

Smart Contract NFT Trading Marketplace

Yuanhao Shen, ys3609

Zijian Xu, zx2413

Yining Ma, ym2928

Sirui Feng, sf3166

GitHub Link: <https://github.com/Kawamiya/NFTMarket>

1. Introduction:

Our DAPP is designed to be user-friendly and intuitive. Users may simply purchase, sell, and list NFT assets through the UI interface. The NFTs are securely kept on the blockchain, preventing them from being tampered with or copied. Furthermore, our DAPP accepts a variety of payment options, including cryptocurrency payments, making it easier to purchase and sell NFTs. One of our DAPP's distinguishing features for users is to mint their own NFTs by uploading digital photos. This gives creators the ability to monetize their work by selling their unique digital assets. Our DAPP also provides a powerful search capability, allowing users to easily identify NFTs that fit their interests. Our DAPP uses smart contracts to assure transaction security and accuracy. These contracts are automatically executed after certain conditions are satisfied, eliminating the need for middlemen and lowering the risk of mistakes or fraud. Overall, our DAPP provides a dependable and efficient platform for exchanging NFT goods, generating opportunities in the digital space for sellers and collectors.

2. Project Design

2.1 Program Structure

Our DAPP involves two parts: a smart contract at the backend and a UI interface at the front end. At the backend, our team developed a smart contract based on ERC-721 standards to achieve operations that satisfied the requirements of an NFT marketplace. ERC-721 is a groundbreaking token standard of NFT development in Ethereum's ecosystem. The token standard defines a common interface for NFTs, and each token is distinct. ERC-721 enables NFTs to be moved between accounts and traded for other currencies. The contract design, deployment, and test are based on the Truffle framework. The development team used the Truffle framework for contract creation, deployment, and testing to ensure the efficiency and security of our smart contract. The Truffle framework manages the complete contract development cycle in a fast and secure way. During the development stage, our team ran tests using Ganache. Ganache is a personal blockchain emulator that enables us to create a blockchain environment for testing smart contract capabilities without requiring a live blockchain network.

As for the front end, our team designed an interactive UI to let users list and trade their digital assets, we use Vue.js, Metamask, and Web3 to connect with our backend smart contract, and use pinata and IPFS to get URL links of the picture uploaded by users.

2.2 Backend

Our team uses the Truffle development framework to build and test our smart contract. Our smart contract includes basic operations like mint tokens, creating NFT items, getting NFT items, listing NFT items and item counts, and trading NFT items.

- **Mint token:** After a user requests to list a new NFT item, the blockchain needs to create a new token onto the network. This process is called mint token. For our 'mintToken' function, we need to provide a token URI which is created by Pinata. The price is defined by the user on the UI page. This function is defined as 'Payable' since it is expected to accept ether or require a gas fee during execution. An important check conducted before creating a new token is to check whether the input token URI has already been present in the network. Also, the function will check if the listing price matches the input price. After successfully creating a new token, the function will first add a new line to the NFT tables stored on the blockchain and finally return the new token ID.
- **Access NFT items:** To access the NFT item information on the blockchain, our team designed the function 'getNftItem' which takes an input token ID and returns the NFT item information that matches the input token ID. The NFT item information includes price, creator, and status information indicating whether the item is already listed('isListed').
- **Trade NFT items:** Trading NFT items involves buying and selling.
 - As for buying functionality, our team designed a function 'buyNft'. The input of this function is a token ID that comes from the user's buying request. In this function, two necessary checks are conducted to make sure the success of later ownership transfer. First, the function checks whether the buying request initiator is already the owner of the NFT under this input token ID. Then, the function will check whether the price of this NFT item is the same as the price listed on the front-end UI. If both tests pass, the function will delist the item. The ownership of this NFT item will be transferred to the buying request sender and the original owner of this NFT item will get paid with the value of the price.
 - The function 'placeNftOnSale' takes the input of the token ID and listing price. The listing price is defined by the sell request sender. First, the function checks if the owner of the NFT item corresponds to the input token ID. Then, the function checks whether the NFT item is already listed on the marketplace. With the success of the check, the function will change the status of this NFT item to listed and set the listing price with the input price value.
- **Burn token:** To permanently remove one NFT item, the function 'burnToken' takes an input token ID and deletes the information related to this token ID. After the execution of this function, the NFT item and token will be removed from the blockchain network. The related information to this token ID will also be deleted.

2.3 Frontend

We designed a user interface frontend website to visualize the functions we have realized in the smart contract. The pages include:

- Market Home page: display the NFTs currently on the market, the user can get detail information on the NFTs from the page and if they click the buy button, the purchasing relevant functions in the smart contract will be executed, the amount of ETH in Metamask will be charged, and the account can receive the ownership.
- User information page: include the NFTs owned by the account that login in the Metamask. For the NFTs that are purchased by the account, there is a function to resale the item and set a new price. For the NFTs created by the account, there is a function to revoke the item from the market.
- CreateNFT page: allow users to upload the NFT object and set a price, after clicking submit, the system will send a request to Pinata and IPFS to get a Hash value and a URL for this NFT object, then we execute the createNFT function in the backend, the user will be asked to agree the transaction and be charged a gas fee. After that, the NFT will be placed on the market.

2.4 Procedure

With the successful compilation of our smart contract, our team deployed the contract on the local test network, Ganache.

Connect Metamask to the local blockchain network. After a successful connection, a Web3 instance will be injected into the browser. This connection will enable Metamask to automatically communicate with the deployed blockchain network. The information Metamask will obtain is the ledger information from the whole network. Accessing a single node on the blockchain network is able to give us the ledger information for the whole network.

After contract migration, we conducted several test cases by running the Truffle test on the file 'NFTMarketplace.test.js'. First, our team tests if the contract is able to create a new NFT. This test is conducted in the section of code with 'describe("Mint token")'. The next test case will show whether the contract is able to list an NFT for sale. This test is conducted in the section 'describe("list all Nft on sale")'. To test whether the contract is able to execute a successful NFT purchase or not and remove an NFT from the sale, the code section starting with 'describe("Buy NFT")' conducts these tests. Lastly, the contract will be tested if it can assert that the NFT ownership remains with the first user account and no Ether was transferred. This test is also included in the code section under 'Buy NFT'.

3. Result

3.1 Test Result

We write test cases in the Truffle test to test the functionality of our smart contract. And all test cases are passed, the execution status is shown below:

```

Compiling your contracts...
=====
> Compiling ./contracts/NFTMarketplace.sol
> Compiling @openzeppelin/contracts/access/Ownable.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling @openzeppelin/contracts/utils/Context.sol
> Compiling @openzeppelin/contracts/utils/Counters.sol
> Compiling @openzeppelin/contracts/utils/Strings.sol
> Compiling @openzeppelin/contracts/utils/introspection/ERC165.sol
> Compiling @openzeppelin/contracts/utils/introspection/IERC165.sol
> Compiling @openzeppelin/contracts/utils/math/Math.sol
> Artifacts written to /var/folders/68/52xbtgjt4772tt6glt2rlgmc0000gn/T/test--19749-jeUNRon8QOHN
> Compiled successfully using:
  - solc: 0.8.19+commit.7dd6d404.Emscripten.clang
string
string

Contract: NFTMarketplace
Mint token
  ✓ owner of first token should be address[0]
  ✓ first token id should point to correct tokenURI
  ✓ should not be possible to create a NFT with used tokenURI (149ms)
  ✓ should have create NFT item
Buy NFT
  ✓ should unlist the item
  ✓ should decrease listed item count
  ✓ should change owner
  ✓ should not change owner if buy owned item (62ms)
list all Nft on sale
  ✓ should list all Nft on sale (52ms)
list owned Nfts by owner
  ✓ should list all owned Nfts
burn token
  ✓ should have 2 token for current account 4
  ✓ should only have 1 token after burn (180ms)

12 passing (3s)

```

Fig 3 Test Results

3.2 User Interface

Our front end contains a total of three pages and one component. The three pages are the Home page (Figure 3.1), the Personal Information page (Figure 3.2), and the Create New NFT page (Figure 3.8). The component is the Topbar component. We also simulate the actions on the front end to further test the functionality of our smart contract.

- The Topbar component contains three jump links and the Metamask account information, which is shown in Figure 3.4.
- On the Home page, we display all the NFTs for sale. When moving the mouse over one of the NFTs, the creator, price and buy button of the Nft will be displayed (Figure 3.5). When the purchase is clicked, it will enter the transaction process. During the transaction, metamask will check if the current user has enough ETH, so we don't do additional validation on the front end. In the back end contract section, we do a validation function that will verify the status of the NFT currently being purchased, whether the purchaser has enough Nft, etc. When the purchase is made, the item will turn to off-sale status and at the same time will no longer be displayed on the front page, which then allows us to enter the personal information page where we can see all the NFT information currently held by the user.

- On the personal information page, when we move the mouse over a held NFT, it will show the creator of the current NFT, the price, and whether to on-shelf/off-shelf. We can take down an Nft that is in on-sale status, after which the item will no longer be displayed on the home page (Figure 3.7). At the same time, we can also shelve an item we have purchased and set a new price when we shelve it (Figure 3.6).
- On the Create NFT page, we can upload images to create a new NFT. The uploaded images will be automatically saved to Pinata, and at the same time pinata will generate a public URL address to return to us, and we will send this URL to our back-end contract to generate a new NFT item (Figure 3.8,3.9, 3.10).

4. Conclusion

In conclusion, our team has successfully developed an NFT Marketplace smart contract that allows users to create and trade NFTs. The smart contract is structured in a way that ensures secure and efficient transactions by implementing various operations such as minting tokens, accessing NFT items, and trading NFT items. Additionally, our team has thoroughly tested the contract using Truffle to ensure its functionality.

We have also designed a user-friendly frontend website that allows them to easily interact with the smart contract. The website features a Home page that displays all the NFTs available for purchase, a Personal Information page where users can view their owned NFTs and make changes to their listings, and a Create NFT page where users can upload their own NFTs to be added to the marketplace.

Overall, we are confident that our NFT Marketplace smart contract and accompanying frontend website could provide a valuable platform for creators and collectors to buy, sell, and trade NFTs with ease and security.

Contribution:

- Zijian Xu and Yuanhao Shen took charge of smart contract design, deployment, and test case design;
- Zijian Xu, Yining Ma took charge of Frontend development and testing;
- Yining Ma and Sirui Feng combine frontend and backend applications, demo the Dapp;
- Yuanhao Shen, Zijian Xu, Yining Ma, Sirui Feng contributed to the project documentation (final report).

Figures

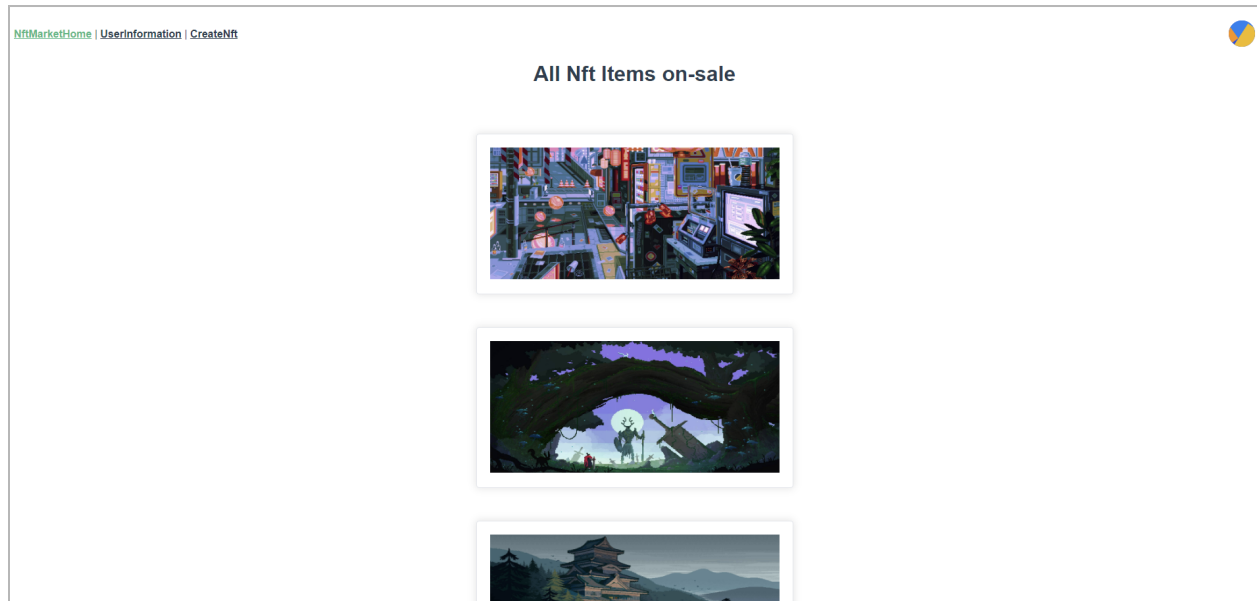


Fig 3.1 Nft marketplace home page

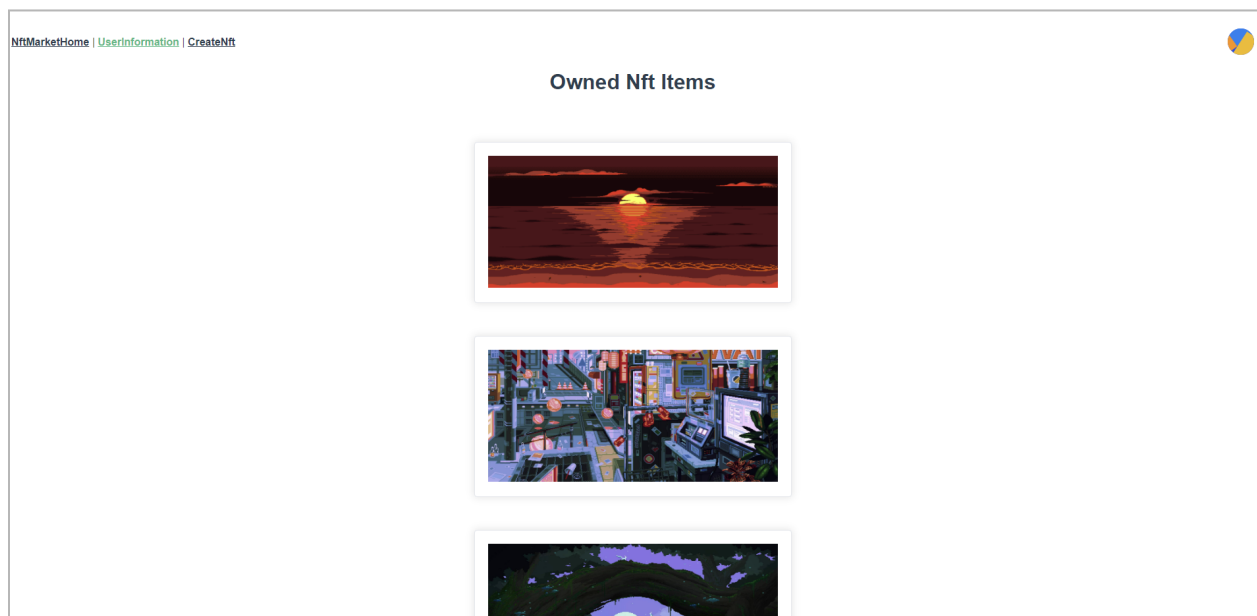


Fig 3.2 User information page

All Nft Items on-sale



creator: 0xc53232D0283E989f2ba29E3c48eA15f0cfF721cA

price: 4 ETH

buy this Nft

Fig 3.3 Home page function

s on-sale



Account: 0xc53232D0283E989f2ba29E3c48eA15f0cfF721cA

Balance: 81.671089138279521442 ETHs

Network Id: 5777

Fig 3.4 Topbar Function

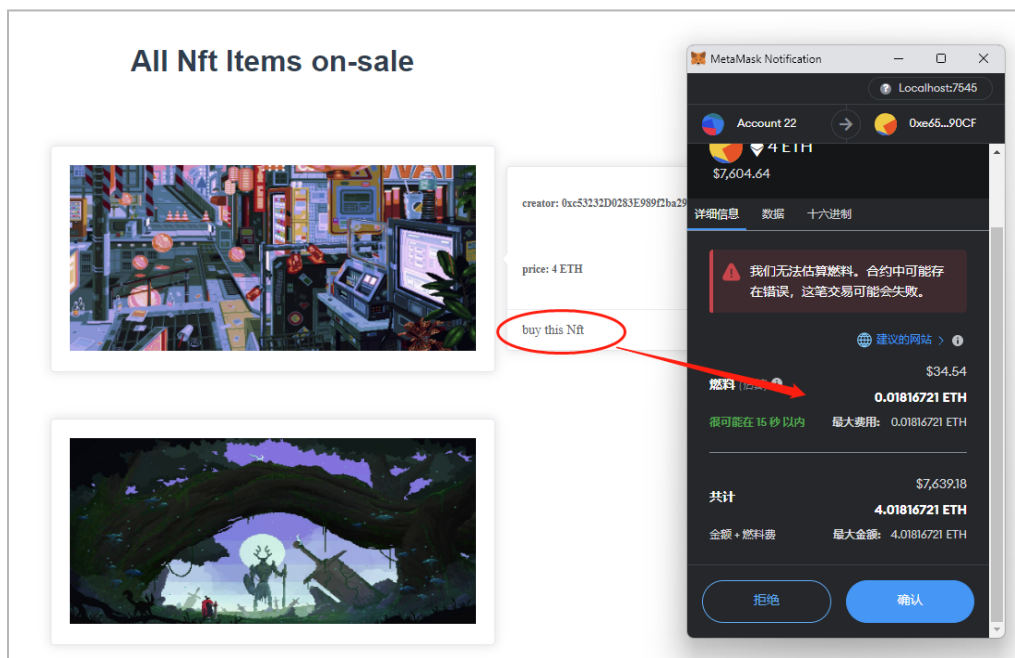


Fig 3.5 Buy an Nft item on the Market page

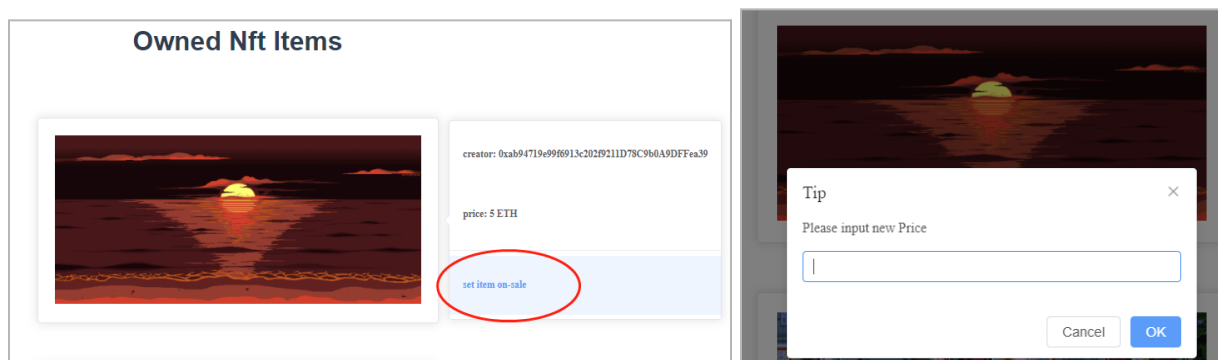


Fig 3.6 Set owned Nft on-sale on UserInformation page

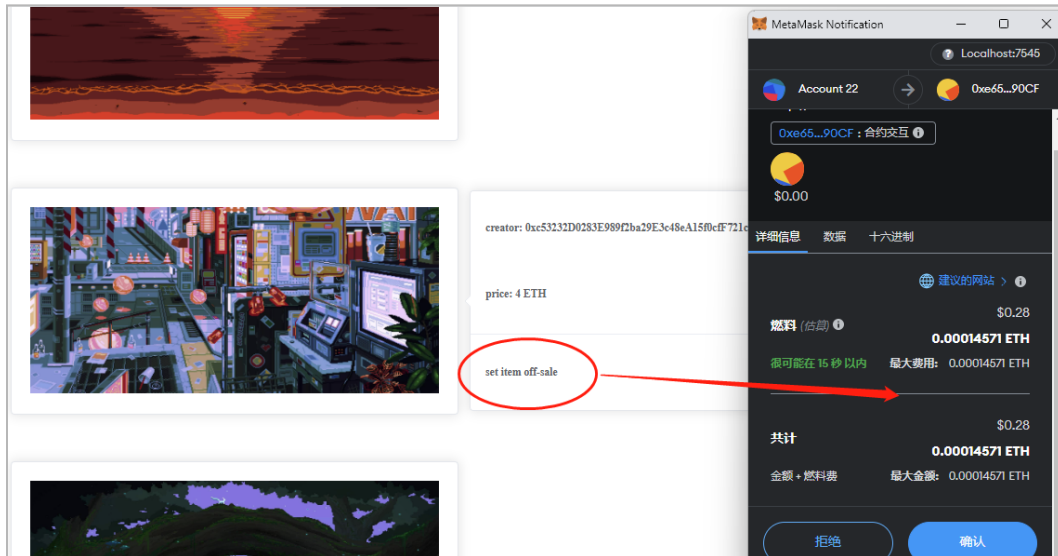


Fig 3.7 Set created Nft off-sale on UserInformation page

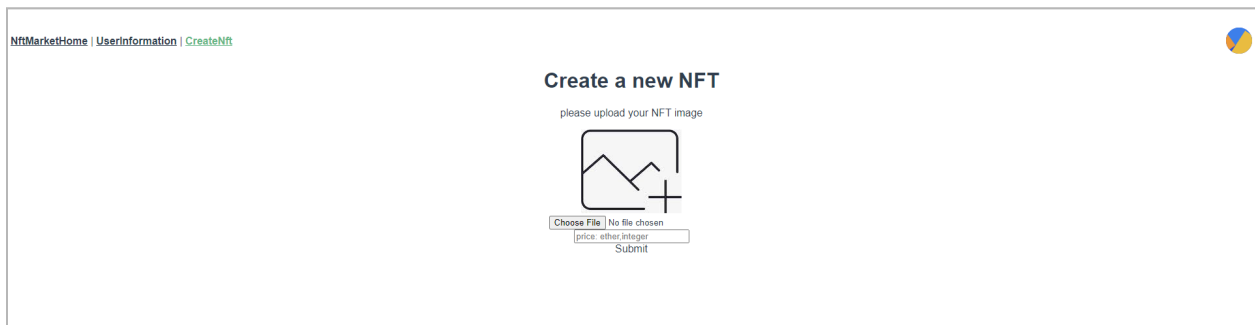



Fig 3.8 Create a new NFT on the CreateNFT page

Create a new NFT

please upload your NFT image

选择文件

plus.png



Withdraw

5

Submit

Fig 3.9 Create a new NFT after uploading a picture


[NftMarketHome](#) | [UserInformation](#) | [CreateNft](#)

Create a new NFT

please upload your NFT image

选择文件

plus.png



Withdraw

5

Submit

Account 5

0xd13...A371

http://localhost:8080

正在发送 ETH

0.025

详细信息

数据

十六进制

建议的网站 >

0.00009808

燃料 (估计)

0.00009808 ETH

很可能在 15 秒 以内

最大费用: 0.00009808 ETH

共计

0.02509808

0.02509808 ETH

金额 + 燃料费

最大金额: 0.02509808 ETH

拒绝

确认

Fig 3.10 Create a new NFT after submitting