NFT Marketplace Smart Contract Development based on ERC-721

Project Report

ELEN E6883: An Introduction to Blockchain Technology - Spring 2023

Project members:

Jiawen Liu(jl6337)

Nina Hsu(hh2961)

Yue Rao(yr2425)

Yuyang Wang(yw3912)

Shutong Zhang(sz3101)

Github Repo: https://github.com/liujiawen0905/ELEN6883-FinalProject

1. Summary of our work

In this project, we aimed to develop a robust and secure NFT marketplace smart contract based on the ERC-721 standard, allowing users to mint, buy, sell, and trade unique digital assets represented as NFTs. We followed a systematic approach, achieved our goal, and ensured that the final product met the requirements and provided a seamless user experience.
We conducted a thorough analysis of the current NFT market and identified the critical features and functionalities that users expect in a modern NFT marketplace. This analysis helped us gain valuable insights into user preferences, security concerns, and other crucial factors that would guide our development process.

2. Procedure and Methodology

Set up the development environment by installing the necessary tools: Visual Studio Code, Node.js, Truffle, Ganache, and MetaMask.
Created a new Truffle project and designed the NFT marketplace smart contract, focusing on understanding and implementing the ERC-721 standard. We chose ERC-721 over the ERC-1155 standard due to its wide adoption and compatibility with various platforms and wallets.
Defined the structure of the smart contract, which includes variables for storing NFT information (name, description, and URI), an array for storing token IDs, and mappings for token ownership, token approvals, and token listings.
Implemented minting, ownership, and metadata functions within the smart contract:

a. The minting function allows users to create new NFTs with unique IDs, names, descriptions, and metadata stored on Web3 API.

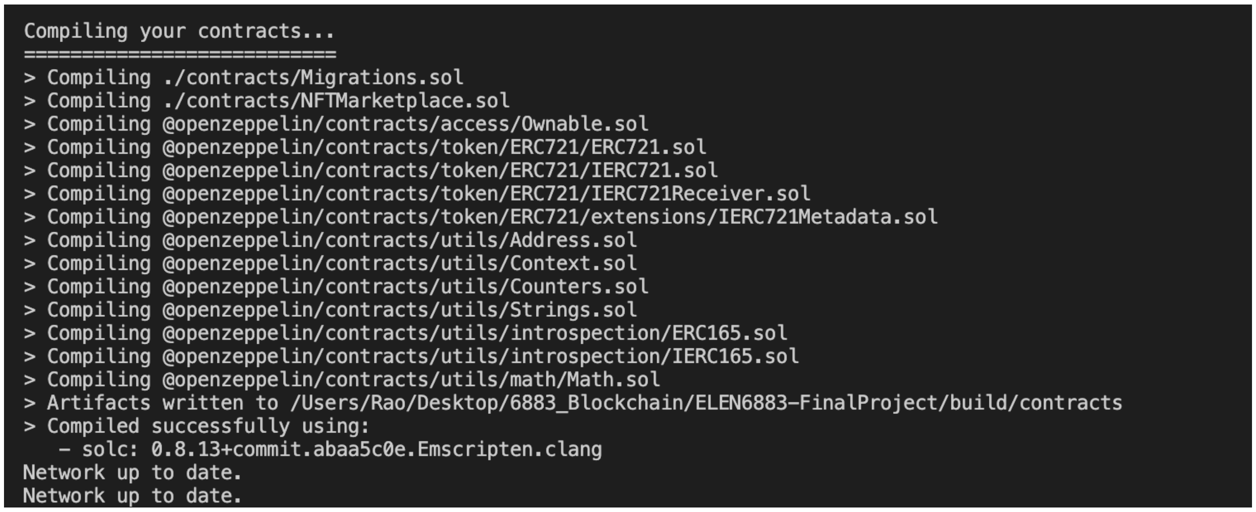
b. The ownership function allows for transferring NFTs between users and managing the approval process for third-party transfers.

c. The metadata function retrieves the NFT's metadata from Web3 API, providing a decentralized storage solution for digital assets.

Deployed the smart contract to a test network (Rinkeby) and tested the contract functionalities using various test cases, such as creating new NFTs, transferring ownership, listing NFTs for sale, removing NFTs from sale, and executing successful and unsuccessful purchases.

3. Showcase of our work

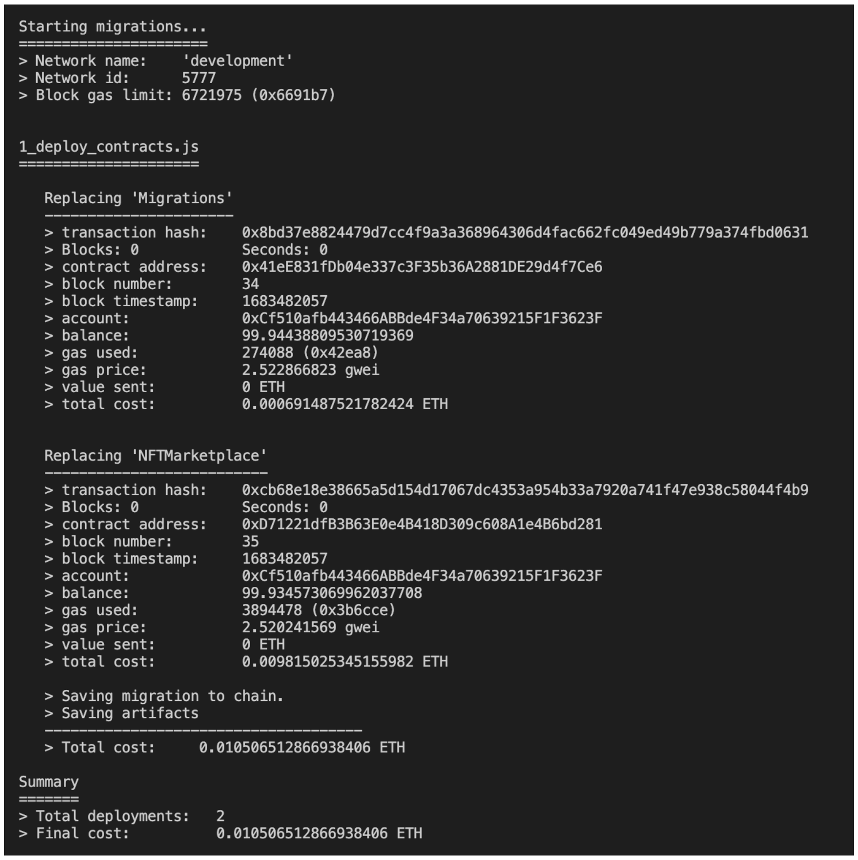
Designing the NFT Marketplace Smart Contract:



```
Compiling your contracts...
===========================
> Compiling ./contracts/Migrations.sol
> 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/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 /Users/Rao/Desktop/6883_Blockchain/ELEN6883-FinalProject/build/contracts
> Compiled successfully using:
   - solc: 0.8.13+commit.abaa5c0e.Emscripten.clang
Network up to date.
Network up to date.
```

Figure 1: Compiling the Smart Contract

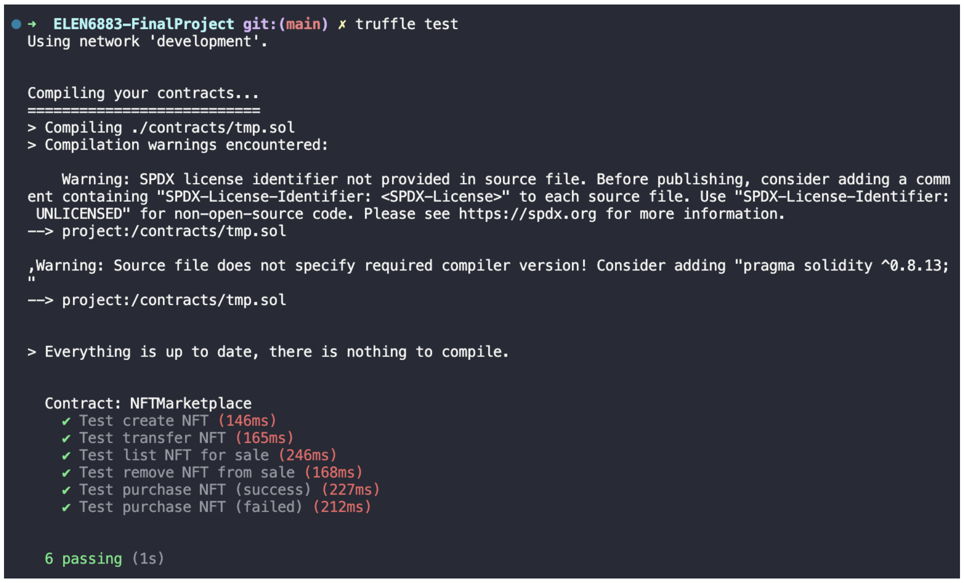
Deploying the NFT Marketplace:



```
Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_deploy_contracts.js
=====================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x8bd37e8824479d7cc4f9a3a368964306d4fac662fc049ed49b779a374fbd0631
   > Blocks: 0            Seconds: 0
   > contract address:    0x41eE831fDb04e337c3F35b36A2881DE29d4f7Ce6
   > block number:        34
   > block timestamp:     1683482057
   > account:             0xCf510afb443466ABBde4F34a70639215F1F3623F
   > balance:             99.94438809530719369
   > gas used:            274088 (0x42ea8)
   > gas price:           2.522866823 gwei
   > value sent:          0 ETH
   > total cost:          0.000691487521782424 ETH


   Replacing 'NFTMarketplace'
   --------------------------
   > transaction hash:    0xcb68e18e38665a5d154d17067dc4353a954b33a7920a741f47e938c58044f4b9
   > Blocks: 0            Seconds: 0
   > contract address:    0xD71221dfB3B63E0e4B418D309c608A1e4B6bd281
   > block number:        35
   > block timestamp:     1683482057
   > account:             0xCf510afb443466ABBde4F34a70639215F1F3623F
   > balance:             99.934573069962037708
   > gas used:            3894478 (0x3b6cce)
   > gas price:           2.520241569 gwei
   > value sent:          0 ETH
   > total cost:          0.009815025345155982 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.010506512866938406 ETH

Summary
=======
> Total deployments:   2
> Final cost:          0.010506512866938406 ETH
```

Figure 2: Deploying the Marketplace

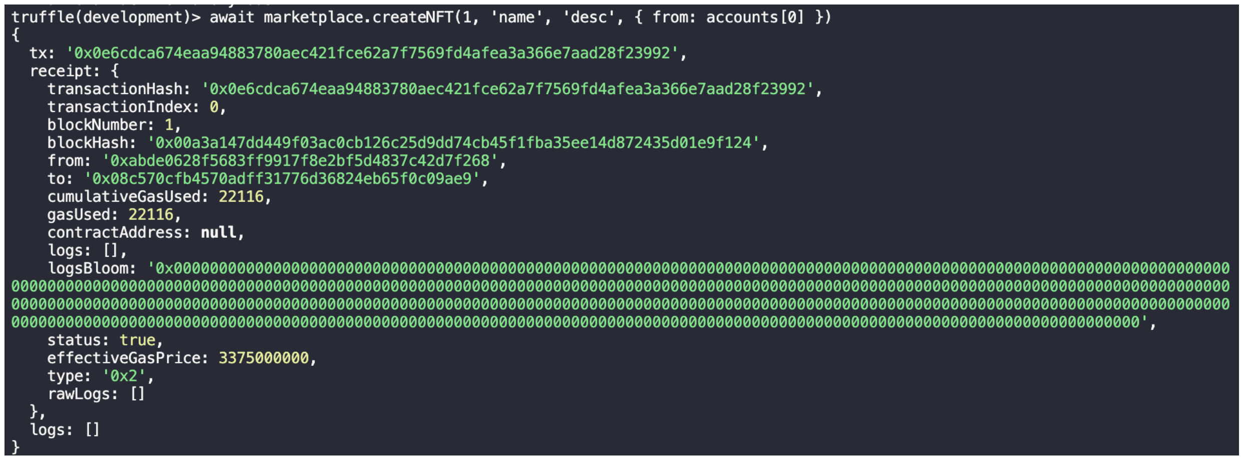
Test cases for the NFT marketplace smart contracts:



```
● ➜  ELEN6883-FinalProject git:(main) ✗ truffle test
  Using network 'development'.


  Compiling your contracts...
  ===========================
  > Compiling ./contracts/tmp.sol
  > Compilation warnings encountered:

      Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comm
  ent containing "SPDX-License-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier:
   UNLICENSED" for non-open-source code. Please see https://spdx.org for more information.
  --> project:/contracts/tmp.sol

  ,Warning: Source file does not specify required compiler version! Consider adding "pragma solidity ^0.8.13;
  "
  --> project:/contracts/tmp.sol


  > Everything is up to date, there is nothing to compile.


    Contract: NFTMarketplace
      ✔ Test create NFT (146ms)
      ✔ Test transfer NFT (165ms)
      ✔ Test list NFT for sale (246ms)
      ✔ Test remove NFT from sale (168ms)
      ✔ Test purchase NFT (success) (227ms)
      ✔ Test purchase NFT (failed) (212ms)


    6 passing (1s)
```

Figure 3: Testing the NFT marketplace smart contracts

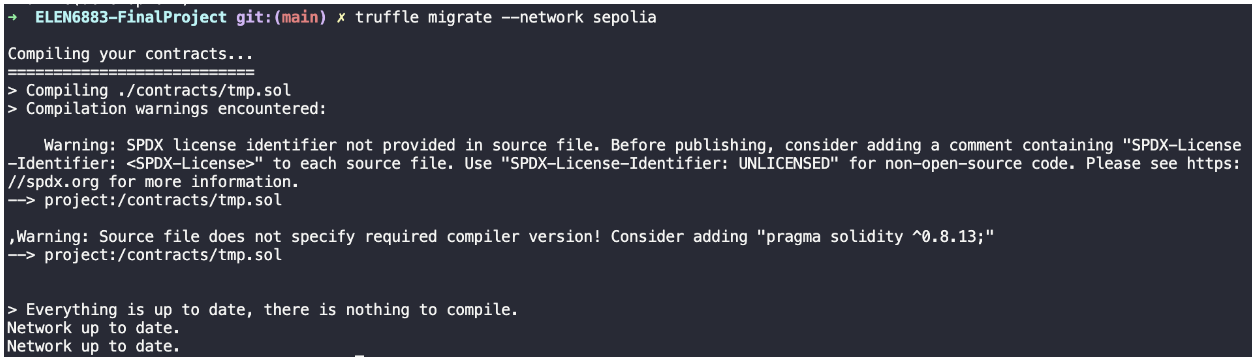
Create NFT with id 1:



```
truffle(development)> await marketplace.createNFT(1, 'name', 'desc', { from: accounts[0] })
{
  tx: '0x0e6cdca674eaa94883780aec421fce62a7f7569fd4afea3a366e7aad28f23992',
  receipt: {
    transactionHash: '0x0e6cdca674eaa94883780aec421fce62a7f7569fd4afea3a366e7aad28f23992',
    transactionIndex: 0,
    blockNumber: 1,
    blockHash: '0x00a3a147dd449f03ac0cb126c25d9dd74cb45f1fba35ee14d872435d01e9f124',
    from: '0xabde0628f5683ff9917f8e2bf5d4837c42d7f268',
    to: '0x08c570cfb4570adff31776d36824eb65f0c09ae9',
    cumulativeGasUsed: 22116,
    gasUsed: 22116,
    contractAddress: null,
    logs: [],
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
    status: true,
    effectiveGasPrice: 3375000000,
    type: '0x2',
    rawLogs: []
  },
  logs: []
}
```

Figure 4: Example test result in truffle console

Deployed our smart contracts to sepolia test network:



```
➜  ELEN6883-FinalProject git:(main) ✗ truffle migrate --network sepolia

Compiling your contracts...
===========================
> Compiling ./contracts/tmp.sol
> Compilation warnings encountered:

    Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License
-Identifier: <SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https:
//spdx.org for more information.
--> project:/contracts/tmp.sol

,Warning: Source file does not specify required compiler version! Consider adding "pragma solidity ^0.8.13;"
--> project:/contracts/tmp.sol


> Everything is up to date, there is nothing to compile.
Network up to date.
Network up to date.
```

Figure 5: Sepolia migration result

We have provided a sample user experience below:

1. User A mints an NFT with a unique ID, name, and description. The NFT is then added to their account, and the metadata associated with the NFT is stored on the blockchain.
2. User A lists their NFT for sale with a specified price. The NFT is now visible on the marketplace, and other users can view and purchase the NFT.

3. User B browses the marketplace and finds User A's NFT. They purchase the NFT using the required amount of Ether, transferring ownership of the NFT to User B.
4. User A receives the Ether from the sale, and User B now owns the NFT. The transaction history of the NFT is updated on the blockchain.

4. Conclusion

Our team successfully developed an NFT marketplace smart contract based on the ERC-721 standard, which allows users to mint, buy, sell, and trade unique digital assets in a secure and efficient manner. The functions are correct, secure, and efficient.

5. Team member contributions

Jiawen Liu: Development environment setup, smart contract design, and testing.

Yuyang Wang, Shutong Zhang: Implementation of minting, ownership, and metadata functions.

Nina Hsu: Smart contract deployment and test case development.

Yue Rao: Project management, documentation, and quality assurance.