DoggyDigiArt: An ERC-721-based NFT Marketplace
ELEN E6883: An Introduction to Blockchain Technology

Han Yang (hy2759), Ziyi Xuan (zx2420), Zhiyuan Liu (zl3208),
Minghui Zhao (mz2866), Shen Gao (sg4140)
Electrical Engineering Department, Columbia University

May 8, 2023

1 Introduction

Non-fungible tokens (NFTs) have emerged as a groundbreaking technology in the world of digital assets, addressing the challenges of scarcity and provenance by assigning unique identifiers to distinguish each asset. This innovative approach has opened up a plethora of possibilities across various industries, including art, gaming, and collectibles, by allowing for secure and verifiable ownership of digital content[1]. This paper presents a case study on the development and analysis of an ERC-721-based NFT marketplace. The study focuses on designing and implementing a secure and efficient smart contract capable of creating and trading non-fungible tokens (NFTs). The proposed smart contract provides a reliable platform where users can mint, buy, sell, and trade NFTs securely and efficiently. The technical aspects of the implementation are outlined, and the marketplace is demonstrated. The findings show that the proposed smart contract is highly reliable in managing NFT transactions, making it a promising platform for future NFT market development. You can find the code for this project on our **GitHub** page: `https://github.com/lzy2022/FinalProject_NFTmarketPlace_6883`

2 Summary of Work

2.1 Objective and Scope

In order to develop a comprehensive understanding of the technical aspect behind the NFT marketplaces, we designed a decentralized marketplace named "DoggyDigiArt". The platform was designed to allow artists to mint and list their dog-related artwork while providing a secure and efficient platform for collectors to browse, bid on, or purchase these pieces. The marketplace was built on the Ethereum network using the ERC-721 standard for non-fungible tokens. It features a user-friendly layout and guarantees a smooth experience for both artists and collectors.

2.2 ERC-721 Standard

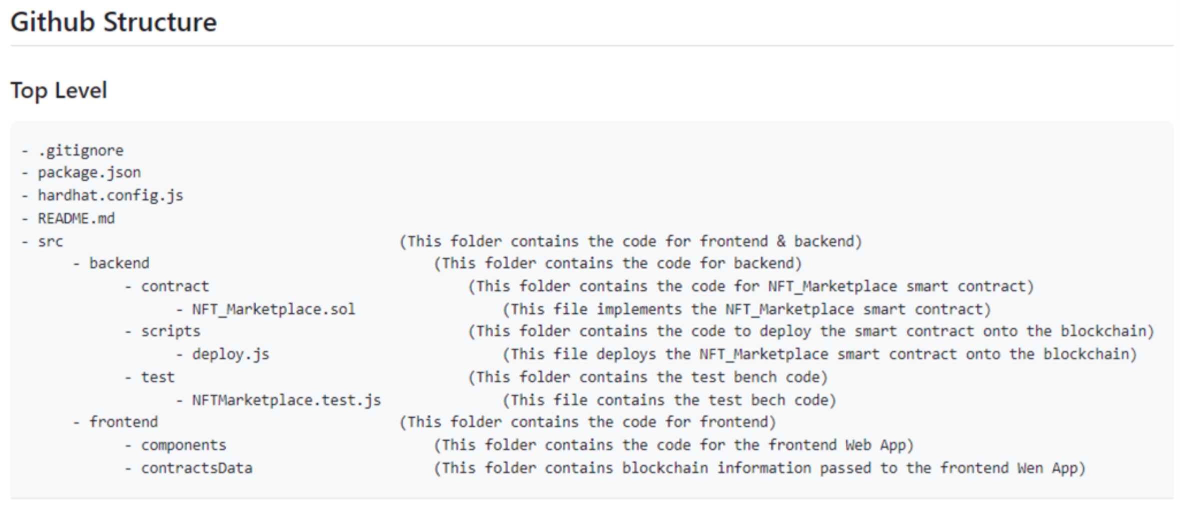
The ERC-721 standard is a crucial component of any NFT marketplace as it enables the creation, transfer, and management of non-fungible tokens (NFTs) on the Ethereum blockchain[2]. These tokens represent unique digital assets such as digital art and collectibles. Compliance with the ERC-721 standard allows developers to create, manage, and trade these tokens, ensuring that each token is one-of-a-kind and cannot be replicated or replaced by another token. In the smart contract proposed for our marketplace, we have designed several functions including createNFT(), listNFTForSale(), removeNFTFromSale(), transferNFT(), and purchaseNFT() which enable users to manage the ownership and sale status of their tokens [3].

- **createNFT()**: The function allows users to mint a new NFT with specified metadata, including the token URI, named ID, name, and description, in order to create a unique digital asset.
- **listNFTForSale()**: Users can put their NFT up for sale by specifying the token ID and desired price, making the NFT available for others to purchase on the marketplace.

- **removeNFTFromSale()**: This function enables NFT owners to remove their token from sale by providing the token ID, effectively cancelling the listing and retaining ownership.
- **transferNFT()**: Token owners can transfer the ownership of their NFT directly to another user by specifying the token ID and the recipient's address, bypassing the marketplace sale process.
- **purchaseNFT()**: Users can buy a listed NFT by providing the token ID and sending the required amount of Ether, which transfers the ownership of the NFT and completes the transaction.

The project comes with a hardhat test bench code at
`\FinalProject_NFTmarketPlace_6883-main\src\backend\test\NFTMarketplace.test.js`
This test bench is built using hardhat local testing blockchain. The code can be run with the command `npx hardhat test`. It would deploy the smart contract to the testing blockchain and test the smart contract using the following test cases:

- **Deployment**: Track the name and symbol of the NFT Marketplace smart contract.
- **Creating NFT**: Track the created NFTs then check their ownership and properties.
- **Transfering NFT without price**:
  - Should transfer NFT1 from User1 to User2
  - Should fail if User1 does not own this NFT
- **User lists an NFT for sale**:
  - Should put the NFT on sale, emit an Offered event
  - Should fail if price is set to zero
  - Should fail if the item is already on sale
- **Recalling an listed NFT (remove from sale)**:
  - Should let User1 recall NFT1
  - Should fail if User1 does not own this NFT
- **Purchasing marketplace items**:
  - Should update item as sold, pay seller, transfer NFT to buyer, charge fees, and emit a Bought event
  - Should fail for invalid item ids, sold items and when not enough ether is paid

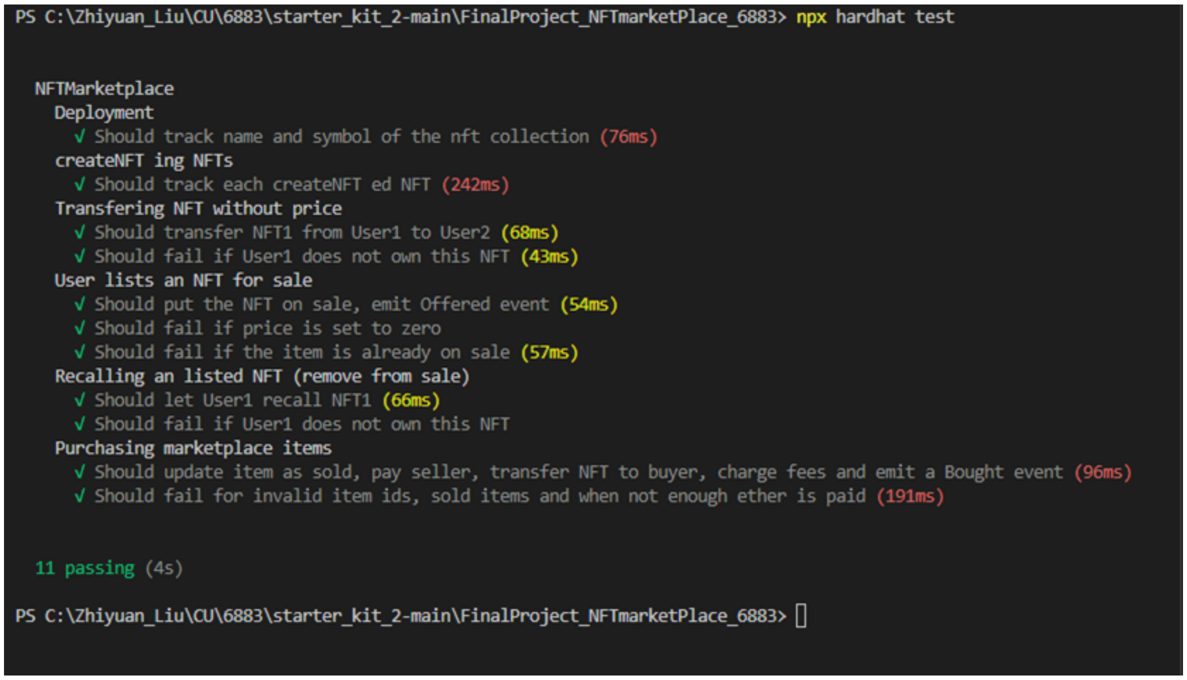


Github Structure

Top Level

```
- .gitignore
- package.json
- hardhat.config.js
- README.md
- src                                (This folder contains the code for frontend & backend)
      - backend                          (This folder contains the code for backend)
            - contract                       (This folder contains the code for NFT_Marketplace smart contract)
                  - NFT_Marketplace.sol          (This file implements the NFT_Marketplace smart contract)
            - scripts                        (This folder contains the code to deploy the smart contract onto the blockchain)
                  - deploy.js                    (This file deploys the NFT_Marketplace smart contract onto the blockchain)
            - test                           (This folder contains the test bench code)
                  - NFTMarketplace.test.js       (This file contains the test bech code)
      - frontend                     (This folder contains the code for frontend)
            - components                 (This folder contains the code for the frontend Web App)
            - contractsData              (This folder contains blockchain information passed to the frontend Wen App)
```

Figure 1: Project Structure



```
PS C:\Zhiyuan_Liu\CU\6883\starter_kit_2-main\FinalProject_NFTmarketPlace_6883> npx hardhat test


  NFTMarketplace
    Deployment
      √ Should track name and symbol of the nft collection (76ms)
    createNFT ing NFTs
      √ Should track each createNFT ed NFT (242ms)
    Transfering NFT without price
      √ Should transfer NFT1 from User1 to User2 (68ms)
      √ Should fail if User1 does not own this NFT (43ms)
    User lists an NFT for sale
      √ Should put the NFT on sale, emit Offered event (54ms)
      √ Should fail if price is set to zero
      √ Should fail if the item is already on sale (57ms)
    Recalling an listed NFT (remove from sale)
      √ Should let User1 recall NFT1 (66ms)
      √ Should fail if User1 does not own this NFT
    Purchasing marketplace items
      √ Should update item as sold, pay seller, transfer NFT to buyer, charge fees and emit a Bought event (96ms)
      √ Should fail for invalid item ids, sold items and when not enough ether is paid (191ms)


  11 passing (4s)

PS C:\Zhiyuan_Liu\CU\6883\starter_kit_2-main\FinalProject_NFTmarketPlace_6883> 
```

Figure 2: Test bench results

2.3 Implementation Process and Methodology

After ensuring compliance with the ERC-721 standard, we implemented the smart contract focusing on security and extensibility using **Solidity**[4] and the **OpenZeppelin** library[3]. Our primary development environment was **Visual Studio Code**[5], while **Node.js**[6] and **Hardhat**[7] facilitated contract compilation, migration, and testing. **Ganache**[8], a personal blockchain platform for Ethereum development, provided a local blockchain for simulating transactions and smart contract interactions.

For the user interface, the front-end was built using JavaScript libraries, **React.js**[9] and **Web3.js**[10], to facilitate the integration between the user interface and the smart contract. React.js manages the dynamic rendering of web applications, and Web3.js enables developers to interact with the Ethereum blockchain, allowing the integration of smart contracts and decentralized applications with web-based user interfaces.

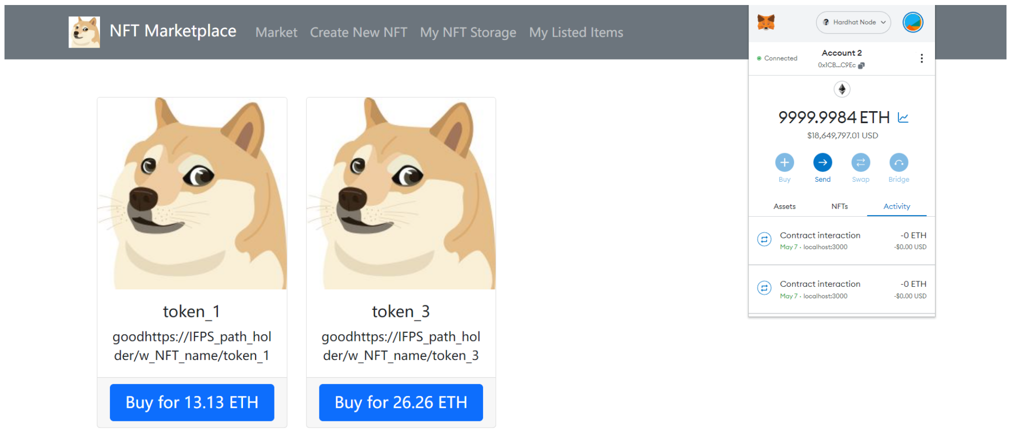
2.4 Results and Evaluation

The DoggyDigiArt marketplace successfully showcased the potential of an efficient and user-friendly platform for trading unique digital assets. The ERC-721 standard for NFTs was demonstrated, particularly in terms of asset uniqueness and ease of integration with the Ethereum network. Throughout the development process, we emphasized creating a visually appealing layout, resulting in a smooth experience for both artists and collectors.

3 Showcase of Work

3.1 NFT Marketplace Overview

As illustrated in Figure 3, the DoggyDigiArt marketplace provides functions for both artists and buyers. Once logged into your personal account, you can view all the listed artworks and the respective prices on the main page. Additionally, you can upload your own work by selecting the "Create Your NFT" option at the top of the page. The "My NFT Storage" option allows you to access the works that you own. By clicking on your profile picture in the upper right corner, you can view your personal information and wallet details at any time.



token_1
goodhttps://IFPS_path_holder/w_NFT_name/token_1

token_3
goodhttps://IFPS_path_holder/w_NFT_name/token_3

9999.9984 ETH
$18,649,797.01 USD

Contract interaction -0 ETH
May 7 · localhost:3000 -$0.00 USD

Contract interaction -0 ETH
May 7 · localhost:3000 -$0.00 USD

Figure 3: Homepage of the DoggyDigiArt marketplace

3.2 User Guide

To access the DoggyDigiArt marketplace, users must follow the steps outlined below:

1. Install the required dependencies:
   - Node.js v16.20.0
   - MetaMask browser extension
2. Run the following commands to set up the environment:
   - `npm install`
   - `npm install --save-dev hardhat@2.8.4`
   - `npm install react-router-dom@6`
   - `npm install ipfs-http-client@56.0.1`
   - `npm i @openzeppelin/contracts@4.5.0`
3. Run the testbench:
   - `npx hardhat test`
4. Set up the GUI:
   - In a terminal, run `npx hardhat node` (this will log the testing accounts and keys)
   - In a new terminal, run `npx hardhat run src\backend\scripts\deploy.js --network localhost`
   - In another new terminal, run `npm run start` (this should open the GUI webpage in the browser)
5. Configure MetaMask:
   - Create and connect to the Hardhat Node using the following setup:
     - Network Name: Hardhat Node
     - New RPC URL: http://127.0.0.1:8545
     - Chain ID: 31337
     - Currency Symbol: ETH
   - Disconnect the blockchain and reconnect if previously connected to the Hardhat Node

- Remove all previous test accounts and import the testing accounts to MetaMask using the corresponding private keys

6. Interact with the platform:

   - In the GUI webpage, click "Connect Wallet" and select the account to connect in Meta-Mask's prompt
   - Create, send, buy, and sell NFTs using the GUI
   - To switch to another testing account, open the "Connected sites" page in the MetaMask account tab, disconnect the localhost page, refresh the webpage, and connect using another account

3.3 Trading Process and Interaction

3.3.1 Token Creation

To create a new NFT token in the DoggyDigiArt marketplace, users need to submit a file and then set the ID, name, and description for the NFT. After confirmation in the Meta-Mask wallet extension, users will have the created NFT stored in their wallet on the DoggyDigiArt marketplace

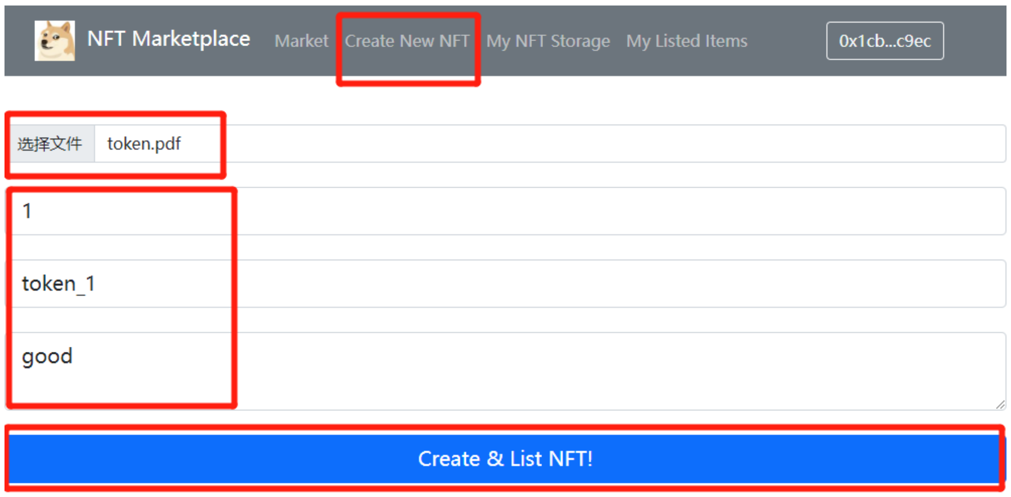


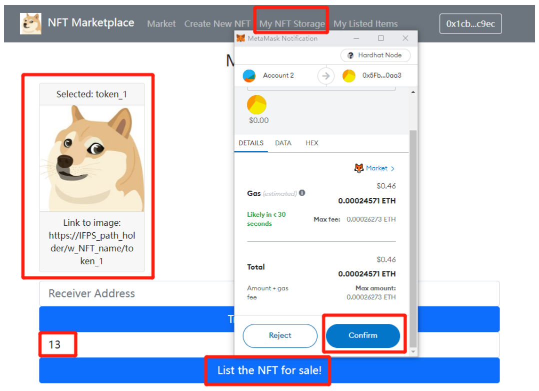
Figure 4: Token Creation

3.3.2 List and Trade NFTs

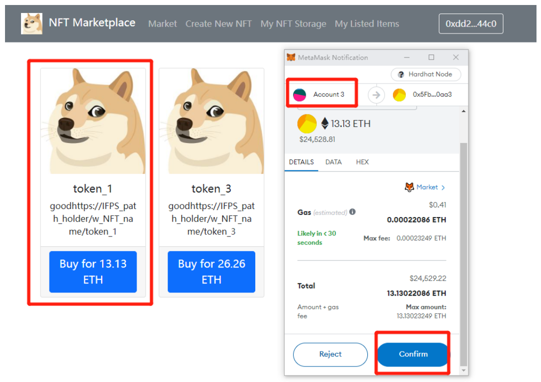
Users can sell or gift an existing NFT by selecting it from their storage page. The selling and buying functions are demonstrated as follows: sellers choose the NFT token from their storage for sale, and the "Name" tag of the selected NFT changes to "Selected". Two confirmations are required from the sellers after they list the NFT and set a price. The NFTs will be available on the DoggyDigiArt marketplace homepage for buyers. Also, buyers are able to both list and recall items in the "My Listed Items" interface. To purchase an NFT, the buyers simply browse the DoggyDigiArt marketplace homepage and click "Buy". The purchasing process is done only after confirming that the buyers have enough ETHs. The purchased NFT token will be displayed in the "My Listed Items" interface, and be removed if the token is sold. (See Figure 5)

3.3.3 Transfer

Users can donate or transfer their NFT tokens by clicking the "Transfer" button, which is shown in Figure 6a. To complete this transaction, users must specify the account address to which the tokens will be transferred. The transferred tokens will then appear in the storage of the selected account.

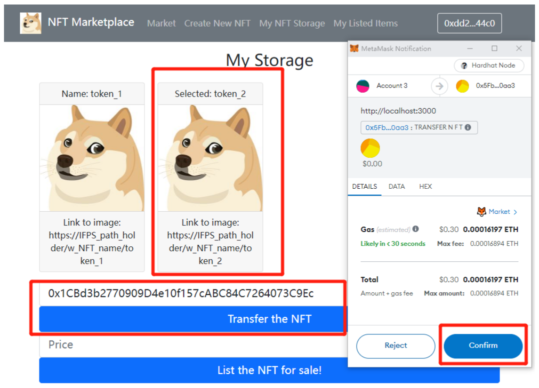


(a) The seller can list the NFTs for sale

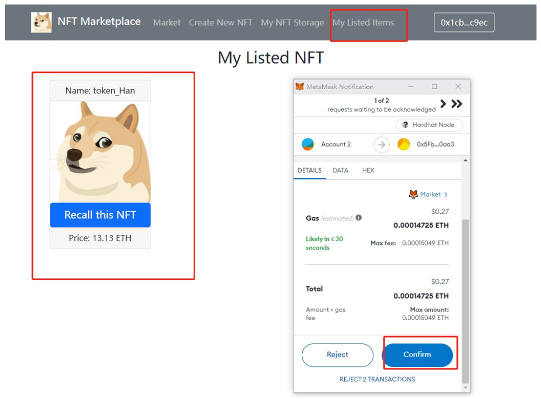


(b) The buyer can purchase the listed NFTs

Figure 5: The homepage of seller and buyer accounts, confirmation is required for all trascations



(a) The user transfers NFT tokens



(b) The user recalls NFT tokens

Figure 6: Transfer and Recall operations in the "My Listed Items" interface

3.3.4 Ownership Management

Users can recall any listed tokens at any time by clicking the "Recall" button under the NFT that needs to be recalled (see Figure 6b). This option is available for unsold NFTs.

4 Conclusion

4.1 Key Findings

The development and analysis of DoggyDigiArt , our ERC-721 based NFT marketplace, successfully demonstrated the potential for creating an efficient and user-friendly platform for trading unique digital assets. Our study revealed that careful attention to user experience and interface design is crucial in driving user adoption and ensuring a positive experience for both artists and collectors. Additionally, the importance of implementing robust security measures and best practices in the smart contract cannot be overstated, as it directly impacts the trustworthiness and overall success of the platform.

4.2 Contributions of Each Team Member

Our team had a clear division of work, and each member **contributed equally** to DoggyDigiArt marketplace. Zhiyuan Liu (zl3208), researched the ERC-721 standard and designed the smart contract, while Han Yang (hy2759) focused on front-end development and user interface design. Minghui Zhao (mz2866) worked on the back-end logic and integrated the smart contract with the user interface. Shen Gao (sg4140) conducted security audits and performance testing to ensure the overall robustness of the platform. Lastly, Ziyi Xuan (zx2420) was responsible for integrating the different components of the project and took the lead in writing the report and coordinating the documentation efforts.

4.3 Future Work and Recommendations

While DoggyDigiArt has shown promising results, there are several potential improvements and expansions that could be explored in future iterations of the project. Some recommendations include:

- Implementing additional features such as NFT royalties, allowing artists to receive a percentage of sales every time their artwork is resold.
- Enhancing the platform's search functionality and incorporating AI-driven recommendations to further personalize user experiences and facilitate content discovery.
- Expanding to other blockchain networks, such as Binance Smart Chain or Polygon, to tap into a wider range of users and reduce transaction costs.

Utilizing what we learned in class, we created an NFT trading platform DoggyDigiArt . Our team gained valuable insights and developed a deeper understanding of the inner workings of the NFT market. We learned about the essential components of an NFT marketplace, such as smart contract design, compliance with the ERC-721 standard, front-end and back-end integration, and security considerations. A key takeaway from this project is the importance of building a secure and efficient smart contract to handle the minting, trading, and ownership transfer of NFTs. This requires careful attention to security best practices and diligent testing to ensure the stability and reliability of the platform.

References

[1] Qin Wang, Rujia Li, Qi Wang, and Shiping Chen. Non-fungible token (nft): Overview, evaluation, opportunities and challenges, 2021.

[2] Seyed Mojtaba Hosseini Bamakan, Nasim Nezhadsistani, Omid Bodaghi, and Qiang Qu. Patents and intellectual property assets as non-fungible tokens; key technologies and challenges. *Scientific Reports*, 12(1):1–13, 2022.

[3] OpenZeppelin. ERC 721 - OpenZeppelin Docs. https://docs.openzeppelin.com/contracts/4.x/api/token/erc721, 2023. Accessed on May 7, 2023.

[4] Ethereum Foundation. Solidity Programming Language. https://soliditylang.org/, 2023. Accessed on May 7, 2023.

[5] Microsoft. Visual Studio Code. https://code.visualstudio.com/, 2023. Accessed on May 7, 2023.

[6] Node.js Foundation. Node.js. https://nodejs.dev/en/learn/, 2023. Accessed on May 7, 2023.

[7] Nomic Foundation. Hardhat. https://hardhat.org/, 2023. Accessed on May 7, 2023.

[8] Truffle Suite. Ganache Documentation. https://trufflesuite.com/docs/ganache/, 2023. Accessed on May 7, 2023.

[9] Facebook. React.js. https://reactjs.org/, 2023. Accessed on May 7, 2023.

[10] Ethereum Foundation. Web3.js Documentation. https://web3js.readthedocs.io/en/v1.8.2/, 2023. Accessed on May 7, 2023.