# Web3 application implementing an NFT ticketing platform

Background and Specification Progress Report

Author: Henry Li
Student ID: 1923074
Supervisor: Dr Agi Kurucz

December 2022

# Contents

# 1. Introduction

## 1.1. Aims & Objectives

An emerging technology in recent years has been the use of blockchain and cryptocurrencies. I am passionate to explore and learn about these technologies and implement them in an application that has real-world use cases that benefit users.

As this will be my first time diving into using these technologies it will require a significant amount of research and self-learning. This project will let me further my knowledge and skills in this area significantly.

The main requirements for this project will be:

- To have a web application that will allow users to purchase tickets for music events
- The ticket should be in the form of an NFT, which is generated by a smart contract on a blockchain
- The application should allow the user to connect their crypto wallet in order to perform transactions
- The application should be user-friendly by having good flow and layout

# 2. Background

## 2.1. Blockchain

A blockchain is a digital form of data storage, where transactions are logged and stored in chronological order. A block consists of many transactions which subsequently form the blockchain. These blocks must be verified by

nodes on the blockchain network in order to be accepted and added to the blockchain. Consensus mechanisms such as Proof of Work (PoW) and Proof of Stake (PoS) are used in order to decide whether blocks are accepted into the blockchain or not. (1)

As blockchains are maintained by multiple nodes, it gives them a decentralised nature as no one single entity owns the blockchain. This gives them special characteristics compared to traditional data storage solutions. One of these key characteristics is the fact that data on the blockchain is immutable, this means once the data is on the blockchain it cannot be modified.

Each block is identified by its own unique cryptographic hash value. The hashing process involves using meta-data such as the checksum of the hash from the preceding block. Therefore, if a malicious actor attempted to modify data on a block it would invalidate the hashes of future blocks, breaking the blockchain. (2)

For there to be an incentive for nodes to validate the network, a transactional fee called a gas fee is paid to the validators when they calculate the hash for the latest block to be added to the network. (3) The Ethereum blockchain is a popular blockchain that incorporates this system, requiring a gas fee to be paid for every transaction that writes data to the blockchain. The gas fee is set by the network's validators (miners), meaning that it can fluctuate depending on supply and demand between the miners and users of the network writing to the blockchain. (4)

## 2.2. Web3

Web2 is the version of the internet most users are interacting with today. This version of the web is dynamic as it allows users to read and write to the web.

This version of the web is dominated by tech giants such as Apple, Meta, Google and Amazon and has led to an explosion of the number of users using the web daily. (5) Data stored on Web2 platforms are centralised, meaning it is owned by a single entity. This has led to privacy and security concerns for users as they have little control over their online information.

Web3 is the next iteration of the web that aims to solve these issues by providing decentralisation of the web. This version of the web is governed by the users themselves and data is mainly stored on peer-to-peer networks such as blockchains. Web3 provides users full ownership of their data, i.e., they can "read-write-own" the web and do not have to rely on central entities such as Google to store their data. (6)

## 2.3. Non-Fungible Tokens

A fungible asset is an asset that can be swapped or traded for an identical object which has the same value; for example, a dollar bill can be exchanged for another dollar bill — they are of equal value.

As opposed to fungible assets, non-fungible assets are different as they cannot easily be swapped or traded for another non-fungible asset as each non-fungible asset is considered as "one-of-a-kind", giving them a collectable nature.

A Non-Fungible Token (NFT) is a blockchain-based record that represents a piece of digital media and proves ownership to its owner. NFTs can represent any type of digital asset — examples include digital artwork, music files and concert tickets. The metadata associated with these NFTs can be stored using a decentralised file system called IPFS (InterPlanetary File System).

NFTs can be purchased and sold through Web3, using cryptocurrencies such as Bitcoin or Ethereum to facilitate the payment. As these NFTs are stored on the blockchain, it means that no user can alter or change the NFT. (7)

## 2.4. Smart Contracts

A smart contract is a piece of programming code that is deployed onto the blockchain. They can include functions that can be executed when specified conditions are met. As smart contracts are deployed onto the blockchain, it means the logic of the code cannot change — if you wanted to change a function in a smart contract it would require another deployment of the contract onto the blockchain. (8)

Smart contracts can be used to create NFTs, this process is called “minting” and involves creating a unique digital asset and publishing it to the blockchain. For Ethereum, there is a widely used ERC-1155 token standard which facilitates functions including minting, transfer and burning (destruction) of tokens. (9)

## 2.5. Web Ticketing Platforms

A web ticketing platform is usually hosted on the World Wide Web and can usually be accessed through a web browser or smartphone application. These platforms allow the user to browse and purchase tickets for any type of event or service.

Once the ticket is purchased, users will usually be able to either collect a physical ticket from a specified location, print the ticket, or show the ticket on an electronic device such as a smartphone.

## 2.6. Security

Purchase of tickets through traditional online web ticketing platforms is typically completed through a credit or debit card transaction, where the customer authorises the merchant to charge the card that they have provided their details for. As these transactions occur over the internet, it requires security measures such as HTTPS (10) to encrypt data being sent between the merchant and customer to prevent data theft. If the merchant does not implement sufficient security measures it poses severe security threats to the customer.

Modern browsers such as Google Chrome have implemented warnings to inform users when they are visiting websites that make them vulnerable to threats such as Man-in-the-Middle (MITM) attacks (11), however users running outdated browsers may not see such warnings and continue to visit these websites and have their information stolen.

In this project I will be using Web3 technologies to avoid these issues associated with Web2 ticketing platforms. Payment will be made through cryptocurrency transactions which have to be signed by the sender's private key to ensure the transaction is not sent maliciously. (12) This also prevents the need for any sensitive information such as credit card numbers to be sent over the internet, which risk being stolen.

## 2.7. Ticket Resale

A major issue concerning current ticketing platforms is the secondary ticket market. Major ticketing companies such as Ticketmaster face frustration and backlash by customers due to users buying a large number of tickets and reselling them on the secondary ticket market at extortionate prices. This practise has been informally named as "price gouging" or "scalping" and has

been a centre of criticism for many ticketing companies in recent years. In 2019 the secondary ticket market was worth $15.19 billion which has prompted the creation of legislation in Europe to help combat the issue. (13)

The use of NFTs can be used to prevent scalping of tickets as the ticket can be represented as a token on the blockchain, proving the ownership to the buyer. These tickets cannot be traded or resold unless functionality is added in the smart contract. As resold functionality is defined in the smart contract, it can be used to limit the price at which tickets are resold for in order to prevent excessive pricing. This also means that resale can occur officially, as opposed to through third party services which lowers the risk of buyers from being victims of fraudulent behaviour.

# 3. Requirements & Specification

## 3.1. Requirements

I have decided to use the Ethereum blockchain for my web application as this is one of more mature blockchains that has more development tools and documentation available. In addition, I will be using the Goerli testnet to develop my application so that no real Ether (ETH) cryptocurrency will need to be used for transactions.

This web application will require the installation of the MetaMask browser extension[1] to facilitate the purchase transactions to the smart contract from the user. MetaMask is available on both Firefox and Chromium based browsers. If the user does not have this extension installed it will direct them an installation page.

[1] https://metamask.io/

### 3.1.1. Content Requirements

- C1 — Application should be able to store music events.
- C2 — User should be able to search for music events by artist, genre or location.
- C3 — User should be able to filter events by genre, location and price.
- C4 — User should be able to select an event to view more details.
- C5 — User should be able to connect their MetaMask wallet.
- C6 — User should be able to buy a ticket for an event.
- C7 — User should be able to list a ticket they have purchased for resale.
- C8 — Application should show a confirmation prompt when buying or reselling a ticket.
- C9 — Application should provide appropriate visual feedback to the user in the case of errors or messages.
- C10 — User should be able to view the tickets they have purchased.
- C11 — Application should have an intuitive and easy to navigate Graphical User Interface (GUI).
- C12 — Application should recommend events based on previous purchases.
- C13 — Application should have an admin interface to modify events

### 3.1.2. Smart Contract Requirements

- S1 — Smart contracts should be deployed onto the Goerli testnet.
- S2 — Smart contract should allow the purchase of an NFT.
- S3 — Smart contract should allow the resale of an NFT.
- S4 — Smart contract should provide the remaining quantity of NFTs available.
- S5 — Smart contract should be able to transfer its balance of Ether to a specified wallet address.
- S6 — Smart contract should be able to mark an NFT as used.
- S7 — Smart contract should be able to associate metadata with an NFT.
- S8 — Smart contracts should be automatically generated by events in the application.

## 3.2. Specification

The priority of each requirement will be assessed; high priority requirements are considered essential for the project, whilst low priority requirements are welcome additions that are not critical. I will implement the requirements in the respective order: high, medium, low.

### 3.2.1. Content Specification

| Requirement Code | Requirement | Specification | Priority |
|---|---|---|---|
| C1 | Application should be able to store music events. | MySQL database will be used to store events, artists, genres, and venues.<br><br>Events fields: name, description, artist ID, venue ID, date & time, price, quantity, contract address, IPFS address.<br><br>Artist fields: name.<br><br>Venue fields: name, city. | High |
| C2 | User should be able to search for music events by artist, genre or location. | Search bar in the navigation bar of the website. Event listeners to detect text changes which then is used to query the database for events. | Medium |
| C3 | User should be able to filter events by genre, location and price. | Filter checkboxes, tick-boxes on the events page. These inputs will be used to query the database for events matching the criteria. | Low |
| C4 | User should be able to select an event to view more details. | User should be able to click an event which takes them to a page for that specific event. | Medium |
| C5 | User should be able to connect their MetaMask wallet. | Button which causes MetaMask to prompt the authorise their wallet with the website. | High |
| C6 | User should be able to buy a ticket for an event. | Button which causes MetaMask prompt to authorise transaction to buy the NFT. | High |

| Requirement Code | Requirement | Specification | Priority |
|---|---|---|---|
| C7 | User should be able to list a ticket they have purchased for resale. | Button which sends a request to the server to list the ticket for resale. Server will interact with the smart contract to accomplish this. | High |
| C8 | Application should show a confirmation prompt when buying or reselling a ticket. | A confirmation modal should appear when buying or reselling a ticket. They can either confirm or cancel the action. | Low |
| C9 | Application should provide appropriate visual feedback to the user in the case of errors or messages. | Alerts should show on the user's webpage. | Medium |
| C10 | User should be able to view the tickets they have purchased. | A page where they can view the purchases they have made before. | High |
| C11 | Application should have an intuitive and easy to navigate Graphical User Interface (GUI). | Web elements should be easy to read and spaced well. The layout should be responsive to accommodate different screen sizes. | Medium |
| C12 | Application should recommend events based on previous purchases. | AI recommender system will take previous purchases and generate recommendations based on a dataset. | Low |
| C13 | Application should have an admin interface to modify events | A password protected page where events can be modified. Changes will be reflected in the database. | Low |

### 3.2.2. Smart Contract Specification

| Requirement Code | Requirement | Specification | Priority |
|---|---|---|---|
| S1 | Smart contracts should be deployed onto the Goerli testnet. | Infura node provider[1] will be used to deploy contracts onto the Goerli blockchain. | High |
| S2 | Smart contract should allow the purchase of an NFT. | Contract should implement the ERC-1155 token standard[2] to create NFTs. The creation of the NFT should require payment. | High |
| S3 | Smart contract should allow the resale of an NFT. | Contract should have a function that marks it for resale and a function to buy the resale ticket. | High |
| S4 | Smart contract should provide the remaining quantity of NFTs available. | Contract function returning remaining quantity. | Medium |
| S5 | Smart contract should be able to transfer its balance of Ether to a specified wallet address. | Contract function to transfer balance to address. | Low |
| S6 | Smart contract should be able to mark an NFT as used. | Contract function to mark NFT as used. | Low |
| S7 | Smart contract should be able to associate metadata with an NFT. | Contract function to set the URI of the token, which refers to its metadata. | Low |
| S9 | Smart contracts should be automatically generated by events in the application. | Endpoint on server that fetches event data from the database and generates smart contracts, updating the contract addresses of the events in the database. | Medium |

| Requirement Code | Requirement | Specification | Priority |
|---|---|---|---|
| 1. https://www.infura.io/<br>2. https://docs.openzeppelin.com/contracts/3.x/erc1155/ | | | |

# 4. Design

## 4.1. Use Cases

In the use case diagram below, we identify the interactions between the actors and the systems, defining the main functionalities of our application.

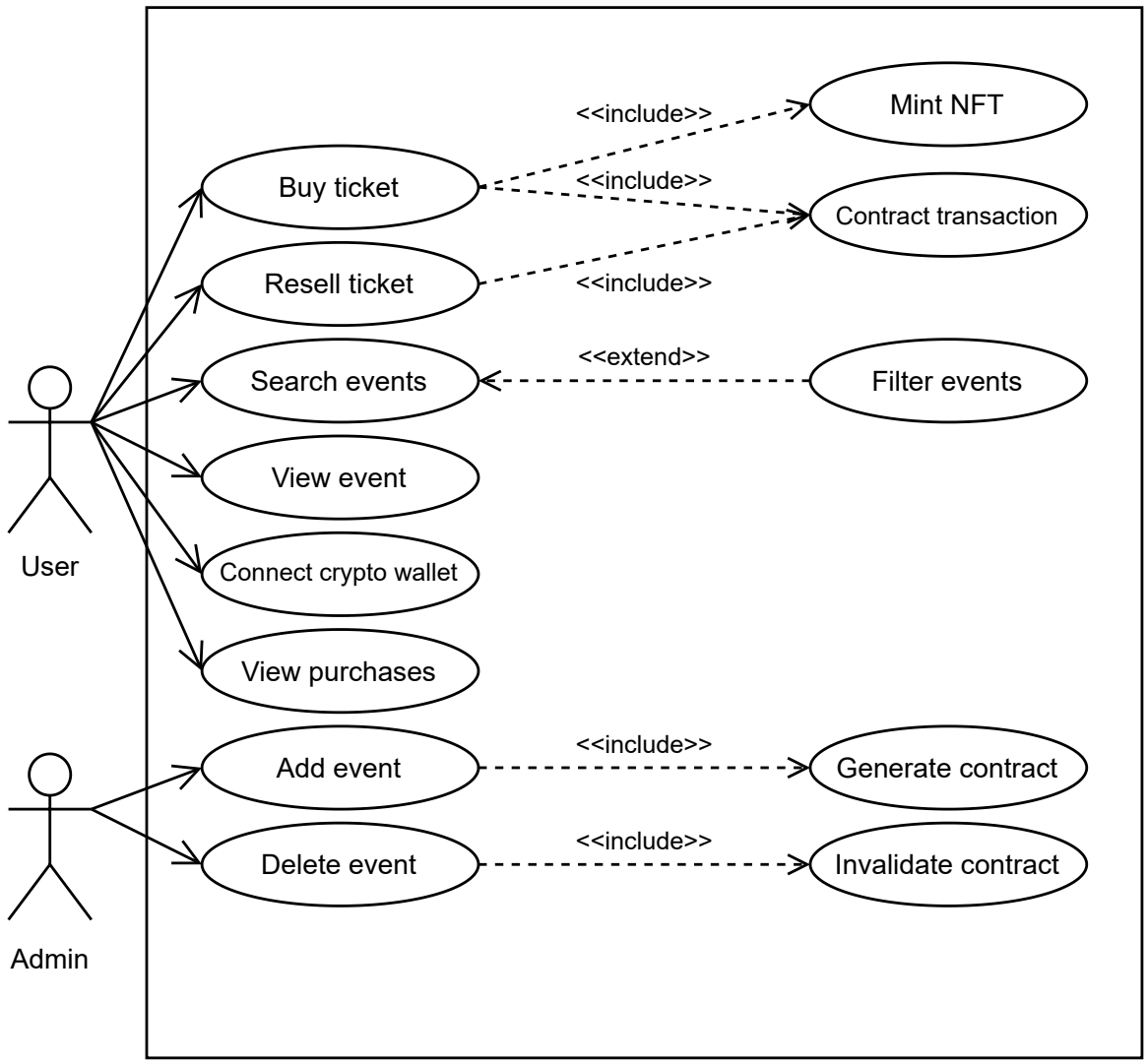


**Figure 4.1: Web3 Application Use Case Diagram**

Users will access the application via a URL on their web browser, they will arrive at the home page of the application where they will be able to navigate to events and connect their crypto wallet. Once users have connected their crypto wallet, they will be able to buy tickets and view their previous purchases, where they also can list a ticket for resale.

Admins will be able to access their interface by a specific URL, where they will need to enter a password for authentication. Once authenticated they can add and delete events.

## 4.2. System Architecture

The system architecture is divided into three main layers. The interactions between the different components of these subsystems can be seen in the diagram below (Figure 4.2). The arrows between the subsystems indicate dependency.

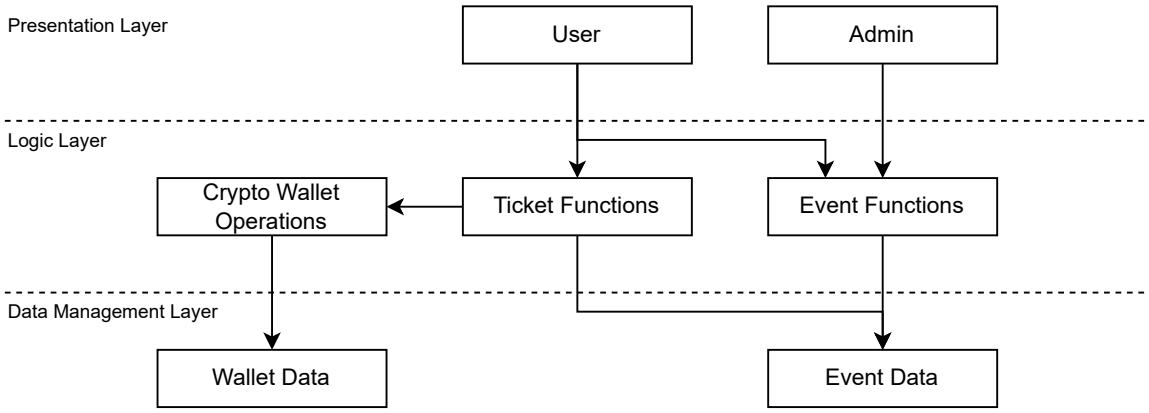


**Figure 4.2: Web3 Application System Architecture Diagram**

The presentation layer represents what the actor will see when they use the application. These are reliant on the logic of the subsystems in the logic layer. Users can interact with event functions (viewing) and ticket functions (buying/reselling) which require the crypto wallet operations. Meanwhile admins can interact with a different subset of event functions (viewing/adding/deleting). Each of the logic layers work on some data which will be

available to interact with either on the application's database or the blockchain.

## 4.3. Data Flow

In the data flow diagram (Figure 4.3) below we can visualise how data moves around in the system for the main use cases.

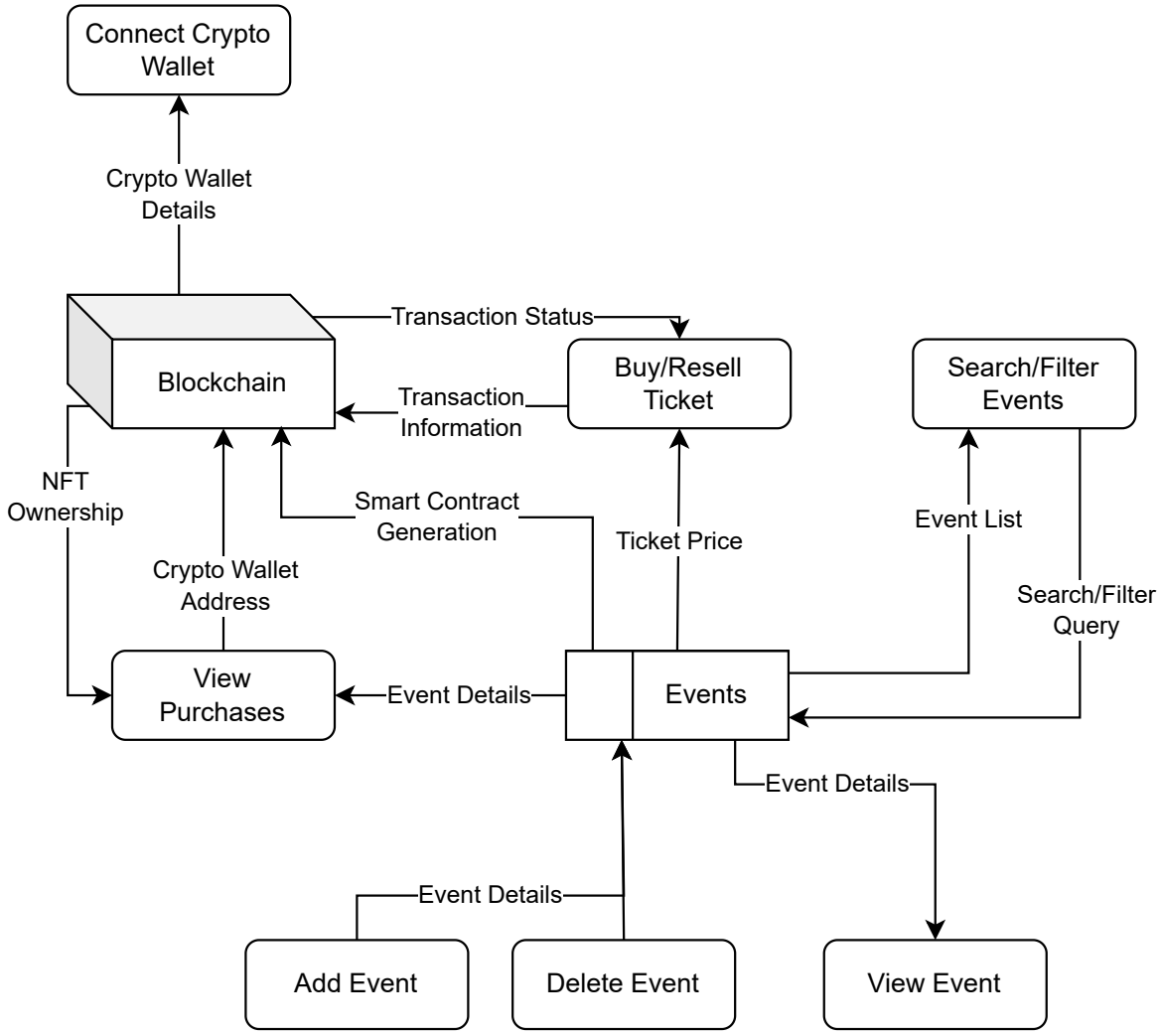


**Figure 4.3: Web3 Application Data Flow Diagram**

The central point of data in this application will be our events, this will be used to generate the smart contracts that will be deployed onto the blockchain which is vital for purchasing NFTs. Although we could theoretically store all the data on the blockchain, it helps to have key

information available within our system itself for quick access. However, we will have to think carefully during implementation about how to deal with potential data inconsistencies between the data on our system and the corresponding data on the blockchain.

The 'user' in this system will be identified by their crypto wallet address as this will be unique on the blockchain. Therefore, there is no requirement to store user data on our database.

The logic behind data retrieval and manipulation in this system will be handled by functions on the smart contract in the case of the blockchain, or through a RESTful API on our backend server in the case of our own database.

## 4.4. Graphical User Interface

The user interface for the application is an essential part of the system and this is primarily what the end users will interact with. There should be a balance between simplicity, flow, ease of use and functionality. Whilst the interface should be friendly for less experienced users, it should avoid restricting functionality for more advanced users. The aim is to have all round good user experience.

As our system is a web-based application, I have designed wireframes for the different views of our website, these can be found in appendix 6.1. As we are using MetaMask extension to initiate transactions with the smart contracts, some of user interaction will be completed through the extension's interface (see appendix 6.2 for an example). Our website should ensure good flow and handoff between the two.

### 4.4.1. Home Page

This is the first page users will see when visiting the website. Here they can view general information and navigate to other pages. A selection of events will be shown on here, with recommended events showing up if they have connected their crypto wallet.

### 4.4.2. Navigation Bar

The navigation bar will be present on all non-admin facing pages. There will be a search bar for events, links to other pages and a button to connect their crypto wallet via MetaMask. When connecting their wallet, a prompt will open in MetaMask asking the user to confirm the action. If they have already connected their wallet, the text of the button will reflect this and instead the button will act as a logout button. Different links will show depending on whether users have connected their wallet or not — for example, a link to the purchases page will not show if they have not connected their wallet.

### 4.4.3. Browse Events Page

Here users will be able to browse the upcoming events they can purchase tickets for. They will have the ability to filter by genre, location and price. When users use the search for events from the navigation bar it will take them to this page with the relevant events that match their search query. They will be able to use the filters with their search query simultaneously.

### 4.4.4. Event Page

This will show key information for the event — its name, date, venue, location, price, description and genres. There will be a purchase button to allow users to buy the ticket. If the event has sold out the button will be disabled with text reflecting it is sold out. If users have not connected their wallet, the

same connect wallet found in the navigation bar will be shown in place of the purchase button instead.

The genres for the events will have their own dedicated pill shaped buttons, when these are clicked it will take them to the browse events page with the respective genre applied in the filters.

### 4.4.5. Purchases Page

This page can only be accessed if the user has connected their wallet, otherwise they will be redirected to the home page.

On this page a list of events and their details the user has purchased will appear. These purchases will have a button to allow users to resell the ticket if they have not done so already. Users will be easily to switch between viewing upcoming and expired events by clicking on the respective tabs near the top of the page. On the top right of the page there will be a button to view their NFTs and their corresponding artwork on OpenSea[2].

### 4.4.6. Admin Page

Similarly to the purchases page admins will be able to switch between viewing upcoming and expired events via their respective tabs. They will have the ability to edit events by clicking on the event to update their details as well as being able to cancel them. A button on the top right of the page will let admins add an event to the application.

### 4.4.7. Admin Add/Edit Event Page

This page will be used for both adding and editing events. Page text and fields will dynamically update depending on which action is taken. Admins can enter and submit the required data which will update the information for

[2] https://opensea.io/

the event in the database. The data entered in these fields will be validated before the data is submitted — if the data for a field is invalid it will show an error message below the field indicating them to amend the issue.

### 4.4.8. Confirmation Modal

This is a simple dialog box that will show with certain actions on the application requiring confirmation. It will display text about the action they are intending to perform where they will have the option to either continue or cancel the action by clicking the respective buttons. This will be used on important actions where making a mistake could cause detrimental effects e.g., accidentally cancelling an event.

# 5. References


1. Cryptopedia. Gemini. [Online] 28 June 2022. https://www.gemini.com/cryptopedia/blockchain-technology-explained/.
2. Sahu, Mayank. upGrad. [Online] 22 November 2022. https://www.upgrad.com/blog/what-makes-a-blockchain-network-immutable/.
3. Frankenfield, Jake. Investopedia. [Online] 27 September 2022. https://www.investopedia.com/terms/g/gas-ethereum.asp/.
4. Development Documentation. Ethereum. [Online] 4 October 2022. https://ethereum.org/en/developers/docs/gas/.
5. Our World in Data. [Online] 2015. https://ourworldindata.org/internet/.
6. Development Documentation. Ethereum. [Online] 26 September 2022. https://ethereum.org/en/developers/docs/web2-vs-web3/.
7. Ethereum. [Online] https://ethereum.org/en/nft/.
8. IBM. [Online] https://www.ibm.com/uk-en/topics/smart-contracts/.
9. Documentation. OpenZeppelin. [Online] https://docs.openzeppelin.com/contracts/4.x/api/token/erc1155/.
10. Cloudflare. [Online] https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/.
11. Hashed Out. The SSL Store. [Online] 14 September 2017. https://www.thesslstore.com/blog/google-chrome-63-tls-interception-warning/.
12. Development Documentation. Ethereum. [Online] 22 November 2022. https://ethereum.org/en/developers/docs/transactions/.
13. Lunny, Oisin. Forbes. [Online] 24 June 2019. https://www.forbes.com/sites/oisinlunny/2019/06/24/the-battle-for-15-19b-secondary-ticket-market-heats-up-with-first-europe-wide-anti-touting-law/.

# 6. Appendix

## 6.1. Graphical User Interface Wireframes

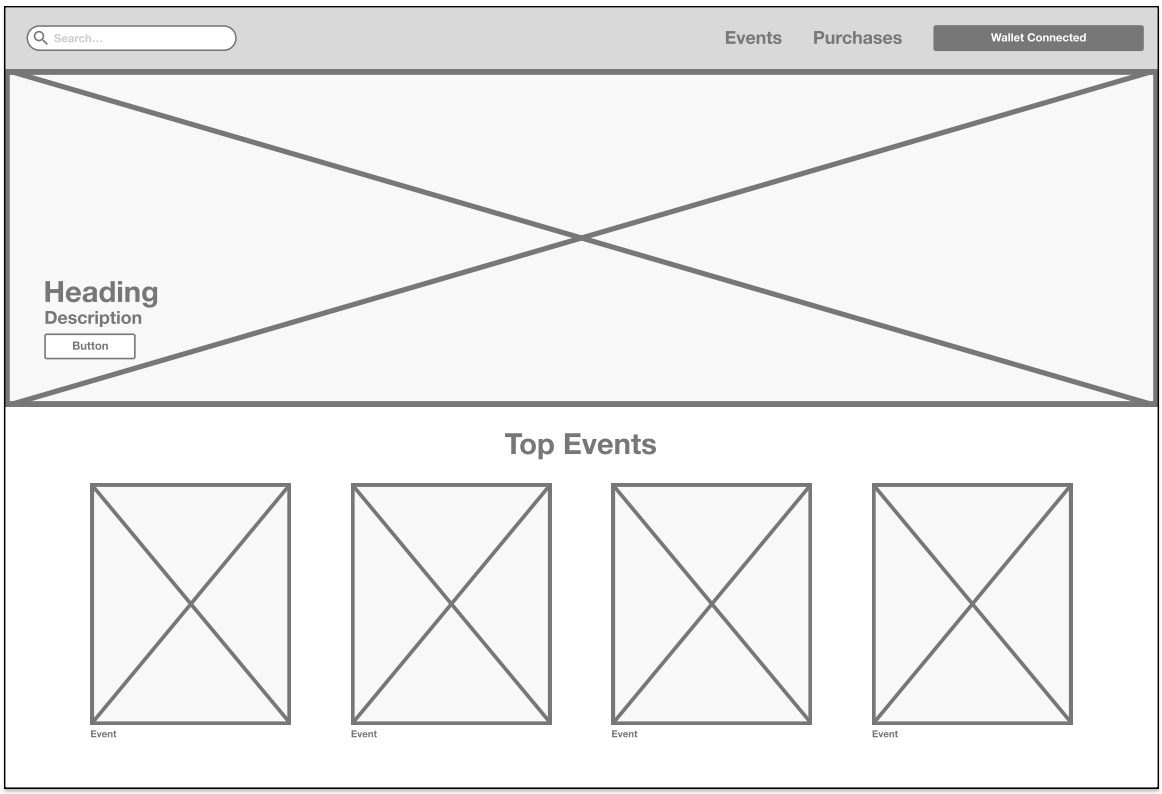


**Figure 6.1.1: Web3 Application Home Page Wireframe**

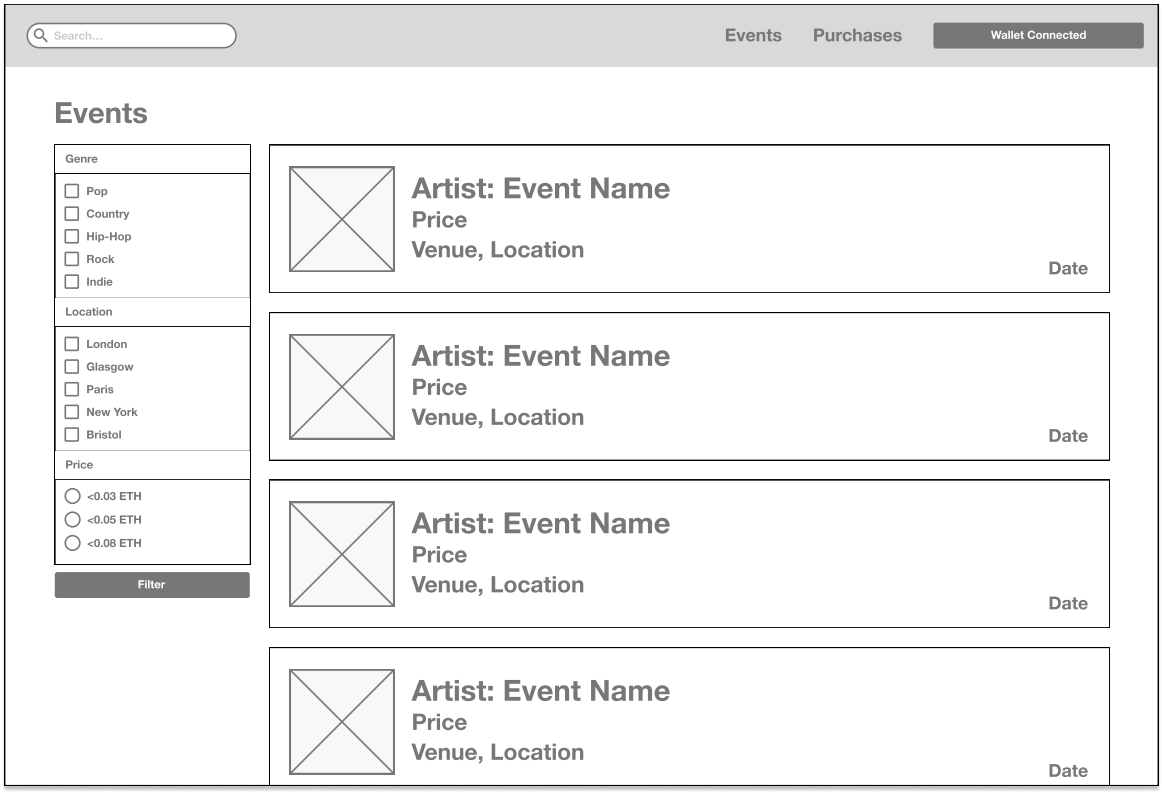


**Figure 6.1.2: Web3 Application Browse Events Page Wireframe**

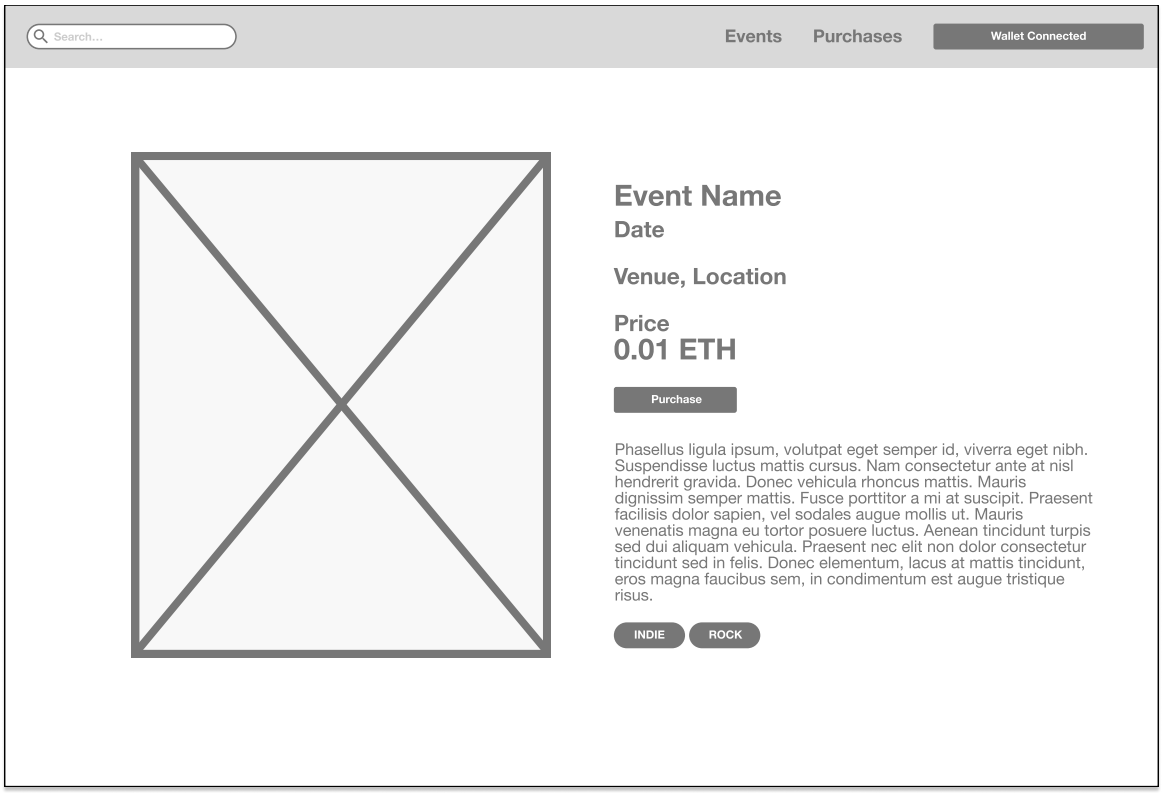


**Figure 6.1.3: Web3 Application Event Page Wireframe**

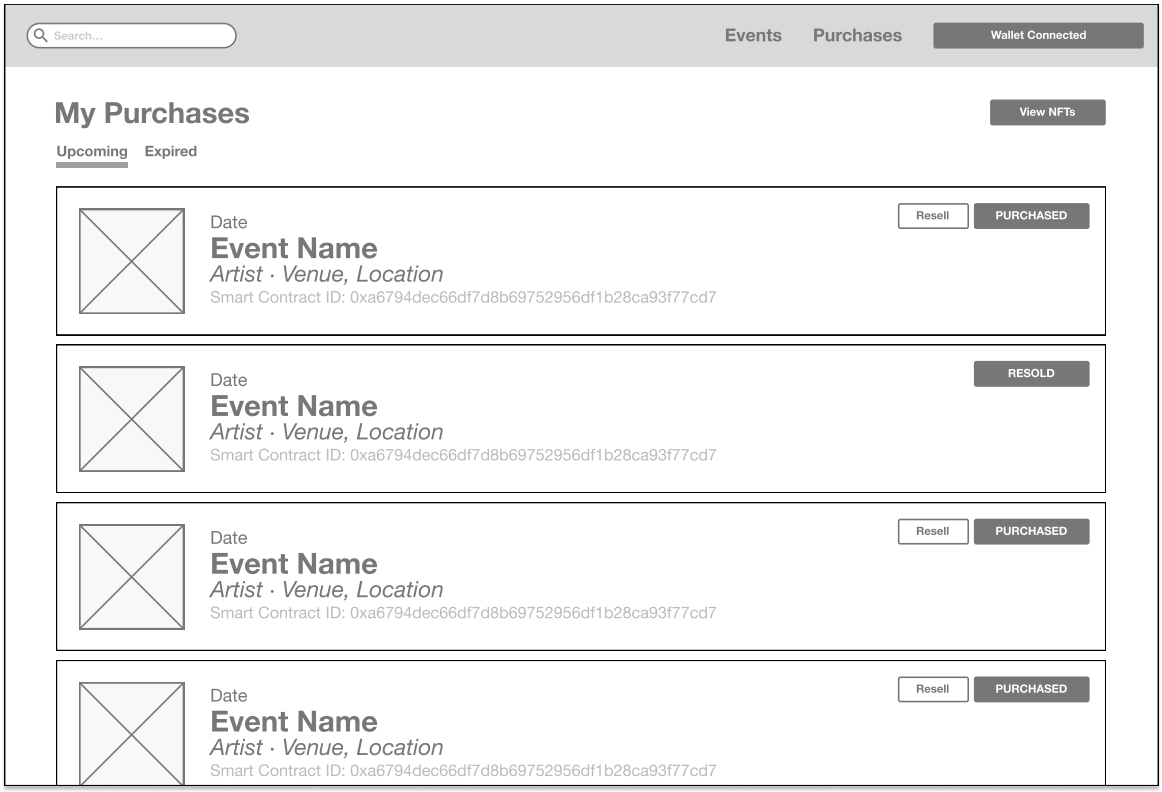


**Figure 6.1.4: Web3 Application Purchases Page Wireframe**

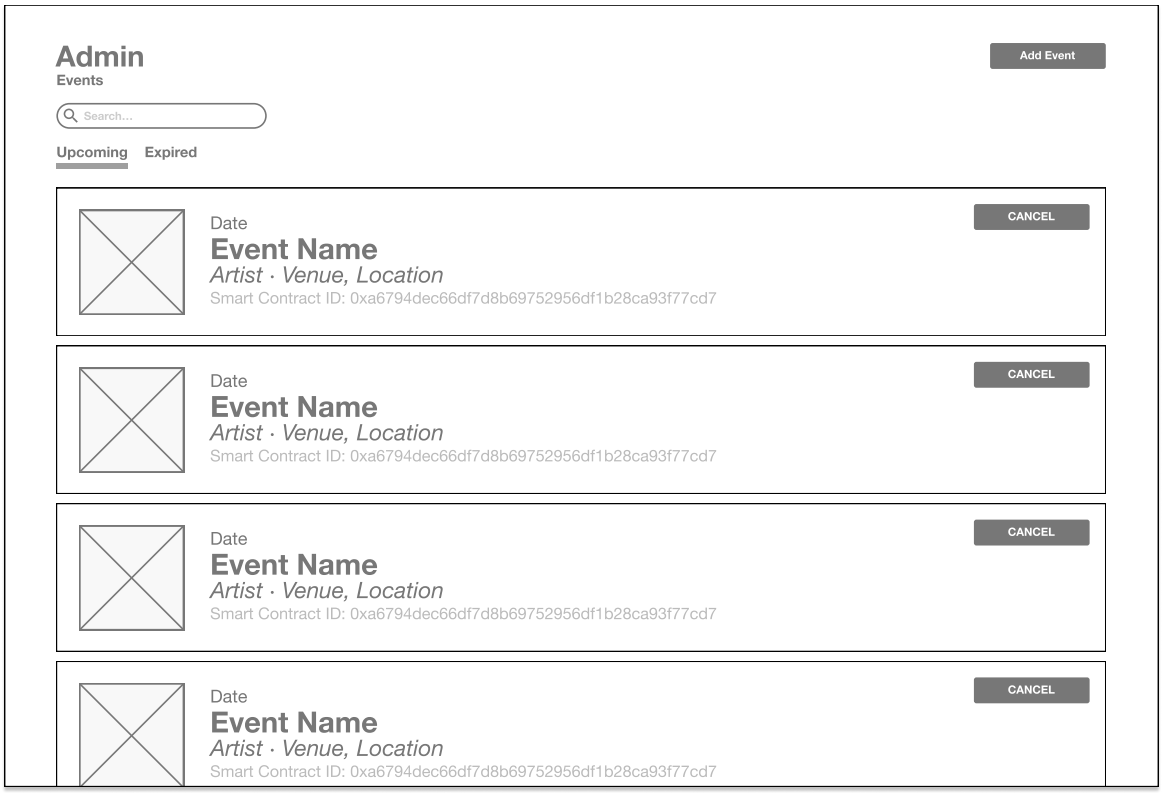


**Figure 6.1.5: Web3 Application Admin Page Wireframe**

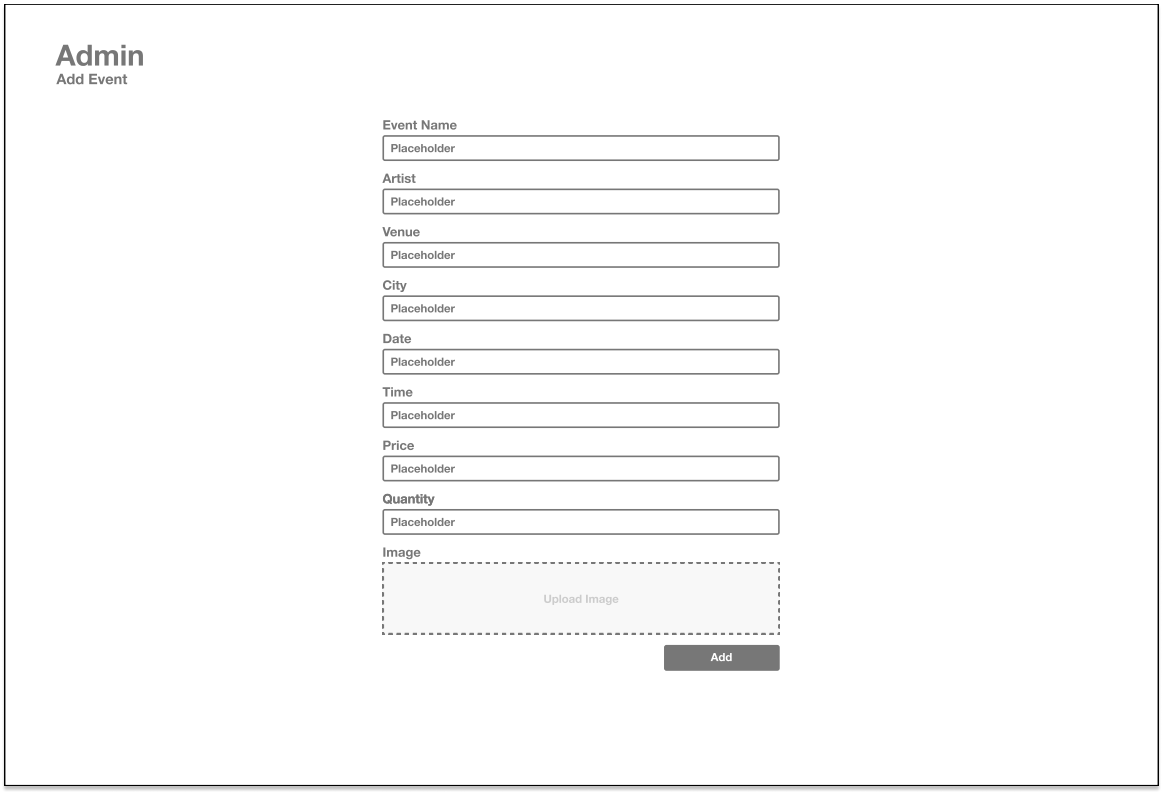


**Figure 6.1.6: Web3 Application Admin Add/Edit Event Page Wireframe**

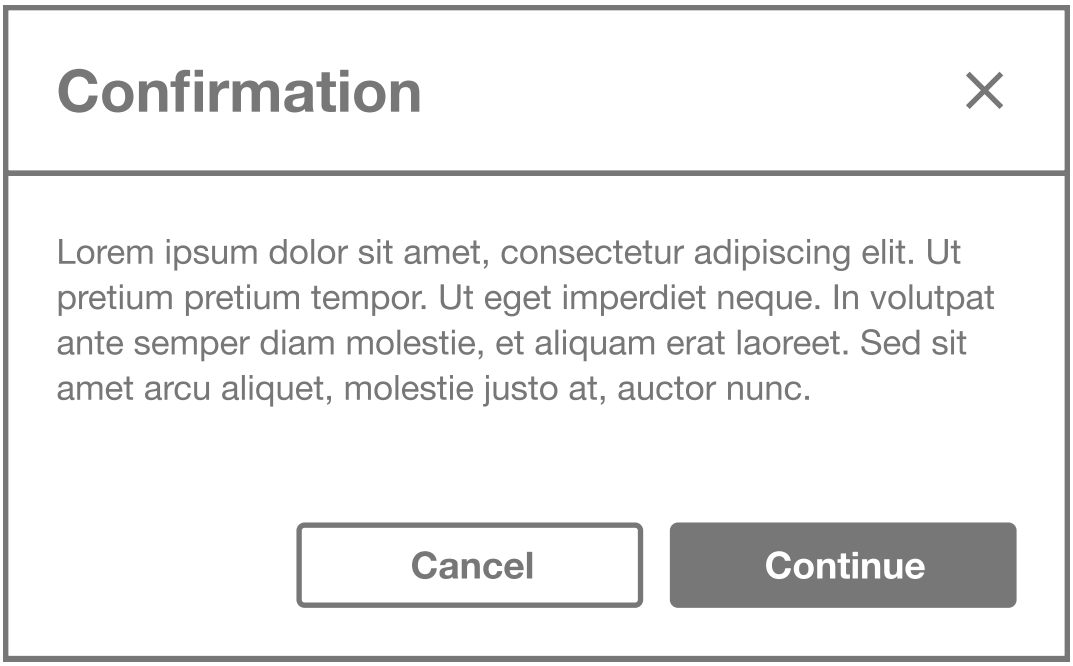


**Figure 6.1.7: Web3 Application Confirmation Modal Wireframe**

## 6.2. MetaMask Transaction Interface

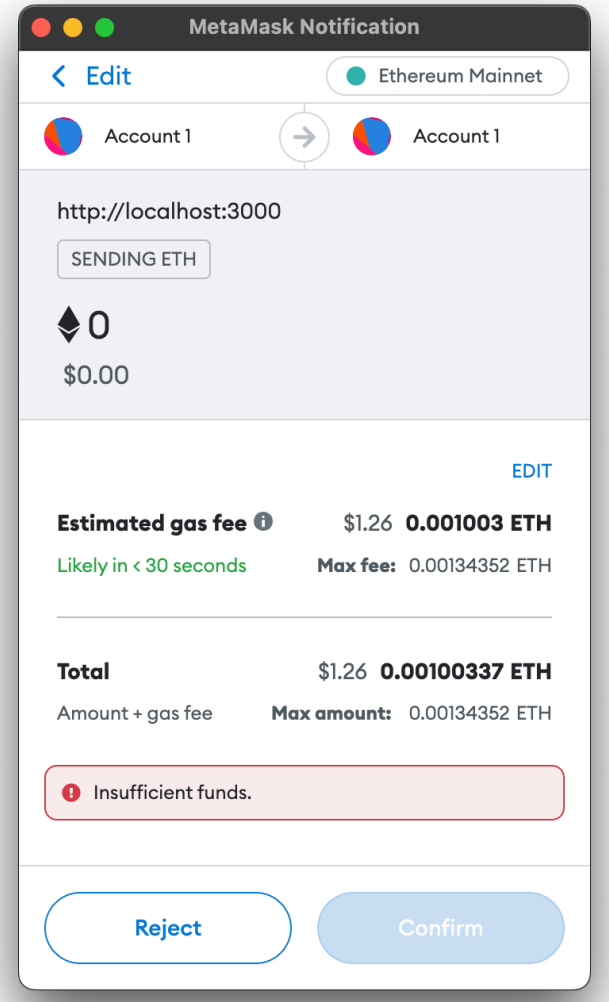


**Figure 6.2: MetaMask Transaction Pop-up**