LTIMindtree

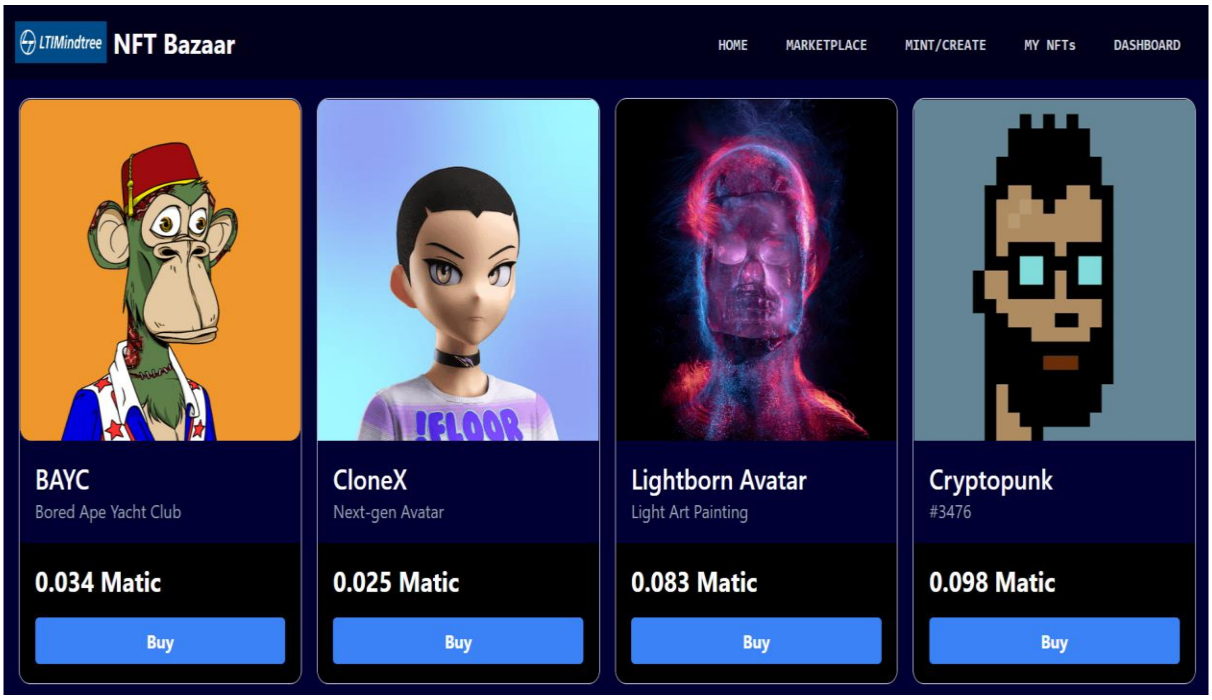
NFT BAZAAR

Play to Learn - (nftbazaar.lntinfotech.com)

Table of Contents

1 Introduction to NFT Bazaar

NFT Bazaar is LTIMindtree's first NFT marketplace, and a cool tool to immerse and dive into the world of NFTs. Essentially demystify the world of NFTs, Web3 and Cryptocurrency. It is a blockchain-based online platform and a premier destination to mint, sell and buy non-fungible tokens (NFTs).



2 Features of NFT Bazaar

NFT Bazaar provides with the following features:

- Mint NFT - It allows user to create their NFT on Marketplace to sell
- Buy NFT - Users can buy NFTs from Marketplace
- Resell NFT - Users can resell their collections to Marketplace
- Dashboard - Page for creators to see their listed and sold tokens
- My NFTs - Page for users to see their collections
- Authentication Flow with MetaMask Wallet
- Transfer NFT ownership to the buyer
- Tutorials for new users for how to get started

3 Application Flow

In this section we will describe 4 flows with respective to the user.

3.1 Authentication Flow

1. Users authenticate using MetaMask wallet, which act as unique ID for users.

3.2 Mint Flow

1. Once a user creates a MetaMask wallet, he/she should be able to upload NFT
2. Basic details of NFT like name, description and price are filled.
3. NFT image Is uploaded to IPFS using nft.storage API which returns the CID of image on IPFS.
4. The CID of image Is tagged with metadata of NFT (name, description) and JSON file is uploaded to IPFS.
5. Minting the NFT will store the NFT on blockchain using Alchemy API of polygon Mumbai after the transaction is confirmed on MetaMask
6. User can see their NFTs on Dashboard and Marketplace page.

3.3 Buy Flow

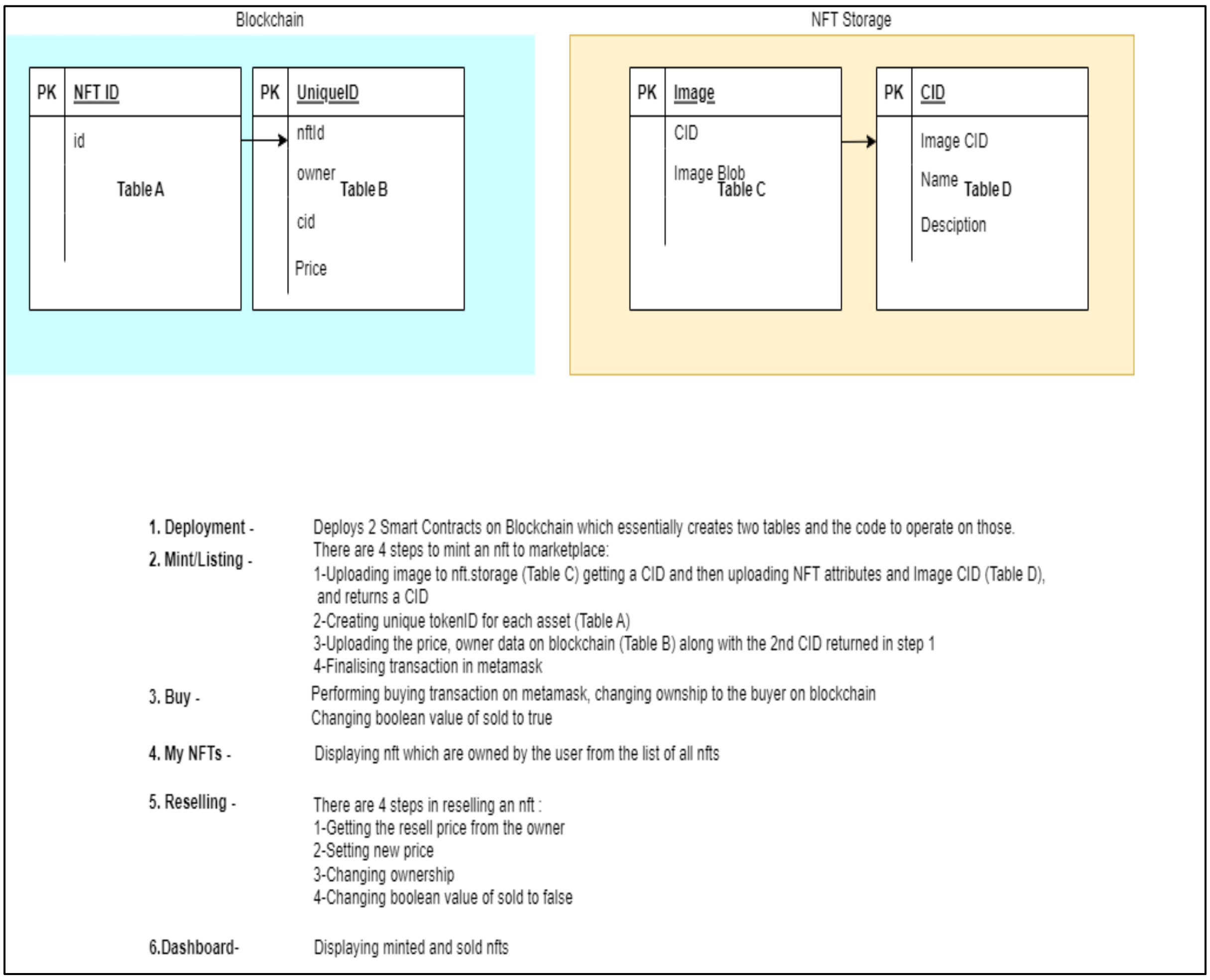
1. The user selects the NFT to purchase
2. Launch MetaMask to perform transaction
3. Buy Token - MATIC Token will be moved to Seller (after deducting the market fee)
4. Ownership will be transferred to the Buyer
5. User to view their purchased NFT under “My NFTs”

3.4 Resell Flow

1. The user can his/her collections under My NFTs page.
2. In each collection there is an option to resell the NFT
3. The user chooses the NFT to resell
4. Confirm the transaction on MetaMask
5. Ownership will be transferred to the Marketplace
6. Resell NFTs will be now visible under Dashboard page

4 Data Flow

1. The diagram captures the data model of the application. The fields of the table and how the relate to each other.
2. The application does not have a traditional database but instead stores data in a blockchain and IPFS storage.
3. More verbose data is stored in IPFS and their references are stored in blockchain.



1. Deployment - Deploys 2 Smart Contracts on Blockchain which essentially creates two tables and the code to operate on those.

2. Mint/Listing - There are 4 steps to mint an nft to marketplace:
1-Uploading image to nft.storage (Table C) getting a CID and then uploading NFT attributes and Image CID (Table D), and returns a CID
2-Creating unique tokenID for each asset (Table A)
3-Uploading the price, owner data on blockchain (Table B) along with the 2nd CID returned in step 1
4-Finalising transaction in metamask

3. Buy - Performing buying transaction on metamask, changing ownship to the buyer on blockchain
Changing boolean value of sold to true

4. My NFTs - Displaying nft which are owned by the user from the list of all nfts

5. Reselling - There are 4 steps in reselling an nft :
1-Getting the resell price from the owner
2-Setting new price
3-Changing ownership
4-Changing boolean value of sold to false

6.Dashboard- Displaying minted and sold nfts

5 Setting up the Project

Please go through this video to get the conceptual understanding of NFT Marketplace before Implementing the project - https://www.youtube.com/watch?v=GKJBEEXUha0

5.1 Prerequisites

1. Node.js version 16 or higher.
2. MetaMask wallet extension installed as a browser extension.

5.2 Technology Stack

The Application uses the following technology stack:

1. **Web application framework- Next.js** - Next.js is an open-source web development framework created by Vercel enabling React-based web applications with server-side rendering and generating static websites.
2. **Solidity development environment- Hardhat** - Hardhat is a development environment for Ethereum software. It consists of different components for editing, compiling, debugging and deploying your smart contracts and dApps, all of which work together to create a complete development environment.
3. **Ethereum Web Client Library- Ether.js** - The ethers.js library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem.

4. **MetaMask Wallet** - MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications.
5. **Alchemy** - Alchemy is a powerful blockchain developer platform providing a suite of developer tools. Developers building apps which interact with Ethereum can use Alchemy's powerful APIs to supercharge their apps, and leverage features not available in vanilla nodes.
6. **IPFS** - The Inter-Planetary File System is a protocol, hypermedia and file sharing peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting IPFS hosts.
7. **Polygon** - Polygon is a protocol and a framework for building and connecting Ethereum-compatible blockchain networks. Aggregating scalable solutions on Ethereum supporting a multi-chain Ethereum ecosystem.

5.3 Initialize the project

1. First, we create a new Next.js app using following command:

```
npx create-next-app nft-bazaar
```

2. Download the nft-bazaar.zip file from this: nft-bazaar.zip
   Note that the code provided along with the tutorial is not being used as we faced certain Issues (Described In section 9).
3. Replace the **package.json** file in the nft-bazaar directory from the provided zip file.
4. Run the following command: **npm install** in the nft-bazaar directory. This will Install all the necessary libraries required to run the project.
5. Install the Tailwind dependencies by running the following command

```
npm install -D tailwindcss@latest postcss@latest autoprefixer@latest
```

6. Next, create the configuration files needed for Tailwind to work with Next.js by running the following command

```
npx tailwindcss init -p
```

7. Configure the **tailwind.config.js** and **styles/global.css** files for frontend from the zip file.
8. Next, initialize a new Hardhat development environment from the root of your project:

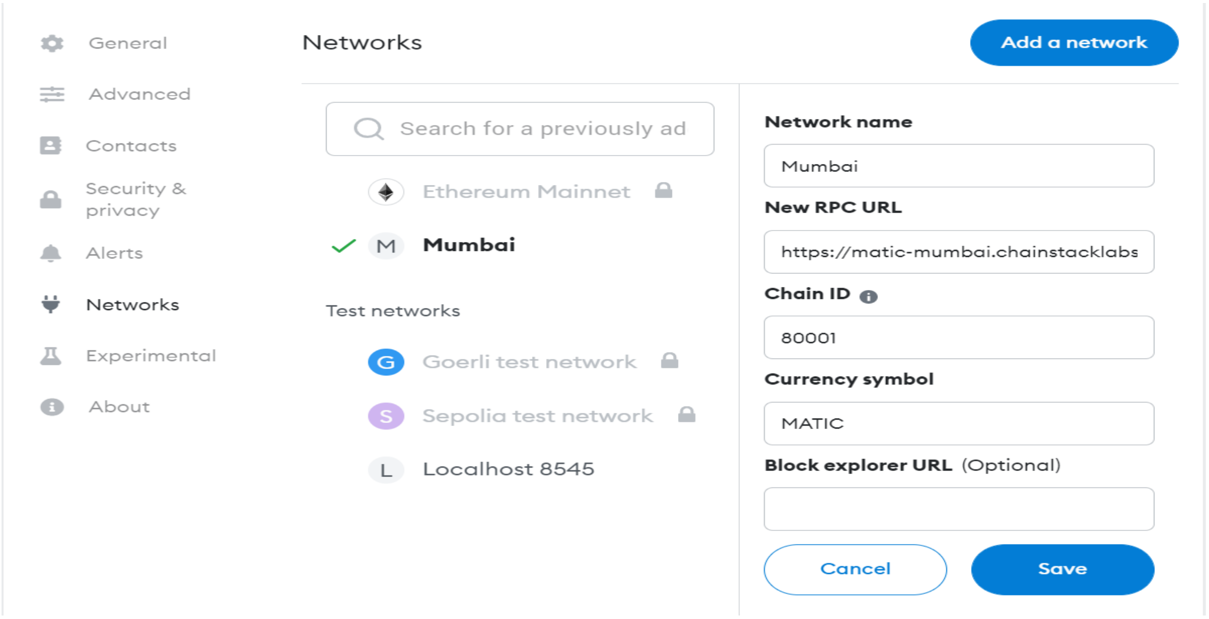
```
npx hardhat
```

9. Configure **hardhat.config.js** file and for polygon Mumbai URL use Alchemy API as it provides with free limit.
10. After configuring hardhat config file, Smart Contract need to be written to perform transactions on Marketplace. NFT.sol and NFTMarket.sol In contracts directory files contain the smart contract written in solidity.
11. Next, write the frontend code in the pages directory and image.js file In pages/api directory to let users interact with browser
12. Next, write the test/Lock.js and scripts/deploy.js and config.js to deploy our application.

5.4 Importing accounts into MetaMask

Add the Polygon Test network in MetaMask account using the following details:

- **Network Name**: Mumbai
- **RPC URL**: https://rpc-mumbai.maticvigil.com
- **Chain ID**: 80001
- **Currency Symbol**: MATIC



Network name: Mumbai
New RPC URL: https://matic-mumbai.chainstacklabs
Chain ID: 80001
Currency symbol: MATIC
Block explorer URL (Optional):

After, adding the network In MetaMask, copy then paste the private Key of our account in the hardhat.config.js file in privateKey variable.

Then some Matic tokens are required to interact with the applications. To get these, visit the Matic Faucet, inputting the MetaMask account address that would like to request the tokens.

5.5 Deploying to Polygon

Now that we have some Matic tokens, we can deploy to the Polygon Mumbai network. To do so, be sure that the address associated with the private key we are deploying the contract with has received some Matic tokens to pay the gas fees for transaction.

To deploy to Matic, run the following command:

```
npx hardhat run scripts/deploy.js --network mumbai
```

After running this command, we will get two address (NFT address and NFTMarket address) which we need to write in **config.js** file and an Artifacts directory get generated.

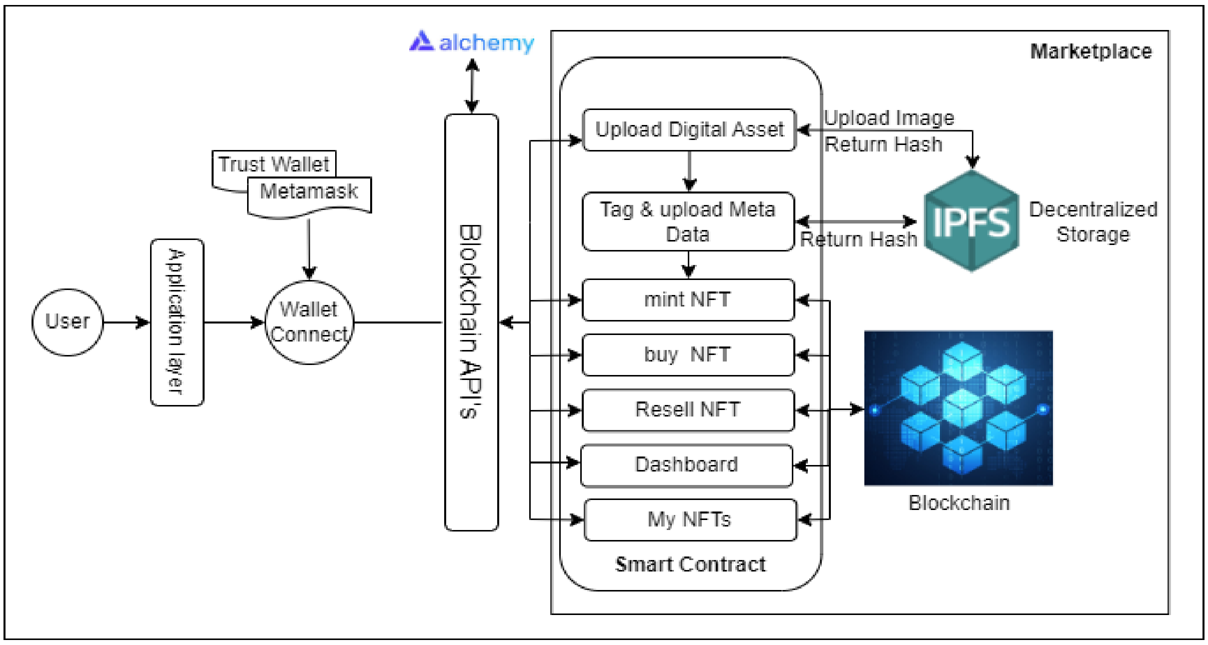
5.6 Executing

To start our application, run the following command:

```
npm run dev
```

It will start our server on localhost with port 3000. To access the website, open the browser with MetaMask wallet and use URL **http://localhost:3000/**.

6 Application Architecture



Technical Implementation of the Flow:

1) Users authenticate using metamask.
2) A user can be a buyer or creator.
3) **Minting (Creator)**:
   a) Basic details of NFT like name, description and price are filled.
   b) NFT image Is uploaded to IPFS which returns the CID of image on IPFS.
   c) The CID of image Is tagged with metadata of NFT (name, description) and JSON file is uploaded to IPFS.
   d) Minting the NFT will store the NFT on blockchain using Alchemy API of polygon Mumbai
   e) User can see their NFTs on Dashboard and Marketplace page.
4) **Buy and Resell (Buyer)**:
   a) User selects the NFT to purchase
   b) User clicks on buy the NFT and confirm the transaction on MetaMask
   c) Users can see their collection on My NFTs page and if they want to resell any NFT they do so by providing the new price.

7 List of API Endpoints

In this we will explain the APIs which we have created in our app, the first 6 APIs are of the smart contract which are required to perform the transactions. Image API Is used to convert the Image and resize them and the last one is for IPFS storage

7.1 mintNFT()

- This endpoint enables the user to mint the ERC721 token. It takes the following NFT attributes along with asset details and stores the data in IPFS
- IPFS Hash then get stored in the blockchain

7.2 createMarketSale()

- This enables the user to purchase the NFT

7.3 fetchMarketItems()

- This enables the marketplace to bring the market items

7.4 fetchMyNFTs()

- This enables the marketplace to bring the items purchased by the user, so each user can view their purchases

7.5 fetchItemsCreated()

- This enables the user to view a list of NFTs created by themselves.

7.6 listNFTForReSell()

- This enables you to put an item for resell

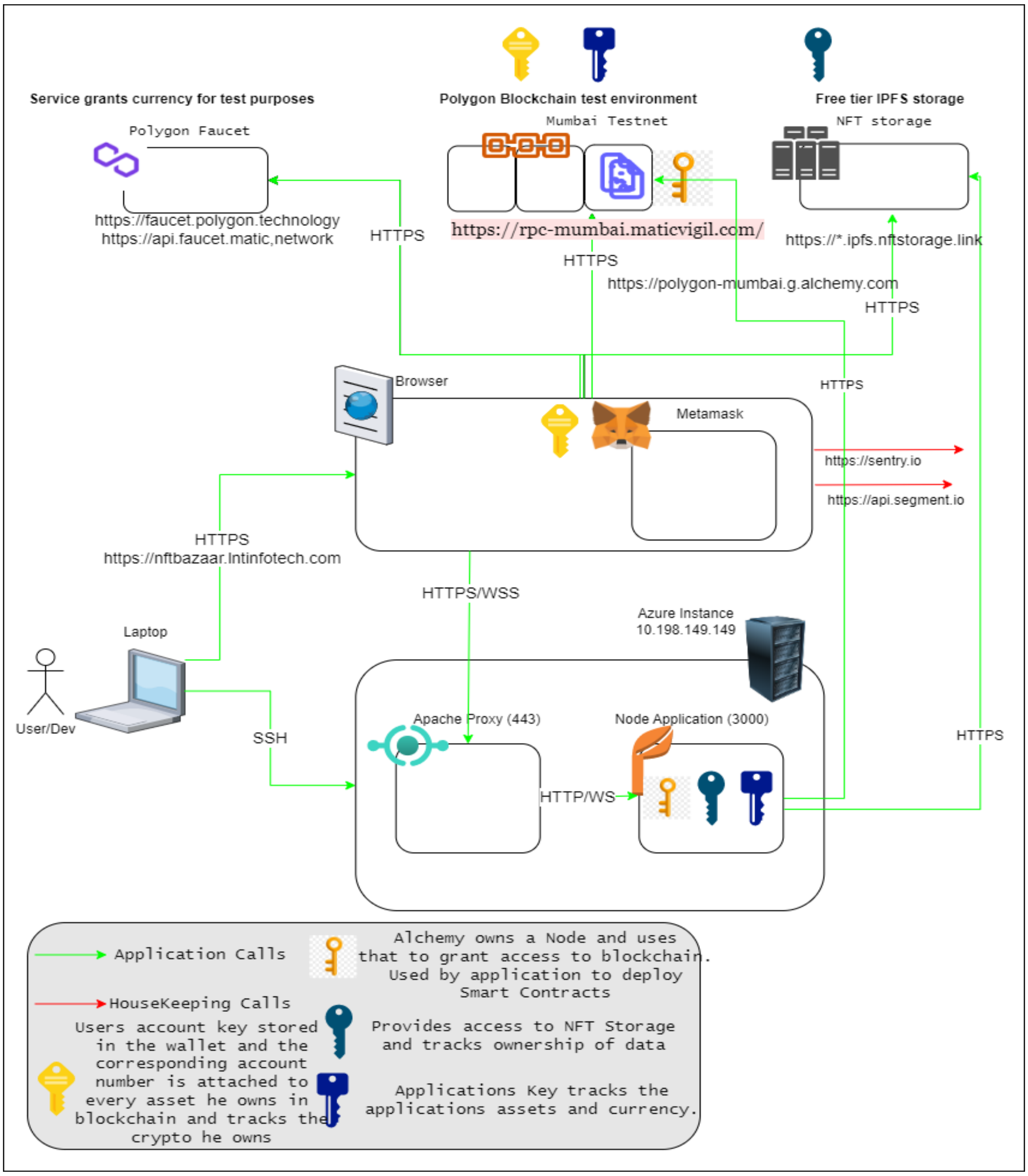
7.7 imageAPI()

- This API Is used to convert the size and format to png format for all Images.

7.8 Nft.storage()

- This enables you to store the images and their details on IPFS

8 Deployment Architecture


Service grants currency for test purposes
Polygon Faucet
https://faucet.polygon.technology
https://api.faucet.matic,network
HTTPS
Polygon Blockchain test environment
Mumbai Testnet
https://rpc-mumbai.maticvigil.com/
HTTPS
https://polygon-mumbai.g.alchemy.com
Free tier IPFS storage
NFT storage
https://*.ipfs.nftstorage.link
HTTPS
Browser
Metamask
HTTPS
https://sentry.io
https://api.segment.io
HTTPS
https://nftbazaar.lntinfotech.com
HTTPS/WSS
Azure Instance
10.198.149.149
Laptop
User/Dev
SSH
Apache Proxy (443)
Node Application (3000)
HTTP/WS
HTTPS

Application Calls

HouseKeeping Calls

Users account key stored in the wallet and the corresponding account number is attached to every asset he owns in blockchain and tracks the crypto he owns

Alchemy owns a Node and uses that to grant access to blockchain. Used by application to deploy Smart Contracts

Provides access to NFT Storage and tracks ownership of data

Applications Key tracks the applications assets and currency.

9 Securing the application

To secure our application using SSL certificates we need to first generate the private key and a CSR file, for this we will be using OpenSSL command in our ubuntu Instance.

The OpenSSL command below will generate a 2048-bit RSA private key and CSR:

```
openssl req -newkey rsa:2048 -keyout NFTBazaar.key -out NFTBazaar.csr
```

Now enter the following details:

- **Country Name** – IN
- **State** – Maharashtra
- **Locality** – Mumbai
- **Organization name** – Larsen & Toubro Infotech Limited
- **Organizational unit name** – GTO
- **Common name** – nftbazaar.lntinfotech.com
- Rest of the fields are optional

Upon completion of this process, you will be returned to a command prompt. You will not receive any notification that your CSR was successfully created. Now, give the CSR file to CA to generate your domain name.

10 Issues Faced during Implementation

10.1 Polygon Mumbai url

While building this project we were not able to use the Infura polygon Mumbai URL as to use that we need to add our credit card, so for that we used Alchemy as an alternate.

10.2 IPFS storage limit

We started this project using IPFS services from Infura with Ipfs-http-client library then have to switch to nft.storage library to store our Images as in Infura there was a limit of 1 GiB free storage.

10.3 Minting NFT on Cloud Instance

After deploying the application on Cloud VM Instance we were not able to add Image to nft.storage as our application was not secured using SSL certificates, so we weren't able to connect to nft.storage from our cloud public IP.

10.4 Adding SSL Certificates to our App

To secure our application we were not able to add the certificates on our node application, so as an alternative we used Apache2 as proxy service. Now when a user tries to access our app he/she hits our apache2 server running on port 443 which then routes them to our node application running on port 3000.

10.5 Loading Image on Browser

While testing our App we found out that Edge, Firefox does not support avif image type so are now converting all the Images to png format as It Is supported by all the browsers, and the other Issue we faced was the Image dimensions, so we are also resizing our all Image to same dimensions. To performs both this task we are **sharp** library from node.js. Since sharp library does not work in browser, we had to ship some functionality to the backend and create a API.