

NFT Application

INTRODUCTION

NFT are non Fungible tokens that are unique digital items managed by blockchain e.g. digital art, digital music, virtual items in a video game, etc. They cannot be replaced by another identical item. There can be an ownership exchange of the item but the item remains immutable.

The project comprises 2 different applications:

1. IPFS file Handler (<https://github.com/ArnavDhiman/ipfs-file-upload-brl>)
2. NFT minter (<https://github.com/ArnavDhiman/nft-minter-brl>)

The applications listed above aims to create NFT by allowing users to upload an image to an IPFS server (to read more about IPFS and its use cases visit this [link](#)).

NOTE: All the transactions that happen on the Ethereum blockchain are done on Ropsten Test Net.

3rd PARTY APPS AND APIs'

The applications use multiple 3rd party APIs' and applications to achieve their goal. Below is a list of all the APIs and applications used and their links.

#	API Name	Link	Use Case
1	MetaMask wallet	https://metamask.io/	This wallet is used to store the mined NFT and pay the Ethereum gas fee. You will need to create an account for this application. Make sure that both chrome extension and mobile apps are installed. The mobile app will be used to view the NFT.
2	Infura IPFS storage	https://infura.io/	This storage allows us to store the uploaded image in an IPFS file storage.
3	Alchemy	https://www.alchemy.com	This API is used to interact with Ethereum blockchain and mint NFTs. You will also require an account on this platform to use its APIs.
4.	Etherscan	https://etherscan.io/	This portal is required to check the status of our transaction. This can

			also be done on Alchemy's web portal but it's better to check the status here .
--	--	--	---

APPLICATION 1 - IPFS FILE UPLOADER (REACT)

This web app is the main UI for the project and solves the following 2 use cases that are required:

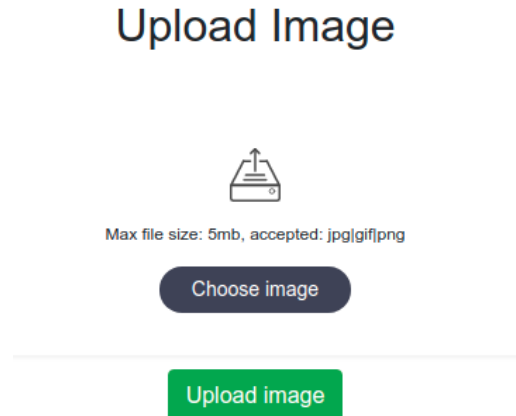
1. Upload the image and metadata JSON file to the IPFS server.
2. Get the IPFS hash of the metadata file and send it to the backend.

This app uses react js (a javascript framework) so before starting the app make sure that **Node** and **NPM** are installed on the computer. Follow the steps here <https://nodejs.org/en/download/> to install Node and <https://www.npmjs.com/get-npm> to update/install NPM.

INTERNAL WORKINGS

Following is how the data flows in the application:

1. The user selects the image to upload using the "Choose image" button on the UI (gray button) and clicks the "Upload image" button (green button).



The image is captured and is converted to an unsigned int buffer. All this happens in the `onDrop()` method. Then this buffer is uploaded to the IPFS storage in `handleUpload()` method. The `handleUpload` method uses `ipfs.infura.io` as the server address and 5001 port internally.

2. When the request to the IPFS server is successful, we receive a hash of the file uploaded. This hash + <https://ipfs.io/ipfs/> gives us the address of the file stored. This link will look like this <https://ipfs.io/ipfs/hash>.
3. Using the above-mentioned IPFS link of the uploaded file with some metadata about the image, we create a new JSON object. This object will look like the following snippet:

```

{
  "attributes": [
    {"size":size},
    {"date_created":"date_created"},
    {"ipfs_hash":"ipfs_hash"}
  ],
  "description":"image description",
  "name":"image name",
  "image":"ipfs hash link to the file"
}

```

Data can be added or removed from the attributes.

4. The user can download this JSON by clicking on the “Download JSON” button (highlighted in red) and the minting process will start when the user clicks on the “MINT NFT” button (highlighted in green).

Upload Image


 Max file size: 5mb, accepted: jpg/gif/png

File Name	File Size	Download	Mint NFT
BRLGroupPhoto	1332025	<input type="button" value="Download JSON"/>	<input type="button" value="MINT NFT"/>



5. The above-mentioned JSON object is then converted to string using `JSON.stringify()` and the resultant string is again converted to buffer using the `Buffer.from()` method. This buffer is then uploaded to the IPFS storage in the `mintNFTHandler()` method and a hash is generated for this file. This hash is then converted to the link as mentioned in step 2 and a POST request is made in `mintRequestHandler()` method to the **NFT minter application** on port 9000.
6. The minting of NFT can take up to 10 seconds. After clicking on the MINT NFT button, check the status of the transaction on alchemy's mempool web portal. The link to the page is <https://dashboard.alchemyapi.io/mempool>

STEPS TO RUN THIS APP

Following steps are required to run this app:

1. Install node and NPM on the machine as stated above.

2. Run `npm install` command on the terminal. This will automatically install all the required react libraries required to run this app.
3. Run `npm start` command on the terminal to start the app.

APPLICATION 2 - NFT MINTER (NODE)

This app is the backend for the project that receives the IPFS hash of the file uploaded in the UI and connects to Alchemy to mint the NFT. It solves the below use cases:

1. Deploy a smart contract on the Ethereum blockchain to mint NFT(ERC-721 non-fungible token standard).
2. Using the smart contract, Mint NFT using Alchemy's API.

This app uses Node js on the server-side and uses Solidity to create the smart contract.

INTERNAL WORKINGS

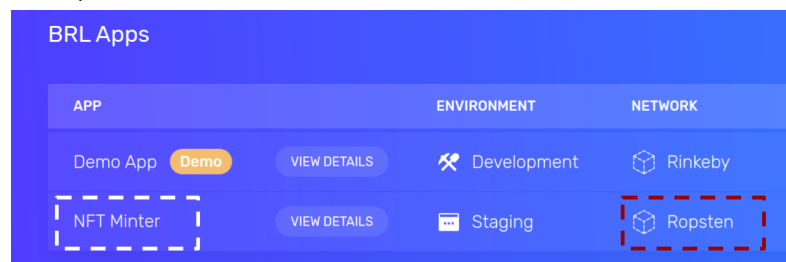
Following is the data flow in this app:

1. The app listens to port number 9000 continuously and when the hash of the metadata file uploaded to the IPFS server is generated, it is sent by the UI app to port 9000 by a POST request. This request is processed by the router method of `/routes/mint-nft.js` file.
2. The router method extracts the hash link from the JSON request and invokes the minter method in `/scripts/mint-nft.js` and passes the link to it.
3. This minter method uses the `contractAddress` i.e. the address of our ERC-721 contract and invokes Alchemy's API with the IPFS link. It also signs the transaction with the Meta Mask wallet's private key.
4. As the blockchain transaction is slow, the status of the transaction can be monitored on the mempool portal (<https://dashboard.alchemyapi.io/mempool>).

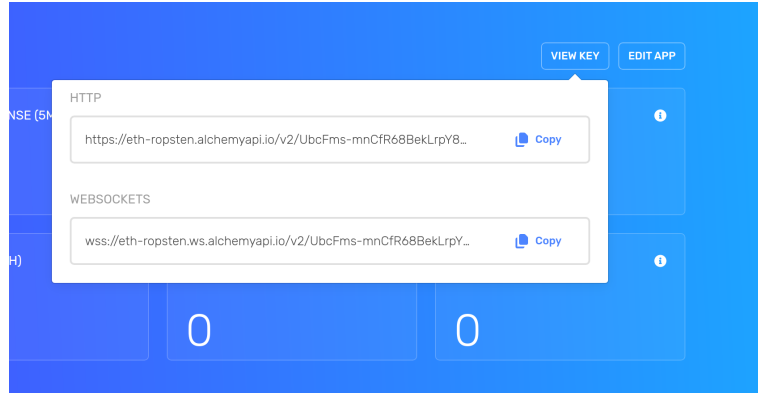
STEPS TO RUN THIS APP

Following are the steps to run the application:

1. Create an `.env` file:
 - a. Create an empty `.env` file in the parent directory of the app.
 - b. Open Alchemy dashboard (<https://dashboard.alchemyapi.io/>) and select the NFT minter app(Marked in White). Make sure that the app is on the Ropsten test net (Marked in Red).



- c. Click on the "View Key" button and copy the HTTP link (1st link here).



- d. In the .env file, create a field named “API_URL” and assign it the link copied in the above step.
- e. Now open the Metamask Chrome extension and click on the : button next to the “Account 1” header and go to account details.
- f. Copy the hash already selected and in the .env file create a field named “PUBLIC_KEY” and assign it the hash copied
- g. Next, repeat the step e and go to account details. Now click on the “export private key” button. This will prompt a window to enter the account password. Enter the password and click on “confirm”.
- h. Doing this will show a hash again. Copy it and in the .env file create “PRIVATE_KEY” and assign it the copied hash.

Note: Do not add the .env file to GitHub as exposing these hash will result in a compromised meta mask wallet.

In the end, the .env file will look like the picture below:

```
API_URL = "https://eth-ropsten.alchemyapi.io/v2/yourAppLink"
PRIVATE_KEY = "YOUR PRIVATE KEY"
PUBLIC_KEY = "YOUR PUBLIC KEY"
```

2. Deploy smart contract
 - a. Run the /scripts/deploy.js file using the following command on the terminal.
`npx hardhat run scripts/deploy.js --network ropsten`
 - b. On successful completion, the following message will appear.
Contract deployed to address: <address of the contract>
 - c. Copy the address of the contract and go to <https://etherscan.io/> to check the status of the contract. It should show a successful message or a pending message. If the status is pending, wait till the transaction is successful.
 - d. Copy the address of the contract deployed in **step c** and in /scripts/mint-nft.js assign `const contractAddress` the address copied.
3. Start the application
 - a. Run `npm install` command on the terminal. This will automatically install all the required react libraries required to run this app.

- b. Run `npm start` command on the terminal to start the app. Make sure both the apps are running simultaneously.

VIEW THE NFT IN META MASK WALLET

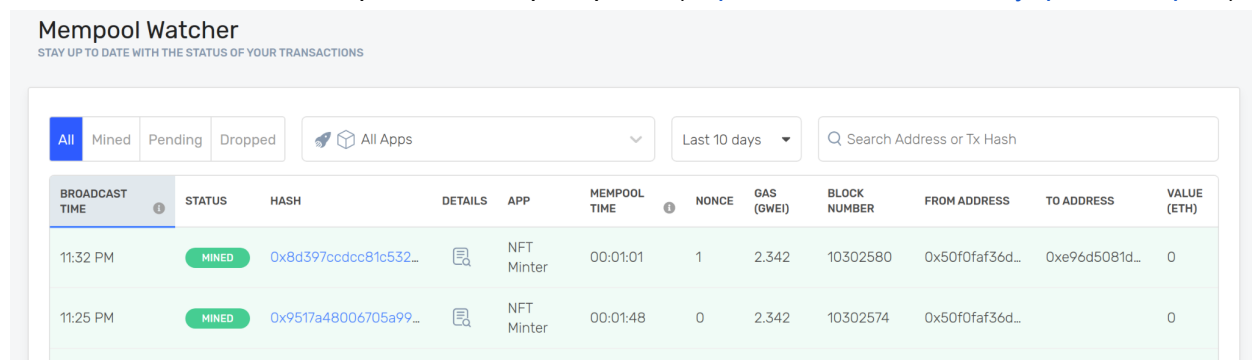
To view NFT on Metamask Wallet, we need an iOS or Android mobile app. Following are the links to download them:

iOS: <https://metamask.app.link/skAH3BaF99>

Android: <https://chrome.google.com/webstore/detail/nkbihfbeogaeaoehlefnkodbefgpgknn>

After downloading the wallets, log in to the app by scanning a QR code that can be shown from the chrome extension. Go to account > settings > advanced > sync with mobile.

Open the app and navigate to the “Collectables” tab in the app. Click on “Add collectible”. To get the address and NFT ID, open the mempool portal (<https://dashboard.alchemyapi.io/mempool>),

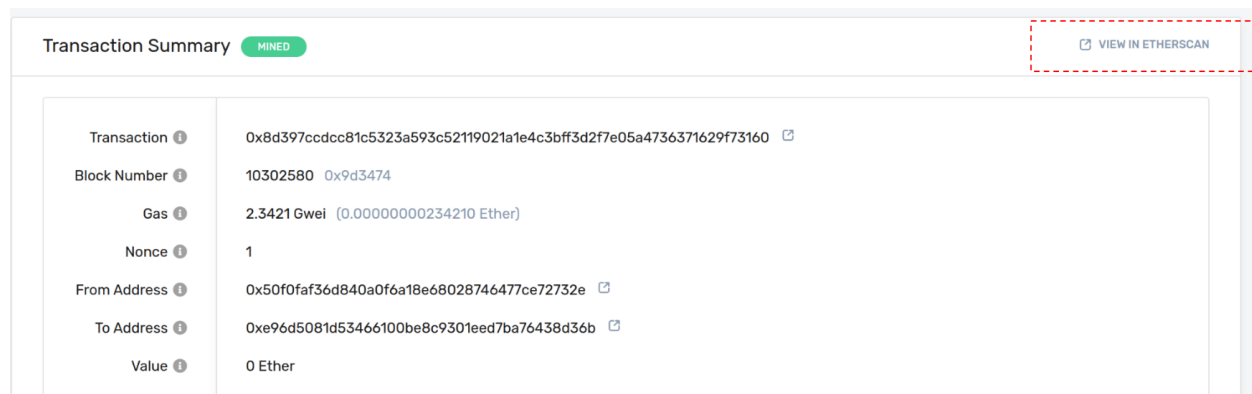


Mempool Watcher
STAY UP TO DATE WITH THE STATUS OF YOUR TRANSACTIONS

Filters: All (selected), Mined, Pending, Dropped. Search: Search Address or Tx Hash. Time: Last 10 days.

BROADCAST TIME	STATUS	HASH	DETAILS	APP	MEMPOOL TIME	NONCE	GAS (GWEI)	BLOCK NUMBER	FROM ADDRESS	TO ADDRESS	VALUE (ETH)
11:32 PM	MINED	0x8d397ccdcc81c532...	[Icon]	NFT Minter	00:01:01	1	2.342	10302580	0x50f0faf36d...	0xe96d5081d...	0
11:25 PM	MINED	0x9517a48006705a99...	[Icon]	NFT Minter	00:01:48	0	2.342	10302574	0x50f0faf36d...		0

Now select the NFT mining transaction and click on the “View in Etherscan” button.



Transaction Summary MINED [VIEW IN ETHERSCAN]

Transaction	0x8d397ccdcc81c5323a593c52119021a1e4c3bff3d2f7e05a4736371629f73160
Block Number	10302580 0x9d3474
Gas	2.3421 Gwei (0.00000000234210 Ether)
Nonce	1
From Address	0x50f0faf36d840a0f6a18e68028746477ce72732e
To Address	0xe96d5081d53466100be8c9301eed7ba76438d36b
Value	0 Ether

This will open the [Etherscan](https://etherscan.io) portal's transaction details page. Copy the Interacted With(To) contract address(Highlighted in Red), note that this is the same address used in NFT minter's [step 2c](#). Input the copied address in the mobile app and the ID of the NFT(Highlighted in Green)

Transaction Details	
Overview	Logs (1) State
[This is a Ropsten Testnet transaction only]	
Transaction Hash:	0x8d397cdcc81c5323a593c52119021a1e4c3bff3d2f7e05a4736371629f73160
Status:	Success
Block:	10302580 1 Block Confirmation
Timestamp:	1 min ago (May-25-2021 06:32:15 AM +UTC)
From:	0x50f0faf36d840a0f6a18e68028746477ce72732e
Interacted With (To):	Contract 0xe96d5081d53466100be8c9301eed7ba76438d36b
Tokens Transferred:	From 0x0000000000000000... To 0x50f0faf36d840a... For ERC-721 TokenID [1] MyNFT (NFT)
Value:	0 Ether (\$0.00)

Refresh your app a few times and you should be able to see the NFT there.

FUTURE WORK

This project has many areas which can improve the overall security and functionality. Below are a couple of areas identified:

1. The application can have a user authentication mechanism that can eliminate unauthorized access to the app.
2. The application, right now, does not use any database to store all the files uploaded to the IPFS server. A database can help us to track all the files uploaded to the IPFS server.

ALL ACCOUNT DETAILS

Gmail, Alchemy

userID : brltestacc@gmail.com
 Password: Q-xwYEmngQBH!3qN

MetaMask

userID: brltestacc@gmail.com
 Password: !7xUt#MK78AsEWW
 MetaMask Secret Phrase: salute custom pumpkin lab special wash upper panel island someone maple like