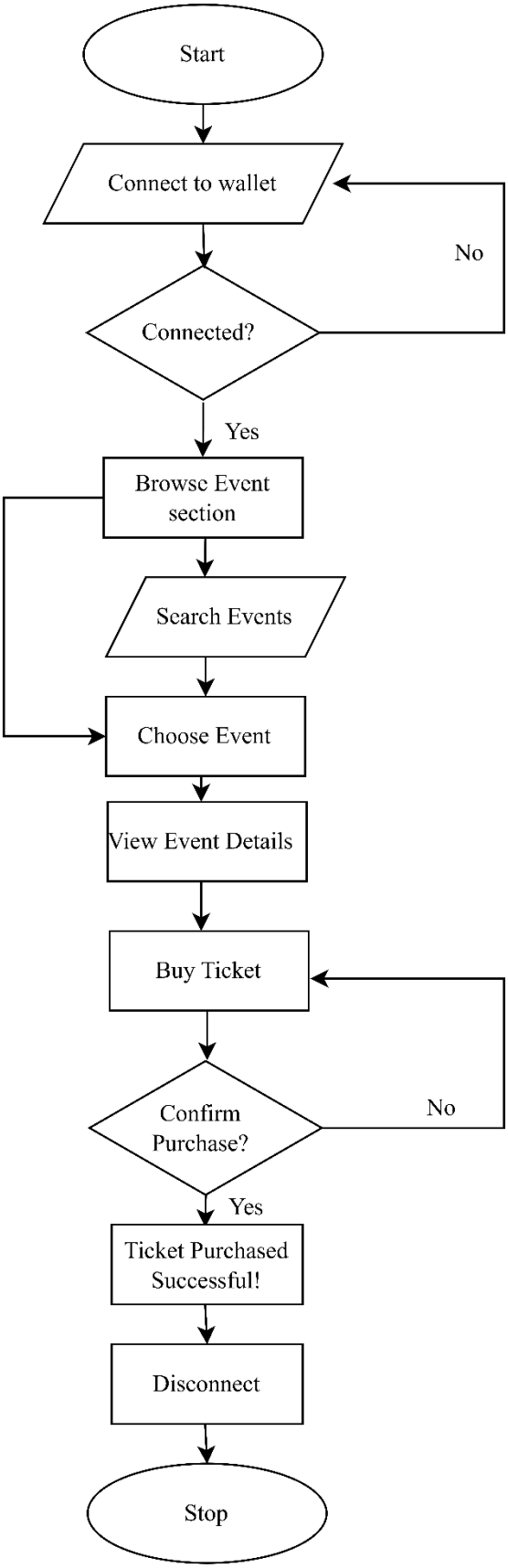


Figure 4.5 Flow chart of ticket purchasing

Event Creation Algorithm:

STEP1: Invoke smart contract function ` function createEvent(eventCid, totalTickets, price) ` when organizer click create event button in dApp.

STEP 2: Contract verify the creator is a registered organizer

If the creator is not a registered organizer, the contract state is reverted.

STEP 3: Contract checks that the event CID is not empty

If the event CID is empty, the contract state is reverted.

STEP 4: Calculate the event creation fee based on total tickets and price.

STEP 5: Contract checks the sent fund is equal to or greater than the event creation fee.

If the sent ether is insufficient, the contract state is reverted.

STEP 6: If validation pass:

Transfer the event creation fee to the owner.

Transfer excess ether back to the organizer.

Assign the event ID to the event.

Create and store the event details in the event mapping.

STEP 7: Event Created successfully

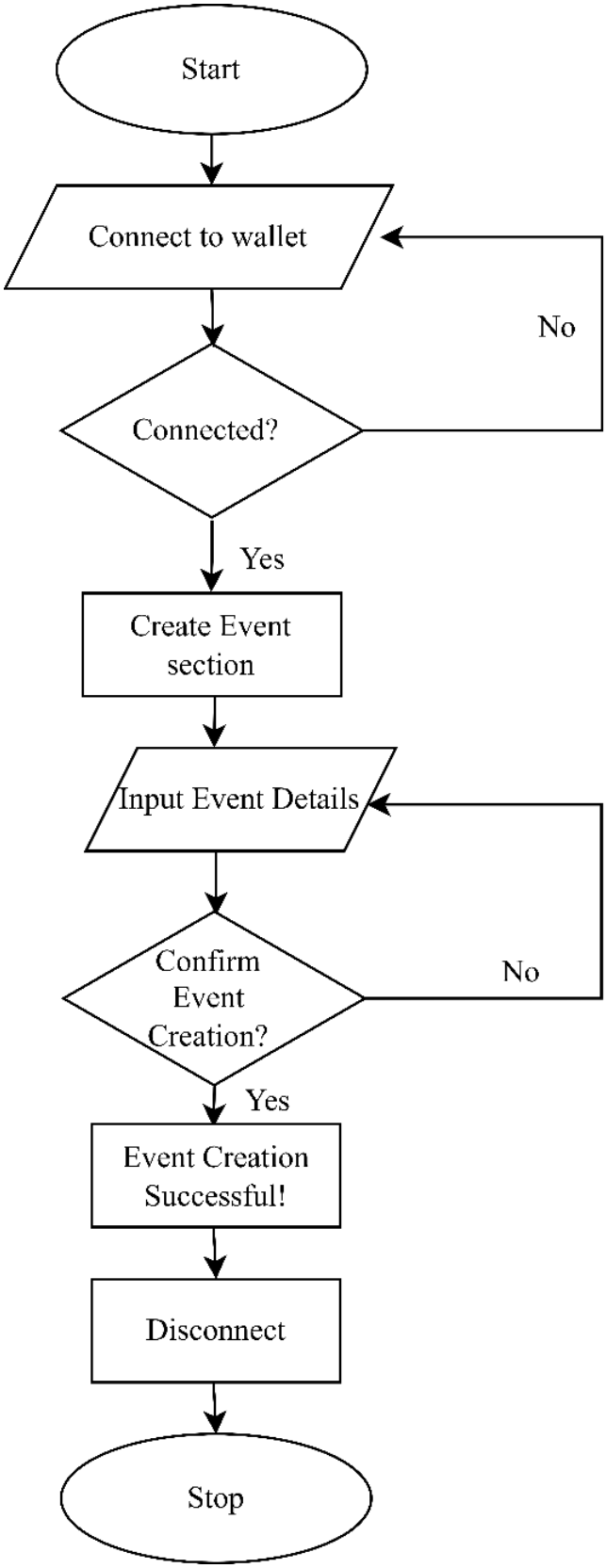


Figure 4.6 Flow chart of event creation

CHAPTER 5
IMPLEMENTATION AND TESTING

5.1. Implementation

In the project, implementation involves coding smart contracts, integrating the application UI with the smart contracts, and thorough testing for functionality. This phase ensures that users can interact with the system smoothly, including purchasing tickets and creating events.

5.1.1. Tools Used

Table 5.1 : Tools Used

Tools	**Purpose**
React and Ether.js	Frontend Library
Solidity	Backend Development
VS Code	IDE
Hardhat	Ethereum Development Environment
IPFS	File Sharing and Storage Protocol
Git and GitHub	Version control and collaborative storing of code
Bootstrap	Styling

5.2. Testing

Testing is essential in the project to ensure quality and functionality. It involves executing components like smart contracts and the dApp to identify errors and ensure compliance with requirements. This rigorous testing guarantees a reliable and user-friendly ticketing platform.

5.2.1. Test Cases for Unit Testing:

In the EthTix project, unit testing focuses on evaluating individual software components like smart contracts and user interfaces to ensure they perform as intended. This approach ensures the correctness of each component, helping to identify and fix any errors or issues early in the development process.

Test Case: Wallet Connection

Table 5.2: Wallet Connection

S.N.	Description	Expected Result	Actual Result	Test Result
1	Connect wallet with valid credentials. Test Wallet Address: "0xfEBdd78e5e478Fe27D55C5B18771581448523D71"	Wallet is connected successfully	Wallet is connected successfully	**Pass**
2	Connect wallet with invalid credentials. Test Wallet Address: ""	Error message: "Please install your wallet to connect wallet"	Error message: "Please install your wallet to connect wallet"	**Pass**

Test Case: Organizer Registration

Table 5.3: Organizer Registration

S.N.	Description	Expected Result	Actual Result	Test Result
1	Register an event organizer with a valid CID. Test CID: "QmRRvXSHVXxtbbjBg2Urq6mtCBTAdLbxriSHXXgC4WpQZB"	Event organizer is registered successfully	Event organizer is registered successfully	**Pass**
2	Register an event organizer with an empty CID. Test CID: ""	Error message: "CID cannot be empty"	Error message: "CID cannot be empty"	**Pass**

Test Case: Organizer Status Check

Table 5.4: Organizer Status Check

S.N.	Description	Expected Result	Actual Result	Test Result
1	Check if a user is an event organizer. Test Input Address: "0xfEBdd78e5e478Fe27D55C5B18771581448523D71"	Returns true if the user is an event organizer, false otherwise	Returns true if the user is an event organizer, false otherwise	**Pass**

Test Case: Event Creation

Table 5.5: Event Creation

S.N.	Description	Expected Result	Actual Result	Test Result
1	Create an event with valid details. Test Data: ipfsCid: "QmVPcDBXAkuwbmK L3bTDCUVEQhxE6pVAEUTDP bR9WdU7qF", totalTickets: 50, priceInWei: 100000000000000, msg.sender: "0xfEBdd78e5e478Fe27D55C5B1 8771581448523D71"	Event is successfully created	Event is successfully created	**Pass**
2	Create an event with an empty CID Test Data: ipfsCid: "", totalTickets: 50, priceInWei: 100000000000000, msg.sender: "0xfEBdd78e5e478Fe27D55C5B1 8771581448523D71"	Error message:" Event CID should not be empty"	Error message:" Event CID should not be empty"	**Pass**

S.N.	Description	Expected Result	Actual Result	Test Result
3	Create an event with total tickets less than or equal to 0. Test Data: ipfsCid: “QmVPcDBXAkuwbmKL3bTDCUVEQhxE6pVAEUTDPbR9WdU7qF”, totalTickets: -5, priceInWei: 100000000000000, msg.sender: “0xfEBdd78e5e478Fe27D55C5B18771581448523D71”	Error message: “Total tickets should be greater than 0”	Error message: “Total tickets should be greater than 0”	**Pass**
4	Create an event as a non-organizer Test Data: ipfsCid: “QmVPcDBXAkuwbmKL3bTDCUVEQhxE6pVAEUTDPbR9WdU7qF”, totalTickets: 50, priceInWei: 100000000000000, msg.sender: “0xF415ab7c5f701cC0f349beb5A349Fa6987e8409e”	Error message:” Only organizer can create event”	Error message:” Only organizer can create event”	**Pass**

S.N.	Description	Expected Result	Actual Result	Test Result
5	Create an event as the owner Test Data: ipfsCid: “QmVPcDBXAkuwbmK L3bTDCUVEQhxE6pVAEUTDP bR9WdU7qF”, totalTickets: 50, priceInWei: 100000000000000, msg.sender: “0x70f03230c919C1D49Ab34cD E6EbEc37331cE57BE”	Error message:” Owner cannot create event”	Error message:” Owner cannot create event”	**Pass**

Test Case: Ticket Purchase

Table 5.6: Ticket Purchase

S.N.	Description	Expected Result	Actual Result	Test Result
1	Buy tickets for an existing event with enough tickets left. Test Data: eventId: 1, totalTicketsToBuy:1, remTickets: 50, msg.sender: “0x0243ed028EE41A5515F4Bf94 8280AfF794592Ba0”, msg.value: 200000000000000, totalPrice: 100000000000000	Tickets are purchased successfully	Tickets are purchased successfully	**Pass**

S.N.	Description	Expected Result	Actual Result	Test Result
2	Buy tickets for an event that does not exist. Test Data: eventId: , totalTicketsToBuy:1, remTickets: 50, msg.sender: “0x0243ed028EE41A5515F4Bf948280AfF794592Ba0”, msg.value: 200000000000000, totalPrice: 100000000000000	Error message: “Event does not exist”	Error message: “Event does not exist”	**Pass**
3	Buy more tickets than available. Test Data: eventId: 1, totalTicketsToBuy:6, remTickets: 5, msg.sender: “0x0243ed028EE41A5515F4Bf948280AfF794592Ba0” msg.value: 800000000000000, totalPrice: 600000000000000	Error message: “Not enough tickets left”	Error message: “Not enough tickets left”	**Pass**
4	Buy tickets as an organizer. Test Data: eventId: 1, msg.sender: “0xfEBdd78e5e478Fe27D55C5B18771581448523D71” creater: “0xfEBdd78e5e478Fe27D55C5B18771581448523D71”	Error message: “Organizers cannot buy tickets for their own event”	Error message: “Organizers cannot buy tickets for their own event”	**Pass**

S.N.	Description	Expected Result	Actual Result	Test Result
5	Buy tickets with not enough ether. Test Data: eventId: 1, totalTicketsToBuy:2, remTickets: 5, msg.sender: "0x0243ed028EE41A5515F4Bf94 8280AfF794592Ba0" msg.value: 100000000000000, totalPrice: 200000000000000	Error message: "Not enough ether sent"	Error message: "Not enough ether sent"	**Pass**

Test Case: Ticket Information

Table 5.7: Ticket Information

S.N.	Description	Expected Result	Actual Result	Test Result
1	Get tickets forparticular user. Test Data: msg.sender: "0x0243ed028EE41A5515F4Bf94 8280AfF794592Ba0"	Returns tickets of the user. ["0x0243ed028EE41A5515F4Bf94 8280AfF794592Ba0", "QmVPcDBXAkuwbmKL3bTDCUVEQhxE6pVAEUTDPbR9WdU7qF",1,3/14/2024,7:59, 100000000000000]	Returns tickets of the user. ["0x0243ed028EE41A5515F4Bf94 8280AfF794592Ba0", "QmVPcDBXAkuwbmKL3bTDCUVEQhxE6pVAEUTDPbR9WdU7qF",1,3/14/2024,7:59, 100000000000000]	**Pass**

Test Case: Organizer Information

Table 5.8: Organizer Information

S.N.	Description	Expected Result	Actual Result	Test Result
1	Get all registered event organizers. Test Data: msg.sender: "0x75fF4eA5947 A9f76ffC0a47c6 Fb0CAc886FbF C37"	Returns array of all event organizers. ["0xfEBdd78e5e478 Fe27D55C5B187 71581448523D71"]	Returns array of all event organizers. ["0xfEBdd78e5e47 8Fe27D55C5B187 71581448523D71"]	**Pass**

Test Case: Ticket Holder Information

Table 5.9: Ticket Holder Information

S.N.	Description	Expected Result	Actual Result	Test Result
1	Get all addresses of ticket holders. Test Data: msg.sender: "0x75fF4eA5947 A9f76ffC0a47c6 Fb0CAc886FbF C37"	Returns array of all ticket holder addresses. ["0x0243ed028 EE41A5515F4Bf94 8280AfF794592Ba0"]	Returns array of all ticket holder addresses. ["0x0243ed028 EE41A5515F4Bf94 8280AfF794592Ba0"]	**Pass**

5.2.2. Test Cases for System Testing

System testing in the EthTix project verifies the entire decentralized ticketing system, ensuring that components like applications, smart contracts, and IPFS work together correctly. This testing confirms that users can perform actions like purchasing tickets and creating events seamlessly. It is conducted after integration testing to ensure the system functions as expected in a real-world environment.

Test Case: Overall System Functionalities

Table 5.10: Overall System Functionalities

S.N.	Description	Expected Result	Actual Result	Test Result
1	Register event organizer with valid CID	Event organizer is registered successfully	Event organizer is registered successfully	**Pass**
2	Check if registered event organizers are correctly identified as organizers	Returns true for registered event organizers, false for others	Returns true for registered event organizers, false for others	**Pass**
3	Create events with valid details by different organizers	Events are created successfully	Events are created successfully	**Pass**
4	Buy tickets for various events by different users	Tickets are purchased successfully.	Tickets are purchased successfully.	**Pass**

S.N.	Description	Expected Result	Actual Result	Test Result
5	Retrieve tickets of various users	Returns tickets of respective users	Returns tickets of respective users	**Pass**
6	Retrieve all registered event organizers	Returns array of all registered event organizers	Returns array of all registered event organizers	**Pass**
7	Retrieve CIDs of all registered event organizers	Returns array of all registered event organizer CIDs	Returns array of all registered event organizer CIDs	**Pass**
8	Retrieve addresses of all ticket holders	Returns array of all ticket holder addresses	Returns array of all ticket holder addresses	**Pass**

5.3. Result Analysis

In the project, result analysis involves evaluating the outcomes of various testing phases, including unit testing, integration testing and system testing. This analysis aims to determine the overall quality and functionality of the decentralized ticketing system. Results from unit testing ensure that individual components like smart contracts and user interfaces perform as intended. Integration testing results verify that these components integrate correctly. System testing results confirm that the entire system works together seamlessly, allowing users to purchase tickets and create events without issues. Overall, result analysis ensures that the EthTix project meets its specified requirements and functions reliably for end users.

CHAPTER 6
CONCLUSION AND RECOMMENDATION

6.1. Conclusion

The project aimed to transform ticketing through the application of Ethereum blockchain technology. This initiative provided valuable insights about the blockchain, improving proficiency in the use of smart contracts. The primary goal was to create a user-friendly decentralized app that leverages blockchain's benefits of transparency, security, and decentralization. The developed application illustrates blockchain's potential to enhance comprehensive solution for creating events and selling events tickets. Despite challenges such as the need for users to understand wallets and cryptocurrencies, the project establishes a solid foundation for a decentralized ticketing platform.

In conclusion, the project's accomplishments highlight the transformative impact of blockchain ticketing by simplifying the processes and providing users with more control over the tickets paving the way for efficient user-centric ticketing industry.

6.2. Future Recommendations

EthTix aspire to reshape the landscape of the ticketing industry using Ethereum blockchain technology providing a secure, transparent, and efficient platform for the creation and purchase of event tickets. Nevertheless, to address existing constraints and enhance the platform's capabilities following implementations are recommended:

Integration of Layer 2 Scaling Solutions:

Investigate the incorporation of Layer 2 scaling solutions like Optimistic rollups, zk-Rollups, or sidechains such as Polygon which can address scalability issues, reduce transaction costs, and improve the overall efficiency of the ticketing platform.

Diversification of Wallets Options:

Extend support beyond MetaMask by integrating various wallets, such as Guarda Wallet, Exodus, etc. to offer users a broader choice in selecting wallets based on their needs.

Ticket Transfer Functionality

Introducing the advanced ticket transfer features, allowing users to easily transfer tickets to others. Smart contracts can facilitate secure and transparent ticket ownership transfer on the blockchain.

Interoperability with other Blockchains:

Implementing interoperability solutions to enable cross-chain functionality, allowing users to interact with the ticketing platform using different blockchain networks.

REFERENCES

[1] P. A. &. G. D. Pavlou, "Building effective online marketplaces with institution-based trust," *Information Systems Research,* pp. 37-59, 2004.

[2] H. K. Gimun Kim, "The causal relationship between risk and trust in the online marketplace: A bidirectional perspective," *Computers in Human Behavior,* vol. 55 (Part B), pp. 1020-1029, February 2016.

[3] S. N. (pseudonymous), "Bitcoin: a peer-to-peer electronic cash system," Online at https://bitcoin.org/bit.

[4] G. Wood, "Ethereum: a secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper,* pp. 1-32.

[5] C. S. S. T. Suporn Pongnumkul, "Performance Analysis of Private Blockchain," in *26th International Conference on Computer Communication and Networks (ICCCN),* 2017.

[6] G. D. M. M. M. O. R. Tonelli, "Smart Contracts Software Metrics: a First Study," arXiv preprint, 2018.

[7] M. W. J. W. D. C. S. (. a. w. V. U. eng Zhang, "Metrics for Assessing Blockchain-based Healthcare Decentralized Apps," 2018.

[8] M. W. a. U. Zdun, "Smart Contracts: Security Patterns in the Ethereum Ecosystem and Solidity".

[9] S. C. M. Sina Rafati Niya, "DeTi: A Decentralized Ticketing Management Platform," Journal of Network and Systems Management, 2022.

[10] J. F. (. t. a. o. t. O. P. w. M. Liu, "Origin Protocol," https://www.originprotocol.com/en/whitepaper, 2019.

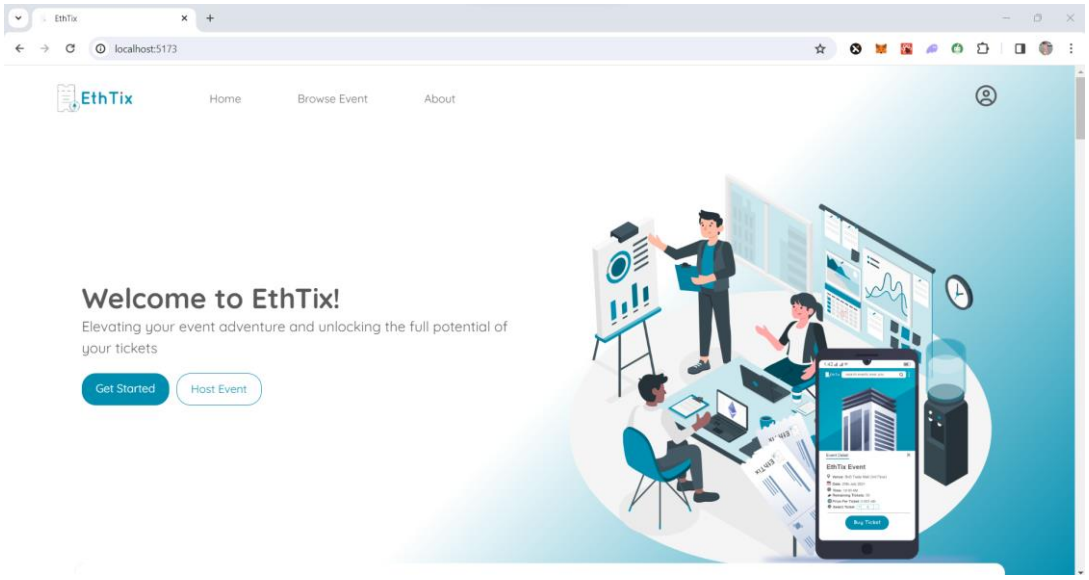
[11] G. F. T. (. o. t. whitepaper), "Guaranteed entrance token smart event ticketing protocol," https://get-protocol.io/files/GET-Whitepaper-GUTS-Tickets-latest.pdf, 2017.

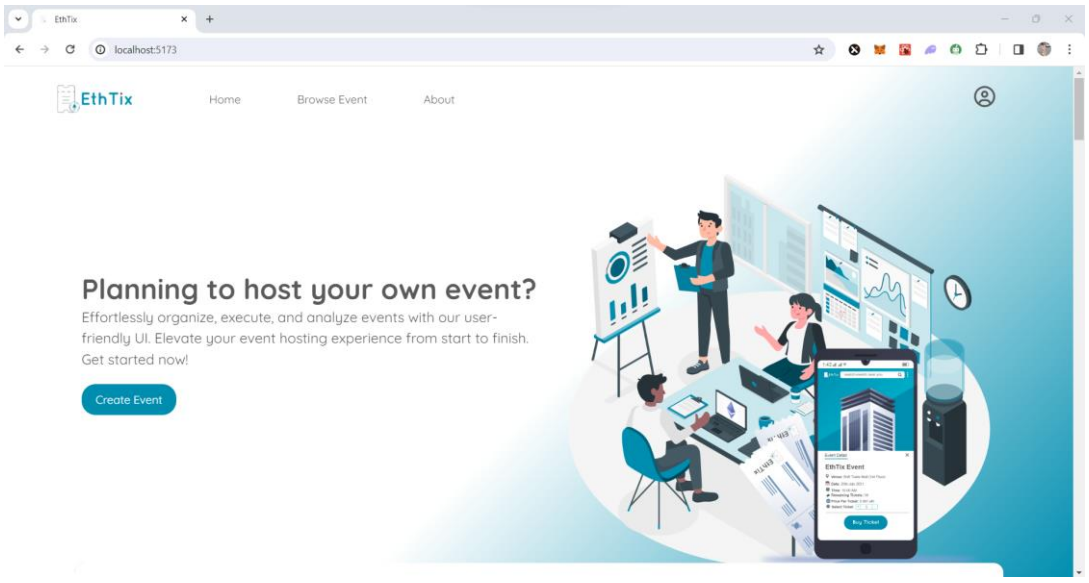
[12] B. (. t. a. o. t. whitepaper), "Blockparty: an event ticketing blockchain protocol," https://cms.goblockparty.com/wpcontent/uploads/2019/04/Blockparty-Event-Ticketing-Whitepaper-v-4.6.pdf, 2018.

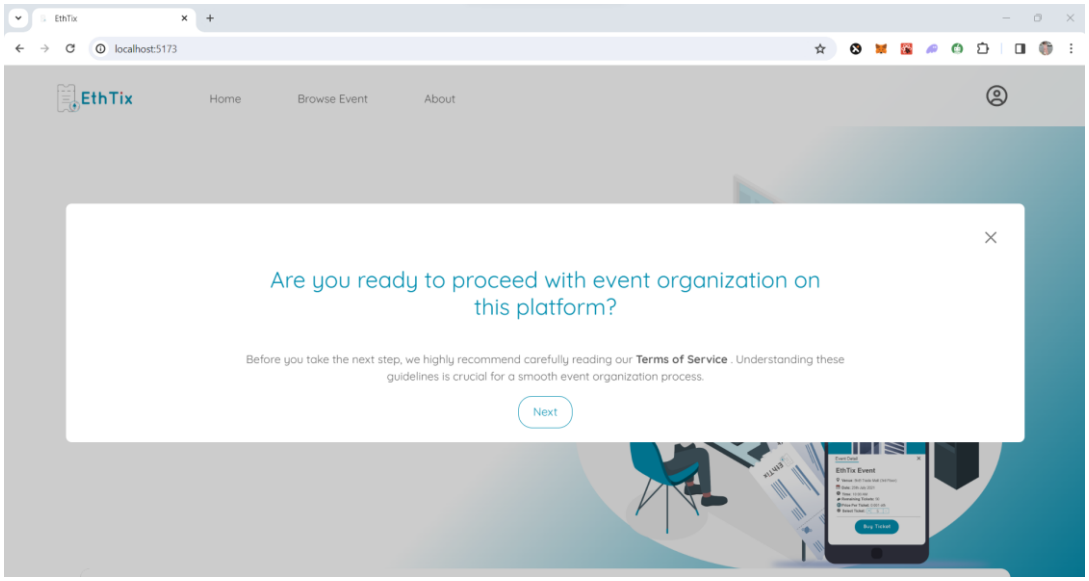
[13] Aventus, "Aventus: a layer-2 blockchain protocol that brings scalability, lower costs, and speed to Ethereum transactions," https://www.aventus.io/, 2021.

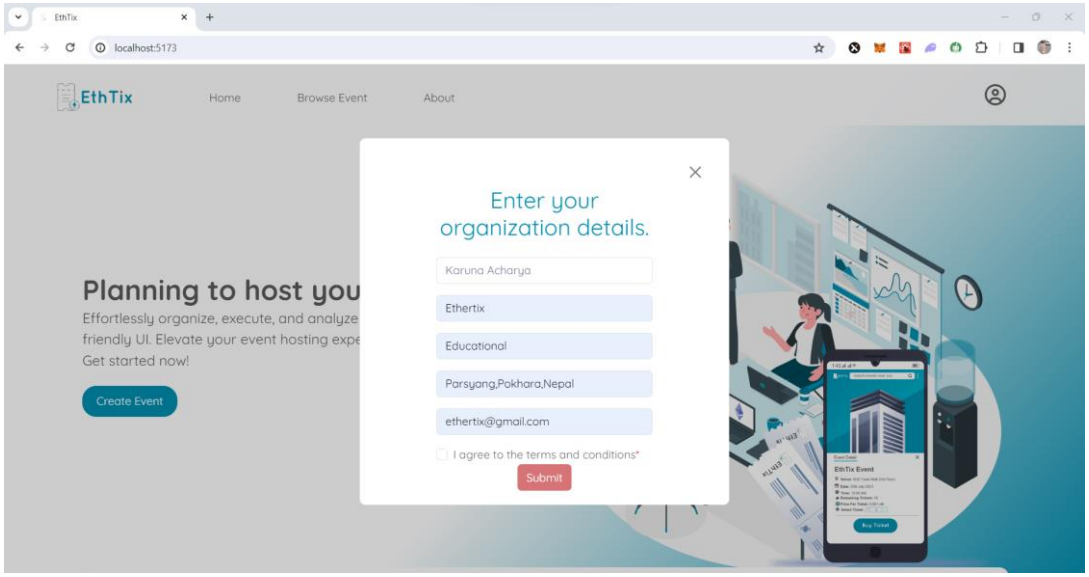
APPENDICES

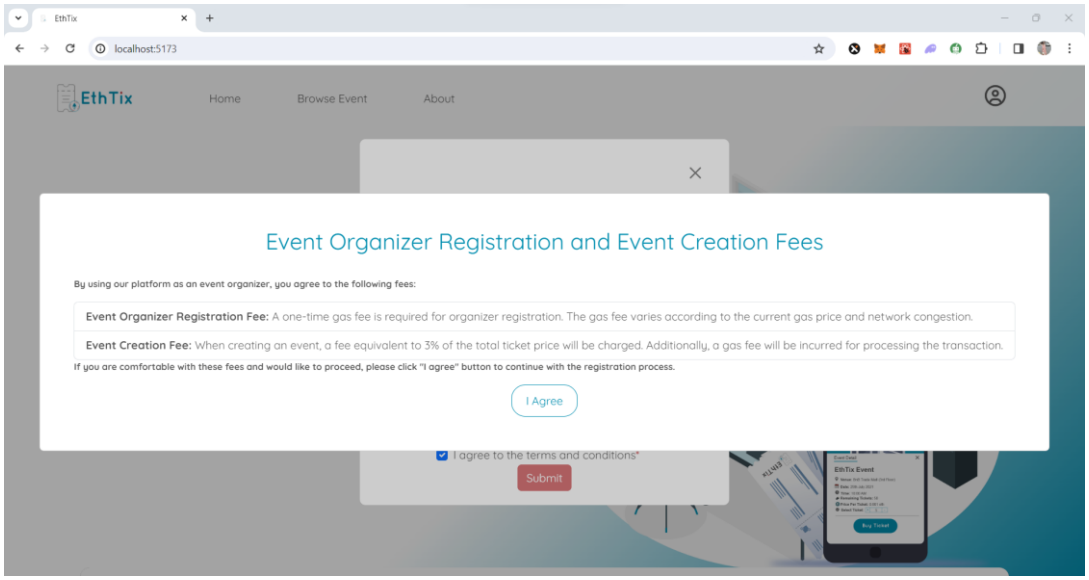
Event Organizer View

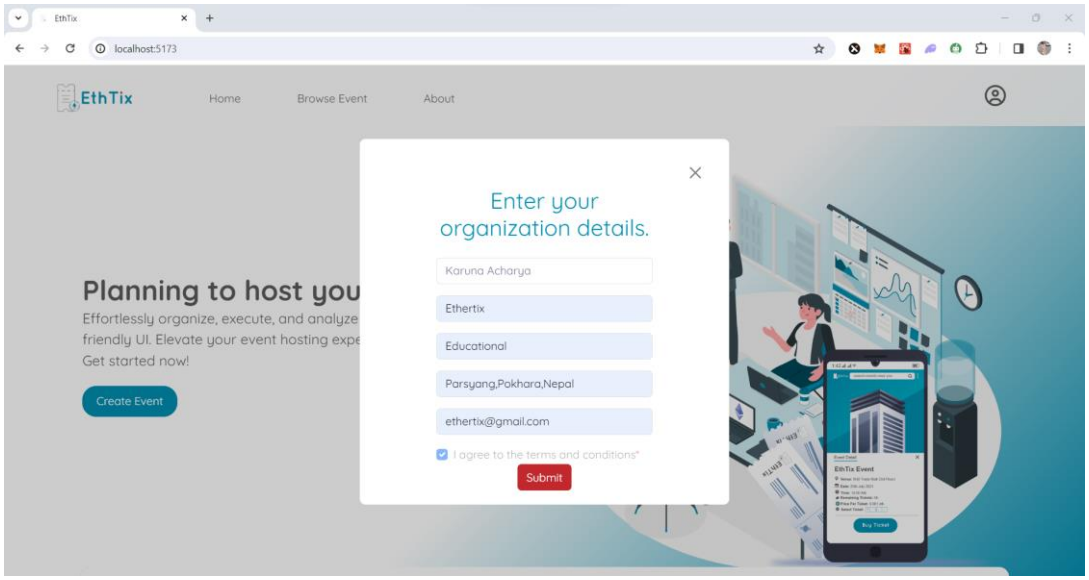


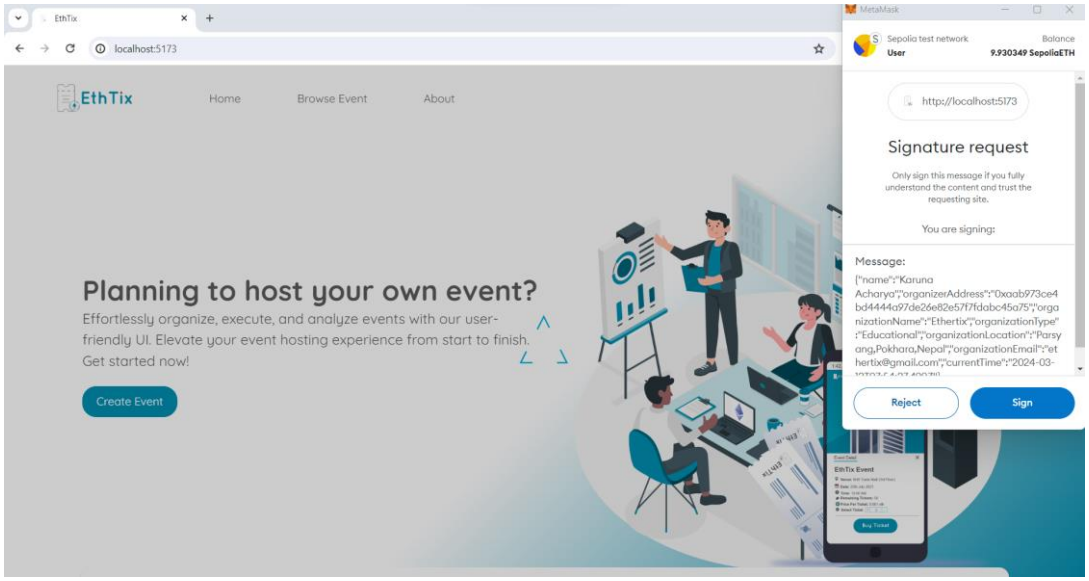


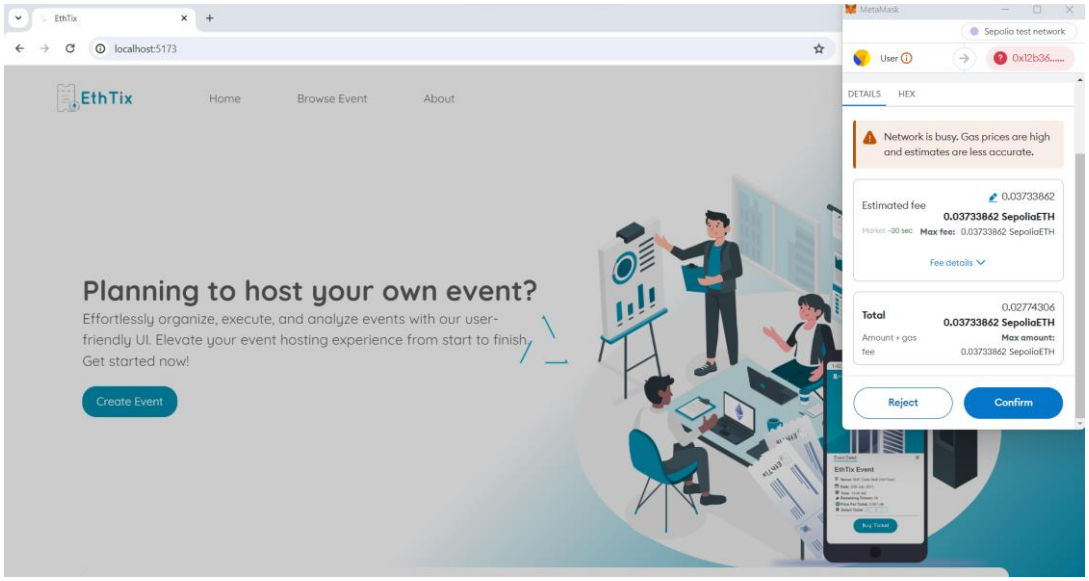

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Are you ready to proceed with event organization on this platform?
Before you take the next step, we highly recommend carefully reading our Terms of Service . Understanding these guidelines is crucial for a smooth event organization process.
Next

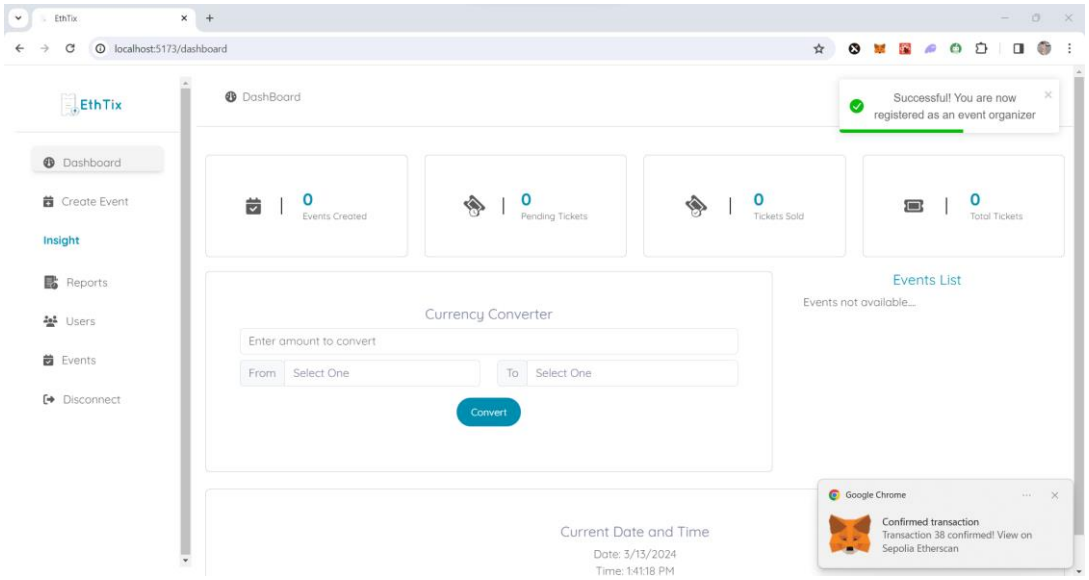

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Planning to host you
Effortlessly organize, execute, and analyze
friendly UI. Elevate your event hosting expe
Get started now!
Create Event
Enter your organization details.
Karuna Acharya
Ethertix
Educational
Parsyang,Pokhara,Nepal
ethertix@gmail.com
I agree to the terms and conditions*
Submit

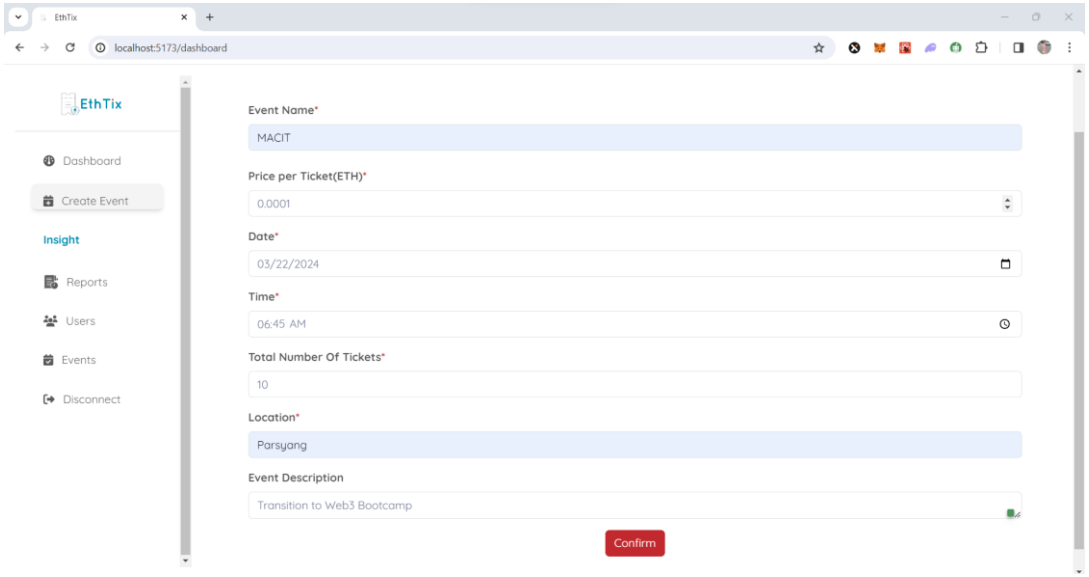

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Event Organizer Registration and Event Creation Fees
By using our platform as an event organizer, you agree to the following fees:
Event Organizer Registration Fee: A one-time gas fee is required for organizer registration. The gas fee varies according to the current gas price and network congestion.
Event Creation Fee: When creating an event, a fee equivalent to 3% of the total ticket price will be charged. Additionally, a gas fee will be incurred for processing the transaction.
If you are comfortable with these fees and would like to proceed, please click "I agree" button to continue with the registration process.
I Agree
I agree to the terms and conditions*
Submit

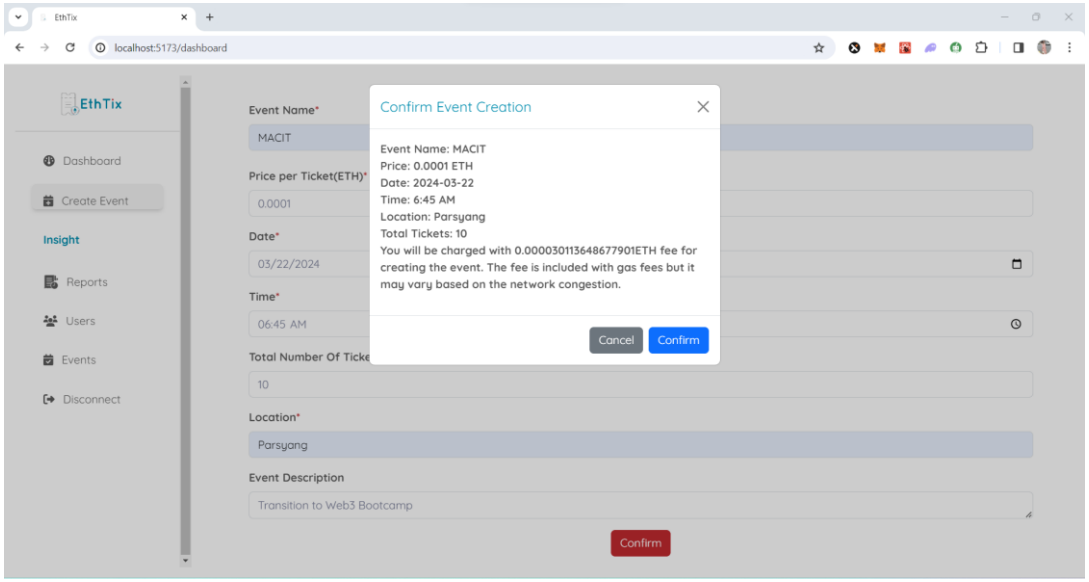

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Enter your organization details.
Karuna Acharya
Ethertix
Educational
Parsyang,Pokhara,Nepal
ethertix@gmail.com
I agree to the terms and conditions*
Submit
Planning to host you
Effortlessly organize, execute, and analyze
friendly UI. Elevate your event hosting expe
Get started now!
Create Event

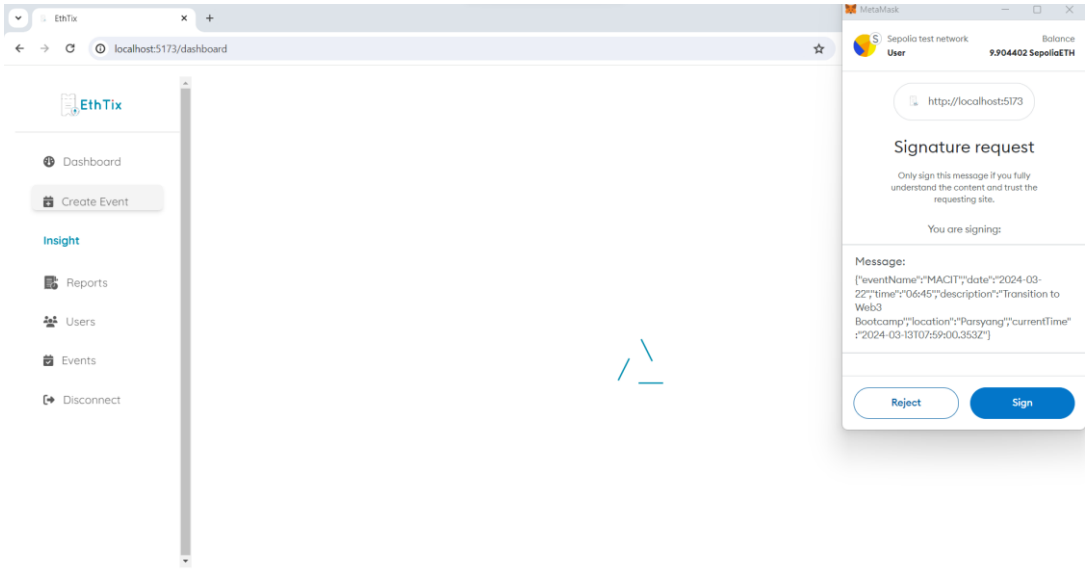

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Planning to host your own event?
Effortlessly organize, execute, and analyze events with our user-friendly UI. Elevate your event hosting experience from start to finish.
Get started now!
Create Event
MetaMask
Sepolia test network
User
Balance
9.930349 SepoliaETH
http://localhost:5173
Signature request
Only sign this message if you fully understand the content and trust the requesting site.
You are signing:
Message:
Reject
Sign

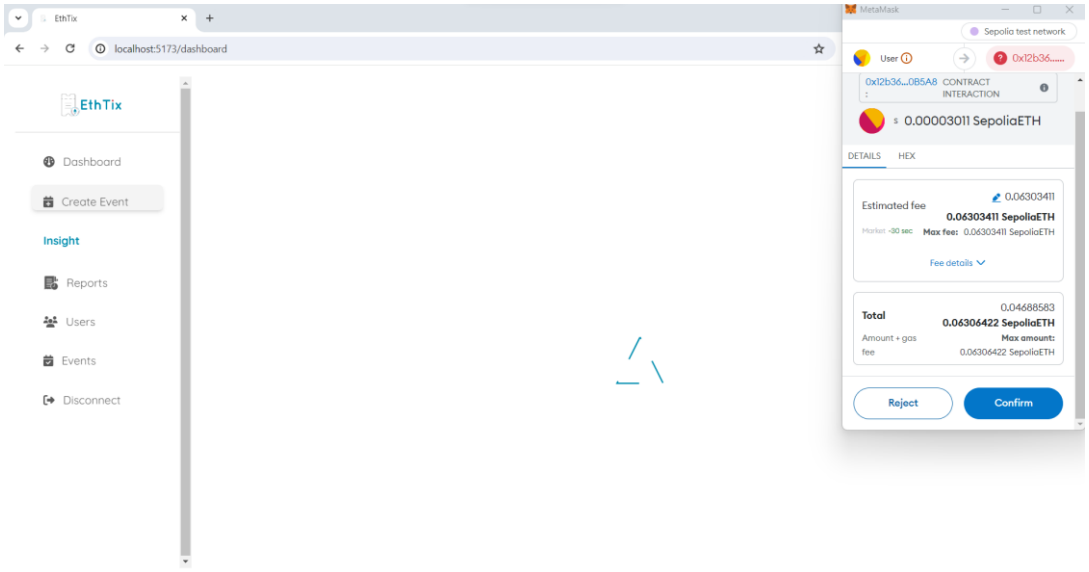

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Planning to host your own event?
Effortlessly organize, execute, and analyze events with our user-friendly UI. Elevate your event hosting experience from start to finish.
Get started now!
Create Event
MetaMask
Sepolia test network
User
DETAILS
HEX
Network is busy. Gas prices are high and estimates are less accurate.
Estimated fee
0.03733862
0.03733862 SepoliaETH
Max fee: 0.03733862 SepoliaETH
Fee details
Total
0.02774306
0.03733862 SepoliaETH
Max amount:
0.03733862 SepoliaETH
Reject
Confirm

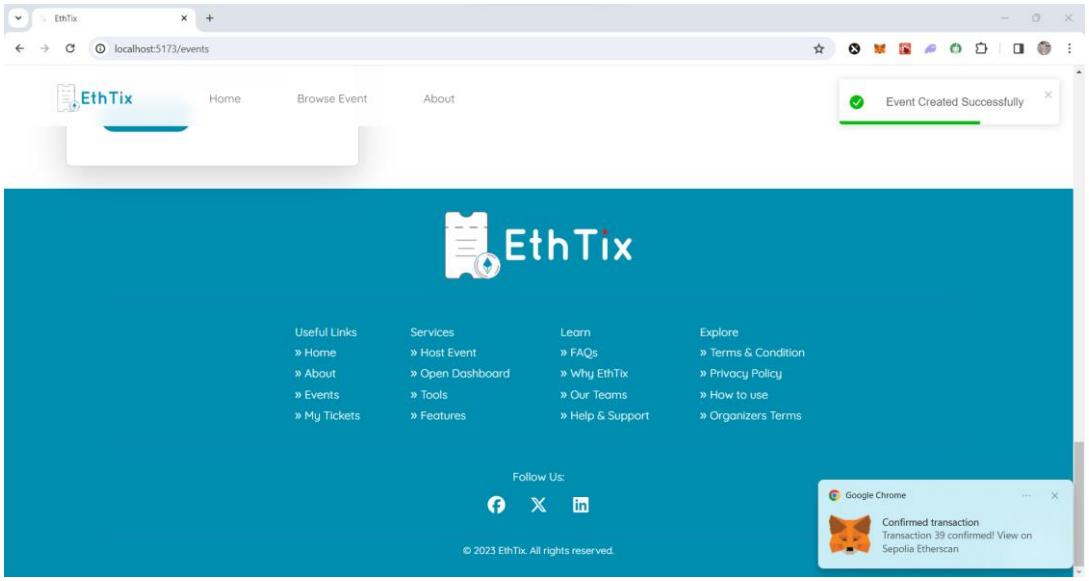

EthTix
localhost:5173/dashboard
EthTix
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
DashBoard
Successful! You are now registered as an event organizer
0
Events Created
0
Pending Tickets
0
Tickets Sold
0
Total Tickets
Currency Converter
Enter amount to convert
From
Select One
To
Select One
Convert
Events List
Events not available...
Current Date and Time
Date: 3/13/2024
Time: 1:41:18 PM
Google Chrome
Confirmed transaction
Transaction 38 confirmed! View on Sepolia Etherscan

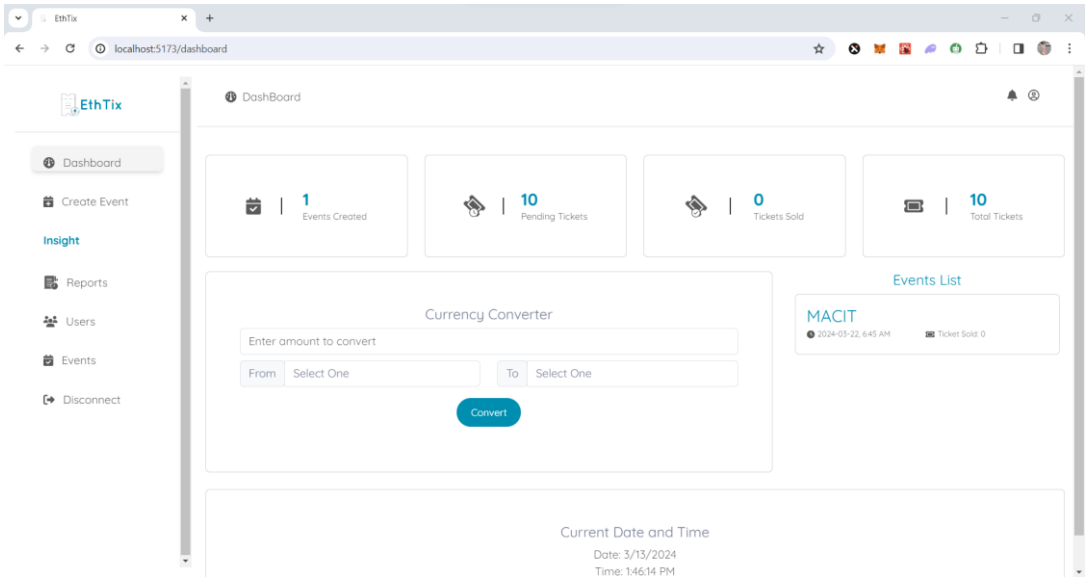

EthTix
localhost:5173/dashboard
EthTix
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
Event Name*
MACIT
Price per Ticket(ETH)*
0.0001
Date*
03/22/2024
Time*
06:45 AM
Total Number Of Tickets*
10
Location*
Parsyang
Event Description
Transition to Web3 Bootcamp
Confirm

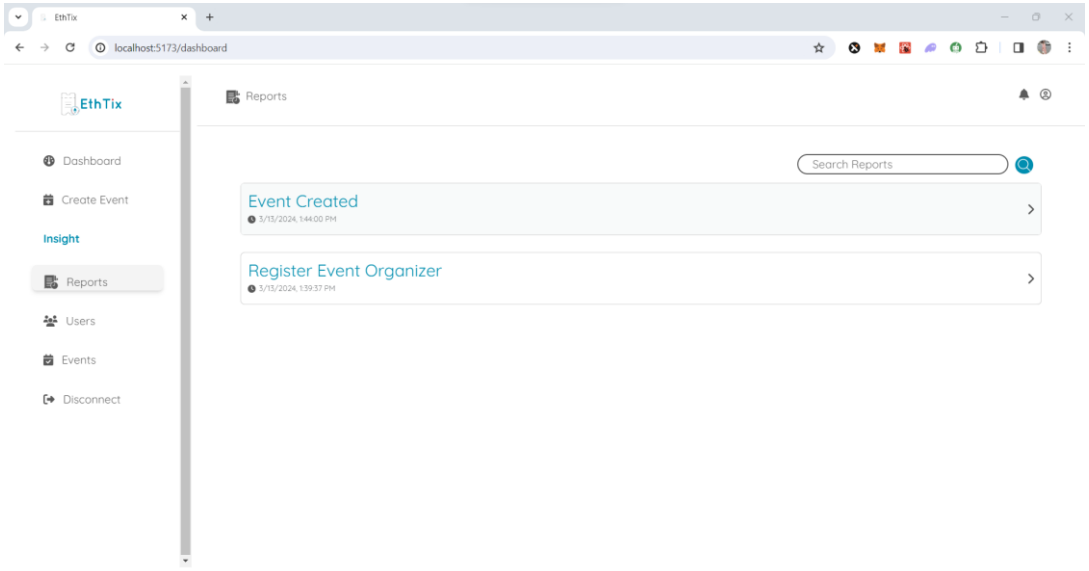

EthTix
localhost:5173/dashboard
EthTix
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
Event Name*
MACIT
Price per Ticket(ETH)*
0.0001
Date*
03/22/2024
Time*
06:45 AM
Total Number Of Ticke
10
Location*
Parsyang
Event Description
Transition to Web3 Bootcamp
Confirm
Confirm Event Creation
Event Name: MACIT
Price: 0.0001 ETH
Date: 2024-03-22
Time: 6:45 AM
Location: Parsyang
Total Tickets: 10
You will be charged with 0.000030113648677901ETH fee for creating the event. The fee is included with gas fees but it may vary based on the network congestion.
Cancel
Confirm

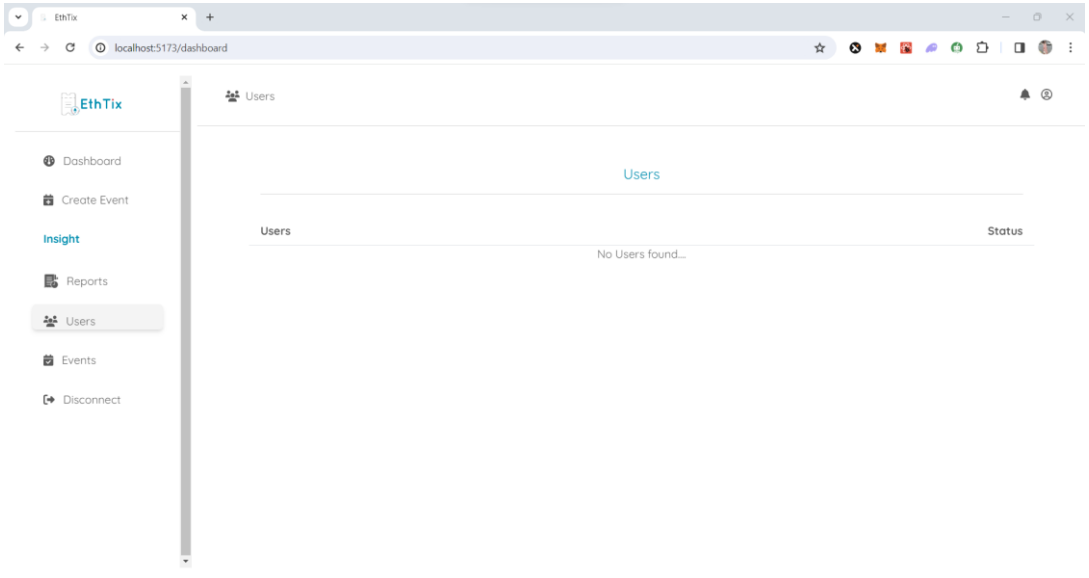

EthTix
localhost:5173/dashboard
EthTix
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
MetaMask
Sepolia test network
User
Balance
9.904402 SepoliaETH
http://localhost:5173
Signature request
Only sign this message if you fully understand the content and trust the requesting site.
You are signing:
Message:
{"eventName":"MACIT","date":"2024-03-22","time":"06:45","description":"Transition to Web3 Bootcamp","location":"Parsyang","currentTime":"2024-03-13T07:59:00.353Z"}
Reject
Sign

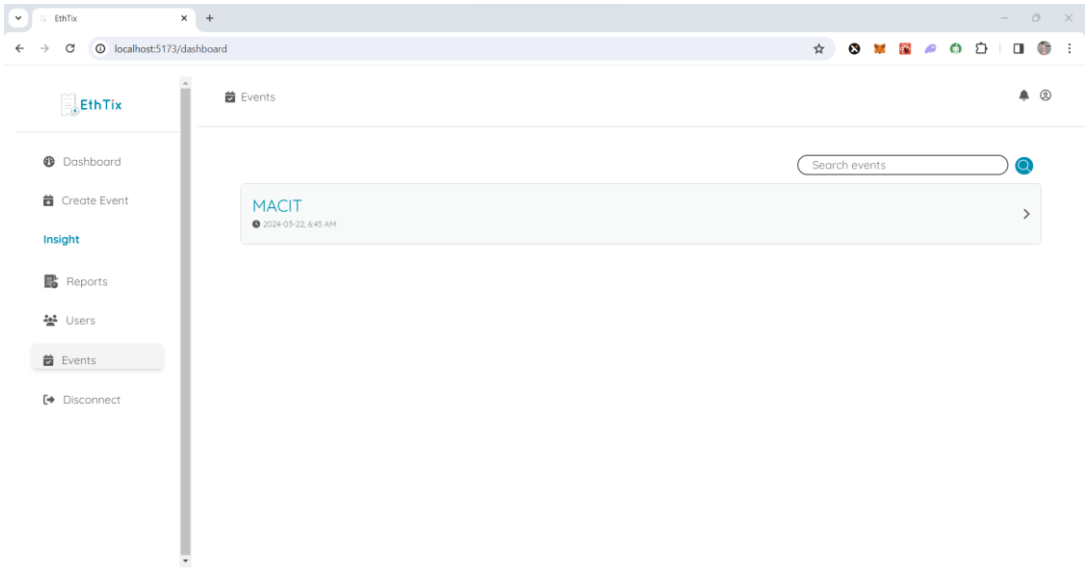

EthTix
localhost:5173/dashboard
EthTix
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
MetaMask
Sepolia test network
User
0x12b36.....
0x12b36...0B5A8 CONTRACT INTERACTION
0.00003011 SepoliaETH
DETAILS
HEX
Estimated fee
0.06303411
0.06303411 SepoliaETH
Max fee: 0.06303411 SepoliaETH
Fee details
Total
0.04688583
0.06306422 SepoliaETH
Amount + gas fee
Max amount:
0.06306422 SepoliaETH
Reject
Confirm


EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
Event Created Successfully
EthTix
Useful Links
» Home
» About
» Events
» My Tickets
Services
» Host Event
» Open Dashboard
» Tools
» Features
Learn
» FAQs
» Why EthTix
» Our Teams
» Help & Support
Explore
» Terms & Condition
» Privacy Policy
» How to use
» Organizers Terms
Follow Us:
© 2023 EthTix. All rights reserved.
Google Chrome
Confirmed transaction
Transaction 39 confirmed! View on Sepolia Etherscan

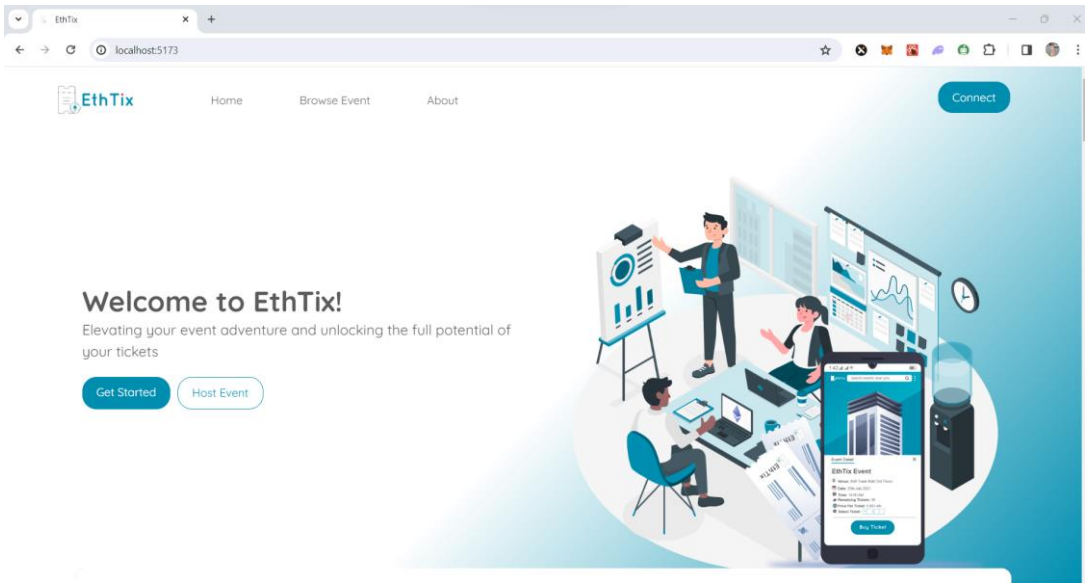

EthTix
localhost:5173/dashboard
EthTix
DashBoard
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
1
Events Created
10
Pending Tickets
0
Tickets Sold
10
Total Tickets
Currency Converter
Enter amount to convert
From
Select One
To
Select One
Convert
Events List
MACIT
Current Date and Time
Date: 3/13/2024
Time: 1:46:14 PM

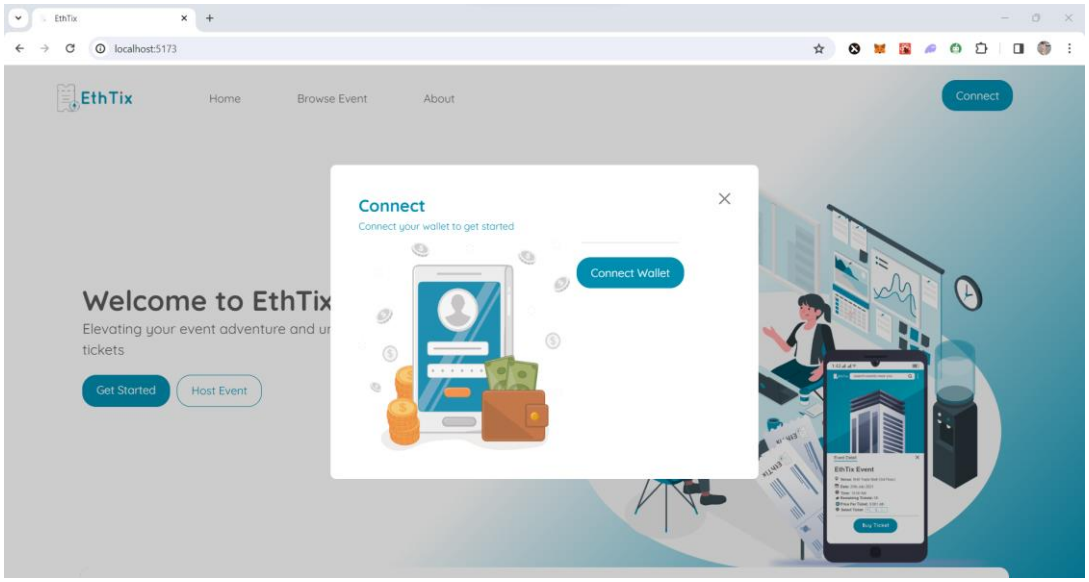

EthTix
localhost:5173/dashboard
EthTix
Reports
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
Search Reports
Event Created
Register Event Organizer

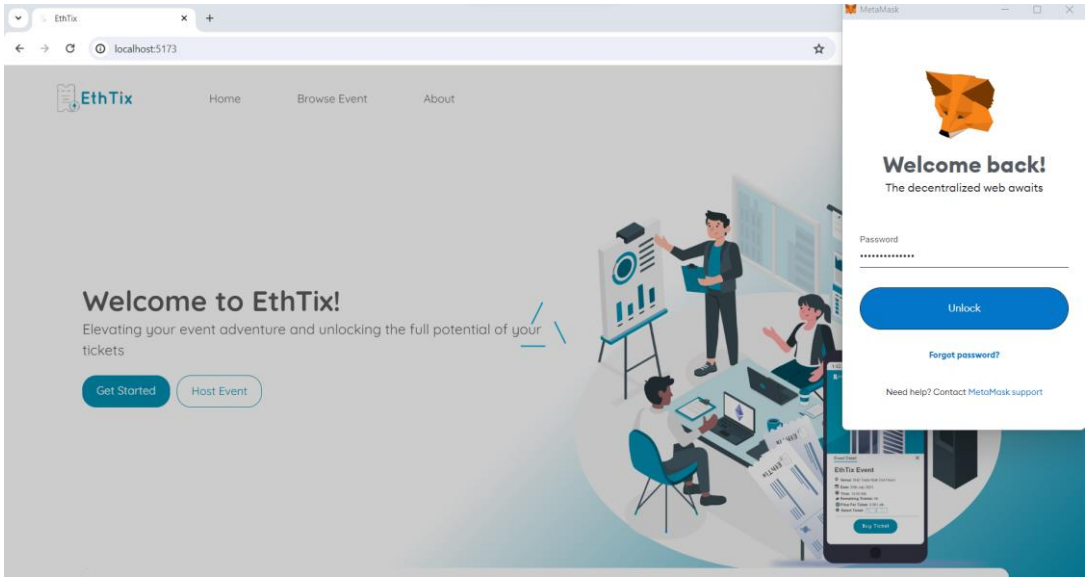

EthTix
localhost:5173/dashboard
EthTix
Users
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
Users
Users
Status
No Users found...

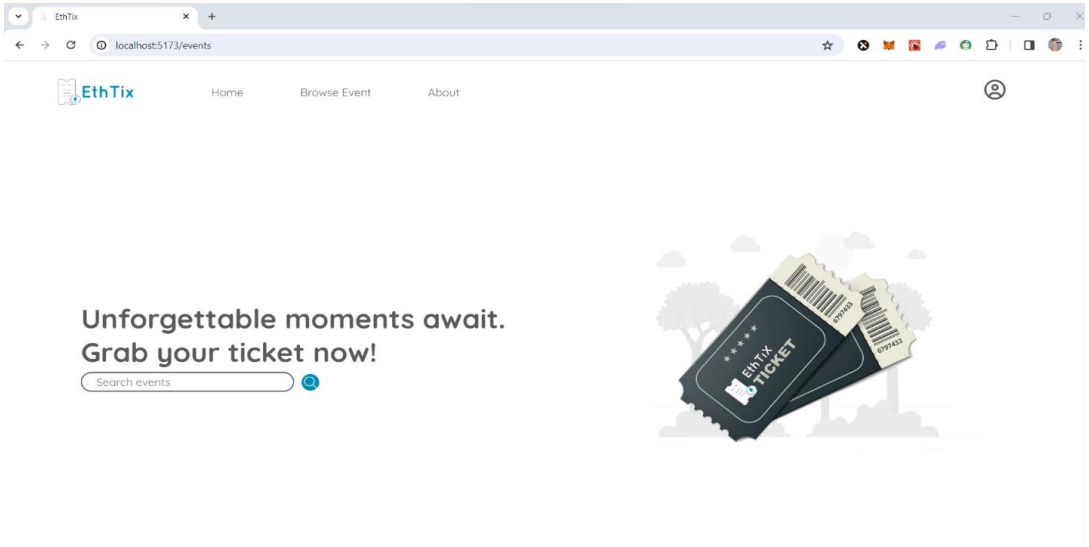

EthTix
localhost:5173/dashboard
EthTix
Dashboard
Create Event
Insight
Reports
Users
Events
Disconnect
Events
Search events
MACIT
2024-03-22, 6:45 AM

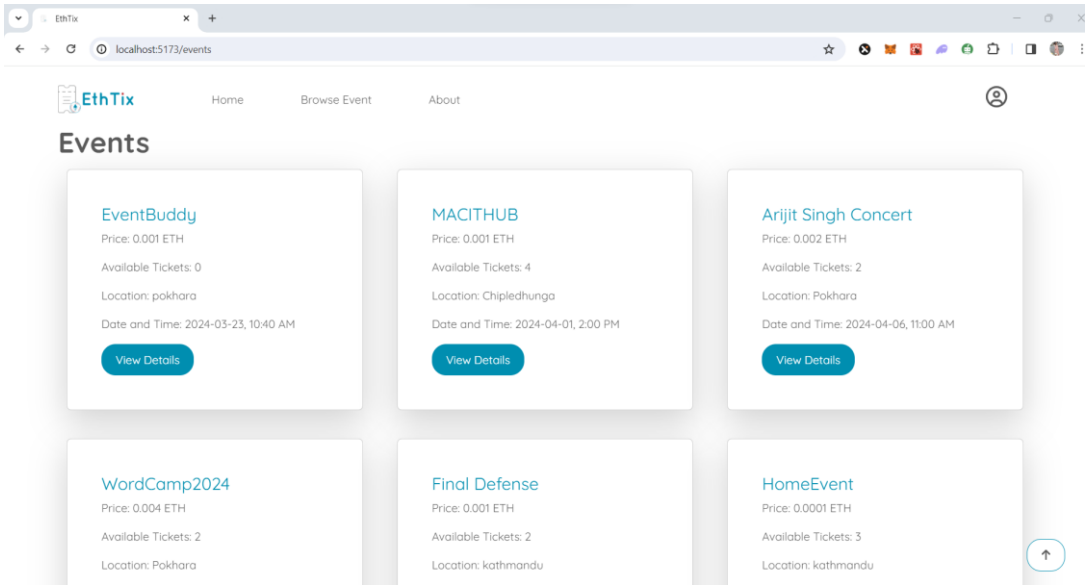
User View

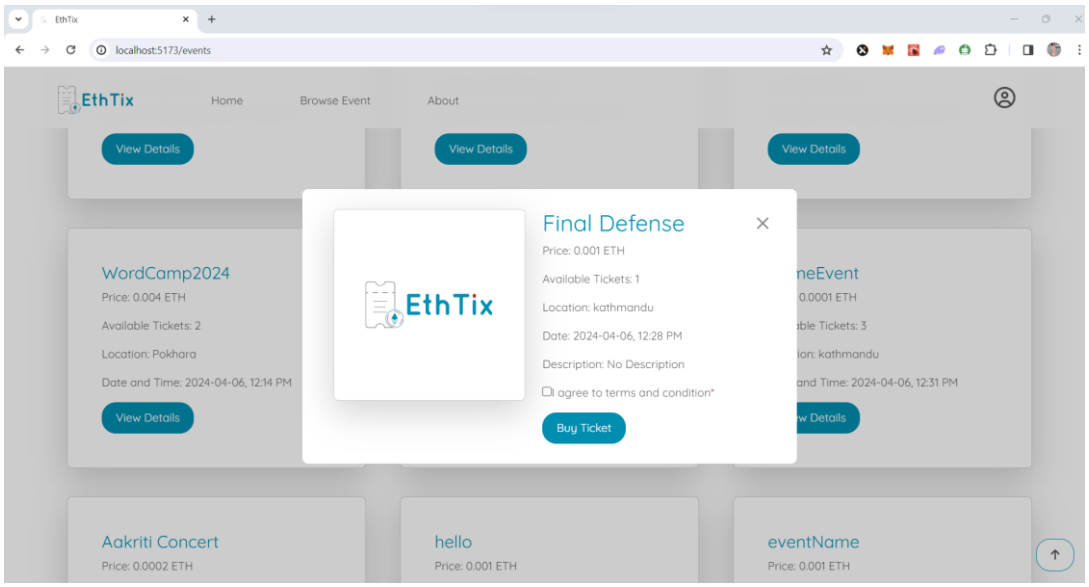

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Connect
Welcome to EthTix!
Elevating your event adventure and unlocking the full potential of your tickets
Get Started
Host Event

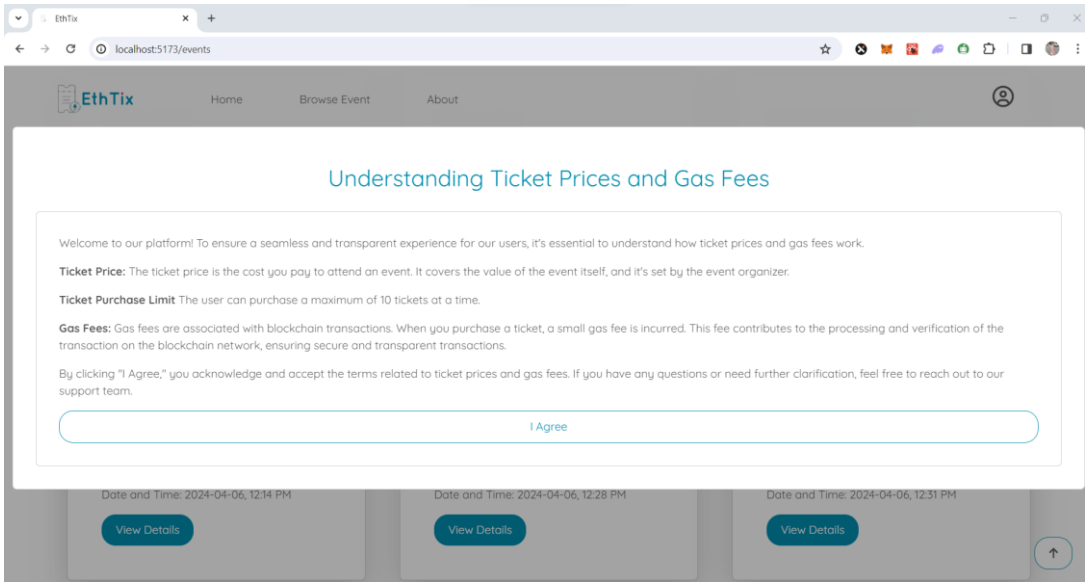

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Connect
Connect
Connect your wallet to get started
Connect Wallet
Welcome to EthTix
Elevating your event adventure and u
tickets
Get Started
Host Event

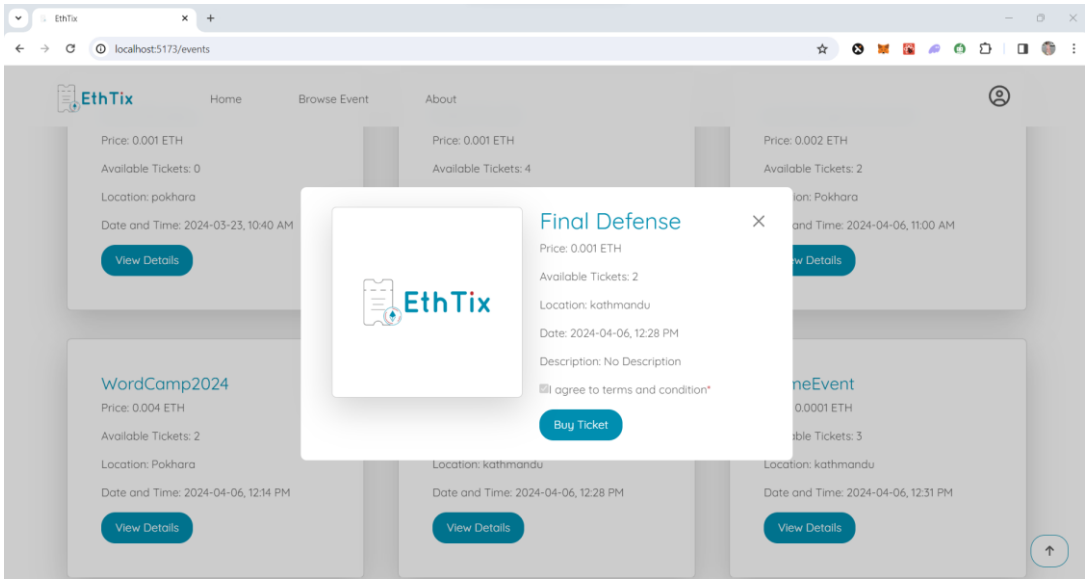

EthTix
localhost:5173
EthTix
Home
Browse Event
About
Welcome to EthTix!
Elevating your event adventure and unlocking the full potential of your tickets
Get Started
Host Event
MetaMask
Welcome back!
The decentralized web awaits
Password
Unlock
Forgot password?
Need help? Contact MetaMask support

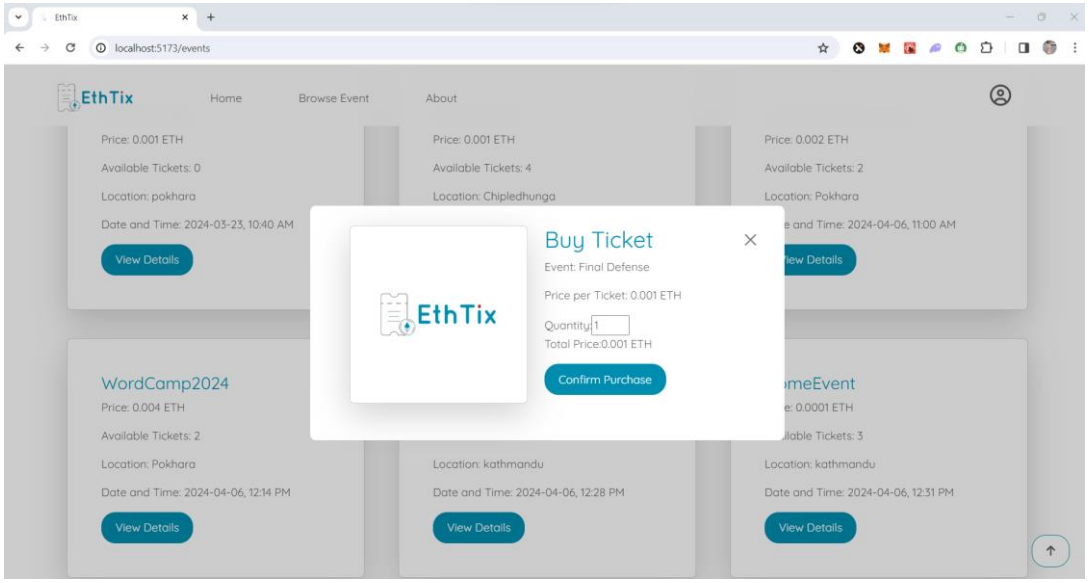

EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
Unforgettable moments await.
Grab your ticket now!
Search events
EthTix
TICKET

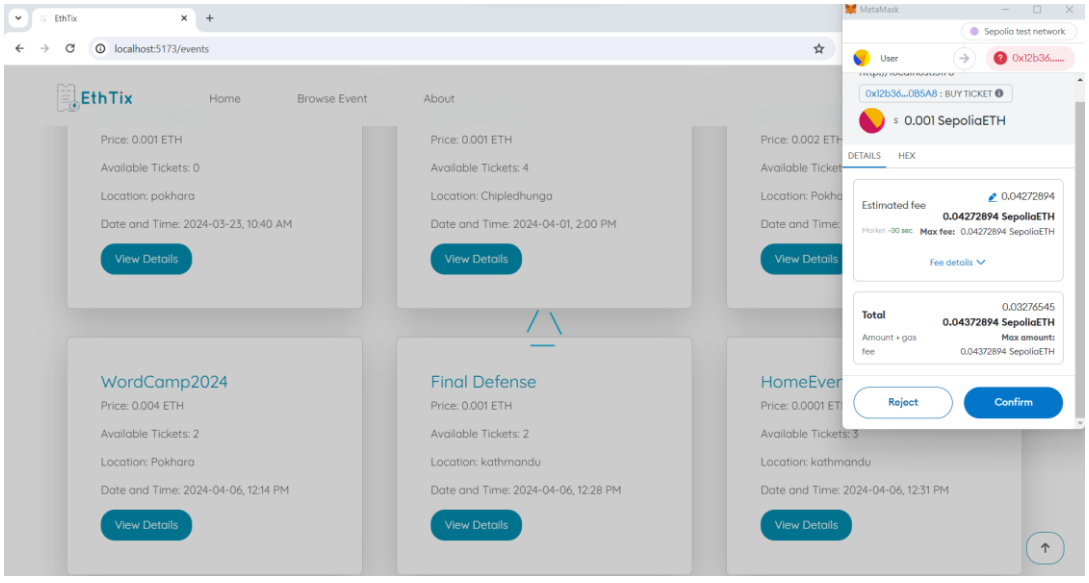

EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
Events
EventBuddy
Price: 0.001 ETH
Available Tickets: 0
Location: pokhara
Date and Time: 2024-03-23, 10:40 AM
View Details
MACITHUB
Price: 0.001 ETH
Available Tickets: 4
Location: Chipledhunga
Date and Time: 2024-04-01, 2:00 PM
View Details
Arijit Singh Concert
Price: 0.002 ETH
Available Tickets: 2
Location: Pokhara
Date and Time: 2024-04-06, 11:00 AM
View Details
WordCamp2024
Price: 0.004 ETH
Available Tickets: 2
Location: Pokhara
Final Defense
Price: 0.001 ETH
Available Tickets: 2
Location: kathmandu
HomeEvent
Price: 0.0001 ETH
Available Tickets: 3
Location: kathmandu

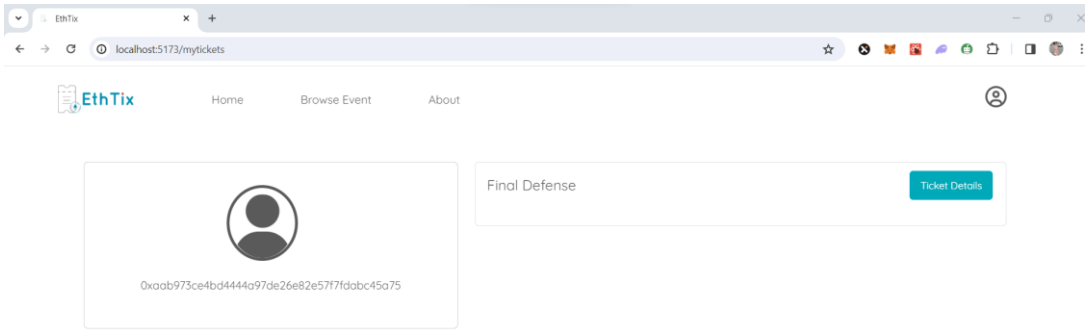

EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
View Details
View Details
View Details
WordCamp2024
Price: 0.004 ETH
Available Tickets: 2
Location: Pokhara
Date and Time: 2024-04-06, 12:14 PM
View Details
EthTix
Final Defense
Price: 0.001 ETH
Available Tickets: 1
Location: kathmandu
Date: 2024-04-06, 12:28 PM
Description: No Description
I agree to terms and condition*
Buy Ticket
neEvent
0.0001 ETH
ble Tickets: 3
ion: kathmandu
and Time: 2024-04-06, 12:31 PM
w Details
Aakriti Concert
Price: 0.0002 ETH
hello
Price: 0.001 ETH
eventName
Price: 0.001 ETH


EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
Understanding Ticket Prices and Gas Fees
Welcome to our platform! To ensure a seamless and transparent experience for our users, it's essential to understand how ticket prices and gas fees work.
Ticket Price: The ticket price is the cost you pay to attend an event. It covers the value of the event itself, and it's set by the event organizer.
Ticket Purchase Limit The user can purchase a maximum of 10 tickets at a time.
Gas Fees: Gas fees are associated with blockchain transactions. When you purchase a ticket, a small gas fee is incurred. This fee contributes to the processing and verification of the transaction on the blockchain network, ensuring secure and transparent transactions.
By clicking "I Agree," you acknowledge and accept the terms related to ticket prices and gas fees. If you have any questions or need further clarification, feel free to reach out to our support team.
I Agree
Date and Time: 2024-04-06, 12:14 PM
View Details
Date and Time: 2024-04-06, 12:28 PM
View Details
Date and Time: 2024-04-06, 12:31 PM
View Details

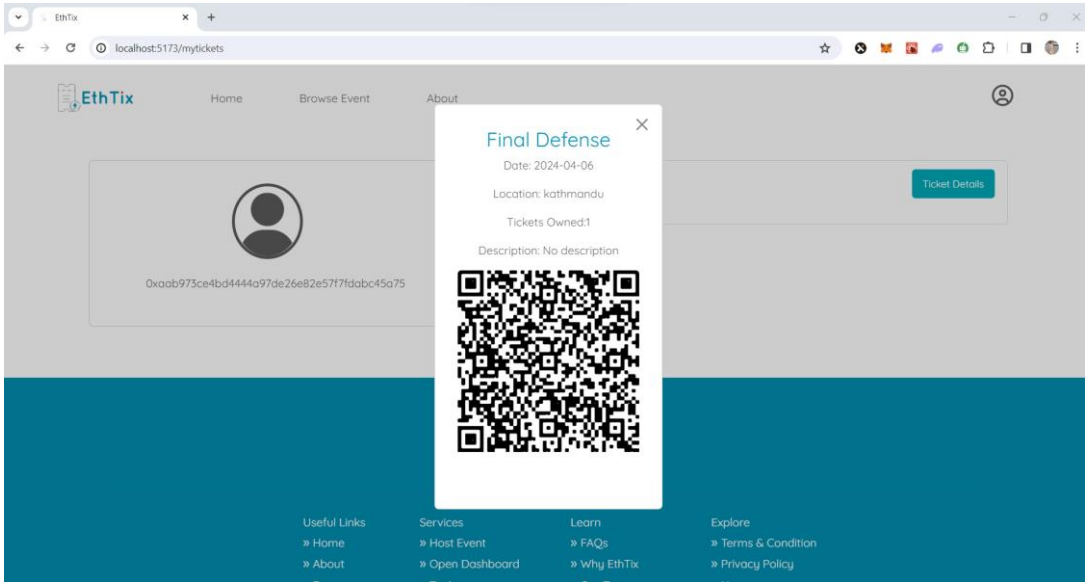

EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
Price: 0.001 ETH
Available Tickets: 0
Location: pokhara
Date and Time: 2024-03-23, 10:40 AM
View Details
Price: 0.001 ETH
Available Tickets: 4
Price: 0.002 ETH
Available Tickets: 2
Final Defense
Price: 0.001 ETH
Available Tickets: 2
Location: kathmandu
Date: 2024-04-06, 12:28 PM
Description: No Description
I agree to terms and condition*
Buy Ticket
WordCamp2024
Price: 0.004 ETH
Available Tickets: 2
Location: Pokhara
Date and Time: 2024-04-06, 12:14 PM
Location: kathmandu
Date and Time: 2024-04-06, 12:28 PM
Location: kathmandu
Date and Time: 2024-04-06, 12:31 PM


EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
Price: 0.001 ETH
Available Tickets: 0
Location: pokhara
Date and Time: 2024-03-23, 10:40 AM
View Details
Price: 0.001 ETH
Available Tickets: 4
Location: Chipledhunga
Price: 0.002 ETH
Available Tickets: 2
Location: Pokhara
Buy Ticket
Event: Final Defense
Price per Ticket: 0.001 ETH
Quantity: 1
Total Price:0.001 ETH
Confirm Purchase
WordCamp2024
Price: 0.004 ETH
Available Tickets: 2
Location: Pokhara
Date and Time: 2024-04-06, 12:14 PM
Location: kathmandu
Date and Time: 2024-04-06, 12:28 PM
Location: kathmandu
Date and Time: 2024-04-06, 12:31 PM

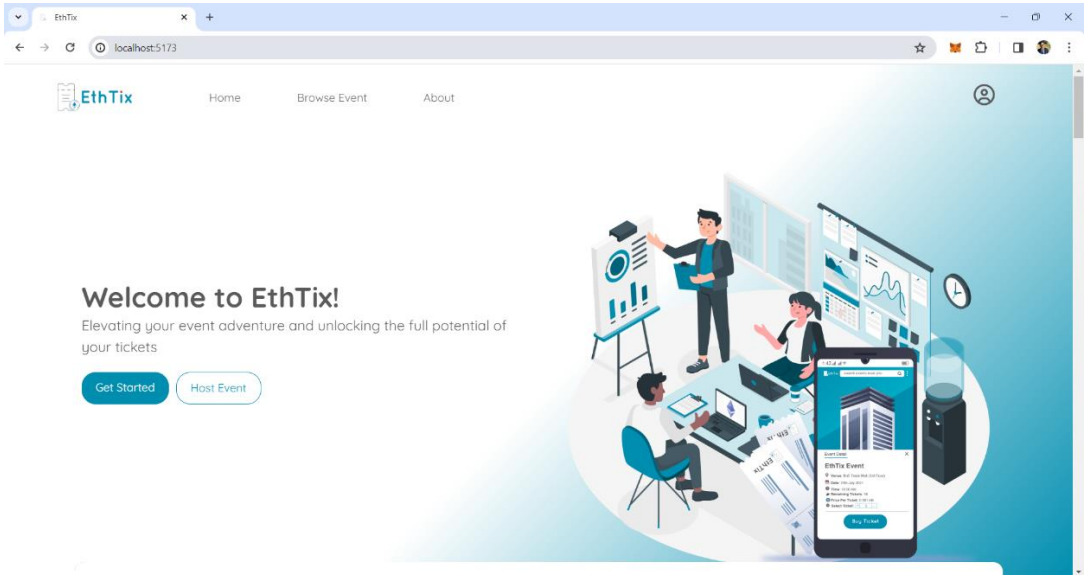

EthTix
localhost:5173/events
EthTix
Home
Browse Event
About
Price: 0.001 ETH
Available Tickets: 0
Location: pokhara
Date and Time: 2024-03-23, 10:40 AM
View Details
Price: 0.001 ETH
Available Tickets: 4
Location: Chipledhunga
Date and Time: 2024-04-01, 2:00 PM
WordCamp2024
Price: 0.004 ETH
Available Tickets: 2
Location: Pokhara
Date and Time: 2024-04-06, 12:14 PM
Final Defense
Price: 0.001 ETH
Available Tickets: 2
Location: kathmandu
Date and Time: 2024-04-06, 12:28 PM
MetaMask
Sepolia test network
User
0x12b36.....
0x12b36...0B5A8 : BUY TICKET
0.001 SepoliaETH
DETAILS
HEX
Estimated fee
0.04272894
0.04272894 SepoliaETH
Max fee: 0.04272894 SepoliaETH
Fee details
Total
0.03276545
0.04372894 SepoliaETH
Amount + gas fee
Max amount:
0.04372894 SepoliaETH
Reject
Confirm

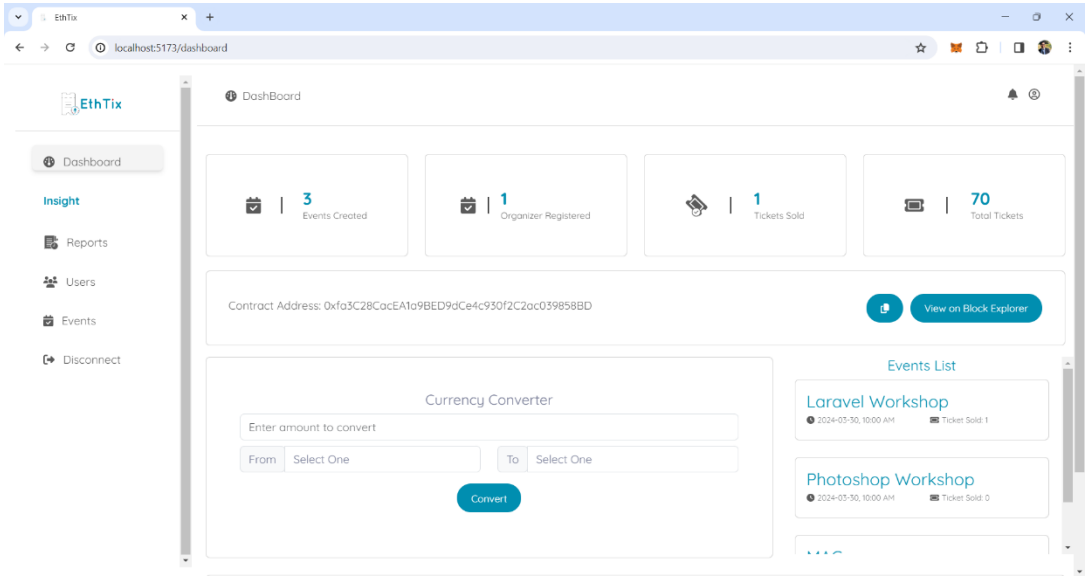

EthTix
localhost:5173/mytickets
EthTix
Home
Browse Event
About
0xaab973ce4bd4444a97de26e82e57f7fdabc45a75
Final Defense
Ticket Details

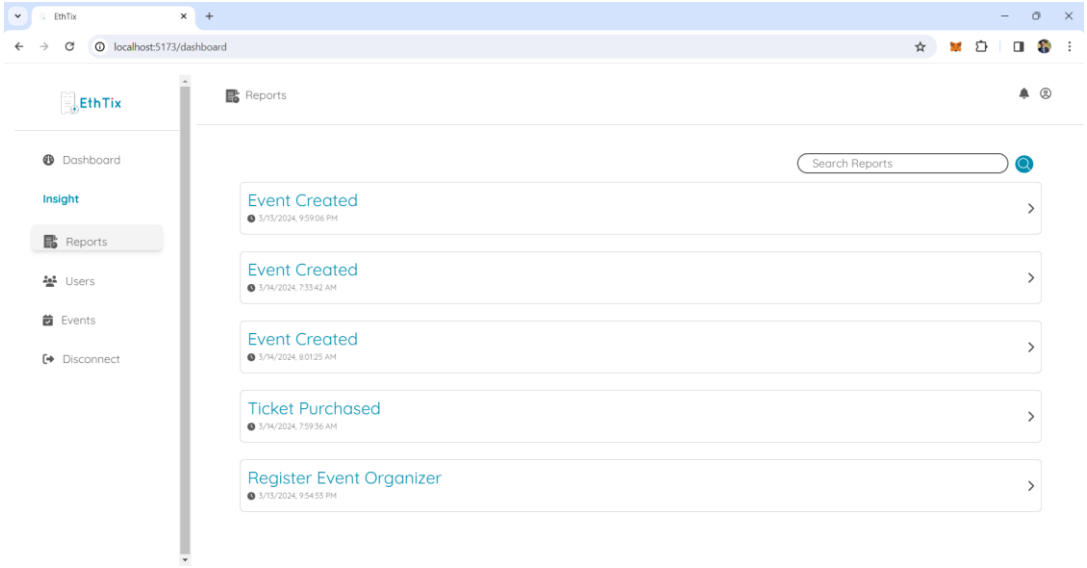

EthTix
Useful Links
» Home
» About
Services
» Host Event
» Open Dashboard
Learn
» FAQs
» Why EthTix
Explore
» Terms & Condition
» Privacy Policy

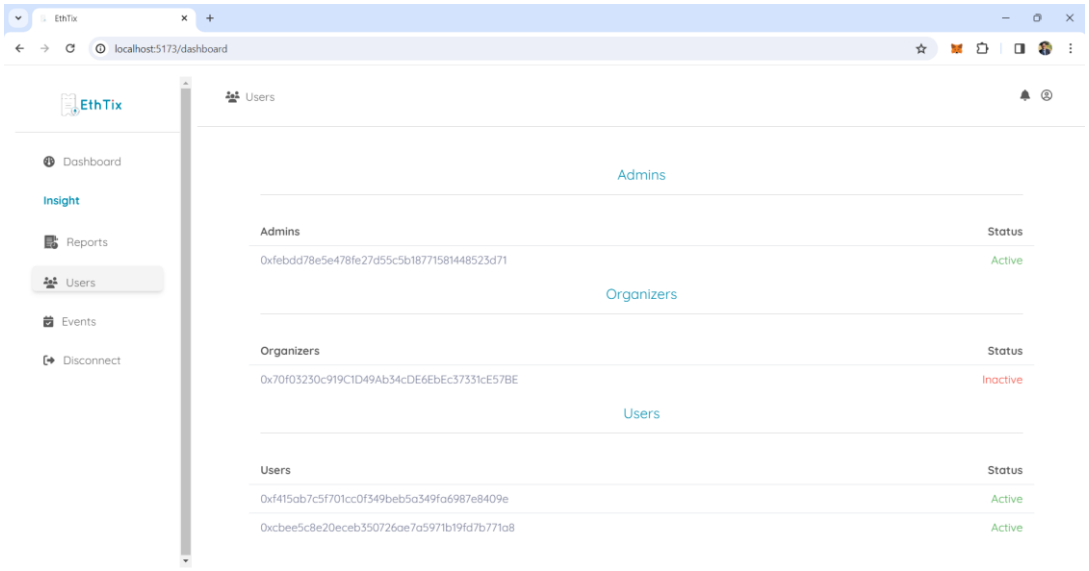

EthTix
localhost:5173/mytickets
EthTix
Home
Browse Event
About
0xaab973ce4bd4444a97de26e82e57f7fdabc45a75
Ticket Details
Final Defense
Date: 2024-04-06
Location: kathmandu
Tickets Owned:1
Description: No description
Useful Links
» Home
» About
Services
» Host Event
» Open Dashboard
Learn
» FAQs
» Why EthTix
Explore
» Terms & Condition
» Privacy Policy

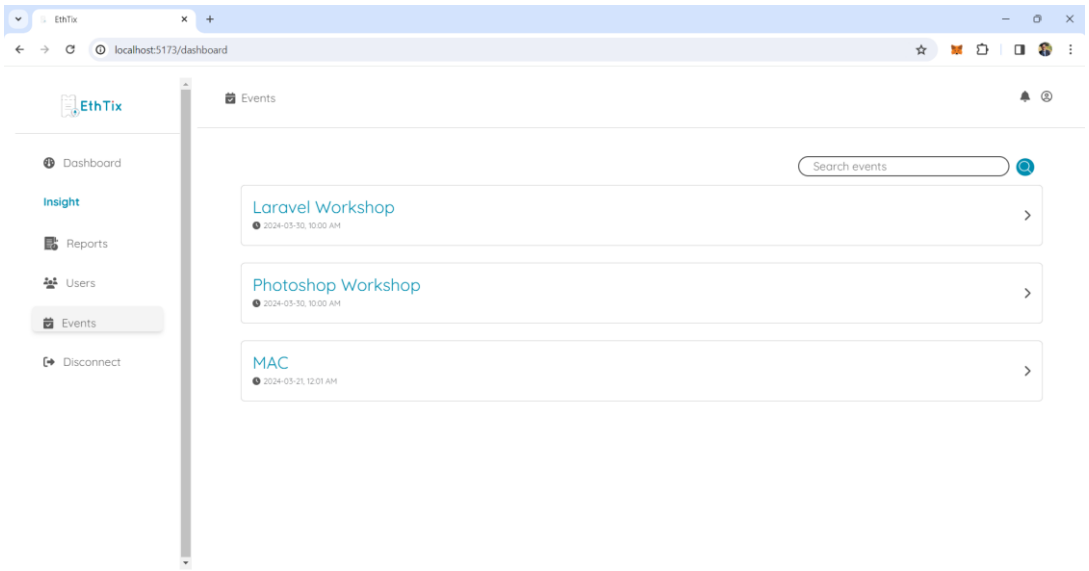
Admin view


EthTix
Home
Browse Event
About
Welcome to EthTix!
Elevating your event adventure and unlocking the full potential of your tickets
Get Started
Host Event


EthTix
DashBoard
Dashboard
Insight
Reports
Users
Events
Disconnect
3
Events Created
1
Organizer Registered
1
Tickets Sold
70
Total Tickets
Contract Address: 0xfa3C28CacEA1a9BED9dCe4c930f2C2ac039858BD
View on Block Explorer
Currency Converter
Enter amount to convert
From
Select One
To
Select One
Convert
Events List
Laravel Workshop
Photoshop Workshop


EthTix
Reports
Dashboard
Insight
Reports
Users
Events
Disconnect
Search Reports
Event Created
Event Created
Event Created
Ticket Purchased
Register Event Organizer


EthTix
localhost:5173/dashboard
EthTix
Users
Dashboard
Insight
Reports
Users
Events
Disconnect
Admins
Admins
Status
0xfebdd78e5e478fe27d55c5b18771581448523d71
Active
Organizers
Organizers
Status
0x70f03230c919C1D49Ab34cDE6EbEc37331cE57BE
Inactive
Users
Users
Status
0xf415ab7c5f701cc0f349beb5a349fa6987e8409e
Active
0xcbee5c8e20eceb350726ae7a5971b19fd7b771a8
Active


EthTix
localhost:5173/dashboard
EthTix
Events
Dashboard
Insight
Reports
Users
Events
Disconnect
Search events
Laravel Workshop
2024-03-30, 10:00 AM
Photoshop Workshop
2024-03-30, 10:00 AM
MAC
2024-03-21, 12:01 AM