香 港 大 學

THE UNIVERSITY OF HONG KONG

Faculty of Engineering

Department of Computer Science

COMP4801 Final Year Project

Interim Report

Blockchain Ticketing System (TiXTacGo)

Date	:	20th January 2024	
Supervisor	:	Dr. Chim Tat Wing	
Students	:	Marsha Melis	(3035683538)
		Lam Yu Hang Julian	(3035687766)
		Lam Pak Hei Jason	(3035685677)
		Kung Hon Kit	(3035687534)

Abstract

Hong Kong, as a cosmopolitan metropolis, attracts many internationally renowned artists who participate in a variety of events. During ticket sales, event ticketing systems are frequently overloaded with customers. Unethical purchasers take advantage by exploiting the systems with bots, scalping tickets, and reselling them at exorbitant prices. To alleviate these issues, this report presents a blockchain ticketing system. A secure, transparent, and decentralized system was developed by incorporating blockchain technologies such as Ethereum, smart contracts, MetaMask, and IPFS. A mobile application for QR code generation was designed to combat ticket fraud by implementing auto refreshing QR codes. To assure two-step verification, verifications are performed prior to generating the QR codes and venue entry. The system can provide all the fundamental functionalities that ticket purchasers, event organizers and super admins require. With the integration of blockchain technologies, it is expected that the challenges encountered by traditional ticketing systems will be mitigated. Comprehensive system testing and necessary improvements will be undertaken prior to the scheduled December demonstration of TiXTacGo. Mobile applications will be developed after the release of the websites.

Acknowledgement

We would like to express our sincere gratitude to our supervisor, Dr. Chim, Tat Wing, for his guidance and support. His incisive feedback and constructive guidance have consistently improved our deliverables. We are truly fortunate to have the opportunity to work under his supervision. We would also like to extend our deepest appreciation to the members of our project team for their collaboration and dedication throughout the development of our project. Their contributions have been invaluable in making this project a success.

Table of Contents

List of Figures

List of Tables

Abbreviations

NFT (Non-Fungible Token) is one of the digital assets in the blockchain network [1]. Each of the NFTs is unique and cannot be separated into different parts. Each of them will have its own value.

Smart contract is the program in the blockchain that stores the data in the blocks of the blockchain [2].

Ethereum is one of the blockchain networks which offers its own cryptocurrency and allows users to create smart contracts and run in the network [3].

ETH is the cryptocurrency issued by Ethereum to enable users to transact under its network [3].

ERC-721 is the standard for NFT in the Ethereum network [4]. Under the standard, every token is unique.

MetaMask is one of the leading crypto wallets in the world which allows users to manage their own cryptocurrency without storing the crypto address in MetaMask [5].

IPFS (InterPlanetary File System) is a protocol to store and transfer data through peer-to-peer networking in a decentralized way similar to blockchain technology [6].

MongoDB is a document database for storing and managing data [7]. It is to implement the use of IPFS.

Python is an object-oriented, high-level programming language for software development [8]. It is a popular language for software development.

Flask is a framework in Python to be used to build website applications [9].

Metadata is the off-chain data of NFT which is stored in the IPFS [10].

Solidity is an object-oriented, high-level programming language to implement smart contracts [11] which is usually used in blockchain.

1 Introduction

Hong Kong, a major metropolis with a diverse population, annually hosts a variety of concerts and music festivals. In 2023, world-known artists and groups such as DJ Alan Walker, BLACKPINK, Charlie Puth, and Jay Chou chose Hong Kong as part of their world tour [12]. Traditional ticketing systems are commonly used to facilitate ticket sales and distribute tickets for various events. However, these systems are fundamentally flawed by vicious speculation, which poses harm to both Event Organizers and Ticket Purchasers.

1.1 Problem Statement

Traditional ticketing systems, despite their wide usage, encounter significant challenges that have adverse effects on various roles of the public. This section will analyse the three major problems, specifically ticket scalping, ticket fraud, and the singular functionality of tickets.

1.1.1 Ticket Scalping

Due to the immense popularity of numerous prestigious events, scalpers purchase large quantities of tickets from traditional ticketing systems, thereby decreasing the number of tickets available to other Ticket Purchasers and increasing overall demand. Subsequently, scalpers proceed to resell these tickets at an inflated price for profit [13].

1.1.2 Ticket Fraud

Traditional tickets are commonly available in two formats: physical or electronic. However, these tickets lack adequate security safeguards and are susceptible to forgery. Due to the lack of regulations on online resale, victims frequently incur losses by not being able to receive a legitimate and valid ticket after the completion of payment [14].

1.1.3 Singular Functionality of Tickets

Tickets distributed by current ticketing systems have limited usage after the conclusion of the event; waste is generated, and the environment is rapidly degraded.

1.2 Project Objectives

This project aims to develop a ticketing system leveraging blockchain technology. The system seeks to revolutionize the ticketing industry by achieving two main goals: eliminating ticket scalping and improving the features and capabilities of tickets. By integrating blockchain technology, the system aims to enhance user experiences, ensure security, and promote transparency in the ticketing procedure.

1.2.1 Prevent Ticket Scalping

To combat ticket scalping, the system capitalizes on the inherent features of blockchain technology, including transparency and immutability. Through the implementation of a distributed ledger, all ticket transactions are securely documented and transparently recorded. The complete transaction history is easily accessible for verification, thereby reducing the possibility for scalpers to partake in fraudulent activities. Blockchain's decentralized infrastructure guarantees the ticket availability and pricing cannot be manipulated by a single entity, which encourages a fair ticketing environment.

In addition, the system incorporates smart contracts, which are conditional agreements that executes automatically according to predefined conditions and rules. These include restrictions on the number of tickets everyone may acquire and the reselling price. Smart contracts reduce dependence on intermediaries and mitigate the risk of unauthorized reselling by automating the issuance and transfer of tickets.

1.2.2 Improve Ticket Functionality

Event organizers can take advantage of the system's dashboard to obtain valuable insights regarding user's behaviour and preferences, allowing them to optimize their services and marketing tactics accordingly. Moreover, the system offers enhanced convenience and personalized experiences to ticket purchasers. Promoting sustained customer engagement and cultivating a positive relationship can be achieved through the provision of incentives, such as discounts, to patrons who purchase tickets on a recurring basis. Also, simultaneous ticket transfers among individuals enhance the adaptability of ticket usage, thereby fostering a ticketing ecosystem that is more streamlined, secure, and user focused.

1.3 Project Motivation

Both ticket purchasers and event organizers are negatively affected by ticket scalping and fraud. For ticket purchasers, the phenomenon of ticket scalping causes them to capitulate to scalpers, resulting in the purchase of overpriced tickets and the risk of purchasing fraudulent tickets. This hinders their enthusiasm to participate in future events, hence posing a threat to the sustainability of the industry. For event organizers, they may suffer reputational harm due to ticket scalping. It creates a negative perception among consumers regarding the arrangements, resulting in customer dissatisfaction.

To tackle the problems associated with traditional ticketing systems, event organizers have used various management strategies, such as limiting the number of transactions per individual and enhancing measures to combat ticket fraud, etc. [15]. Nevertheless, the aforementioned issues persist and have no indication of diminishing. Therefore, it is imperative to undertake a comprehensive revamp of the existing ticketing system to address the problems and ensure sustainable development within the industry.

1.4 Project Solution

To find a suitable solution to the problems, we aim to develop a system that can bring improvements and benefits in critical domains, namely: 1) Enhanced security, 2) Transparency, 3) Trust, 4) Increased management capabilities, and 5) Sustainability.

The utilization of blockchain technology allows a distinctive and secure way to purchase tickets. From customers engaging in transactions to organizers hosting events, user satisfaction and effectiveness can be boosted due to its stable and dependable environment. Hence, a more robust and secure environment should be established to crack down on ticket scalping while enhancing data privacy protection.

2 Project Methodology

This section describes the system's workflow and provides an explanation for the technical aspects of its implementation.

2.1 Roles

2.1.1 Ticket Purchaser

A **Ticket Purchaser** is an individual who purchases tickets for events. They access and interact with the ticket purchaser-facing website to browse events, choose tickets, and submit online payments using their crypto wallet.

2.1.2 Event Administrator

An **Event Administrator** is an individual who authenticate tickets for events. They authenticate the **Ticket Purchasers**' ticket through QR code. **Ticket Purchasers** are only allowed to enter the event after ticket authentication.

2.1.3 Event Organizer

The **Event Organizer** is the individual responsible for utilizing the event organizer-facing website to distribute event tickets to **Ticket Purchasers**. They submit event details and establish ticket inventories through the website to seek approval from the **Super Administrator**. Additionally, they can oversee the ticket sales process through an interactive dashboard.

2.2 Deliverables

Figure 1 illustrates the interaction between 1) end users, 2) front-end websites and applications, 3) back-end data storage, and 4) Ethereum blockchain.

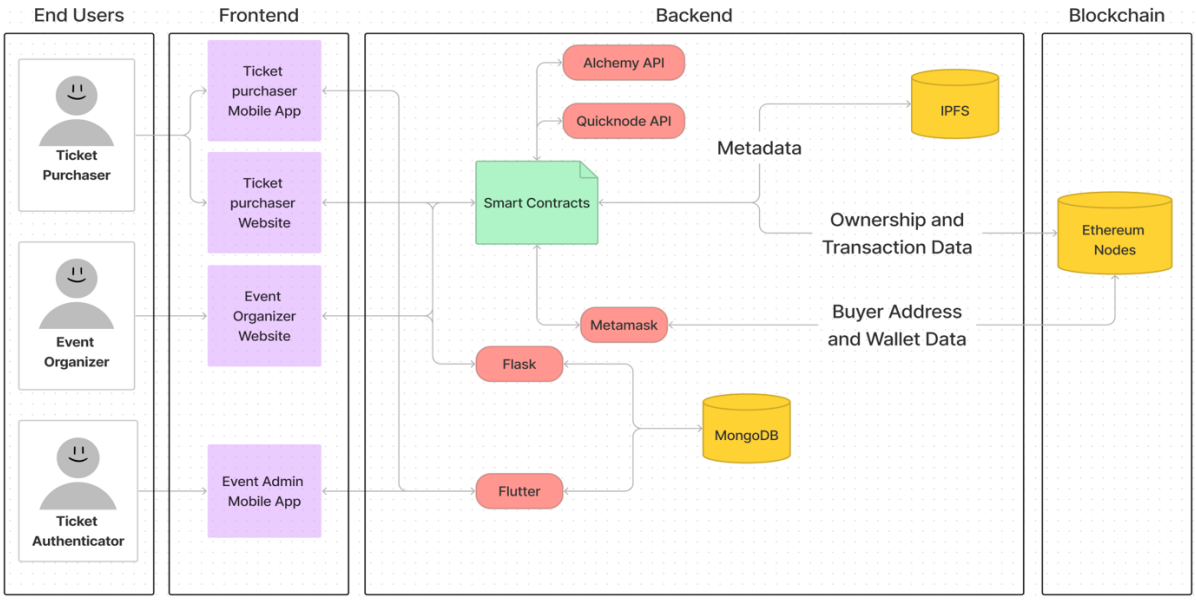

***Figure 1**: High-level Overview of Technical Implementations of the Blockchain Ticketing System*

By implementing smart contracts for identity verification, ticket purchasers will be able to connect the ticket purchaser-facing website with their crypto wallet (MetaMask). The smart contract will be linked with MetaMask to fetch relevant information from the cryptocurrency wallet of the ticket purchasers. Then, ticket purchasers may trigger the minting button to mint the TiXTacGo tickets, which indicates a purchase of event tickets.

Ticket purchasers will be able to connect their cryptocurrency wallet with the ticket purchaser-facing mobile app. Purchased TiXTacGo tickets will be shown in the app upon the establishment of a successful connection. After selecting the TiXTacGo ticket, a QR code will be generated for the purpose of ticket authentication. This QR code automatically refreshes every second to enhance security and prevent unauthorized sharing. The activation of the smart contract will be initiated after the QR code is scanned by the event administrator mobile app, resulting in the deactivation of the TiXTacGo ticket that has been utilized.

Event organizers could use the event organizer-facing website to access and review the TiXTacGo ticket sales data. Data of the TiXTacGo tickets will be fetched from the Ethereum blockchain and shown on the dashboard. Besides, event organizers could use the website to submit details of the events and tickets to be onboarded.

2.2.1 TiXTacGo Tickets

TiXTacGo tickets in the Ethereum blockchain with the ERC-721 Non-Fungible Token (NFT) Standard will be created and minted through the implementation of smart contracts. Each unique TiXTacGo ticket will consist of on-chain transaction data and off-chain Metadata:

- On-chain Transaction Data:
    1) Token ID
    2) Contract Address
    3) Owner Address
    4) Creator Address
- Off-chain Metadata:
    1) Event Date
    2) Event Time
    3) Event Organizer
    4) Event Name
    5) Event Venue
    6) Gate
    7) Section
    8) Row
    9) Seat

2.2.2 Websites

2.2.2.1 Ticket Purchaser Website

The ticket purchaser website will serve the purpose of general ticket purchase for the ticket purchaser. It will consist of the following functions:

1) User Log-in by Crypto Wallet Connection
2) Event Listing
3) Ticket Buying Confirmation
4) Ticket Minting Smart Contract
5) User Ticket Collection Listing

2.2.2.2 Event Organizer Dashboard

The dashboard will serve the purposes of data analytics and event details submission for the event organizer. It will consist of the following functions:

1) User Registration
2) User Log-in
3) User Account Reset
4) Event Details Submission
5) Ticket Details Submission
6) Data Analytics Dashboard
7) Event and Ticket Details Confirmation by Super Admin

2.2.3 Mobile Application

2.2.3.1 Ticket Purchaser Ticket Authentication Application

The ticket purchaser ticket authentication application will serve the purpose of generating QR code tickets for the ticket purchaser. It will consist of the following functions:

1) User Log-in by Crypto Wallet Connection
2) Event Listing
3) User Ticket Listing
4) QR Code Ticket Generation from TiXTacGo Ticket
5) Auto-refresh QR Code Ticket in the Regular Time Frame

2.2.3.2 Event Administrator Ticket Authentication Application

The event administrator ticket authentication application will verify the authenticity of QR code tickets generated by the ticket purchaser ticket authentication application. It will consist of the following functions:

1) Scanning QR Code Tickets by Mobile Camera
2) Ticket Burning Smart Contract

2.3 Technology Implementation

2.3.1 Front-end Development

Before starting on the front-end code development, our own design system will be created in Figma to visualize the layout of the websites and the mobile applications. To reduce design and development time, we plan to utilize the pre-styled CSS framework Bootstrap rather than starting from scratch.

2.3.1.1 Ticket Purchaser Website and Event Organizer Dashboard

As our project is based heavily on Python, a Python templating engine Jinja2 will be used to integrate with our backend. It allows simplicity in producing dynamic HTML content with JavaScript while optimizing web applications for improved performance.

2.3.1.2 Ticket Purchaser and Event Administrator Ticket Authentication Application

The two mobile applications will be implemented based on Flutter, a cross-platform framework that allows us to deploy the application on various platforms such as Android and iOS. Flutter can deliver high-quality applications while executing them very fast, this high and efficient performance benefits us in terms of the main usage of the applications, which is to allow generating and scanning of QR code in a short period of time.

2.3.2 Back-end Development

2.3.2.1 Flask

Flask will be used for developing our Python-based websites and Web Server Gateway Interface (WSGI) [16] will be used to run the websites. As a powerful backend infrastructure and lightweight web micro-framework [16, 17], it even has great compatibility with Python which allows us to connect to the blockchain ecosystem seamlessly.

2.3.3 Data Storage

2.3.3.1 MongoDB

MongoDB is a document database for storing and managing data [7]. Data in MongoDB is in JSON-like format [7]. This feature allows the database to directly map the current programming language with ease and convenience. MongoDB could be used in the Blockchain

Ticketing System to store the login information of event organizers, as well as the sales data that is not included in the Ethereum blockchain.

2.3.3.2 IPFS

InterPlanetary File System (IPFS) is a protocol to store and transfer data through peer-to-peer networking in a decentralized way similar to blockchain technology [8]. In the Ethereum blockchain, the block size is limited and not large enough to store all the data of an NFT, IPFS will be used to store the Metadata of the TiXTacGo tickets of the Blockchain Ticketing System. The decentralized feature of IPFS allows the Metadata can be stored in IPFS nodes securely and ensures the Metadata is tamper-proof.

2.3.4 Crypto Wallet

2.3.4.1 MetaMask

To connect ticket purchasers' cryptocurrency wallet, MetaMask, will be used in both the ticket purchaser website and ticket authentication application. Ticket purchasers' MetaMask are connected to blockchain for interaction, such as storing the TiXTacGo tickets minted by ticket purchasers and amending the used TiXTacGo tickets.

2.3.5 Blockchain

The decentralized, traceable, and transparent feature of blockchain [18] will play an essential role in the Blockchain Ticketing System to fulfil the objective of the system. TiXTacGo tickets sold by the system should be 1) tamper-proof to prevent counterfeit tickets, 2) able to limit the resale price to control the ticket scalping on secondary sales, and 3) traceable on ticket resale history for the ease of data analysis.

2.3.5.1 Ethereum

Prior to selecting a blockchain for the system, several characteristics will be evaluated: 1) popularity, 2) transaction fees, 3) scalability, and 4) level of decentralization. A comparative analysis of multiple blockchains has been conducted to evaluate the benefits and downsides. Table 1 shows the comparison of features between Ethereum, Solana, Polygon PoS, and Polygon zkEVM.

	Ethereum	**Solana**	**Polygon PoS**	**Polygon zkEVM**
Gas Fee **(per transaction)**	26 ETH Gwei (0.86 USD) [19]	0.0001 SOL (0.000027 USD) [20]	88.6 MATIC Gwei (0.00097 USD) [21]	0.6997 MATIC Gwei (0.0000076 USD) [22]
Scalability **(1st = Highest)**	4th	1st	3rd	2nd
Transaction Speed	12.8 TPS [19]	4739 TPS [23]	29.4 TPS [21]	0.2 TPS [24]
No. of Validators	945,129 [25]	3,400 [26]	100 [27]	Permissionless Sequencer and Aggregator [28]
Security	High security but potential risk from PoS	A newer platform with security risks & vulnerabilities	Lower security than Ethereum as it is developed on layer 2	Low security in off-chain transactions but high security in on-chain transactions
Decentralization **(1st = Highest)**	1	3	4	2
No. of Unique Address in Blockchain	245,342,698 [29]	39,750,000 [30]	354,695,682 [31]	441,723 [32]
Consensus Algorithm	Proof of Stake (PoS) [33]	Proof of History (PoH) [34]	Proof of Stake (PoS) [35]	Proof of Efficiency (PoE) [36]

Table 1: *Comparison of features between Ethereum, Solana, Polygon PoS and Polygon zkEVM*

2.3.5.2 Smart Contract

For the Blockchain Ticketing System's primary functions to be executed, Solidity-based smart contracts will be developed. To interact with the Ethereum blockchain, smart contracts will utilize several APIs, including the Alchemy NFT API to retrieve NFT data [37] and MetaMask API to request ticket purchasers' wallet address [38].

3 Current Progress

With regards to what had been done, development of TiXTacGo tickets, user interface (UI) and back-end development of both ticket purchaser and event administrator ticket authentication application have been completed. Furthermore, the UI of the ticket purchaser website has been designed.

3.1 TiXTacGo Tickets Development

By implementing smart contracts, TiXTacGo tickets have been generated and distributed on the Ethereum blockchain using the ERC-721 Non-Fungible Token (NFT) Standard.

Scripts for uploading metadata to IPFS via Pinata, deploying smart contracts, and minting tickets have been developed and executed to ensure the successful creation and minting of TiXTacGo tickets. Simultaneously, the smart contract for the generation of TiXTacGo tickets has been determined and updated to reflect the metadata standard.

3.1.1 File Structure

The file structure for blockchain development is divided into several files, each servicing an independent purpose as illustrated in Figure 2. The “contracts” folder includes the Solidity smart contract for TiXTacGo tickets. The “scripts” folder contains the scripts written in Javascript that allows for the execution of various functions. The “deploy.js” folder deploys the compiled “TiXTacGoTicket.sol” smart contract using the Alchemy API. The “create-ticket.js” folder initially transfers contents of the “Tickets” collection stored in MongoDB, along with the TiXTacGo ticket image for a particular event, to IPFS as metadata. It then executes the “mint” function from “mint-ticket.js” to generate and upload the ticket to Ethereum blockchain.

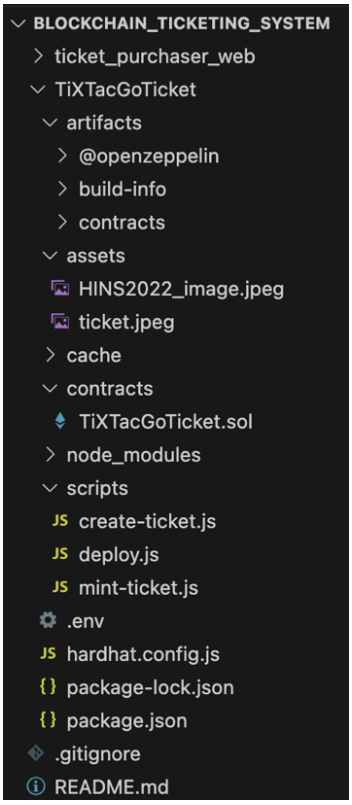

***Figure 2**: Blockchain Development Project Structure*

3.1.2 Smart Contract Development

As shown in Figure 3, TiXTacGo ticket's smart contract, "TiXTacGoTicket.sol" is developed using Solidity with the use of smart contracts of "ERC721.sol" and "ERC721URIStorage.sol" provided by Open Zeppelin library.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

import {ERC721} from "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import {ERC721URIStorage} from "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

contract TiXTacGoTicket is ERC721URIStorage {
    uint256 private _tokenIds;

    constructor() ERC721("TiXTacGoTicket", "Ticket") {}

    function mintTicket(address buyer, string memory tokenURI)
        public
        returns (uint256)
    {
        uint256 newTicketId = _tokenIds++;
        _mint(buyer, newTicketId);
        _setTokenURI(newTicketId, tokenURI);

        return newTicketId;
    }
}
```

***Figure 3**: Code Snippet of "TiXTacGoTicket.sol"*

“TixTacGoTicket.sol” is deployed for testing purposes on the Ethereum testnet, “Sepolia”, prior to its market release. “TixTacGoTicket.sol” will be deployed to the mainnet of Ethereum blockchain once the system is commercially viable. The transaction hash is obtained when “TiXTacGoTicket.sol” is deployed, as illustrated in Figure 4. Figure 5 demonstrates how the Alchemy’s analytics dashboard visualizes the deployment analytics.

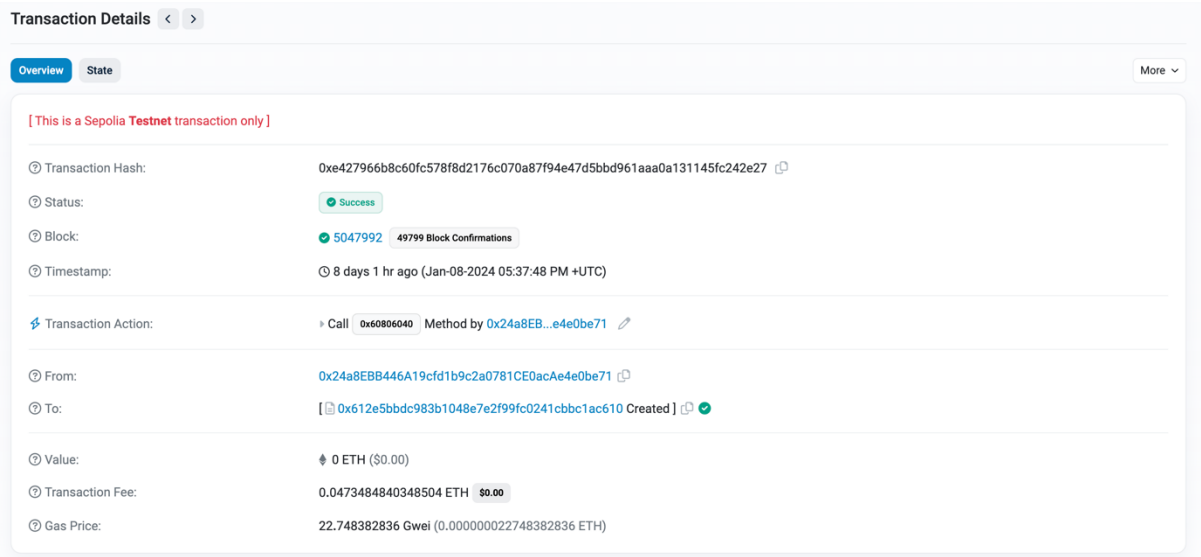

***Figure 4**: Deployed “TiXTacGoTicket.sol” Smart Contract Transaction Details*

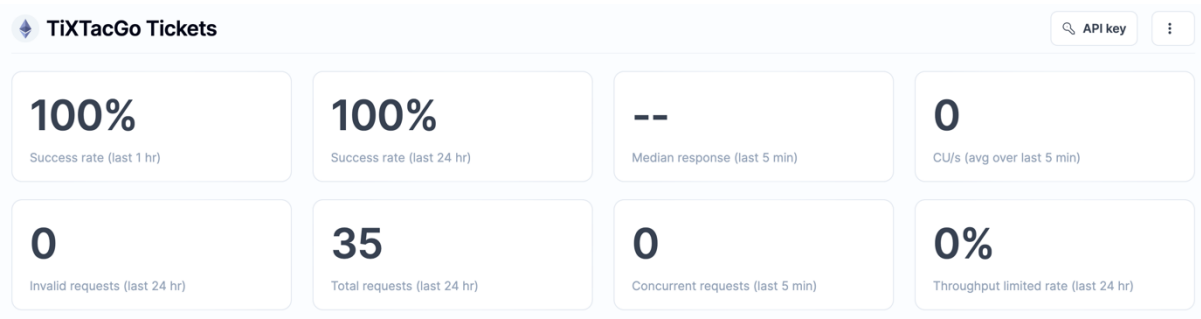

***Figure 5**: Deployment Analytics in Alchemy’s Analytics Dashboard*

3.1.3 Data Model for TiXTacGo Tickets

The "Tickets" collection in MongoDB is designed to store event-specific metadata to facilitate the generation of TiXTacGo tickets for minting purpose. Each data record preserves the fundamental information associated to a ticket, as illustrated in Figure 6. This information comprises:

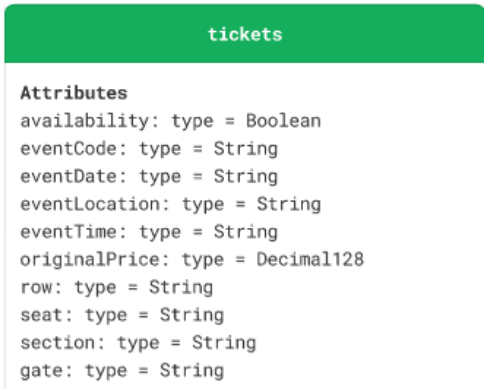

***Figure 6**: Data Model for "Tickets" Collection*

```
_id: ObjectId('65a03b21069875fa66420d3d')
availability: true
eventCode: "HINS2022"
eventDate: "13-12-2021"
eventLocation: "Hong Kong Coliseum"
eventTime: "20:00 HKT"
originalPrice: 0.05
row: "17"
seat: "86"
section: "43"
gate: "Red"
```

```
_id: ObjectId('65a6b6cf925f30aaae7d3a0a')
availability: true
eventCode: "HINS2022"
eventDate: "13-12-2021"
eventLocation: "Hong Kong Coliseum"
eventTime: "20:00 HKT"
originalPrice: 0.05
row: "1"
seat: "87"
section: "43"
gate: "Red"
```

***Figure 7**: Example Data for "Tickets" Collection*

3.1.4 Metadata Standard

Using the Pinata API, “create-ticket.js” transfers the ticket data from MongoDB to IPFS as metadata. Figure 8 illustrates how the uniform metadata standard was defined in “create-ticket.js” prior to minting. To define the metadata standard of each TiXTacGo ticket, the script receives the data of each event from the “Tickets” collection in MongoDB and uses it as variables.

```
const data = JSON.stringify({
    "description": EventDescription,
    "image": `ipfs://${CID}`,
    "name": EventCode,
    "attributes": [
        { "trait_type": "Event Code", "value": `${EventCode}`},
        { "trait_type": "Event Date", "value": `${date}`},
        { "trait_type": "Event Time", "value": `${time}`},
        { "trait_type": "Event Location", "value": `${location}`},
        { "trait_type": "Door/Gate", "value": `${gate}`},
        { "trait_type": "Section", "value": `${section}`},
        { "trait_type": "Row", "value": `${row}`},
        { "trait_type": "Seat", "value": `${seat}`},
        { "trait_type": "Original Price", "value": `${price} ETH`}
    ]
});
```

***Figure 8**: Metadata Standard of TiXTacGo Tickets Defined in “create-ticket.js”*

As shown in Figure 9 and 10, the image and metadata are subsequently uploaded to IPFS via the Pinata API.

```
const res = await axios.post("https://api.pinata.cloud/pinning/pinFileToIPFS", formData, {
    maxBodyLength: "Infinity",
    headers: {
        'Content-Type': `multipart/form-data; boundary=${formData._boundary}`,
        'Authorization': `Bearer ${JWT}`
    }
});
console.log(res.data)
return res.data.IpfsHash
```

***Figure 9**: Scripts to Upload TiXTacGo Ticket Image to IPFS*

```
const res = await axios.post("https://api.pinata.cloud/pinning/pinJSONToIPFS", data, {
    maxBodyLength: "Infinity",
    headers: {
        'Content-Type': `application/json`,
        'Authorization': `Bearer ${JWT}`
    }
})
console.log("Metadata uploaded, CID:", res.data.IpfsHash)
console.log(res.data)
return res.data.IpfsHash
```

Figure 10: *Scripts to Upload Metadata to IPFS*

The Content Identifier (CID) indicating the location of the metadata is received following its transmission to IPFS. Figure 11 displays the metadata that was uploaded to IPFS in the form of JSON.


```
{"description":"HINS2022
Concert","image":"ipfs://QmcY2efyw9GXd277gmu3eZKZVyhaUWny6wXK37Q5G1L1ee","name":"HINS2022","at
tributes":[{"trait_type":"Event Code","value":"HINS2022"},{"trait_type":"Event
Date","value":"13-12-2021"},{"trait_type":"Event Time","value":"20:00"},{"trait_type":"Event
Location","value":"Hong Kong Coliseum"},{"trait_type":"Door/Gate","value":"Red"},
{"trait_type":"Section","value":"43"},{"trait_type":"Row","value":"17"},
{"trait_type":"Seat","value":"86"},{"trait_type":"Original Price","value":"0.05 ETH"}]}
```


Figure 11: *Metadata of TiXTacGo Ticket Uploaded to IPFS*

3.1.5 TiXTacGo Tickets Minting Function

Once the metadata has been uploaded to IPFS, the minting function located in “mint-ticket.js” will be executed to mint the TiXTacGo ticket. The transaction details, including the address of the tickets to be minted to and from, gas fee, Application Binary Interface (ABI) and nonce of the smart contract are initially defined. The script for minting TiXTacGo ticket is shown in Figure 12.

```javascript
async function mintTicket(tokenURI) {
    const nonce = await web3.eth.getTransactionCount(PUBLIC_KEY, 'latest'); //get latest nonce

    const tx = {
        'from': PUBLIC_KEY,
        'to': contractAddress,
        'nonce': nonce,
        'gas': 500000,
        'data': TiXTacGoTicketContract.methods.mintTicket(PUBLIC_KEY, tokenURI).encodeABI()
    };

    const signPromise = web3.eth.accounts.signTransaction(tx, PRIVATE_KEY)

    signPromise
        .then((signedTx) => {
            web3.eth.sendSignedTransaction(
                signedTx.rawTransaction,
                function (err, hash) {
                    if (!err) {
                        console.log("The hash of the transaction is: " + hash)
                    } else {
                        console.log(
                            "Transaction Error: " + err)
                    }
                }
            )
        })
        .catch((err) => {
            console.log(" Promise failed:", err)
        })
}
```

***Figure 12**: Minting Function in "mint-ticket.js"*

The transaction hash is obtained when a TiXTacGo ticket is minted. As illustrated in Figure 13, the transaction details of the minted TiXTacGo ticket are visible. Additionally, the minted TiXTacGo ticket containing the metadata is shown in Figure 14.

Transaction Details

Overview | Logs (2) | State

[This is a Sepolia **Testnet** transaction only]

Transaction Hash:	0x848e74263efa4fce6b585a65b73d152f51f4db4cf68fc97cb501a32eb5887660
Status:	Success
Block:	5068095 29881 Block Confirmations
Timestamp:	4 days 23 hrs ago (Jan-11-2024 08:18:12 PM +UTC)
Transaction Action:	Call 0x5388842c Method by 0x24a8EB...e4e0be71 on 0x612E5B...BC1aC610
From:	0x24a8EBB446A19cfd1b9c2a0781CE0acAe4e0be71
Interacted With (To):	0x612E5BBdc983b1048e7e2F99Fc0241cBBC1aC610
ERC-721 Tokens Transferred:	ERC-721 Token ID [8] TiXTacGoTick... (Ticket...) From 0x000000...00000000 To 0x24a8EB...e4e0be71
Value:	0 ETH ($0.00)
Transaction Fee:	0.001067905567236771 ETH $0.00
Gas Price:	8.378292711 Gwei (0.000000008378292711 ETH)

***Figure 13**: Transaction Details of the Minted TiXTacGo Ticket*

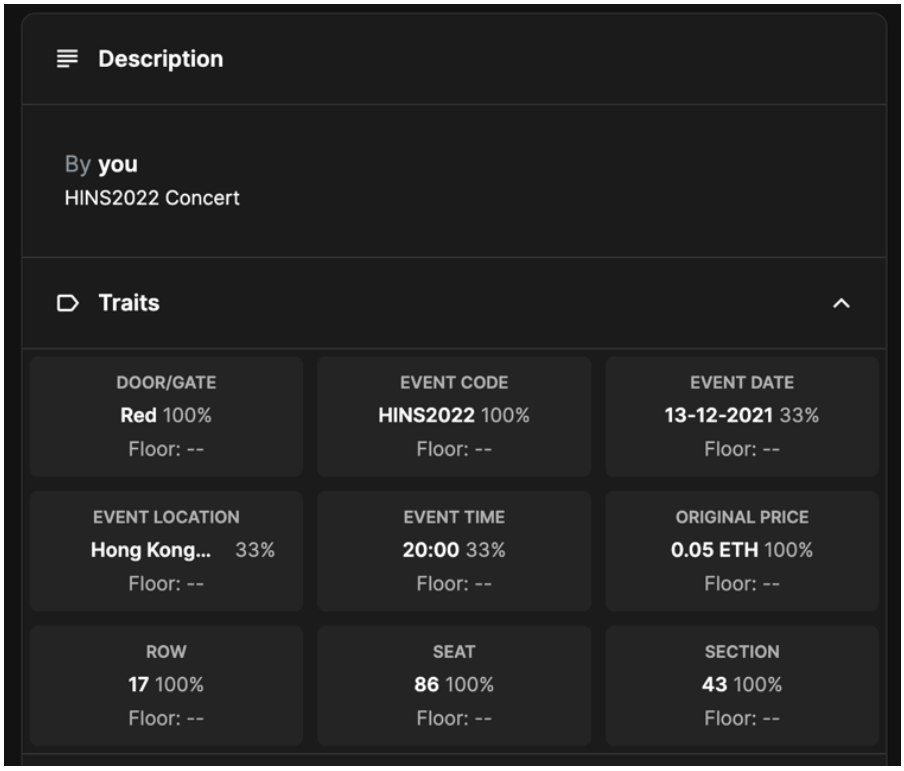

***Figure 14**: Minted TiXTacGo Ticket with Metadata showing in OpenSea, a well-known NFT Transaction Platform*

3.2 Mobile Application Development

Regarding the mobile applications, both the UI of ticket purchaser and event administrator ticket authentication application have been developed. Additionally, the connection between the ticket purchaser application and MetaMask has been established effectively, enabling data exchange between the application and the cryptocurrency wallet. The application could retrieve and display information such as wallet details and the corresponding tickets.

3.2.1 File Structure

Figure 15 illustrates the various sections comprising the structure of mobile applications, the major sections for development are as follows:

- **Assets:** Consists of fonts, icons, and images used throughout the application. It assists the application in organizing and managing static assets.
- **Lib**
    - **Bloc:** Contains the Business Logic Component (BLoC) files, which are responsible for state management and business logic execution within the application through data flow control.
    - **Components:** For improved maintainability, contains UI components that are directly reusable across multiple screens, such as icons and cards.
    - **Config:** Contains configuration files that define the application's themes, environment variables, constants, and routing.
    - **Models:** Contains classes that specify the behavior and data structures of various entities.
    - **MongoDB:** Contains files necessary for configuring the database connection, execute queries, and manipulate data.
    - **Screens:** Each screen folder corresponds to a distinct user interface (UI) screen, such as login, home, and profile screen, and represents a separate section of the application.
- **Pubspec.yaml:** The application configuration file, which defines parameters including dependencies, packages, and other project-specific settings.

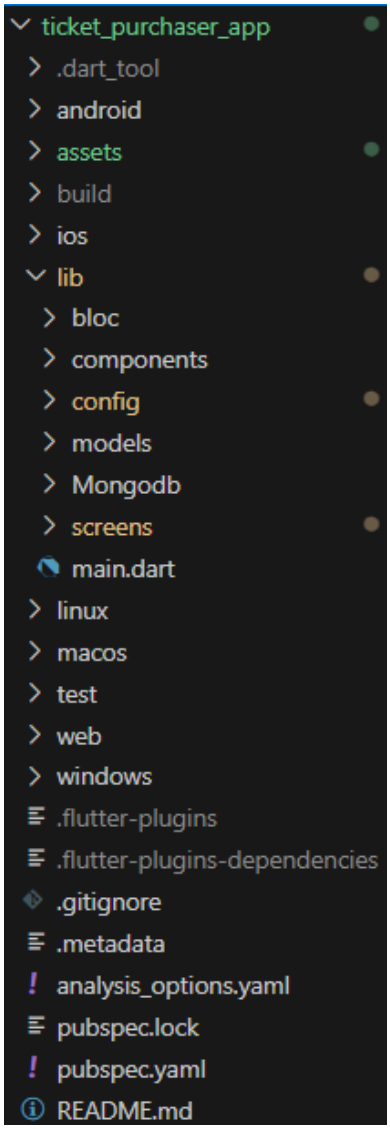

Figure 15: *Mobile Application Project Structure*

3.2.2 Ticket Purchaser Ticket Authentication Application

Ticket purchaser's ticket authentication application consists of four screens. MetaMask installation and support on the electronic device is a prerequisite for this application. Ticket purchasers can effortlessly access the application via MetaMask, as shown in Figure 16. By verifying the wallet address, a collection of tickets may be retrieved together with the specifics of the event. A unique QR code containing the ticket details and a secure object ID will be generated for each ticket upon selection; for security purposes, this QR code will be automatically refreshed every second.

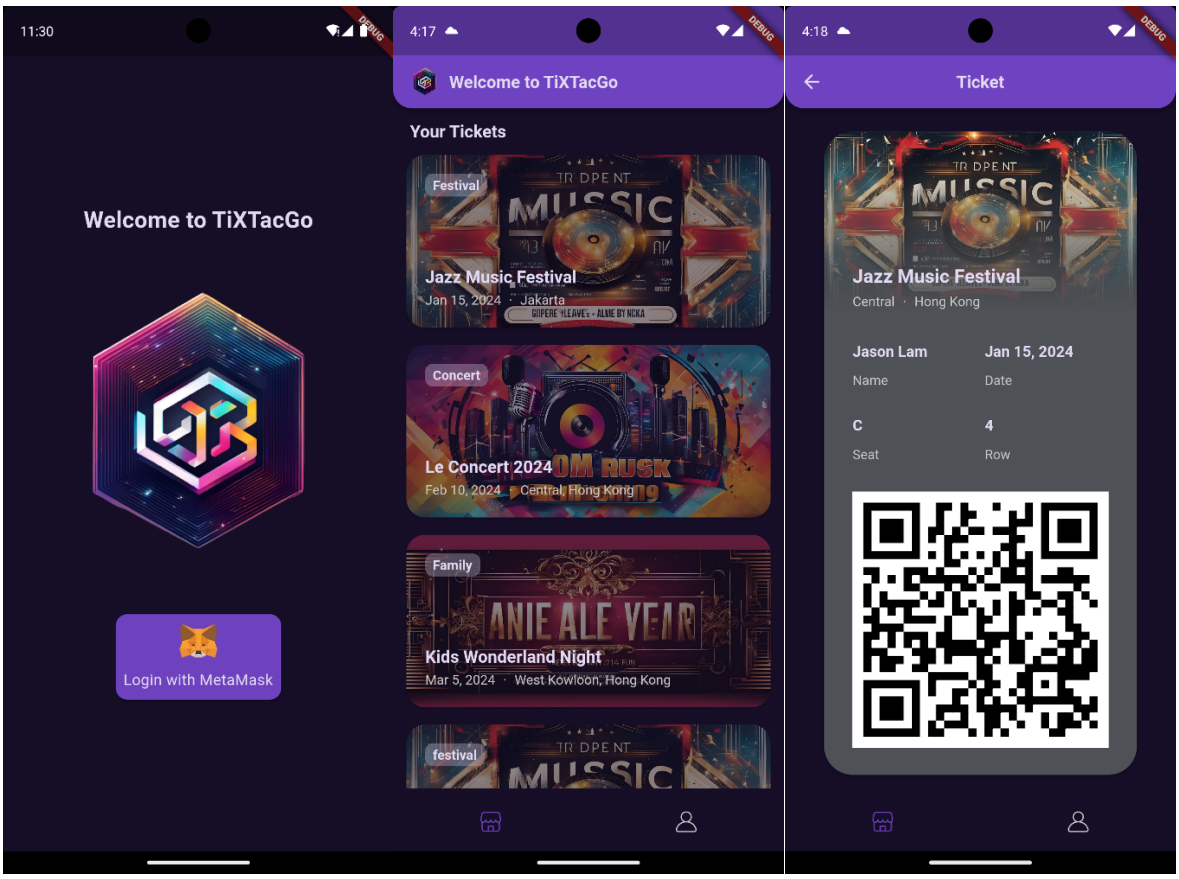

Figure 16: *Login Page (left) Ticket Listing Page (center) Event-Specific Ticket Page (right)*

3.2.3 MetaMask Authentication

After establishing a connection, the application obtains the MetaMask response, which contains the Uniform Resource Identifier (URI). It then initiates the URI redirection to MetaMask and provides MetaMask with a cryptographic signature to obtain authorization (see Figure 17).

If the response indicates a successful session connection, the wallet address of the ticket purchaser will be retrieved to get TiXTacGo ticket information stored in the wallet. Following this, the application will be authenticated and linked to the user's MetaMask wallet data and services.

```
SessionData?
    sessionData = //send signature to metamask to get authorize
    await walletConnectorService.authorize(
        resp, event.signatureFromBackend);
if (sessionData != null) {
  final String walletAddress = NamespaceUtils.getAccount(
    sessionData.namespaces.values.first.accounts.first,
  );
  debugPrint("WALLET ADDRESS - $walletAddress");
  emit(WalletAuthorizedState(
      message: AppConstants.connectionSuccessful,
       walletAddress: walletAddress));
  if (resp.session.isCompleted) {
    final String walletAddress = NamespaceUtils.getAccount(
      sessionData.namespaces.values.first.accounts.first,
    );
    debugPrint("WALLET ADDRESS - $walletAddress");
    //now again go to app ans check for message sign in request
    bool canLaunch =
        await walletConnectorService.onDisplayUri(uri);
    if (!canLaunch) {
      emit(WalletErrorState(
          message: AppConstants.metamaskNotInstalled));
    } else {
      //now send signature to metamask to get signed
      final signatureFromWallet =
          await walletConnectorService.sendMessageForSigned(
              resp,
              walletAddress,
              sessionData.topic,
              event.signatureFromBackend);
      if (signatureFromWallet != null &&
          signatureFromWallet != "") {
        emit(WalletReceivedSignatureState(
            signatureFromWallet: signatureFromWallet,
            signatureFromBk: event.signatureFromBackend,
            walletAddress: walletAddress,
            message: AppConstants.authenticatingPleaseWait));
```

***Figure 17**: Code Snippet of the MetaMask Authentication Logic*

3.2.4 Event Administrator Ticket Authentication Application

The primary function of the event administrator's ticket authentication application is to validate the QR code displayed on the ticket purchaser's application. The MongoDB database stores all logon credentials for event administrators. To activate the device camera, event administrators may log into the system and click the "Scan QR Code" button as shown in Figure 18.

A confirmation popup will appear once the ticket has been verified. The object ID and ticket information contained in the QR code assist in determining the ticket's validity. The QR code will not be validated, and an error message will appear if the specified information is not accurate.

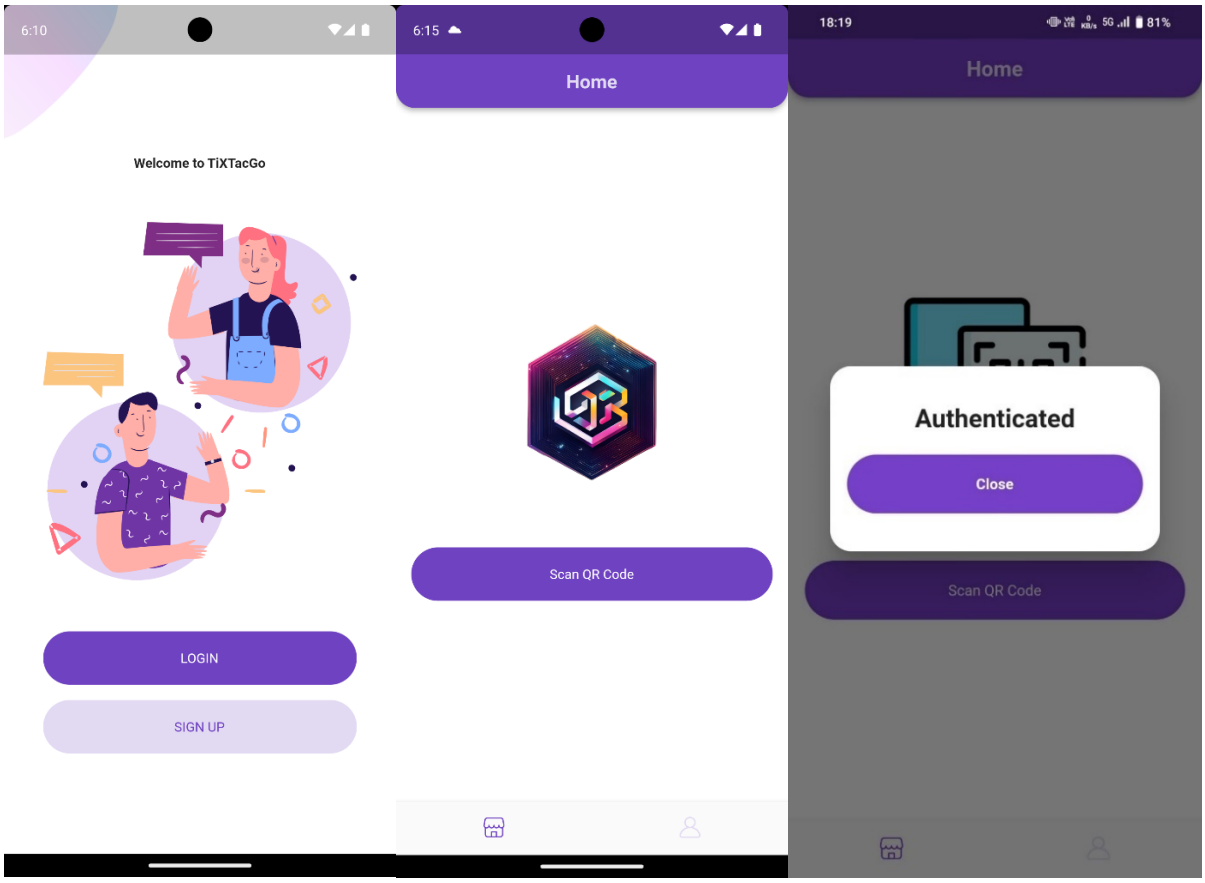

Figure 18: *Login Page (left) Home Page (center) Scanner Authenticated Page (right)*

3.3 Website Design

The UI for both the ticket purchaser and event organizer have been designed using Figma.

3.3.1 Ticket Purchaser Website

All available events are published on the website for ticket purchasers (see Figure 19), allowing them to easily navigate through events, select desired tickets, and complete online transactions securely by connecting their browser to MetaMask. The event listing page provides a comprehensive overview of upcoming events, featuring the most popular event in the upper section of the website.

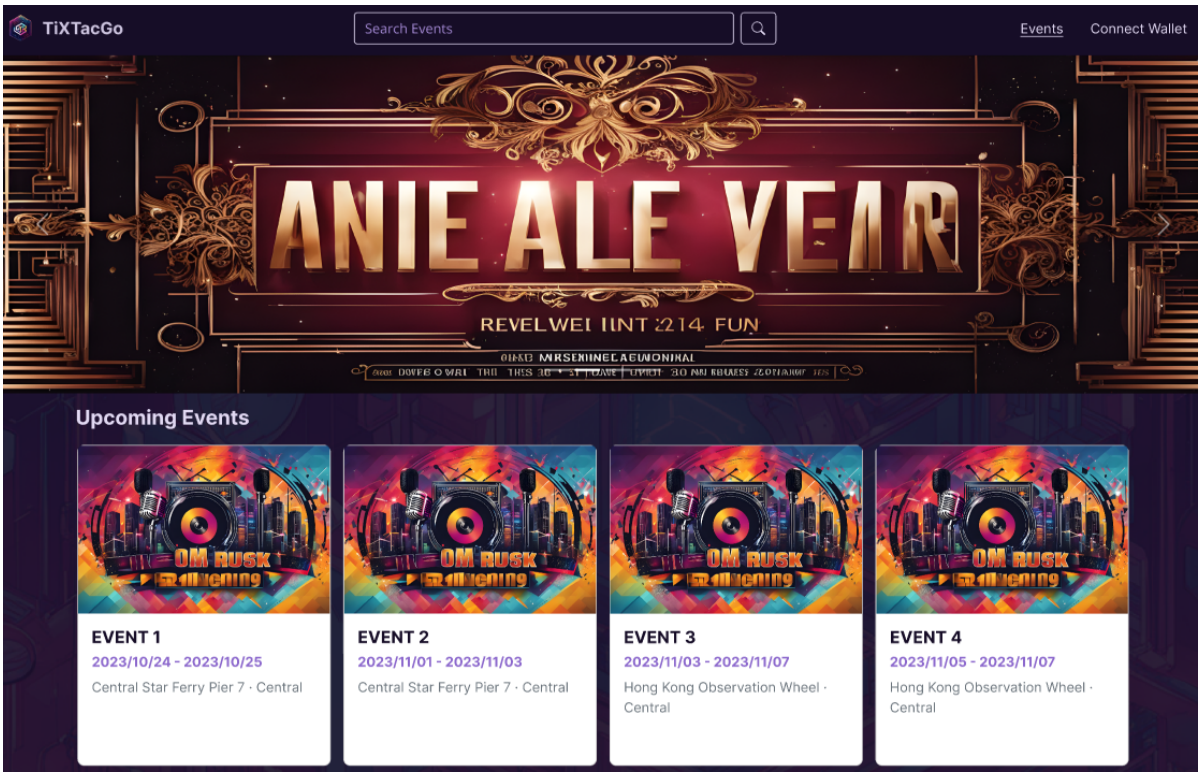

Figure 19 *Event Listing Page*

By selecting a desired event, purchase rules and ticket information are displayed (see Figure 20). Purchasers can select their desired session of the event and seating zone before purchasing tickets using MetaMask. This website reduces the time needed to manually input purchasing details, such as name and credit card information.

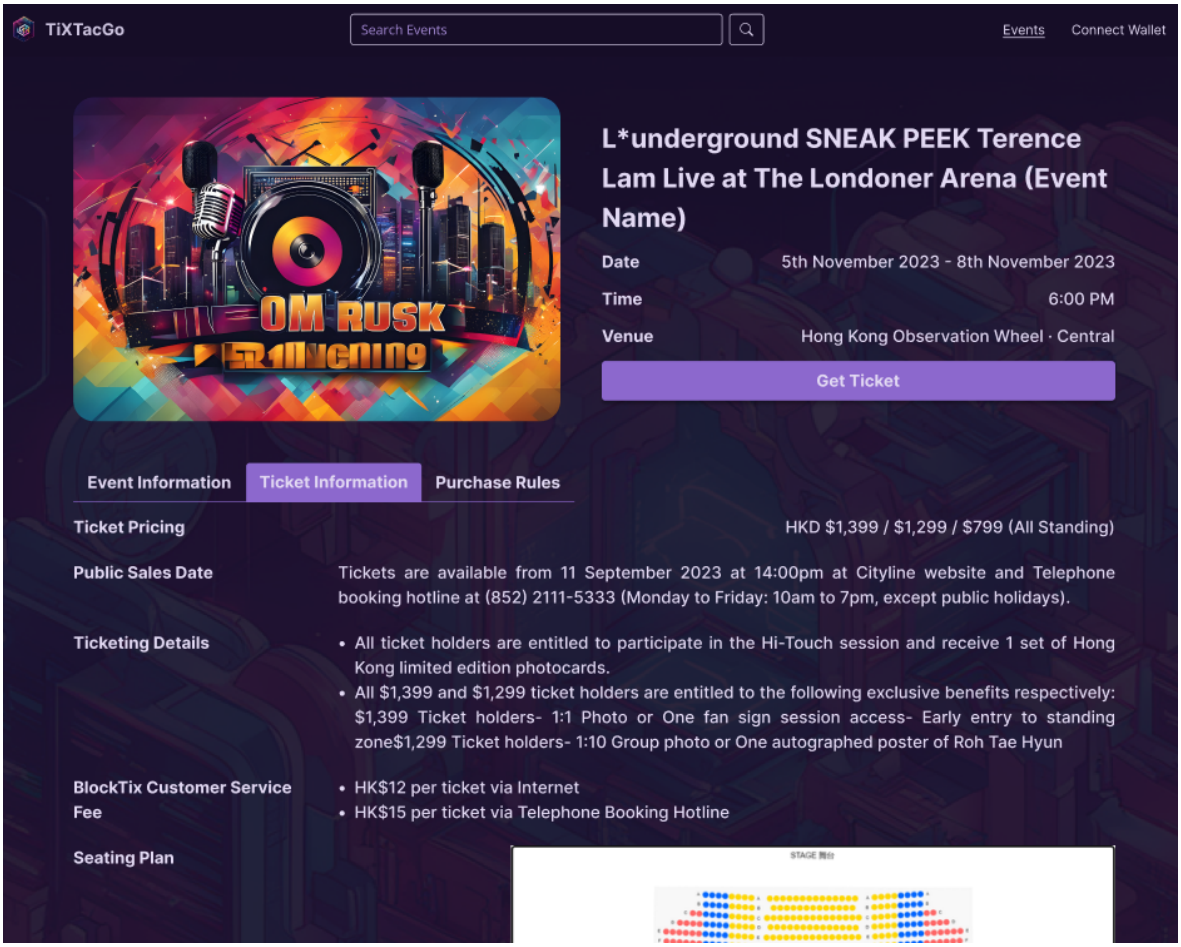

Figure 20 *Event and Ticket Details Page*

3.3.2 Event Organizer Website

The website for event organizers is a dashboard for monitoring sales and uploading event and ticket information. These data are submitted to the smart contract and stored in IPFS as metadata (see Figure 1). When an event is scheduled, event organizers register or sign into their accounts to submit relevant event information. After the event details have been approved by the super admin, the ticket submission page appears (see Figure 21). Figure 21 illustrates how the website enables event organizers to submit essential event details such as date and time, venue, zone pricing, and availability. This user-friendly interface is expected to enhance the ease of event submission and ticket administration.

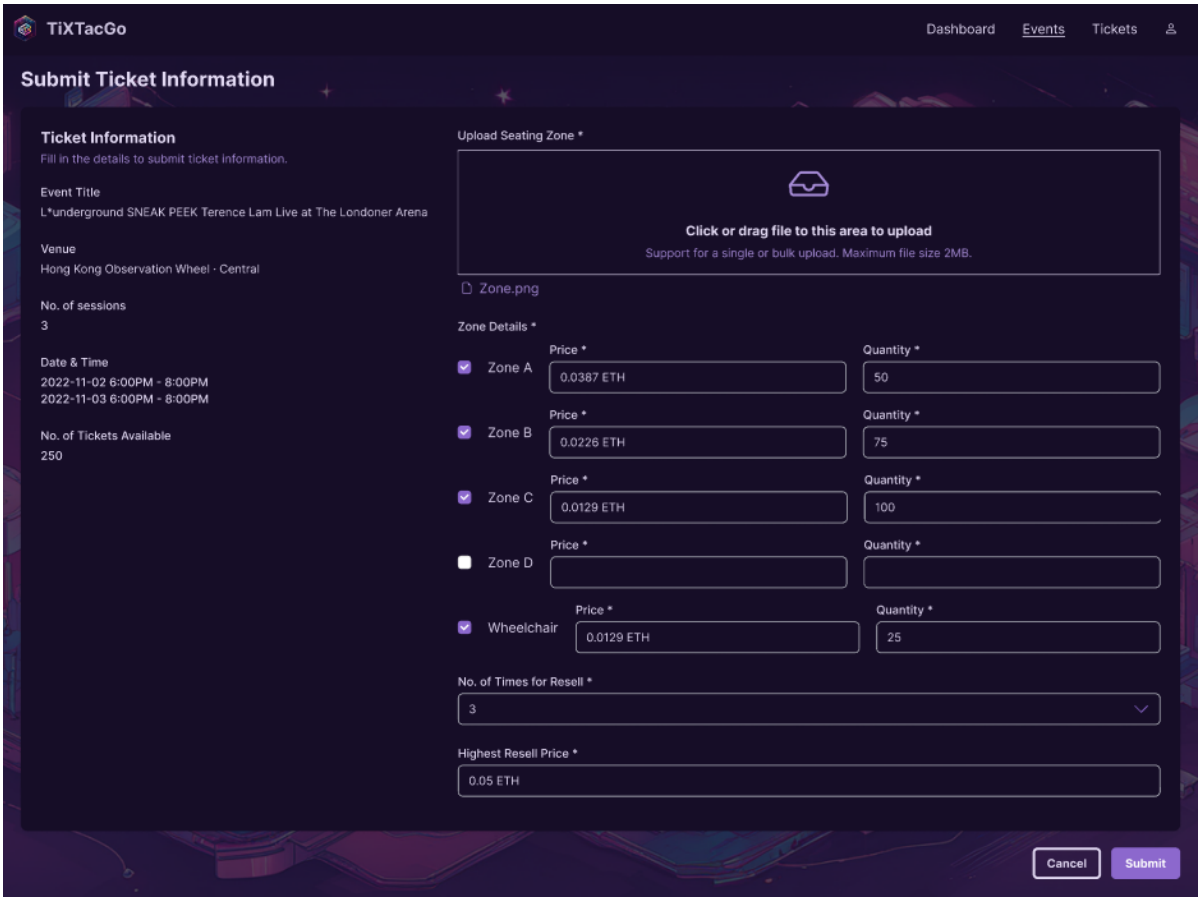

Figure 21 *Ticket Details Submission Page*

4 Future Plan

Table 2 outlines the upcoming project timeline. In subsequent phases, the continuous development of the website will incorporate the implementation of user interface (UI), functionalities, and the dashboard moving forward. Furthermore, the development of the verification logic for QR codes will be initiated.

Date	Details	Status
December 15 – February 10, 2023	· Token retrieval functionalities · Integration of sales data with dashboard · Transaction & transfer of TiXTacGo tickets to purchasers · MongoDB integration with mobile applications · Front-end implementation of mobile applications	In Progress
February 11, 2024	· Front-end implementation of websites · MongoDB integration with websites · Build Relevant Test Cases for Trials	Under planning
March 15, 2024	· Auto refresh functionality for QR codes · Final review & improvement of websites · Implementation to production environment for results	Under planning
March 30, 2024	· Final review of the system	Under planning
April 11, 2024	· Finalized tested implementation	Under planning

Table 2: *Proposed Future Project Timeline*

5 Conclusion

Traditional ticketing systems in Hong Kong offer comprehensive ticketing services for events such as those featuring foreign artists. These systems are essential for facilitating the efficient sale of tickets for various events. Nevertheless, ticket scalping and fraud occur frequently, reducing customer satisfaction and system dependability. The objective of the project is to combine blockchain technology with existing ticketing systems to create a secure, transparent, and decentralized ticketing ecosystem. The workflow and UI for TiXTacGo have been demonstrated while the workflow illustrates in a sequential manner how ticket purchasers, event organizers, and super admins access the system components to perform various actions.

This report has highlighted the implementation of TiXTacGo that uses ETH, MetaMask, smart contracts, and IPFS to create a decentralized system. Also, this system took advantage of using auto-refreshed QR codes to verify the origin and identity of the e-tickets. Research suggests that the integration of these cutting-edge technologies reduces the possibility of ticket scalping and fraud.

In the future development of TiXTacGo, simple and intuitive websites will be developed to benefit ticket purchasers and event organisers. To enhance the secure nature of smart contracts, it may be beneficial to consider their security implementation in both the websites and mobile applications. Investigating how to update the QR code's status once it has been used could be effective in reducing both storage space and verification time. If TiXTacGo can promptly verify the purchaser's identity by extracting the absolute minimum information from ticket purchasers, such as wallet and token ID, peak transaction delay times may be drastically reduced.

6 References

[1] R. Sharma. "Non-Fungible Token (NFT): What It Means and How It Works," Investopedia. Apr. 06, 2023. [Online]. Available: https://www.investopedia.com/non-fungible-tokens-nft-5115211

[2] IBM. "What Are Smart Contracts On Blockchain?," IBM. Accessed: Oct. 23, 2023. [Online]. Available: https://www.ibm.com/topics/smart-contracts#:~:text=Next%20Steps-,Smart%20contracts%20defined,intermediary's%20involvement%20or%20time%20loss

[3] Ethereum.org. "Learn About Ethereum," Ethereum.org. Accessed: Oct. 23, 2023. [Online]. Available: https://ethereum.org/en/learn/#what-is-crypto-ethereum

[4] OpenZeppelin. "ERC721," OpenZeppelin. Accessed: Oct. 23, 2023. [Online]. Available: https://docs.openzeppelin.com/contracts/4.x/erc721#:~:text=ERC721%20is%20a%20standard%20for,across%20a%20number%20of%20contracts

[5] MetaMask. "A Crypto Wallet & Gateway To Blockchain Apps," MetaMask. Accessed: Oct. 23, 2023. [Online]. Available: https://metamask.io/

[6] IPFS. "What Is IPFS," IPFS. Accessed: Sep. 27, 2023. [Online]. Available: https://docs.ipfs.tech/concepts/what-is-ipfs/

[7] MongoDB. "Why Use MongoDB And When To Use It?," MongoDB. Accessed: Sep. 28, 2023. [Online]. Available: https://www.mongodb.com/why-use-mongodb

[8] Python. "What Is Python? Executive Summary," Python. Accessed: Oct. 23, 2023. [Online]. Available: https://www.python.org/doc/essays/blurb/

[9] A. Saini. "An Easy Introduction To Flask Framework For Beginners," Analytics Vidhya. Aug. 23, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/flask-python/#:~:text=Flask%20is%20used%20for%20developing,Lightweight

[10] Solidity. “Contract Metadata,” Solidity. Accessed: Oct. 23, 2023. [Online]. Available: https://docs.soliditylang.org/en/latest/metadata.html

[11] Solidity. “Solidity,” Solidity. Accessed: Oct. 23, 2023. [Online]. Available: https://docs.soliditylang.org/en/v0.8.21/

[12] am730. “演唱會經濟｜新加坡又贏成亞洲第一勝香港　吸巨星開個唱一場賺多少？” am730. Jul. 12, 2023. [Online]. Available: https://www.am730.com.hk/財經/演唱會經濟-新加坡又贏成亞洲第一勝香港-吸巨星開個唱一場賺多少-/387759

[13] C. F. Cheung. “Ticket Scalping.” Legislative Council of the Hong Kong Special Administrative Region. Accessed: Jan. 30, 2019. [Online]. Available: https://www.legco.gov.hk/research-publications/english/essentials-1819ise04-ticket-scalping.htm

[14] Anti-Deception Coordination Centre. “Beware of Online Concert Ticket Trading Fraud.” Anti-Deception Coordination Centre. Nov. 23, 2022. [Online]. Available: https://www.adcc.gov.hk/en-hk/alerts-detail/alerts-1595362836003168257.html

[15] J. Jeffers. “How Enterprises Can Protect Against Ticket Scalping**.**” Security Boulevard. Jan. 18, 2023. [Online]. Available: https://securityboulevard.com/2023/01/how-enterprises-can-protect-against-ticket-scalping/

[16] Hemanthhari2000. “Introduction To Micro Web Framework Flask.” Medium. April. 04, 2021. [Online]. Available: https://medium.com/featurepreneur/introduction-to-micro-web-framework-flask-78de9289270b

[17] T. Nandam. “Implementing Flask Application Using Object Oriented Programming (OOP’s) [Part-1].” Hashnode. Nov. 15, 2021. [Online]. Available: https://nandamtejas.hashnode.dev/implementing-flask-application-using-object-oriented-programming-oops-part-1

[18] IBM. “Benefits Of Blockchain.” IBM. Accessed: Sep. 26, 2023. [Online]. Available: https://www.ibm.com/topics/benefits-of-blockchain

[19] Etherscan. “Ethereum (ETH) Blockchain Explorer.” Etherscan. Accessed: Sep. 26, 2023. [Online]. Available: https://etherscan.io/

[20] Parcl. “What Are Blockchain Gas Fees? A Comprehensive Guide To Gas Fees On The Solana Blockchain.” Parcl. Mar. 20, 2023. [Online]. Available: https://www.parcl.co/blog/what-are-blockchain-gas-fees#:~:text=As of March 2023%2C the,equivalent to a few cents

[21] PolygonScan. “Polygon PoS Chain Explorer.” PolygonScan. Accessed: Sep. 26, 2023. [Online]. Available: https://polygonscan.com/

[22] Polygon zkEVM. “Polygon ZkEVM Blockchain Explorer Average Gas Price Chart.” Polygon zkEVM. Accessed: Sep. 26, 2023. [Online]. Available: https://zkevm.polygonscan.com/chart/gasprice

[23] Solana Explorer. “Explorer | Solana.” Solana Explorer. Accessed: Sep. 26, 2023. [Online]. Available: https://explorer.solana.com/

[24] Polygon zkEVM. “Polygon ZkEVM Explorer.” Polygon zkEVM. Accessed: Sep. 26, 2023. [Online]. Available: https://zkevm.polygonscan.com/

[25] BeaconScan. “Statistics - Validators.” BeaconScan. Accessed: Sep. 26, 2023. [Online]. Available: https://beaconscan.com/stat/validator

[26] Solana Foundation. “Solana Foundation Validator Health Report: March 2023.” Solana Foundation. Mar. 23, 2023. [Online]. Available: https://solana.com/news/validator-health-report-march-2023

[27] Polygon Labs. “Validator Decentralization: Protecting The Network, Securing The Future.” Polygon Labs. Nov. 9, 2022. [Online]. Available: https://polygon.technology/blog/validator-decentralization-protecting-the-network-securing-the-future

[28] Polygon Labs. “Polygon ZkEVM Protocol.” Polygon Labs. Accessed: Sep. 26, 2023. [Online]. Available: https://wiki.polygon.technology/docs/zkEVM/protocol/protocol-components/

[29] Etherscan. “Ethereum Unique Addresses Chart.” Etherscan. Accessed: Sep. 26, 2023. [Online]. Available: https://etherscan.io/chart/address

[30] A. Zirojevic. “Solana Smashes $20 Barrier As Active Users Surge; What’s Next For SOL Price?” Finbold. May. 29, 2023. [Online]. Available: https://finbold.com/solana-smashes-20-barrier-as-active-users-surge-whats-next-for-sol-price/

[31] PolygonScan. “Polygon PoS Chain Unique Addresses Chart.” PolygonScan. Accessed: Sep. 26, 2023. [Online]. Available: https://polygonscan.com/chart/address

[32] Polygon zkEVM. “Polygon zkEVM Unique Addresses Chart.” Polygon zkEVM. Accessed: Sep. 26, 2023. [Online]. Available: https://zkevm.polygonscan.com/chart/address

[33] Ethereum.org. “Consensus Mechanisms.” Ethereum.org. Jan. 14, 2023. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/

[34] Solana Foundation. “Proof of History: How Solana Brings Time To Crypto.” Solana Foundation. Nov. 30, 2021. [Online]. Available: https://solana.com/news/proof-of-history

[35] Polygon Labs. “What is Polygon PoS?” Polygon Labs. Accessed: Sep. 26, 2023. [Online]. Available: https://wiki.polygon.technology/docs/pos/what-is-polygon-pos/

[36] Polygon Labs. “What is Polygon ZkEVM?” Polygon Labs. Accessed: Sep. 26, 2023. [Online]. Available: https://rise.articulate.com/share/p8w6t759TahpdxqMUrUI6tvPWaF8YcBJ#/lessons/EvHYIjRjrV_2GLkhNpZMeBdFv3tPpTaG

[37] Ethereum.org. “Javascript API Libraries.” Ethereum.org. Accessed: Jul. 12, 2023. [Online]. Available: https://ethereum.org/en/developers/docs/apis/javascript/

[38] MetaMask. “Home | MetaMask Developer Documentation.” MetaMask. Accessed: Sep. 27, 2023. [Online]. Available: https://docs.metamask.io/