NFT Marketplace with ERC721

Haobo Liu (hl3645), Haiwen Zhang (hz2846)

1 Summary of Work

1.1 Introduction

In our project, we implemented a version of NFT Marketplace using smart contract development under the standard of ERC-721. This development helps us to understand the development process of smart contracts in Ethereum, as well as the NFT minting and trading process. For the end results, we successfully developed a version of NFT marketplace and deployed it to the test network. Our smart contract was able to allow users (with account) to mint, trade, and sell their NFTs. In the rest of this report, we will cover some key points in detail.

1.2 Procedure and Methodology

After carefully evaluating the requirements for Project Topic 1: NFT Marketplace Smart Contract Development, we decided to focus on ERC-721 in our project, since it is a more typical token standard that only deals with Non-Fungible Token. To plan out this project, we followed the guide for the test plan to first write tests about our development, which led us to define the necessary functions needed in our smart contract: mint, list, buy, and cancel. And then we started to create the smart contract with needed functions.

Meanwhile, while we researched on the ERC-721 standard, we found out it is a open script written in solidity that covers the basic behaviors and data types of NFT by *OpenZeppelin*. After another round of research, we discovered importing and referencing ERC721 in NFT Marketplace smart contracts is a common practice among many examples we saw. Therefore, we installed @openzeppelin/ERC-721 in our project dependency and leveraging some pre-defined functions to help our development.

1.3 Results

In the end of our development, we made a NFT Marketplace smart contract that allows users to mint NFT, list NFT for sale, cancel listing, and transfer NFT. All those functionalities follow the requirements of Ethereum environment, and more importantly, ERC-721 standard. To use our Smart Contract, we developed our smart contract into the Test Network. Users can access our smart contract using the Ethereum enabled console.

2 Work Showcase

Series of Screenshots of a user trying to use our smart contract following the behaviors in Test.
We are using two accounts to showcase.
Account 1: 0xA15d6562217EDCb3CC044C4e35A2125136cBa8aE
Account2: 0xb824bD108f3d843D88c134F8366f57ec64DCF785

We are using account 1 to mint an NFT with token ID 1 and list it with price 1 wei and listing ID 1. Then, we are going to use the second account to buy the token.

```
truffle(development)> nft.mint('0xA15d6562217EDCb3CC044C4e35A2125136cBa8aE',1, "name of 1", "description of 1")
{
  tx: '0xa3eac74609427f247c4655fd93cf5cce603c46dc6825730a10d0f19047d40fe8',
  receipt: {
    transactionHash: '0xa3eac74609427f247c4655fd93cf5cce603c46dc6825730a10d0f19047d40fe8',
    transactionIndex: 0,
    blockNumber: 5,
    blockHash: '0x98e09ba9d9ec88a6f3d1168d95cbdf7a6eace14b8dc29e59a6fa87081872e894',
    from: '0xa15d6562217edcb3cc044c4e35a2125136cba8ae',
    to: '0x26c595934c6052309e75f11465e2ba385cce1404',
    cumulativeGasUsed: 118407,
    gasUsed: 118407,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000004000000000000000000000000000800000000000080000004000000000000000000000000000002000000000000000000080000000000000000
00000010100000000000000000080000000000000000000000000000000000000000000000000000000000000000000080020000000000000000000000000000
00000000000000000000000002000000000001000000000000000000000000000000000000000006000000000000000000000000000000000000000000000000
000000000000000000',
    status: true,
    effectiveGasPrice: 3108705269,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0x26c595934c6052309e75F11465e2BA385cCe1404',
      blockHash: '0x98e09ba9d9ec88a6f3d1168d95cbdf7a6eace14b8dc29e59a6fa87081872e894',
      blockNumber: 5,
      logIndex: 0,
      removed: false,
      transactionHash: '0xa3eac74609427f247c4655fd93cf5cce603c46dc6825730a10d0f19047d40fe8',
      transactionIndex: 0,
      id: 'log_f8d8db36',
      event: 'Transfer',
      args: [Result]
    }
  ]
}
```

```
truffle(development)> nft.getTokenData(1)
[
  'name of 1',
  'description of 1',
  name: 'name of 1',
  description: 'description of 1'
]
```

If we try to mint another NFT with the same token ID, we will get the following error.

```
truffle(development)> nft.mint('0xA15d6562217EDCb3CC044C4e35A2125136cBa8aE',1, "name of 1", "description of 1")
Uncaught:
Error: VM Exception while processing transaction: revert Token already exists -- Reason given: Token already exists.
```

```
truffle(development)> nft.listToken(1, 1, 1)
{
  tx: '0x566ae3706e5105d24cc58b03843810645b8aaabb4a39a9cc4888dab278fa9d00',
  receipt: {
    transactionHash: '0x566ae3706e5105d24cc58b03843810645b8aaabb4a39a9cc4888dab278fa9d00',
    transactionIndex: 0,
    blockNumber: 6,
    blockHash: '0x55d84d74352d9c9f5b17fc33cd82d45e1036504a5cd4f1a95967d3672d043b9f',
    from: '0xa15d6562217edcb3cc044c4e35a2125136cba8ae',
    to: '0x26c595934c6052309e75f11465e2ba385cce1404',
    cumulativeGasUsed: 122713,
    gasUsed: 122713,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000401000000000000000000000000008000000000000008000000400000000000000000000000000000000000000000000000000000000
000001010000000000000000000800000000000000000000000000000000000000000000000000000000000008002000000000000000000000000000
000000000000000000000000200000000000010000000020000000000004000000000000000000040000000000000000000000000000000000000000
000000000000000000',
    status: true,
    effectiveGasPrice: 3035297684,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0x26c595934c6052309e75F11465e2BA385cCe1404',
      blockHash: '0x55d84d74352d9c9f5b17fc33cd82d45e1036504a5cd4f1a95967d3672d043b9f',
      blockNumber: 6,
      logIndex: 0,
      removed: false,
      transactionHash: '0x566ae3706e5105d24cc58b03843810645b8aaabb4a39a9cc4888dab278fa9d00',
      transactionIndex: 0,
      id: 'log_b42a4fb6',
      event: 'Transfer',
      args: [Result]
    }
  ]
}
```

```
truffle(development)> nft.getListing(1)
[
  '0',
  '0xA15d6562217EDCb3CC044C4e35A2125136cBa8aE',
  '1',
  '1',
  status: '0',
  seller: '0xA15d6562217EDCb3CC044C4e35A2125136cBa8aE',
  tokenId: '1',
  price: '1'
]
```

```
truffle(development)> nft.cancel(1)
{
  tx: '0x2ee2c94854f1f2c3d337c80aaebce170e8d82fb294eb555f9b7d74aef7d8b1e0',
  receipt: {
    transactionHash: '0x2ee2c94854f1f2c3d337c80aaebce170e8d82fb294eb555f9b7d74aef7d8b1e0',
    transactionIndex: 0,
    blockNumber: 7,
    blockHash: '0x38a8fe88547d57442ad93cc62e280f1d965d4e27b45c2c8e721a87460e1af359',
    from: '0xa15d6562217edcb3cc044c4e35a2125136cba8ae',
    to: '0x26c595934c6052309e75f11465e2ba385cce1404',
    cumulativeGasUsed: 64515,
    gasUsed: 64515,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000004010000000000000000000000000800000000000080000004000000000000000000000000000000000000000000000000000000000000000000
00000010100000000000000000080000000000000000000000000000000000000000000000000000000000000000000000008002000000000000000000000000
00000000000000000000000002000000000001000000002000000000004000000000000000000040000000000000000000000000000000000000000000000000
000000000000000000',
    status: true,
    effectiveGasPrice: 2970828506,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0x26c595934c6052309e75F11465e2BA385cCe1404',
      blockHash: '0x38a8fe88547d57442ad93cc62e280f1d965d4e27b45c2c8e721a87460e1af359',
      blockNumber: 7,
      logIndex: 0,
      removed: false,
      transactionHash: '0x2ee2c94854f1f2c3d337c80aaebce170e8d82fb294eb555f9b7d74aef7d8b1e0',
      transactionIndex: 0,
      id: 'log_354aed9b',
      event: 'Transfer',
      args: [Result]
    }
  ]
}
```

After removing the NFT from sale using the cancel function. The status of the listing will be changed to 2 (CANCELED). Now, other accounts won't be able to purchase the NFT.

```
truffle(development)> nft.getListing(1)
[
  '2',
  '0xA15d6562217EDCb3CC044C4e35A2125136cBa8aE',
  '1',
  '1',
  status: '2',
  seller: '0xA15d6562217EDCb3CC044C4e35A2125136cBa8aE',
  tokenId: '1',
  price: '1'
]
```

Now, we are using the second account to purchase the token.

```
truffle(development)> nft_.buyToken(1, { value: 0 });
Uncaught:
Error: VM Exception while processing transaction: revert Payment has to the same as price -- Reason given: Payment has to the sa
me as price.
```

```
truffle(development)> nft_.buyToken(1, { value: 1 });
{
  tx: '0x7052c6f25456cb05cd22309926bc8cbdd8ab2d08005cc1438e46e5cf8652953c',
  receipt: {
    transactionHash: '0x7052c6f25456cb05cd22309926bc8cbdd8ab2d08005cc1438e46e5cf8652953c',
    transactionIndex: 0,
    blockNumber: 9,
    blockHash: '0xd5c45a32a5f73b1a0be879bf80919b0ffc08c4f1ef83fd292c27cbee454ca82d',
    from: '0xb824bd108f3d843d88c134f8366f57ec64dcf785',
    to: '0x26c595934c6052309e75f11465e2ba385cce1404',
    cumulativeGasUsed: 76183,
    gasUsed: 76183,
    contractAddress: null,
    logs: [ [Object] ],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000004010000000000000000000000000800000000000080000004000000000000000000000000000000000000000000000000000040000000000000000
00000020100000000000000000080000000000000000000000000000000000000000200000000000000000000000000000008000000000000000000000000000
00000000000000000000000002000000000000000000002000000000004000000000000000000040000000000000000000000000000000000000000000000000
000000000000000000',
    status: true,
    effectiveGasPrice: 2862477720,
    type: '0x2',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0x26c595934c6052309e75F11465e2BA385cCe1404',
      blockHash: '0xd5c45a32a5f73b1a0be879bf80919b0ffc08c4f1ef83fd292c27cbee454ca82d',
      blockNumber: 9,
      logIndex: 0,
      removed: false,
      transactionHash: '0x7052c6f25456cb05cd22309926bc8cbdd8ab2d08005cc1438e46e5cf8652953c',
      transactionIndex: 0,
      id: 'log_298d7ceb',
      event: 'Transfer',
      args: [Result]
    }
  ]
}
```

Finally, let us confirm the owner is now indeed account 2.

```
truffle(development)> let owner = await nft_.ownerOf(1);
undefined
truffle(development)> console.log(owner)
0xb824bD108f3d843D88c134F8366f57ec64DCF785
```

Then, we deployed our smart contract to the sepolia test network.

```
C:\Users\zhang\Desktop\Ethereum_NFT_Market-main\Ethereum_NFT_Market-main>truffle migrate --network sepolia
```

```
Migrations dry-run (simulation)
===============================
> Network name:    'sepolia-fork'
> Network id:      11155111
> Block gas limit: 30000000 (0x1c9c380)


2_deploy_contracts.js
=====================

   Deploying 'Migrations'
   ----------------------
   > block number:        3437954
   > block timestamp:     1683476560
   > account:             0x83B8A7d24006571Ddb257290B42FC47aad5D81cE
   > balance:             0.116725816
   > gas used:            250212 (0x3d164)
   > gas price:           2 gwei
   > value sent:          0 ETH
   > total cost:          0.000500424 ETH


   Deploying 'NFT'
   ---------------
   > block number:        3437955
   > block timestamp:     1683476561
   > account:             0x83B8A7d24006571Ddb257290B42FC47aad5D81cE
   > balance:             0.11042242
   > gas used:            3151698 (0x301752)
   > gas price:           2 gwei
   > value sent:          0 ETH
   > total cost:          0.006303396 ETH


   Deploying 'Market'
```

Then, we briefly test the smart contract by minting a token and listing it.

```
PS C:\Users\zhang\Desktop\Ethereum_NFT_Market-main\Ethereum_NFT_Market-main> truffle console --network sepolia
truffle(sepolia)> NFT.deployed().then((instance) => {nft = instance })
undefined
truffle(sepolia)> nft.mint("0x83B8A7d24006571Ddb257290B42FC47aad5D81cE", 1, "name of token 1", "description of token 1")
{
  tx: '0x12b1e125d31caaa22f2db01b8afdb018b9cecbfb2a923891daff6fe309f947c2',
  receipt: {
    blockHash: '0xf62fe75576a2d25b31fe98f770c0cc3fc8176f903a37b78f3957ded29c71ca70',
    blockNumber: 3438011,
    contractAddress: null,
    cumulativeGasUsed: 762703,
  },
  logs: [
    {
      address: '0x259865E346858F7a2eFe777EDA2547B76783E7E8',
      blockHash: '0xf62fe75576a2d25b31fe98f770c0cc3fc8176f903a37b78f3957ded29c71ca70',
      blockNumber: 3438011,
      logIndex: 1,
      removed: false,
      transactionHash: '0x12b1e125d31caaa22f2db01b8afdb018b9cecbfb2a923891daff6fe309f947c2',
      transactionIndex: 6,
      id: 'log_411e66fd',
      event: 'Transfer',
      args: [Result]
    }
  ]
}
```

```
truffle(sepolia)> nft.getTokenData(1)
[
  'name of token 1',
  'description of token 1',
  name: 'name of token 1',
  description: 'description of token 1'
]
```

```
truffle(sepolia)> nft.listToken(1,1,1)
truffle(sepolia)> {
  tx: '0x108116b0089a33f4dd82c1351a28a6ce428aa4eba94e224d83994c6c4a5f9c83',
  receipt: {
    blockHash: '0x322ca540ae0b0606d515a553b207b3037ab4ac0c1d1bdec803fb856e64d9e5a0',
    blockNumber: 3438019,
    contractAddress: null,
    cumulativeGasUsed: 185713,
    effectiveGasPrice: 2000000000,
    from: '0x83b8a7d24006571ddb257290b42fc47aad5d81ce',
    gasUsed: 122713,
    logs: [ [Object] ],
    logsBloom: '0x0000000000000000000000002000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0040000000000040000000000000008000000000000000000040000000000000000000000000000002000000000000000000000004000000000000000000001000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000020004000000000000000000000000000000020000000
000000000000000000000000000000000000000000004000000000000000000040000000000020000000000000000000000000000000000',
    status: true,
    to: '0x259865e346858f7a2efe777eda2547b76783e7e8',
    transactionHash: '0x108116b0089a33f4dd82c1351a28a6ce428aa4eba94e224d83994c6c4a5f9c83',
    transactionIndex: 3,
    type: '0x0',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      address: '0x259865E346858F7a2eFe777EDA2547B76783E7E8',
      blockHash: '0x322ca540ae0b0606d515a553b207b3037ab4ac0c1d1bdec803fb856e64d9e5a0',
      blockNumber: 3438019,
      logIndex: 0,
```

3 Conclusion

Overall, we think this project helps us to recognize and practice the ideas of smart contract and NFT through developing such smart contracts ourselves. The outcome of this project is an easy-to-use NFT Marketplace that everyone can access through the Test Network with the power of blockchain.

3.1 Team Member Contribution

Haiwen Zhang: Coding of the main functions in NFT.sol, Deploying and Demo-ing of our final product, and Final Report.

Haobo Liu: Test Scripts of the smart contract, Final report, and Plan and research for ERC-721 implementation of NFT Marketplace.

Reference

[1] https://docs.openzeppelin.com/contracts/2.x/api/token/erc721

Appendix

[1] Github: https://github.com/chrisliuuuu/NFT_Marketplace