NFT Marketplace

A PROJECT REPORT

Submitted in partial fulfilment of the

requirement for the award of the degree

of

BACHELOR OF TECHNOLOGY (B.Tech)

in

Information Technology

by

Lavi Jain
199302211

Department of Information Technology

MANIPAL UNIVERSITY JAIPUR

JAIPUR-303007 RAJASTHAN, INDIA

JULY 2023

DEPARTMENT OF INFORMATION TECHNOLOGY
MANIPAL UNIVERSITY JAIPUR, JAIPUR – 303 007 (RAJASTHAN), INDIA

10/07/2023

CERTIFICATE

This is to certify that the project titled **NFT MARKETPLACE** is a record of the bonafide work done by **Lavi Jain**(199302211) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech) in **Information Technology of Manipal University Jaipur, during the academic year 2022-23.**

Ms Smaranika Mohapatra
Project Guide, Dept of Information Technology
Manipal University Jaipur

Dr. Pankaj Vyas
HOD, Dept of Information Technology
Manipal University Jaipur

10/07/2023

CERTIFICATE

This is to certify that the project entitled **NFT MARKETPLACE** was carried out by **LAVI JAIN** (199302211) at **MANIPAL UNIVERSITY JAIPUR** under my guidance during **January, 2023** to **June, 2023**.

Ms Smaranika Mohapatra
Assistant Professor,

Manipal University Jaipur

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Pankaj Vyas, HOD of the Department of Information Technology, and Ms. Smaranika Mohapatra, my project supervisor, for their invaluable guidance and support throughout my B.Tech final year project. I am truly grateful for Dr. Vyas' visionary leadership and continuous encouragement, as well as Ms. Mohapatra's expertise and valuable insights. I would also like to thank the teachers of the Department of Information Technology for their dedication and knowledge, which have greatly contributed to my project's success. Additionally, I am grateful to my classmates, friends, and family for their constant support and motivation. Without the contributions of these exceptional individuals, I would not have been able to accomplish this project. Thank you all for your unwavering support and belief in my abilities

ABSTRACT

One problem that caught my attention and motivated me to undertake a project on NFTs (Non-Fungible Tokens) is the need for secure and authenticated digital asset ownership in today's digital landscape. As the digital world continues to grow, the issue of proving the authenticity and uniqueness of digital assets has become increasingly challenging. This poses significant risks, such as copyright infringement and intellectual property disputes, as it is difficult to verify the ownership and origin of digital assets. However, with the emergence of NFTs, a potential solution arises. NFTs leverage the power of blockchain technology to create verifiable and immutable records of ownership, enabling secure transactions and authentication of digital assets.

Recognizing the significance of this problem and the transformative potential of NFTs, I embarked on a project to delve into the intricacies of this technology. My aim was to contribute to the advancement of secure digital asset ownership and pave the way for innovative solutions in the digital realm. By addressing this problem, I sought to empower creators and users alike, fostering trust and confidence in the digital asset ecosystem.

My journey in developing an NFT marketplace began with a comprehensive understanding of smart contracts. I immersed myself in learning the principles and concepts behind smart contracts and conducted rigorous testing of my code using the Remix IDE. This foundational knowledge provided me with a solid understanding of how smart contracts function and their significance in NFT marketplaces.

Building upon this knowledge, I proceeded to develop the frontend of the marketplace using React.js. This allowed me to create a visually appealing and user-friendly interface that caters to the diverse needs of marketplace users. By employing intuitive design principles and seamless navigation, I aimed to enhance the overall user experience and facilitate effortless engagement with the NFT marketplace.

Simultaneously, I recognized the importance of robust user authentication mechanisms. To achieve this, I integrated Zero Knowledge Proofs (ZKPs) into the backend of the marketplace. Leveraging the Polygon ID SDK and the Iden3JS library in Node.js, I implemented ZKPs to ensure secure and private user authentication. By verifying that users are above the age of 13 and human, I aimed to maintain a trustworthy and reliable environment within the marketplace.

To efficiently store user sign-in details and associated data, I employed MongoDB as the database technology. This choice allowed for efficient data management, ensuring reliable access to user information while maintaining data security. Additionally, I leveraged decentralized storage through the use of IPFS (InterPlanetary File System), specifically utilizing services like Pinata. This approach ensured the secure storage of the images associated with the NFTs, guaranteeing the integrity and accessibility of digital assets within the marketplace.

By combining these technologies and methodologies, I developed an NFT marketplace that prioritizes user privacy, security, and seamless data storage. This comprehensive approach aimed to establish a robust and immersive user experience, fostering trust and confidence in the ownership and exchange of digital assets.

LIST OF FIGURES

Contents

CHAPTER 1:- INTRODUCTION

1.1:- BACKGROUND AND MOTIVATION

The rise of non-fungible tokens (NFTs) in recent years has sparked a global phenomenon that has transformed the way we perceive and interact with digital assets. NFTs have emerged as a groundbreaking technology that allows for the tokenization and ownership of unique digital items, ranging from artwork and music to virtual real estate and collectibles. What are NFTs you ask? NFTs are unique digital tokens that use blockchain technology to establish authenticity, ownership, and scarcity of a particular digital item, whether it be an artwork, video clip, music file, or even a tweet. Unlike cryptocurrencies such as Bitcoin or Ethereum, which are fungible and can be exchanged on a one-to-one basis, NFTs are indivisible and irreplaceable, making them one-of-a-kind digital assets. With the rise of NFTs, theres a need to develop NFT marketplaces.

NFT marketplaces are online platforms where creators can sell their NFTs and collectors can purchase them. These marketplaces serve as the backbone of the NFT market, providing a place for buyers and sellers to connect and facilitating the transaction process.

Currently, there are a number of NFT marketplaces in operation, each with their own strengths and weaknesses. Some of the most successful marketplaces include OpenSea and SuperRare, which have seen significant growth in the past year. According to Dappradar, NFT market's has reached its highest trading volume in June of 2022. The NFT market generated around $24.7 billion worth of organic trading volume in 2022 across blockchain platforms and marketplaces,

As real-world use cases of NFTs continue to expand and gain momentum, there is a strong likelihood of a rapid rise in the number and popularity of NFT marketplaces. The increasing adoption of NFTs in industries such as art, music, sports, gaming, and collectibles demonstrates the tangible value and practical applications of these digital assets. As more artists, creators, and businesses recognize the benefits of tokenizing and selling their unique digital creations, the demand for NFT marketplaces will surge. The marketplace will serve as vital platform for artists and collectors to showcase, trade, and monetize their NFTs, contributing to the growing ecosystem surrounding non-fungible tokens.

1.2 PROBLEM STATEMENT

The objective of this project is to develop an NFT marketplace that allows users to seamlessly buy, sell, browse, and create NFTs listed on a blockchain smart contract. However, a critical challenge to address is the prevention of spam and bots from accessing the application. The presence of spam and bot activity can hinder genuine user engagement, undermine the marketplace's integrity, and create an unfair trading environment. Therefore, a robust and effective method must be implemented to ensure that the NFT marketplace remains secure, reliable, and free from spam and bot interference. By addressing this issue, the project aims to provide users with a trusted and user-friendly platform for NFT transactions while fostering a genuine and authentic community of artists, collectors, and enthusiasts.

1.3 OBJECTIVES

- **Enable Buying NFTs**: The primary objective is to develop a user-friendly marketplace where individuals can browse and purchase NFTs from various creators and artists. This feature will ensure that users have access to a wide range of digital assets and artworks, fostering a vibrant marketplace for buying NFTs.

- **Support Selling NFTs**: The platform will empower users to sell their NFTs easily and efficiently. By providing a streamlined process for listing and showcasing NFTs, sellers will have the opportunity to monetize their digital creations and reach a global audience of potential buyers.

- **Facilitate NFT Minting**: The marketplace will offer the functionality to mint new NFTs. This feature will allow creators to tokenize their digital assets, transforming them into unique and tradable items. By supporting NFT minting, the platform will encourage artists, content creators, and businesses to participate in the NFT ecosystem.

- **User Sign-In and Sign-Up**: To ensure a personalized and secure experience, the marketplace will provide user sign-in and sign-up functionality. This will enable users to create accounts, access their personalized dashboards, and track their NFT transactions conveniently.

- **Credential Verification Using PolygonID**: The marketplace will incorporate PolygonID for credential verification. By leveraging this solution, users can securely verify their identity and establish trust within the platform. This will help prevent fraudulent activities and enhance the overall security of the marketplace.

In conclusion, the project aims to create an NFT marketplace that offers users the ability to buy, sell, and mint NFTs. Additionally, it will provide user-friendly features such as sign-in, sign-up, and verification of credentials using PolygonID. By achieving these objectives, the project aims to establish a trusted and accessible platform where users can actively engage in NFT transactions and contribute to the flourishing NFT ecosystem.

1.4 METHODOLOGY/PLANNING OF WORK

1. **Ideation Phase**: - The ideation phase involves researching, brainstorming, selecting, refining, and prototyping ideas for the NFT marketplace, considering feasibility, user feedback, and market impact to create a prioritized roadmap for development. Also involves deciding the final functionalities of the app.

2. **Designing Phase**: - The design phase involves translating the selected ideas into a tangible blueprint for the NFT marketplace. This phase focuses on creating a detailed design that encompasses the user interface, functionality, architecture, and database structure. It involves wireframing, creating mockups, defining user flows, and designing the overall visual aesthetics of the marketplace. The design phase lays the foundation for the subsequent development and implementation stages.

3. **Adding Functionality**: - The adding functionality phase focuses on implementing the desired features and functionalities identified during the design phase of the NFT marketplace. This phase involves writing code, integrating APIs, developing backend systems, and creating databases to enable the planned functionality. It includes tasks such as user authentication, NFT listing, buying and selling capabilities, minting NFTs, payment processing, and integration with blockchain smart contracts. The functionality phase ensures that the marketplace meets the defined requirements and provides a seamless user experience for all intended actions and interactions.

4. **Testing**: - The testing phase is a crucial stage in the development of the NFT marketplace, where various tests are conducted to ensure the quality, functionality, and usability of the platform. situation and identifying bugs if any.

5. **Finalization**: - The finalization phase is the concluding stage in the development of the NFT marketplace, where the project is prepared for deployment and release.

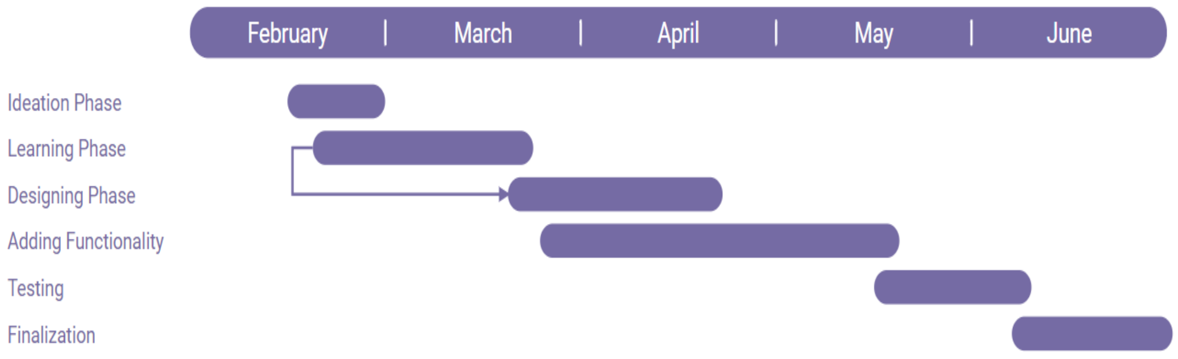


Figure 1:- Gantt Chart

CHAPTER 2: - BACKGROUND MATERIAL

2.1 Overview of NFTs and Blockchain Technology

NFTs are tokens that we can use to represent ownership of unique items. They let us tokenize things like art, collectibles, even real estate. Ownership of an asset is secured by the Ethereum blockchain – no one can modify the record of ownership or copy/paste a new NFT into existence. NFTs stands for non-fungible token, Non-fungible is an economic term that you could use to describe things like your furniture, a song file, or your computer. These things are not interchangeable for other items because they have unique properties. Fungible items, on the other hand, can be exchanged because their value defines them rather than their unique properties. For example, ETH or dollars are fungible because 1 ETH / $1 USD is exchangeable for another 1 ETH / $1 USD, whereas in NFTs every NFT is unique and so non-fungible.

NFTs give the ability to assign or claim ownership of any unique piece of digital data, trackable by using Ethereum's blockchain as a public ledger. Ownership of NFTs is managed through the unique ID and metadata that no other token can replicate. NFTs are minted through smart contracts that assign ownership and manage the transferability of the NFT's. When someone creates or mints an NFT, they execute code stored in smart contracts that conform to different standards, such as ERC-721. This information is added to the blockchain where the NFT is being managed.

The minting process, from a high level, has the following steps that it goes through:

- Creating a new block
- Validating information
- Recording information into the blockchain

NFT's have some special properties:

Each token minted has a unique identifier that is directly linked to one Ethereum address.

They're not directly interchangeable with other tokens 1:1. For example 1 ETH is exactly the same as another ETH. This isn't the case with NFTs.

Each token has an owner and this information is easily verifiable. They live on the specific Blockchain network and can be bought and sold on any that specific blockchain-based NFT market. In other words, if you own an NFT: You can easily prove you own it.

Proving you own an NFT is very similar to proving you have ETH in your account. For example, let's say you purchase an NFT, and the ownership of the unique token is transferred to your wallet via your public address.

The token proves that your copy of the digital file is the original. Your private key is proof-of-ownership of the original.

The content creator's public key serves as a certificate of authenticity for that particular digital artefact.

The creators public key is essentially a permanent part of the token's history. The creator's public key can demonstrate that the token you hold was created by a particular individual, thus contributing to its market value (vs a counterfeit). Another way to think about proving you own the NFT is by signing messages to prove you own the private key behind the address.

As mentioned above, your private key is proof-of-ownership of the original. This tells us that the private keys behind that address control the NFT. A signed message can be used as proof that you own your private keys without revealing them to anybody and thus proving you own the NFT as well!

No one can manipulate it in any way. You can sell it, and in some cases this will earn the original creator resale royalties. Or, you can hold it forever, resting comfortably knowing your asset is secured by your wallet on Ethereum.

2.2 Self Sovereign Identity Solution : Polygon ID

What is self-sovereign identity?

Self-sovereign identity (SSI) is a model for managing digital identities in which individuals or businesses have sole ownership over the ability to control their accounts and personal data. Individuals with self-sovereign identity can store their data to their devices and provide it for verification and transactions without the need to rely upon a central repository of data. With self-sovereign identity, users have complete control over how their personal information is kept and used.

Self-sovereign identity is made up of claims, proofs and attestations:

- A claim is an assertion of identity made by the user.
- Proofs are the forms or documents that act as evidence for a claim. For example, a proof could be a passport or birth certificate.
- An attestation, or validation, is when the other party validates the claim is true. Attestations can be stored in the user's device and are typically machine readable.

Leading blockchain infrastructure, Polygon unveiled an Identity solution Polygon ID, a decentralized identity infrastructure stack that will solve the digital trust issue currently present in the DeFi space. Under an open-source license, the Polygon ID solution will allow developers to fill blockchain space with self-sovereign, decentralized, and private identity solutions for users. The Polygon ID leverages zero-knowledge proofs to interact with smart contracts, which requires off-chain credentials. It includes a diploma certificate, driver's license, or national ID. In the process, Polygon claims that no third party can access users' data and their privacy will stay safe.

The public release adds four new tools to the Polygon ID Toolset:

- Verifier SDK (for verifiers)
- Issuer Node (for issuers)
- Wallet SDK (for wallet builders)
- Wallet App (for devs and end users)

Through the Polygon ID infrastructure stack, "anyone can become an issuer (e.g., KYC providers, DAOs, etc.), verifier (e.g., dApps) or holder (i.e., users) of a Web3 identity,".

Polygon ID, with the help of zero-knowledge proofs, lets users prove their identity without the need of exposing their private information. This ensures both the **Freedom of Expression** and **Privacy by Default** (User's identities are secured by zero-knowledge cryptography).

Every identity is identified by a unique identifier called DID (Decentralized Identifier). Every identity-based information is represented via a Verifiable Credentials (VCs). In the simplest terms, a VC represents any type of information related to an individual/enterprise/object. The VC could be as simple as the age of the entity or the highest degree held by it. It could be a membership certificate issued by a DAO, for instance.

The toolset made available by Polygon ID is fully compliant with the W3C standards. We have a definition spec. for the Polygon ID DID method.

The architecture of the framework is composed of three modules: Identity Holder, Issuer, and Verifier. These three, together, form what we call the Triangle of Trust. Let us see what role each entity plays in Polygon ID.

1. **Identity Holder**: An entity that holds claims in its Wallet. A VC, as mentioned above, is issued by an Issuer to the Holder. The Identity Holder generates zero-knowledge proofs of the VCs issued and presents these proofs to the Verifier, which verifies that the proof is authentic and matches specific criteria.
2. **Issuer**: An entity (person, organization, or thing) that issues VCs to the Holders. VCs are cryptographically signed by the Issuer. Every VC comes from an Issuer.
3. **Verifier**: A Verifier verifies the proof presented by a Holder. It requests the Holder to send a proof based on the VCs they hold in their wallet. While verifying a proof, the Verifier performs a set of checks, for example that the VC was signed by the expected Issuer and that the VC matches the criteria requested by the Verifier. The simplest examples of a Verifier is a Bar that wants to verify if you are over 18. In the real world, the Identity Holder would need to provide an ID and show all their personal information. With Polygon ID, they only need to pass a proof.

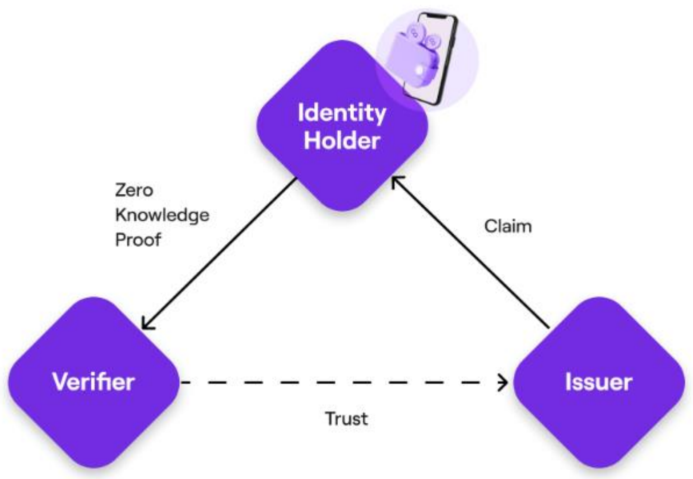


Figure 2 :- Triangle of trust

A core concept here is the trust that must exist between a Verifier and an Issuer: the fact that the information contained inside a VC is cryptographically verifiable doesn't guarantee its truth. The Issuer must be a trusted and reputable party so that Verifier can consume the VCs originated by that Issuer.

2.3 Technologies and Languages

1. **ReactJS :-**

   ReactJS tutorial provides basic and advanced concepts of ReactJS. Currently, ReactJS is one of the most popular JavaScript front-end libraries which has a strong foundation and a large community. ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM. We can use ReactJS on the client and server-side as well as with other frameworks. It uses component and data patterns that improve readability and helps to maintain larger apps

2. **Axios :-**

   Axios is a Javascript library used to make HTTP requests from node.js or XMLHttpRequests from the browser and it supports the Promise API that is native to JS ES6

3. **Reduxjs:-**

   Redux Toolkit simplifies state management in JavaScript applications by creating a centralized store. Components can access the store by dispatching actions and retrieving state data. The store is set up using the configureStore function, which handles middleware, reducers, and configuration options. Actions and reducers are generated using the createSlice function, streamlining the process. Components can access the store's state using the useSelector hook from react-redux, selecting specific data. By leveraging Redux Toolkit, developers can efficiently manage state and enable components to interact with the store, resulting in cleaner and more organized code.

4. **React-Qr-Code :-**

   React-QR-Code is a library that allows developers to easily generate QR codes in React applications. It provides a simple and customizable component that takes in data as a prop and generates a QR code representation of that data. The component can be easily integrated into React projects, allowing for the display of QR codes for various purposes such as sharing links, contact information, or authentication tokens. React-QR-Code offers options to customize the size, color, and error correction level of the generated QR codes. It simplifies the process of generating QR codes in React applications, enabling developers to enhance user experience and facilitate seamless data sharing.

5. **Lottie :-**

   Lottie is an animation rendering library open-sourced by Airbnb. It enables designers to ship animations on any platform (mobile apps, website, etc.) quickly as sending static assets. Lottie animation is animation used in this process and it is a high-quality, small, interactive file that works on any device and can scale up or down without pixelation.

6. **Bootstrap5 :-**

   Bootstrap is a free and open-source collection of CSS and JavaScript/jQuery code used for creating dynamic websites layout and web applications. Bootstrap is one of the most popular front-end frameworks which has really a nice set of predefined CSS codes. Bootstrap uses different types of classes to make responsive websites. Bootstrap 5 was officially released on 16 June 2020 after several months of redefining its features.
   Bootstrap is a framework that is suitable for mobile-friendly web development. it means the code and the template available on bootstrap are applicable to various screen sizes. It is responsive for every screen size.

7. **PolygonID Verifier SDK(iden3js) :-**

   Iden3js is a JavaScript library developed by Iden3, a company focused on decentralized identity solutions. Iden3js provides a set of tools and functionalities for building decentralized applications (dApps) that leverage self-sovereign identity and zero-knowledge proofs.
   With Iden3js, developers can integrate identity management features into their applications, enabling users to have full control over their personal data and identities. It supports the creation, verification, and revocation of decentralized identifiers (DIDs) and allows for secure interactions with blockchain networks.
   Iden3js also includes functionality for working with zero-knowledge proofs (ZKPs), a cryptographic technique that allows proving the validity of a statement without revealing the underlying data. This can be useful for building privacy-preserving applications and conducting verifiable computations.
   The library offers support for various ZKP systems such as zkSNARKs and zkSTARKs, allowing developers to create and verify proofs in a user-friendly manner. It also provides utilities for working with decentralized data storage and decentralized exchanges.
   By utilizing Iden3js, developers can integrate decentralized identity and privacy-enhancing features into their applications, contributing to a more secure and user-centric digital environment.

8. **React-Router-Dom :-**

   React Router Dom is an npm package that enables you to implement dynamic routing in a web app. It allows you to display pages and allow users to navigate them. It is a fully-featured client and server-side routing library for React. React Router Dom is used to build single-page applications i.e. applications that have many pages or components but the page is never refreshed instead the content is dynamically fetched based on the

URL. This process is called Routing and it is made possible with the help of React Router Dom

9. **NodeJS:-**

Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The ability to run JavaScript code on the server is often used to generate dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, as opposed to using different languages for the server- versus client-side programming.

10. **Mongoose:-**

Mongoose is a JavaScript object-oriented programming library that creates a connection between MongoDB and the Node.js JavaScript runtime environment.

11. **Socket.io:-**

Socket.IO is an event-driven library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers. It consists of two components: a client, and a server. Both components have a nearly identical API.

12. **BcryptJS :-**

Bcryptjs is a JavaScript library that provides a simple and straightforward way to hash and compare passwords using the bcrypt algorithm. It is commonly used in web development to securely store and authenticate user passwords. bcryptjs allows developers to hash passwords with a salt, making the resulting hashes more resistant to common cryptographic attacks like rainbow table attacks. Additionally, bcryptjs includes a convenient method for comparing a password input by a user with a stored hashed password, enabling secure password verification. Overall, bcryptjs is a popular choice for password hashing and verification in JavaScript applications.

13. **Cors :-**

CORS (Cross-Origin Resource Sharing) is a mechanism that allows web browsers to make cross-origin requests, which means requesting resources from a different domain than the one the web page originated from. CORS is a security feature implemented in web browsers to protect users from potential cross-site scripting (XSS) and cross-site request forgery (CSRF) attacks.To handle CORS in a JavaScript application, you can use a CORS library or middleware. One popular library for handling CORS in Node.js applications is cors.

14. **Jsonwebtoken :-**

The "jsonwebtoken" library is a popular npm package used for working with JSON Web Tokens (JWT) in JavaScript or Node.js applications. JWTs are a compact and self-contained way to securely transmit information between parties as a JSON object. The "jsonwebtoken" library provides functions for creating, signing, verifying, and decoding JWTs.

15. **Express :-**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

16. **OpenZeppelin :-**

OpenZeppelin is an open-source framework to build secure smart contracts. OpenZeppelin provides a complete suite of security products and audit services to build, manage, and inspect all aspects of software development and operations for decentralized applications.

17. **Ethers :-**

The ethers.js library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem. It is often used to create decentralized applications (dapps), wallets (such as MetaMask and Tally) and other tools and simple scripts that require reading and writing to the blockchain.

18. **Hardhat :-**

Hardhat is an Ethereum development environment for professionals. It facilitates performing frequent tasks, such as running tests, automatically checking code for mistakes or interacting with a smart contract.

19. **Alchemy :-**

I utilized the Alchemy platform to interact with the testnet environment of Sepoliaeth. By leveraging Alchemy's powerful tools and infrastructure, I was able to seamlessly connect to Sepoliaeth's testnet network and perform various operations such as deploying smart contracts, sending transactions, and querying blockchain data. Alchemy's user-friendly interface and robust API capabilities enabled me to effectively test and interact with Sepoliaeth's decentralized ecosystem, ensuring a smooth development and testing experience.

Languages Used:-

1. **Solidity :-**

Solidity is a statically-typed curly-braces programming language designed for developing smart contracts that run on Ethereum

2. **Zero Query Language :-**

   The ZK Query Language is used to provide a simple way for developers to design customised authentication requirements based on someone's credentials.

3. **JavaScript:-**

   JavaScript is a dynamic programming language that brings interactivity and functionality to web pages. It allows developers to add behaviors, handle events, manipulate the content of the page, make network requests, perform calculations, and much more. JavaScript plays a crucial role in creating interactive web applications, enabling user interaction and enhancing the overall user experience.

4. **Html:-**

   HTML (Hypertext Markup Language) provides the structure and content of a web page. It uses tags to define elements such as headings, paragraphs, images, links, forms, and more. HTML acts as the backbone of a webpage, organizing and presenting information to the user.

5. **Css:-**

   CSS (Cascading Style Sheets) is used to control the presentation and layout of HTML elements. It defines how the elements should be displayed, specifying properties like color, size, font, positioning, and responsiveness. CSS enables developers to create visually appealing and consistent designs across different devices and screen sizes.

CHAPTER 3: - System Architecture and Design

3.1 High-Level System Architecture

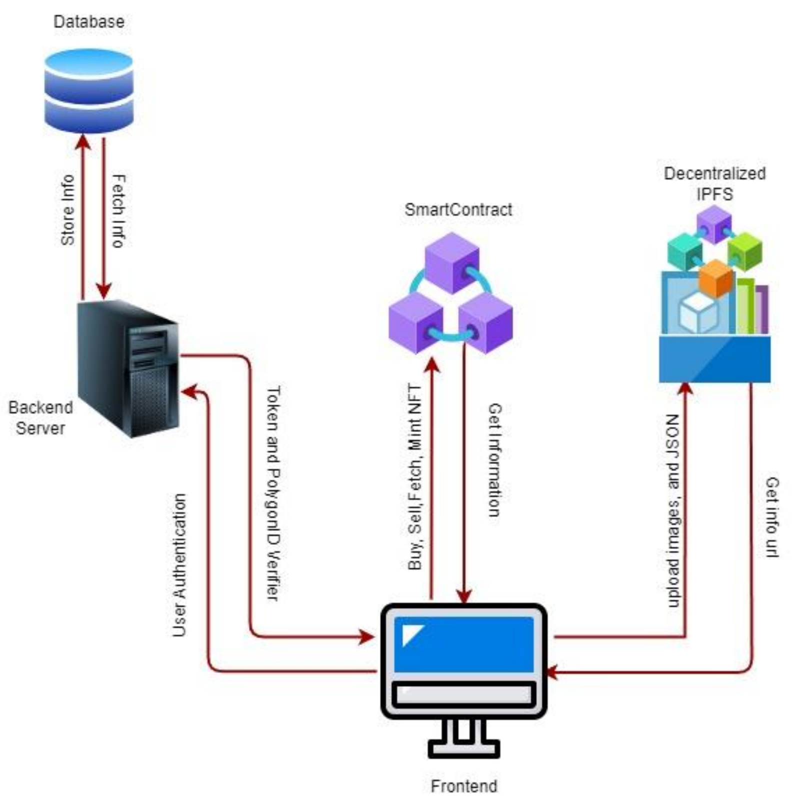


Figure 3:- High Level System Architecture

At a high level, the system architecture of a full stack website includes a frontend, a backend, a smart contract for NFTs, IPFS for storing images and metadata, and a database for user information.

Frontend:- The Frontend is the user-facing part of the website that users interact with. It consists of HTML, CSS, and JavaScript and is responsible for rendering the user interface. Users can view NFTs and interact with various features provided by the website.

Backend:- It serves as the intermediary between the frontend and other components of the system. It handles requests from the frontend and processes them accordingly. In our system, the backend is also responsible for generating the Verifier metadata(Using Polygon ID) for authenticating the user, and then signing them in.

SmartContract:- The smart contract serves as a mechanism to fetch and mint NFTs. It retrieves NFT information from its storage and enables the creation of new NFTs by minting them, thus allowing users to access and interact with unique digital assets in a decentralized manner.

IPFS:- For storing images and metadata related to NFTs, the system utilizes IPFS (InterPlanetary File System). IPFS provides a decentralized and distributed file storage system. When a user uploads an image and its associated metadata, the backend stores them on IPFS, which returns a unique URL representing the content's location in the IPFS network. This URL is then stored in the smart contract, linking the NFT to its corresponding image and metadata.

DataBase:- Additionally, the backend is connected to a database where it stores user information. This database contains user profiles, authentication details, transaction history, and other relevant data. The backend communicates with the database to store and retrieve user information as needed.

Overall, this high-level system architecture demonstrates the flow of data and interactions between the frontend, backend, smart contract, IPFS, and database. It enables users to interact with NFTs through the website's frontend, while the backend handles the necessary communication and data storage operations with the smart contract, IPFS, and database.

3.2 Frontend Design and Development

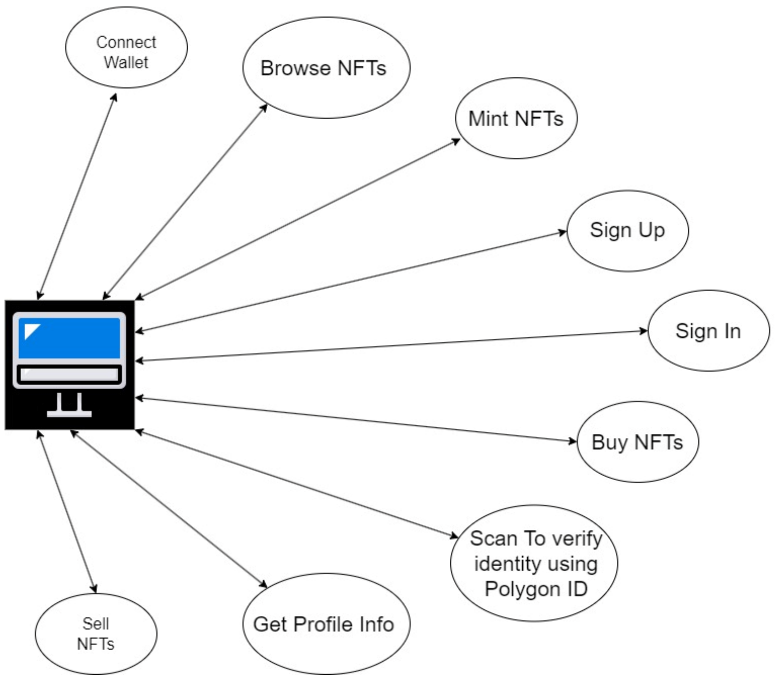


Figure 4:- Frontend Functionalities

Frontend design and development involves creating the user-facing components of a website or application. It encompasses the visual and interactive elements that users see and interact with. Frontend design focuses on creating an intuitive and aesthetically pleasing user interface (UI) using HTML, CSS, and JavaScript. Designers use their creativity and understanding of user experience (UX) principles to craft engaging layouts, typography, colors, and graphics. Frontend developers then implement the design, translating it into functioning code that brings the UI to life.

The functionalities of the frontend for NFT Marketplace includes: -

1. **Connect Wallet**: -

   The user must be able to connect his/her crypto wallet to the website, for the purpose of executing transactions involving smart contracts.

2. **Browse NFTs: -**

   The website should provide its user a showcase of all the currently listed NFTs for sale, the user must be able to view details like name, description, price, owner of the NFT and can then thus make an informed decision regarding the purchase.

3. **Buy NFTs: -**

   The user on deciding to purchase an NFT must be able to buy the NFT from the website.

4. **Sell NFTs**: -

   If a user wants to Sell his/her NFTs the website must allow the user to do so.

5. **Get Profile Info**: -

   The website must display the user information like Name, email, image, no. of NFTs owned, total NFTs amount, wallet address and the owned nft details.

6. **Scan To verify/identity using Polygon ID**: -

   The website before sign in/signup must ask user to verify using PolygonID his/her credentials by scanning a QR code from the webpage using an Self sovereign identity wallet like PolygonID wallet which has the required credentials.

7. **Sign In**: -

   The user must be able to sign in to the website by providing an email and password.

8. **Sign Up**: -

   The user must be able to sign up by providing his/her details like Name, email, password.

9. **Mint NFTs**: -

   The user should be able to create/mint a new NFT through the smart contract by providing details like image for the NFT, name, description and price of the NFT.

3.3 Backend Design and Development

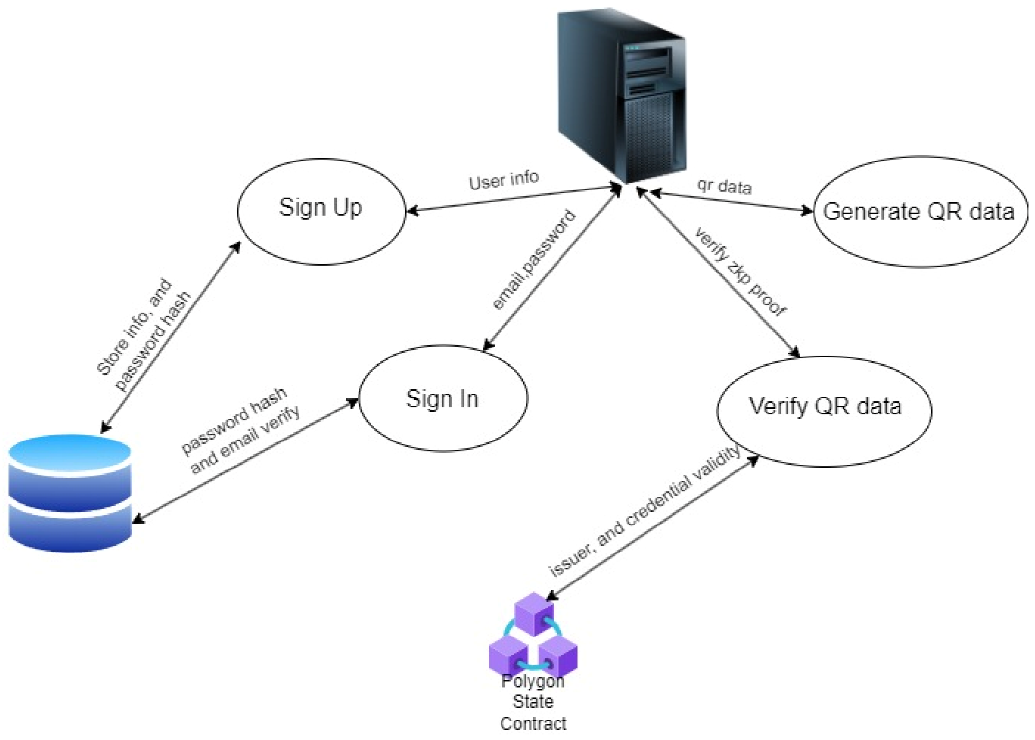


Figure 5 Backend Design

Backend design and development involves building the server-side components of a website or application. It focuses on handling business logic, data storage, and server-side operations. The design and implement of APIs enable communication between the frontend and backend, Also it involves working with databases to store and retrieve data efficiently, ensuring data integrity and security. Also involves, optimizing the performance, handle authentication and authorization, and implement server-side validation and error handling.

The functionalities of the Backend for NFT Marketplace includes: -

1. **Sign In** :-

   The server receives an API post request involving email and password of the user, it then verifies the password from the hash stored in the database, if verified it responds with an json web token which can be used by the user for various authenticated tasks. If not verified, it responds with email or password seems to be wrong.

2. **Sign Up**: -

   The server receives an API post request containing various user information like name, email, password, etc. and then it checks whether the email is unique. If unique, it stores the information on the database and responds with status 200 to the request. If account created successfully, the user must be able to sign in using the credentials now.

3. **Generate QR data**: -

   On receiving request to generate QR data the server uses the sessionId mentioned in the request and zero query language format specified in the function to generate a response containing information for the QR data that needs to be sent to the user.

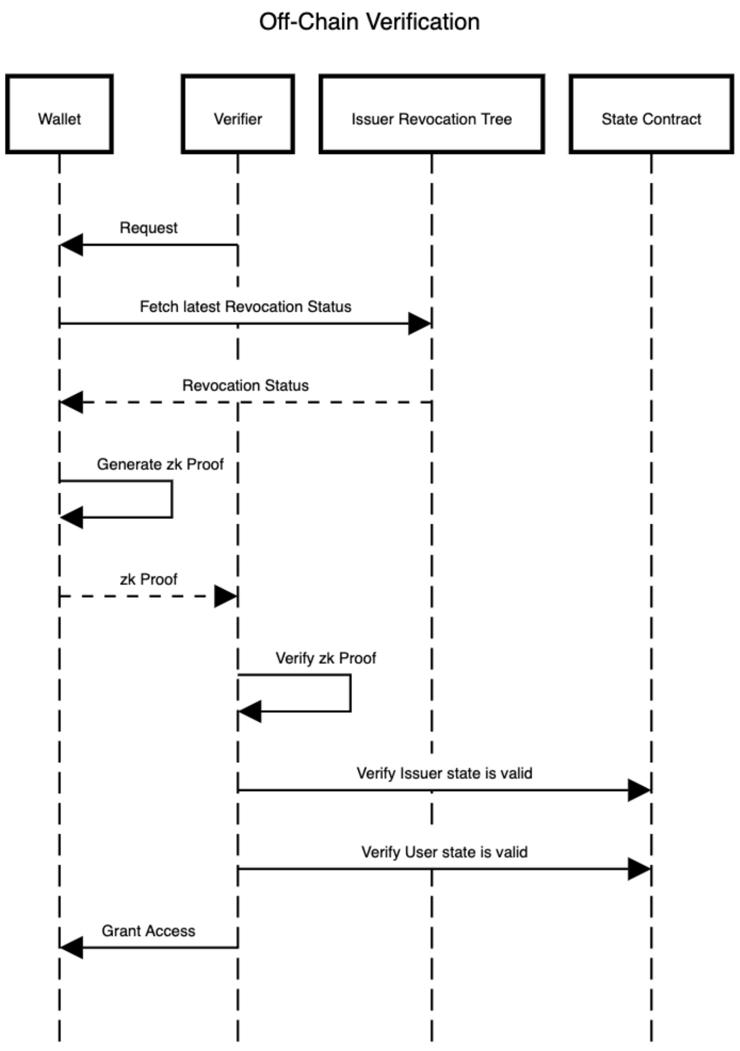


Figure 6 Flow for verifying proof

4. **Verify QR data**: -

   On receiving a request by the wallet to verify the claim of a credential it verifies the zero Knowledge proof received , it also checks the state of the polygon smart contract to check whether the issuer credentials and the credentials provided by the user are not revoked, using the iden3js library.

3.4 SmartContract Design and Development

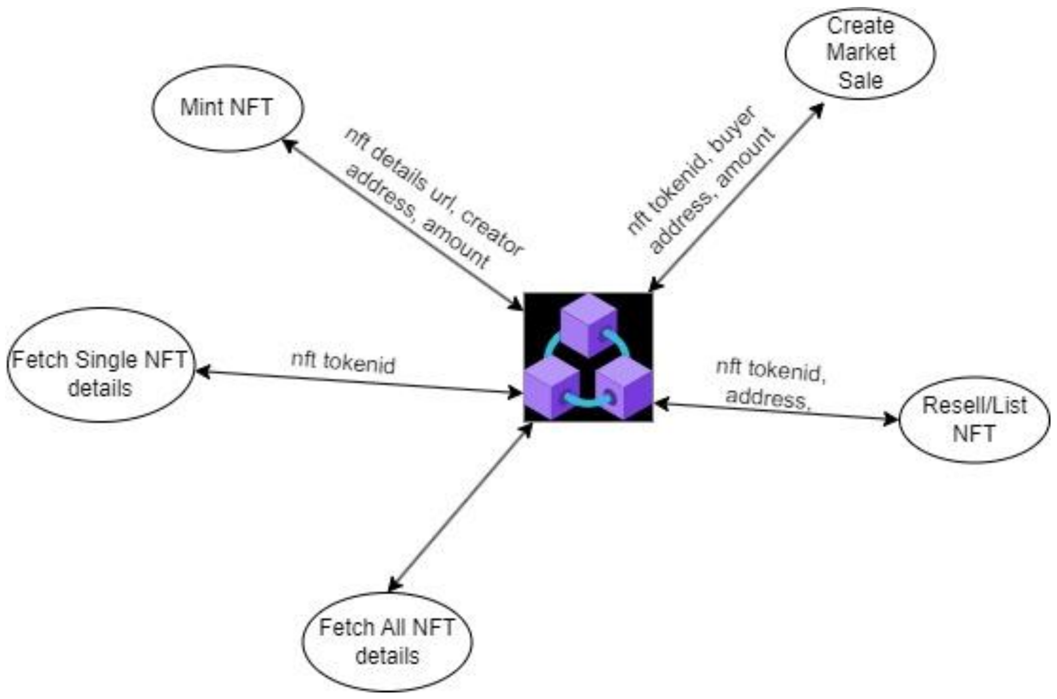


Figure 7 SmartContract Functionalities

Smart contract design and development involves creating self-executing contracts that run on a blockchain network. It requires careful planning and consideration of the contract's purpose and functionality. Designing a smart contract involves defining the contract's data structures, functions, and events, as well as specifying the contract's interactions with external systems and users. Solidity, a programming language specifically designed for smart contracts, is commonly used for development.

The functions of the SmartContract include:-

1. **NFT Minting**:

   The process of minting an NFT involves creating a unique digital asset on a blockchain network. This is done by interacting with a dedicated smart contract designed for NFTs. During the minting process metadata, and ownership details are defined. Once minted, the NFT receives a distinct identifier(tokenId) and becomes part of the blockchain's immutable ledger, verifying its authenticity and ownership.

2. **Single NFT Details Retrieval**:

   To retrieve details of a specific NFT, one can query the associated smart contract using its unique identifier. By sending a request to the blockchain smart contract with the tokenId, the smart contract responds with the relevant information, including metadata, owner. This enables users or applications to access and display specific information about an NFT, facilitating various functionalities such as showcasing artwork, description, or historical data.

3. **All NFT Details Retrieval**:

   Fetching all NFT details involves retrieving information about multiple NFTs. By interacting with the smart contract, it is possible to query for all the NFTs registered under for sale. This provides a comprehensive overview of the available NFTs, including their individual attributes and metadata. Such functionality enables applications, galleries, or marketplaces to showcase or search through the complete collection of NFTs.

4. **NFT Reselling/Listing**:

   Reselling or listing an NFT involves making it available for sale or auction on a marketplace or platform. This process necessitates connection to a marketplace smart contract and providing relevant information, such as the NFT's tokenId, price. By listing the NFT, the owner signals their intention to transfer ownership to a potential buyer. This enables other users to browse, search, and make purchase offers, facilitating secondary market transactions.

5. **Market Sale Creation**:

   Creating a market sale involves initiating a transaction to transfer ownership of an NFT from the seller to the buyer. This requires interaction with the marketplace's smart contract and providing details such as the buyer's address, the NFT's tokenId. The smart contract validates the transaction and facilitates the ownership transfer by updating the NFT's ownership records. This secure and transparent process ensures a smooth exchange of NFTs within the marketplace environment.

CHAPTER 4: - IMPLEMENTATION

4.1 Frontend Website and Polygon ID Authentication

Home Page: - The website loads and prompts the user to connect his/her metamask wallet to the website. After connecting it shows user the list of NFTs currently for sale, the ipfs url is fetched from the smart contract using which we get other details including the NFTs name, price and image which is fetched from the Interplanetary File Storage System Pinata.

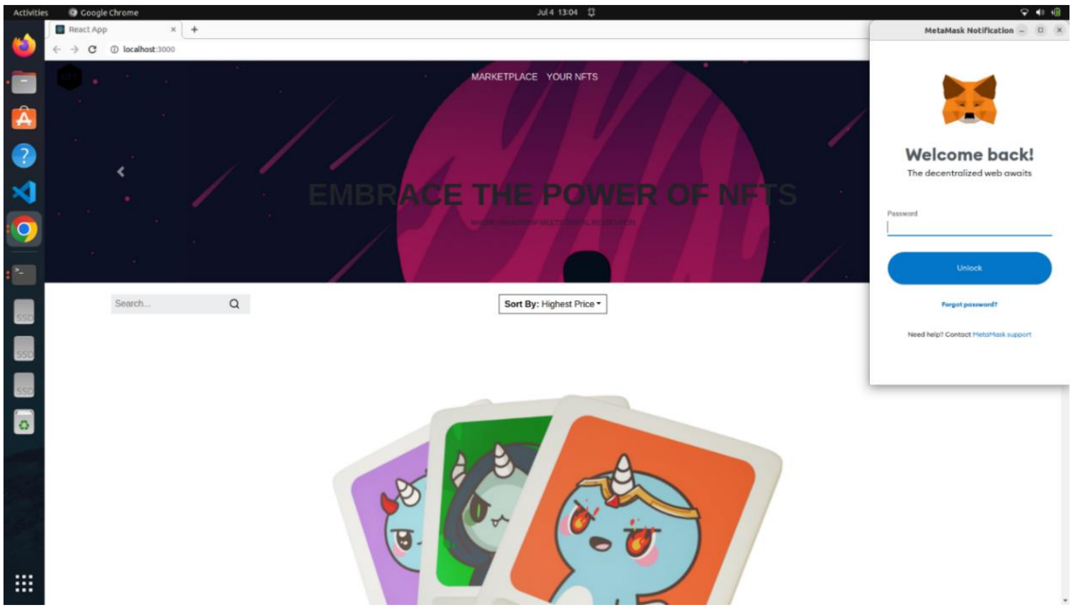


Figure 8 Prompting User to connect Wallet

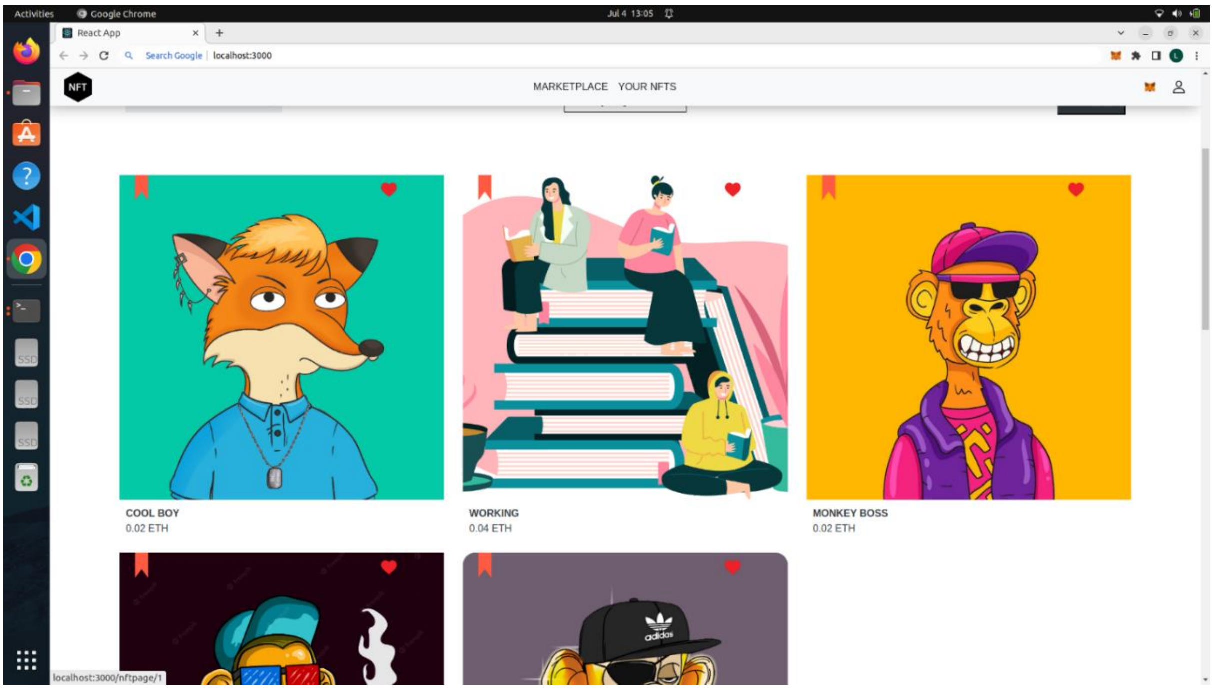
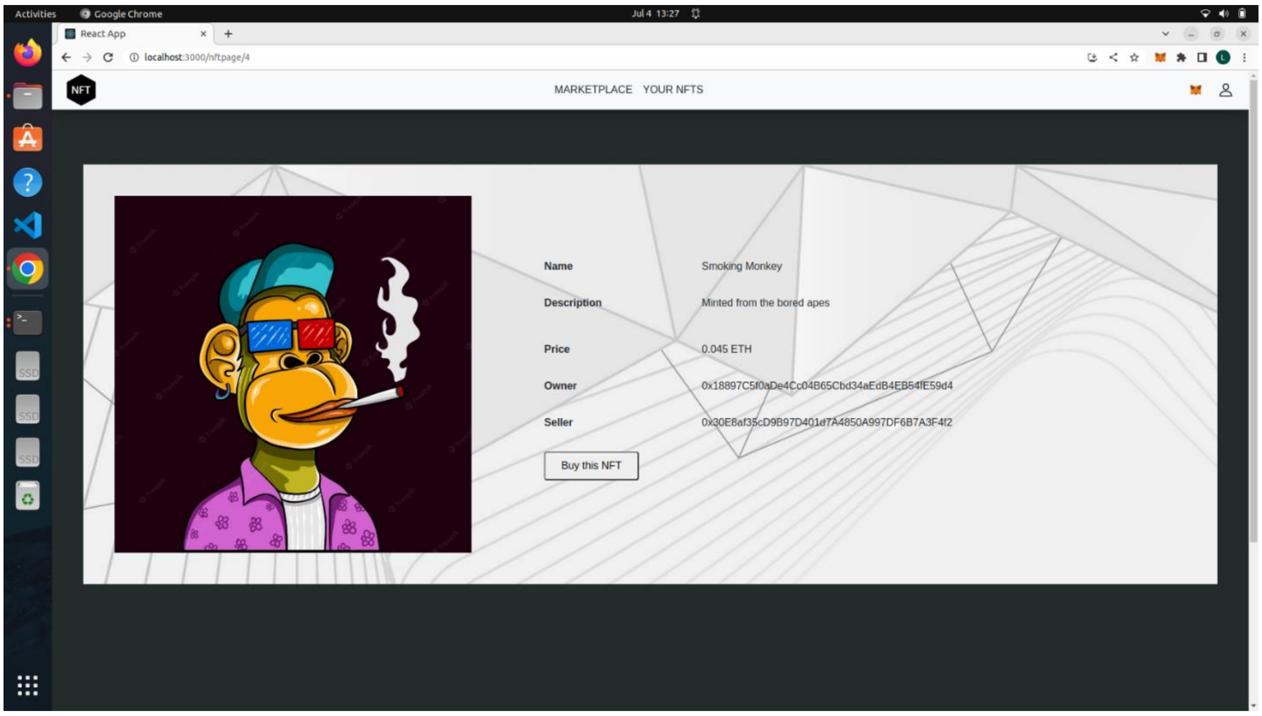


Figure 9:- Showing NFTs for Sale

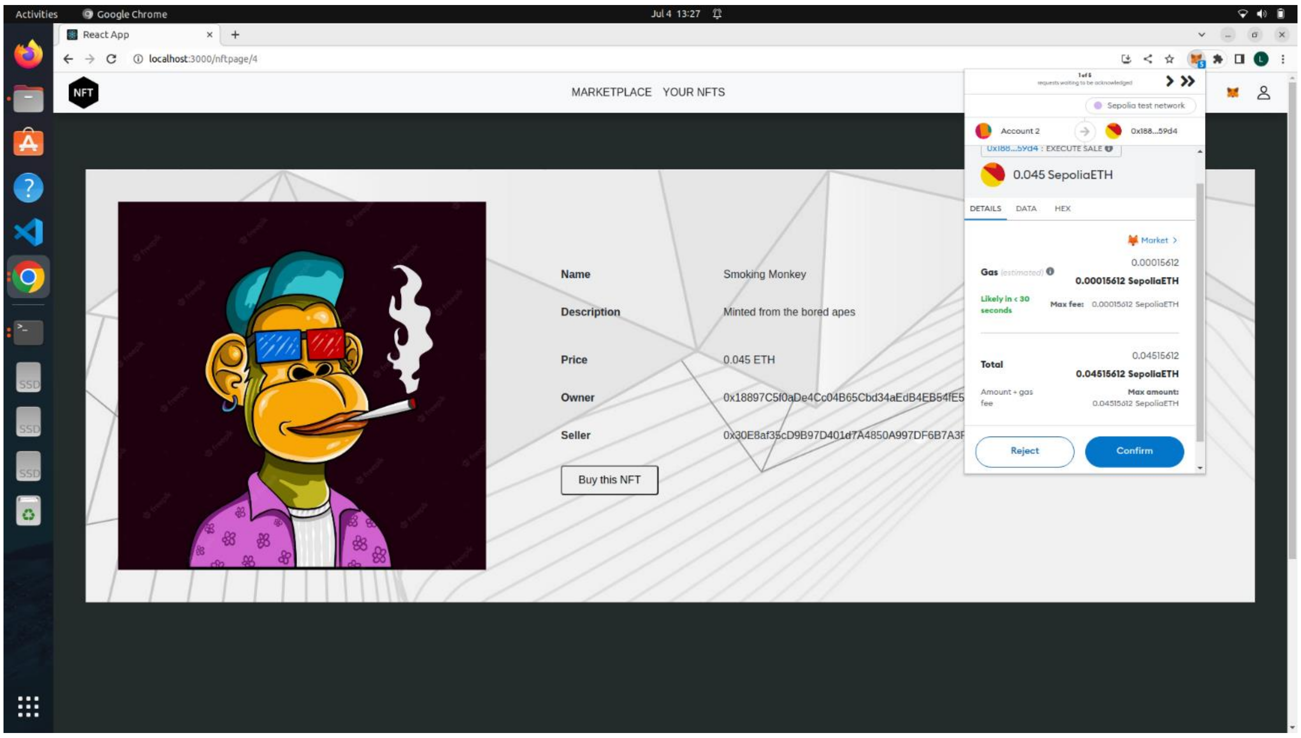
Viewing NFT details :-

When the user clicks on any NFT it redirects them to nft details page which shows them information regarding the nft like name,price,owner,seller and also a button which enables the user to buy the given NFT.



Name	Smoking Monkey
Description	Minted from the bored apes
Price	0.045 ETH
Owner	0x18897C5f0aDe4Cc04B65Cbd34aEdB4EB54fE59d4
Seller	0x30E8af35cD9B97D401d7A4850A997DF6B7A3F4f2

Figure 10 Viewing Single NFT



Name	Smoking Monkey
Description	Minted from the bored apes
Price	0.045 ETH
Owner	0x18897C5f0aDe4Cc04B65Cbd34aEdB4EB54fE5
Seller	0x30E8af35cD9B97D401d7A4850A997DF6B7A3F

0.045 SepoliaETH

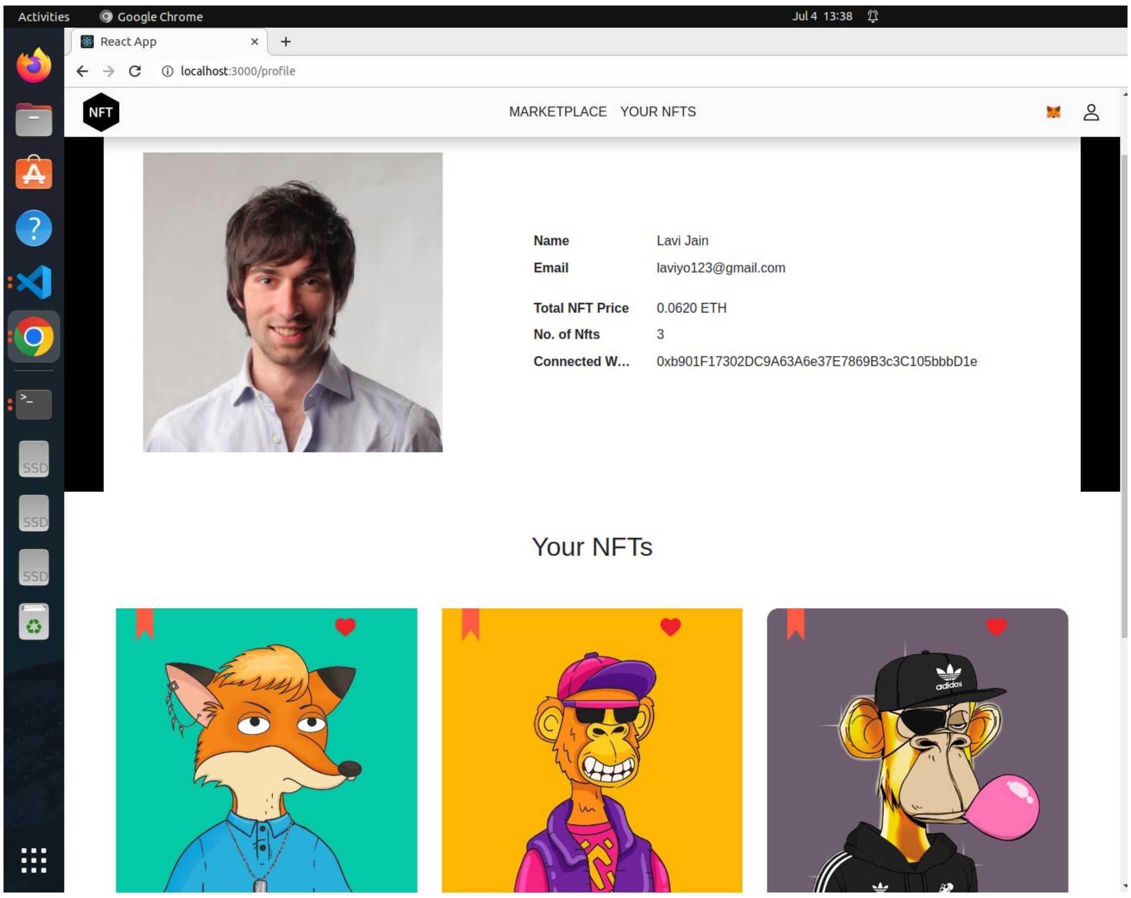
Gas (estimated) 0.00015612 **0.00015612 SepoliaETH**
Likely in < 30 seconds **Max fee:** 0.00015612 SepoliaETH

Total 0.04515612 **0.04515612 SepoliaETH**
Amount + gas fee **Max amount:** 0.04515612 SepoliaETH

Figure 11 User Buying the NFT

Profile Page: -

After Sign in, it shows user's information like name, email and image also shows some stats like no. of NFTs, total NFT price, connected wallet address.



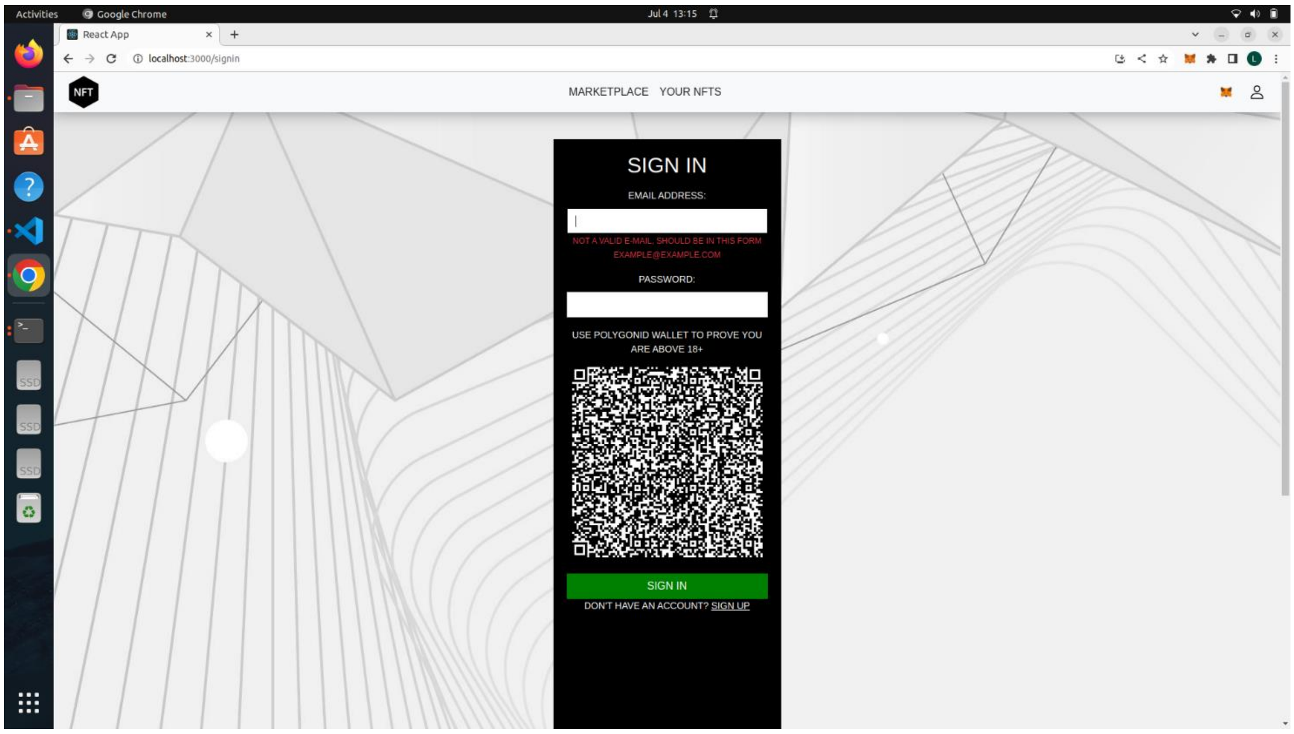
Name	Lavi Jain
Email	laviyo123@gmail.com
Total NFT Price	0.0620 ETH
No. of Nfts	3
Connected W...	0xb901F17302DC9A63A6e37E7869B3c3C105bbbD1e

Your NFTs

Figure 12 User Profile Page

Sign In: -

The sign in page allows user to type in there email and password to login to the website. The textboxes has checks to ensure that the typed email address is a valid email id, It also shows a qr code which allows user to use his/her polygon ID wallet to verify a credential issued by some trusted issuer. This is done to ensure protection against spams and bots using the website. The sign in buttons sends an axios post request to the backend and receives an json web token which it stores in the local storage for further using the authenticated features.



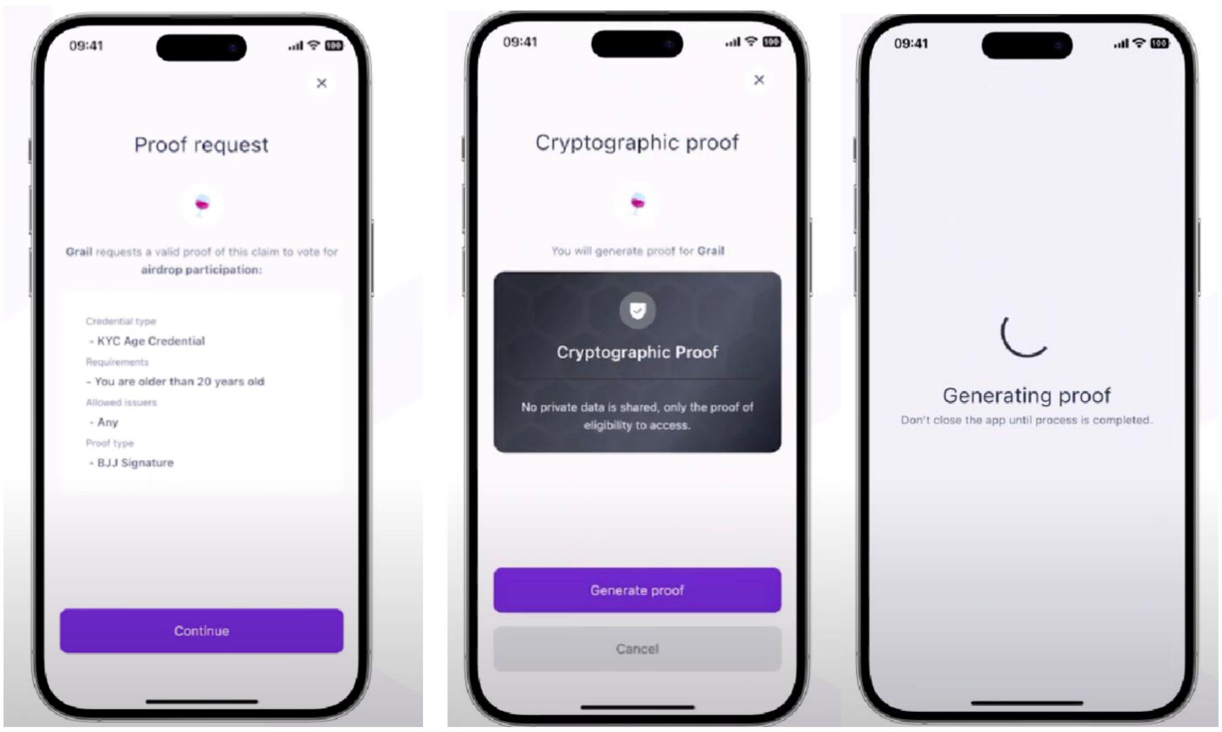
NOT A VALID E-MAIL. SHOULD BE IN THIS FORM EXAMPLE@EXAMPLE.COM

USE POLYGONID WALLET TO PROVE YOU ARE ABOVE 18+

Figure 13 User Sign In

Polygon Id Wallet :-

We use the polygon ID wallet to show that we own a verified credential. On scanning the Qr code in the sign in page the wallet generates a cryptographic proof using zero knowledge methods to show that we are above 18. It then calls the callback function in the backend server to verify.



Grail requests a valid proof of this claim to vote for **airdrop participation:**

Credential type
- KYC Age Credential

Requirements
- You are older than 20 years old

Allowed issuers
- Any

Proof type
- BJJ Signature

You will generate proof for **Grail**

Cryptographic Proof

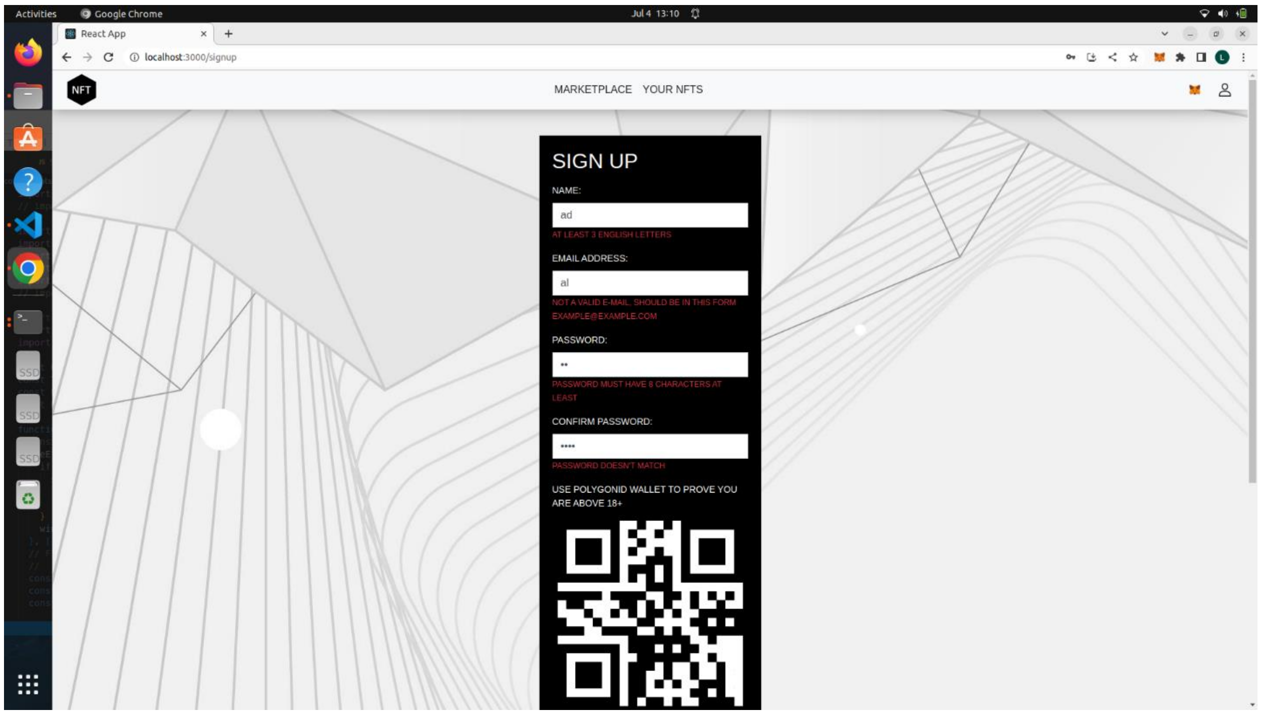
No private data is shared, only the proof of eligibility to access.

Don't close the app until process is completed.

Figure 14 Generating Proof for Age

Sign Up: -

The sign up page allows user to type in there name , email address, password, and verify credential using the polygon ID wallet to verify if the credential is issued by some trusted issuer. This is done to ensure protection against spams and bots using the website.



SIGN UP

NAME: ad

AT LEAST 3 ENGLISH LETTERS

EMAIL ADDRESS: al

NOT A VALID E-MAIL, SHOULD BE IN THIS FORM EXAMPLE@EXAMPLE.COM

PASSWORD: ••

PASSWORD MUST HAVE 8 CHARACTERS AT LEAST

CONFIRM PASSWORD: ••••

PASSWORD DOESN'T MATCH

USE POLYGONID WALLET TO PROVE YOU ARE ABOVE 18+

Figure 15 User Sign Up

Creating an NFT: -

This page allows user to create an NFT, it requires user to provide NFT a name, description, image and a price in ETH, The website stores the image and metadata in the IPFS Pinata which returns a url which is provided to the smart contract for the generation of the NFT.

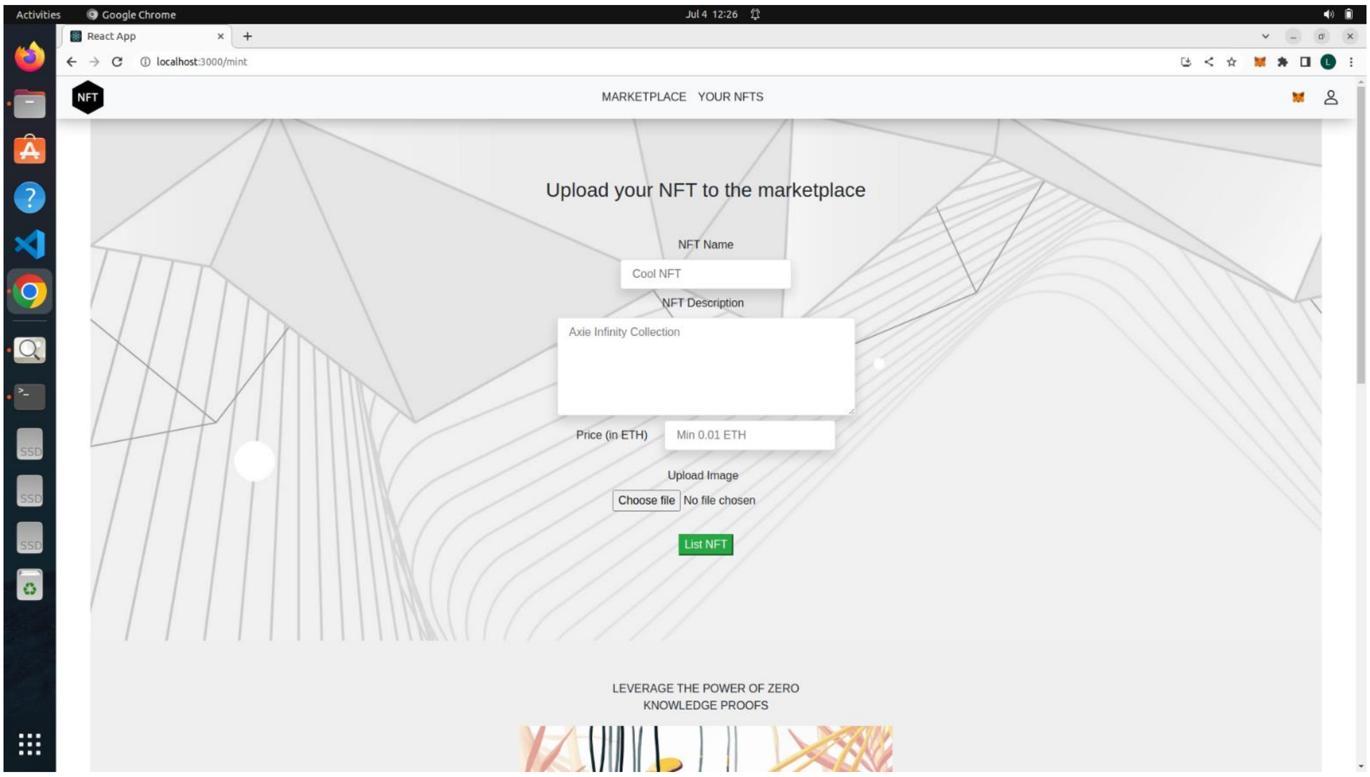


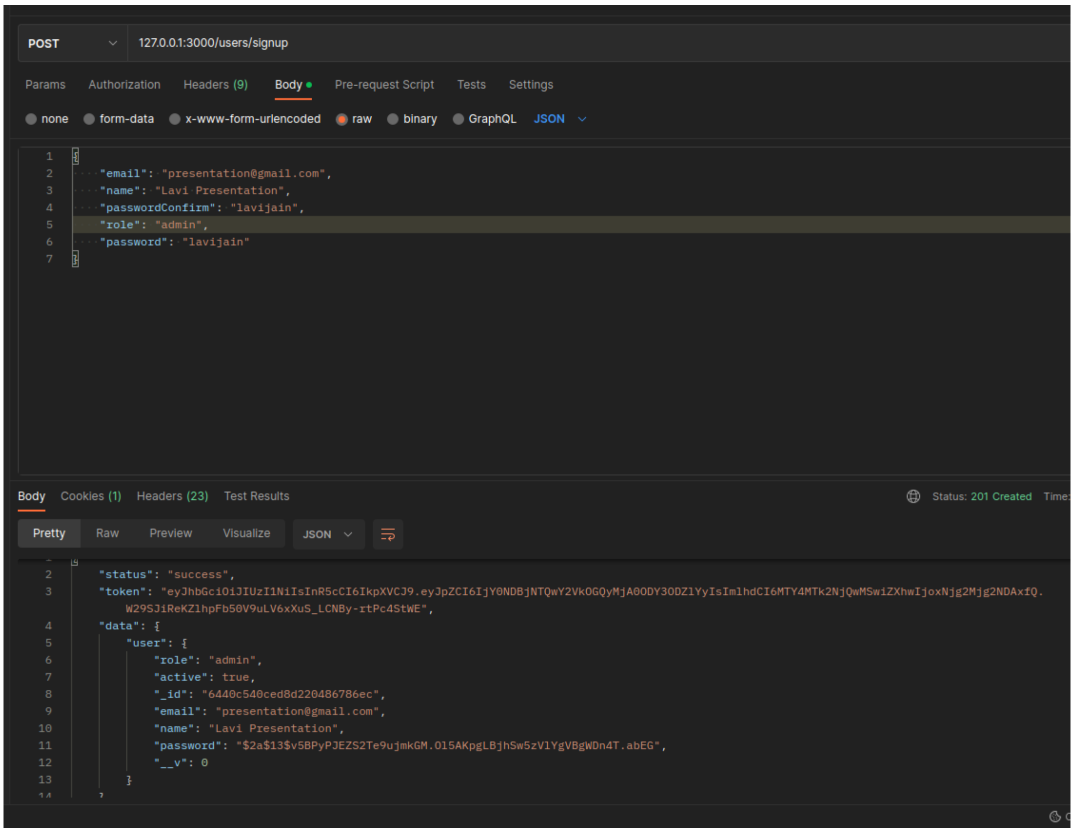
Figure 16 Minting an NFT

4.2 Backend API and Database

The backend API has the following end points:-

Api endpoint	**Request type**	**Functionality**
/users/signup	POST	It receives user details such as name, email, password and stores them in the database
/users/signin	POST	It recieves email and password and verifies the credentials from the database password hash and returns with the json web token which can be used by user during the current session.
/verify/polygonid	GET	It generates information using the zero query language and iden3 library to generate data which the user uses to generate Qr code in order to verify credentials.
/verify/callback	POST	It is called by the polygon id wallet when verifying the proof. It verifies the proof and calls the polygon state contract to check whether the specified credentials are valid and not revoked and also check if the issuer of the credential is valid.
/users/:id	GET	It returns the user details of the id specified in the parameter.

We used Mongo DB as the database for this project. This Database has only a single Table of users which stores user details like name, email, password hash.


POST 127.0.0.1:3000/users/signup

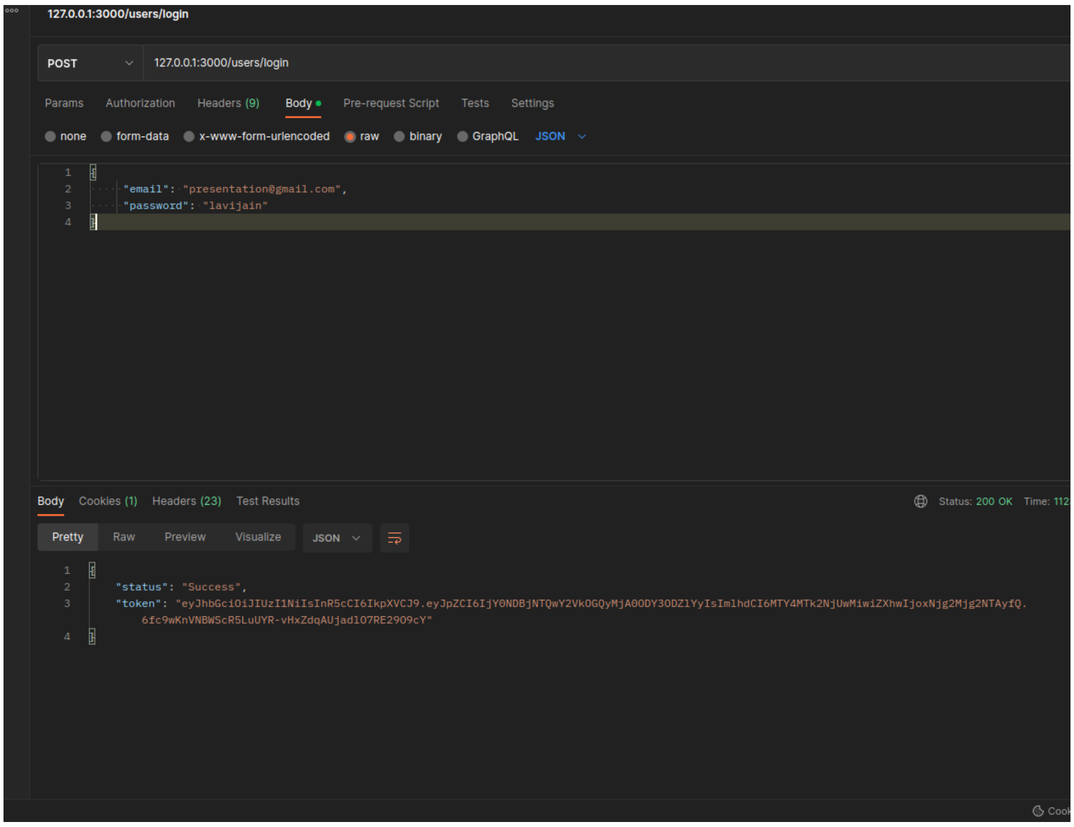
Body: raw JSON

```
{
    "email": "presentation@gmail.com",
    "name": "Lavi Presentation",
    "passwordConfirm": "lavijain",
    "role": "admin",
    "password": "lavijain"
}
```

Status: 201 Created

```
{
    "status": "success",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0NDBjNTQwY2VkOGQyMjA0ODY3ODZlYyIsImlhdCI6MTY4MTk2NjQwMSwiZXhwIjoxNjg2Mjg2NDAxfQ.
        W29SJiReKZlhpFb50V9uLV6xXuS_LCNBy-rtPc4StWE",
    "data": {
        "user": {
            "role": "admin",
            "active": true,
            "_id": "6440c540ced8d220486786ec",
            "email": "presentation@gmail.com",
            "name": "Lavi Presentation",
            "password": "$2a$13$v5BPyPJEZS2Te9ujmkGM.Ol5AKpgLBjhSw5zVlYgVBgWDn4T.abEG",
            "__v": 0
        }
    }
```

Figure 17 Backend Signup


127.0.0.1:3000/users/login

POST 127.0.0.1:3000/users/login

Body: raw JSON

```
{
    "email": "presentation@gmail.com",
    "password": "lavijain"
}
```

Status: 200 OK

```
{
    "status": "Success",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY0NDBjNTQwY2VkOGQyMjA0ODY3ODZlYyIsImlhdCI6MTY4MTk2NjUwMiwiZXhwIjoxNjg2Mjg2NTAyfQ.
        6fc9wKnVNBWScR5LuUYR-vHxZdqAUjadl07RE2909cY"
}
```

Figure 18 Backend Login

4.3 Smart Contract Using Solidity

We use the ethers library to connect with the smart contract deployed on SepoliaETH testnet. After connecting the metamask wallet to the website we call the following functions from the smartcontract:

1. **CreateNFT**:

   The process of minting an NFT involves creating a unique digital asset on a blockchain network. This is done by interacting with a dedicated smart contract designed for NFTs. During the minting process metadata, and ownership details are defined. Once minted, the NFT receives a distinct identifier(tokenId) and becomes part of the blockchain's immutable ledger, verifying its authenticity and ownership.

2. **Single NFT Details Retrieval**:

   To retrieve details of a specific NFT, one can query the associated smart contract using its unique identifier. By sending a request to the blockchain smart contract with the tokenId, the smart contract responds with the relevant information, including metadata, owner. This enables users or applications to access and display specific information about an NFT, facilitating various functionalities such as showcasing artwork, description, or historical data.

3. **All NFT Details Retrieval**:

   Fetching all NFT details involves retrieving information about multiple NFTs. By interacting with the smart contract, it is possible to query for all the NFTs registered under for sale. This provides a comprehensive overview of the available NFTs, including their individual attributes and metadata. Such functionality enables applications, galleries, or marketplaces to showcase or search through the complete collection of NFTs.

4. **NFT Reselling/Listing**:

   Reselling or listing an NFT involves making it available for sale or auction on a marketplace or platform. This process necessitates connection to a marketplace smart contract and providing relevant information, such as the NFT's tokenId, price. By listing the NFT, the owner signals their intention to transfer ownership to a potential buyer. This enables other users to browse, search, and make purchase offers, facilitating secondary market transactions.

5. **Market Sale Creation**:

   Creating a market sale involves initiating a transaction to transfer ownership of an NFT from the seller to the buyer. This requires interaction with the marketplace's smart contract and providing details such as the buyer's address, the NFT's tokenId. The smart contract validates the transaction and facilitates the ownership transfer by updating the NFT's ownership records. This secure and transparent process ensures a smooth exchange of NFTs within the marketplace environment.

4.4 IPFS Storage of Images and JSON

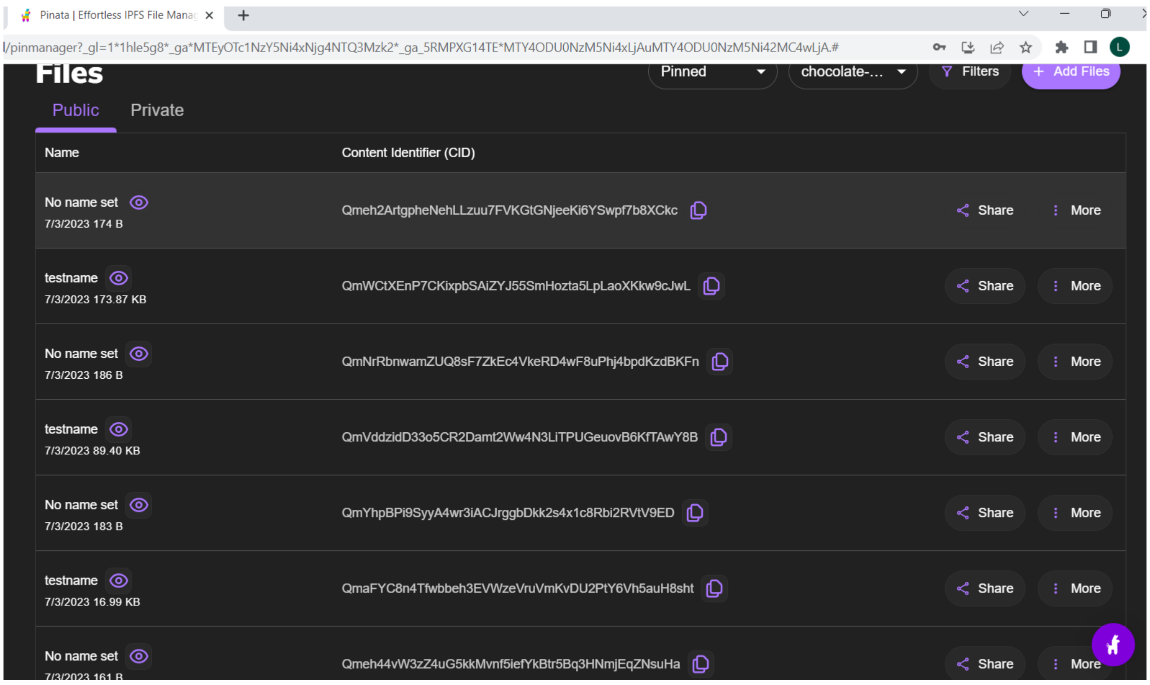
We have 2 functions here: -

1. **UploadImageToIPFS**: -

   This function is called when the user uploads an image in the upload NFT page. It stores the image on the IPFS and returns an CID (hash) which is used to reference the uploaded image. Using the cid we create an URL like https://ipfs.io/ipfs/CID and then call UploadJSONtoIPFS which takes this URL as parameter.

2. **UploadJSONtoIPFS**: -

   This function takes the ipfs URL as a parameter and is called when the user clicks on mint NFT in the upload NFT page. It creates a json object having NFT name, price, description, and ipfs URL and then uploads this object to the IPFS. The url returned along with the CID is used and is what is stored in the smart contract to make the NFT tamper proof.


Pinata | Effortless IPFS File Manag
l/pinmanager?_gl=1*1hle5g8*_ga*MTEyOTc1NzY5Ni4xNjg4NTQ3Mzk2*_ga_5RMPXG14TE*MTY4ODU0NzM5Ni4xLjAuMTY4ODU0NzM5Ni42MC4wLjA.#
Files
Pinned
chocolate-...
Filters
+ Add Files
Public
Private

Name	Content Identifier (CID)		
No name set 7/3/2023 174 B	Qmeh2ArtgpheNehLLzuu7FVKGtGNjeeKi6YSwpf7b8XCkc	Share	More
testname 7/3/2023 173.87 KB	QmWCtXEnP7CKixpbSAiZYJ55SmHozta5LpLaoXKkw9cJwL	Share	More
No name set 7/3/2023 186 B	QmNrRbnwamZUQ8sF7ZkEc4VkeRD4wF8uPhj4bpdKzdBKFn	Share	More
testname 7/3/2023 89.40 KB	QmVddzidD33o5CR2Damt2Ww4N3LiTPUGeuovB6KfTAwY8B	Share	More
No name set 7/3/2023 183 B	QmYhpBPi9SyyA4wr3iACJrggbDkk2s4x1c8Rbi2RVtV9ED	Share	More
testname 7/3/2023 16.99 KB	QmaFYC8n4Tfwbbeh3EVWzeVruVmKvDU2PtY6Vh5auH8sht	Share	More
No name set 7/3/2023 161 B	Qmeh44vW3zZ4uG5kkMvnf5iefYkBtr5Bq3HNmjEqZNsuHa	Share	More

Figure 19:- Uploaded data to IPFS

CHAPTER 5: - Results and Analysis

5.1 RESULTS

This project aimed to develop a full stack marketplace using modern web technologies. The marketplace provides a platform for users to buy and sell various nfts securely and efficiently. The development stack consisted of React.js for the frontend, Node.js with Express.js for the backend, IPFS for file storage, and Solidity smart contracts with OpenZeppelin for blockchain integration.

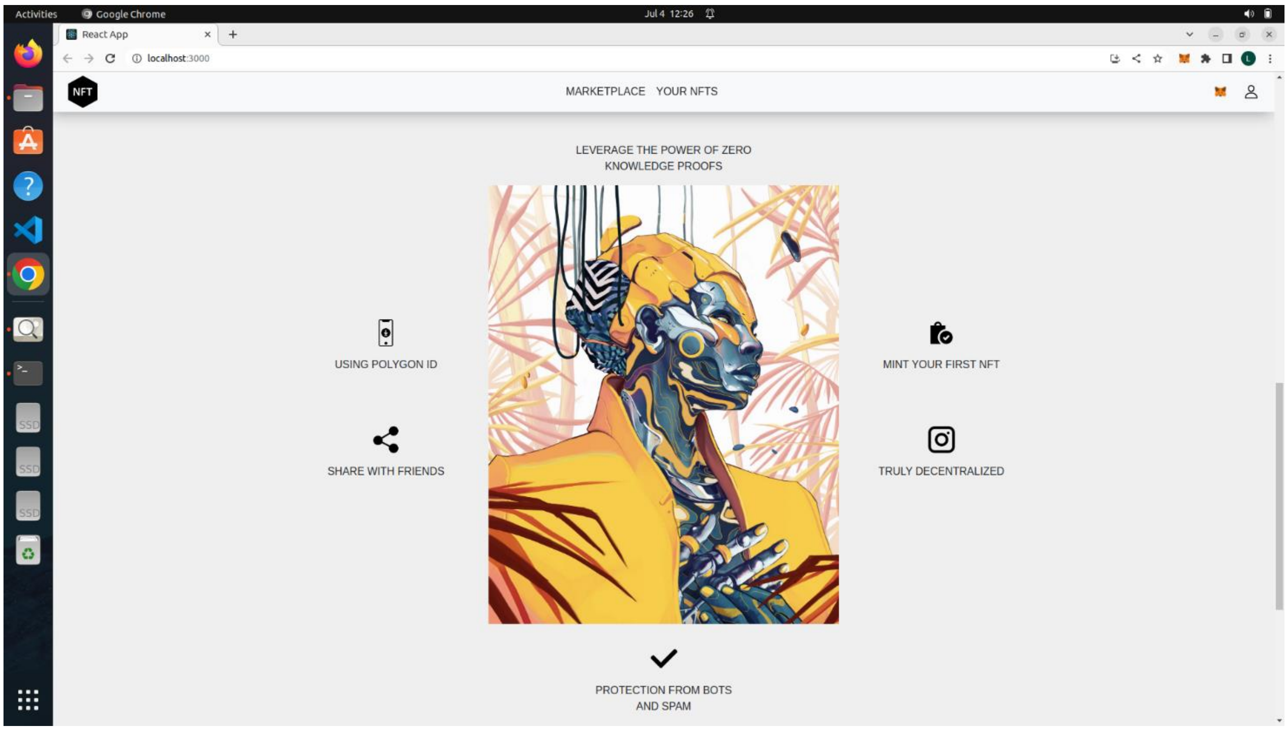


Figure 20 Running Website

Frontend Development with React.js:

The frontend of the marketplace was built using React.js, a popular JavaScript library for building user interfaces. React's component-based architecture facilitated the creation of reusable UI elements, resulting in a responsive and dynamic marketplace. The frontend components were designed to provide an intuitive user experience with features such as nfts listings, search functionality, user authentication.

Backend Development with Node.js and Express.js:

The backend of the marketplace was developed using Node.js, a powerful JavaScript runtime, and Express.js, a flexible web application framework. These technologies provided a robust and scalable foundation for handling various backend functionalities, including polygonId verifiable credential for Age verification. Express.js middleware was utilized for routing, request handling, and implementing authentication and authorization mechanisms.

File Storage with IPFS: In order to enable efficient and decentralized file storage, the InterPlanetary File System (IPFS) was integrated into the marketplace. IPFS allowed users to upload and retrieve files associated with their products, ensuring a reliable and distributed storage solution. By leveraging IPFS, the marketplace achieved greater resilience, data availability, and reduced reliance on a centralized server for file hosting.

Blockchain Integration with Solidity Smart Contracts and OpenZeppelin: To enhance trust and transparency in the marketplace, smart contracts written in Solidity were utilized. These contracts defined the business logic governing nft listings, transactions, and user interactions. OpenZeppelin, a widely-used library for secure smart contract development, was employed to ensure best practices and prevent vulnerabilities. The integration of smart contracts enabled secure and tamper-proof transactions, escrow services, and dispute resolution mechanisms.

5.2 Comparison With Existing NFT Marketplaces

As the popularity of NFTs has grown, the NFT market has climbed to more than INR 3 trillion in value, according to The 2021 NFT Market Report released by blockchain data company Chain analysis. Some of the most popular and well-rated NFT marketplaces where you can buy and sell these digital assets are OpenZeppelin , WazirX. I have implement the same features for buying, selling, and creating an NFT but there is one major problem with the existing NFT marketplace i.e they cant prevent one of the most common problem of web2 and also now web3 Spams and bots. The problems are: -

Scalping and Price Manipulation: Bots can be programmed to rapidly purchase popular NFTs during a sale or auction, creating artificial scarcity and driving up prices. This can lead to frustration among genuine collectors who miss out on acquiring desirable NFTs at reasonable prices.

Market Saturation: Spammers and bots can flood the NFT marketplace with low-quality or irrelevant NFTs, making it difficult for users to discover genuine and valuable creations. This dilution of quality can hinder the growth and reputation of the NFT ecosystem.

Automated Art Generation: Bots can be programmed to generate vast quantities of NFTs automatically, flooding the market with low-effort or algorithmically produced artwork. This can devalue the artistic and creative aspect of NFTs and reduce the overall quality of available offerings.

Trading and Transaction Manipulation: Bots can engage in unfair trading practices, such as front-running or sniping, to gain advantages over human traders. They can also manipulate trading volumes and prices, creating artificial market movements that can mislead investors.

Sybil Attacks: Spammers can create numerous fake accounts to boost the visibility or popularity of certain NFTs artificially. This can distort the perception of demand and influence buyer behavior.

Network Congestion: High bot activity can overload blockchain networks, causing congestion and transaction delays. This can adversely affect the user experience and increase transaction costs.

To combat spam and bots on my website, I have implemented off-chain verification using the Polygon ID SDK. This verification process takes place before users can sign in or sign up, ensuring that only genuine individuals can access the platform's features. Polygon ID, being a relatively new technology, adds an additional layer of security and authenticity to the verification process. In the future, I plan to incorporate on-chain verification, which will further enhance the system's resilience against bots, making it a more secure and trustworthy environment for users.

CHAPTER 6: - Conclusions and Future Scope

6.1 Conclusion

In conclusion, the development of my full stack NFT marketplace has been a significant accomplishment, bringing together modern web technologies and blockchain integration to create a secure and user-friendly platform. By leveraging technologies such as React.js, Node.js, and Express.js, I have built a robust and dynamic user interface that provides an intuitive experience for buyers, sellers, and collectors. The integration of IPFS for file storage has ensured reliable and decentralized data management, allowing users to securely upload and access files associated with their NFTs. Additionally, the incorporation of Solidity smart contracts with OpenZeppelin has enhanced trust and transparency, enabling secure and tamper-proof transactions within the marketplace. To combat spam and bot activities, I have implemented the Polygon ID SDK for off-chain verification, ensuring that only genuine individuals can sign up and access the platform. This proactive measure enhances the security and authenticity of user accounts, fostering a trustworthy environment for NFT enthusiasts. Looking ahead, I plan to further strengthen the system by implementing on-chain verification, leveraging the immutability of the blockchain to enhance security and transparency. This future enhancement will reinforce the marketplace's resilience against bots and fraudulent activities, providing users with even greater confidence in their transactions. Overall, my full stack NFT marketplace project demonstrates the potential of technology and blockchain integration in transforming the digital marketplace landscape, by creating a bot free and secure platform.

6.2 FUTURE SCOPE

1. **On Chain Verification Support: -**

   With on-chain verification, the marketplace gains the advantage of utilizing the immutability and transparency of the blockchain. User identities can be securely validated, ensuring that only genuine individuals can participate in transactions and interact with the platform. By leveraging the Polygon ID SDK, the verification process becomes streamlined and efficient, enhancing the user experience while maintaining a high level of security. This integration of on-chain verification using the Polygon ID SDK establishes a robust framework that instills trust and authenticity within the NFT marketplace. Users can have confidence in the integrity of their interactions, knowing that their identities and transactions are validated and secured by the power of blockchain technology.

2. **Caching System for IPFS data: -**

   As the NFT marketplace utilizes IPFS for file storage, implementing a caching system for IPFS data can greatly improve the platform's performance and user experience. By caching frequently accessed files or metadata locally, the system can reduce the reliance on fetching data from the IPFS network repeatedly. This results in faster load times, reduced latency, and a smoother browsing experience for users. Additionally, a caching system can help alleviate the strain on the IPFS network and contribute to more efficient data retrieval.

3. **Adding Support for different chains: -**

   Expanding the NFT marketplace's compatibility and interoperability by adding support for different blockchain networks is essential for broader adoption and increased liquidity. By integrating with additional chains such as Ethereum, Binance Smart Chain, or others, users can have more flexibility in terms of token standards, transaction speeds, and cost optimization. This allows a wider range of users to participate in the marketplace and engage with NFTs using their preferred blockchain networks.

4. **Adding more NFT standards support like ERC 1155 :-**

   To cater to a diverse range of NFT projects and their specific requirements, it is important to expand the support for different NFT standards. Incorporating standards like ERC 1155, which allows for the creation of both fungible and non-fungible tokens within a single contract, enables greater flexibility and utility in the marketplace. This expansion broadens the types of assets that can be traded and opens up possibilities for innovative use cases, attracting a wider user base and fostering a more vibrant ecosystem.

REFERENCES

Web

[1] Polygon ID Verifier SDK, https://0xpolygonid.github.io/tutorials/verifier/

[2] ReactJS, https://react.dev/

[3] express JS, https://expressjs.com/

[4] Decentralized Identity, https://www.linkedin.com/pulse/building-trust-transparency-nft-marketplaces-identity-atkinson/

[5] Zero Knowledge Proof, https://www.circularise.com/blogs/zero-knowledge-proofs-explained-in-3-examples

[6] IPFS, https://developers.cloudflare.com/web3/ipfs-gateway/concepts/ipfs/

[7] Pinata, https://www.pinata.cloud/

[8] SmartContract and Solidity, https://www.baeldung.com/smart-contracts-ethereum-solidity

[9] JsonWebToken, https://www.digitalocean.com/community/tutorials/nodejs-jwt-expressjs

[10] BcryptJs, https://www.npmjs.com/package/bcryptjs

[11] Redux Toolkit, https://redux-toolkit.js.org/

[12] Lottie, https://creattie.com/blog/how-add-lottie-to-react

[13] Ethers, https://dev.to/yakult/a-tutorial-build-dapp-with-hardhat-react-and-ethersjs-1gmi

[14] OpenZeppelin, https://www.openzeppelin.com/

[15] Iden3Js , https://github.com/iden3/iden3js