student # 20032970

supervisor: John Sapsford

School of Computer Science

BSc Computer Science

Computer Science Project - 6WCM0029

Final Project Report

Election dApp

Muhammad Hamza Saquib Malik

2021

Abstract

Blockchain technology provides a unique solution to the problems associated with elections, public blockchains are fully auditable and transparent and can be used to implement the business logic required to run the election as well as be used as a data store.

The public blockchain referenced is Ethereum and business logic is implemented through its programming language Solidity, creating a smart contract which is stored on the blockchain.

The aim of this project is to justify the use of blockchain technology to be used within software responsible to conduct an election as well as develop a deployable smart contract containing all the business logic required to run the election and declare a winner. The front-end of the application aims to aid the individual conducting the election and users in voting and watching the election as it proceeds

Ballots are in the form if non-fungible tokens, created using the ERC-721 interface, in turn allowing anybody to view every aspect of ballot creation and assignment while also maintaining the anonymity of voters.

Contents

[Abstract 1](#_Toc81502634)

[1.0 Project Plan 5](#_Toc81502635)

[1.1 Resources and Skills Required 6](#_Toc81502636)

[1.2 Risks 6](#_Toc81502637)

[1.3 Legal, Social, Ethical, and Professional Issues 7](#_Toc81502638)

[1.4 Analysis 7](#_Toc81502639)

[2.0 Background 9](#_Toc81502640)

[2.1 Introduction 9](#_Toc81502641)

[2.2 Bitcoin and Blockchain Technology 9](#_Toc81502642)

[2.3 Wallets, Transactions & Keys 10](#_Toc81502643)

[2.4 Blocks 10](#_Toc81502644)

[2.5 Mining 10](#_Toc81502645)

[2.5.1 Proof of Work 11](#_Toc81502646)

[2.6 Ethereum 11](#_Toc81502647)

[2.6.1 Smart Contracts 11](#_Toc81502648)

[2.6.2 Ethereum Virtual Machine 12](#_Toc81502649)

[2.6.3 Gas 12](#_Toc81502650)

[2.6.4 ERC-20 Standard Tokens 13](#_Toc81502651)

[2.6.5 ERC-721 Standard – Non-Fungible Token (NFT) 13](#_Toc81502652)

[2.6.6 Decentralized Applications 13](#_Toc81502653)

[2.6.7 Oracles 13](#_Toc81502654)

[2.7 Blockchain Based Election Systems 13](#_Toc81502655)

[3.0 Design and Development 15](#_Toc81502656)

[3.1.1 Methodology 15](#_Toc81502657)

[3.1.2 Version Control 15](#_Toc81502658)

[3.1.3 Libraries Used (Package.JSON) 15](#_Toc81502659)

[3.2 Smart Contract Design 16](#_Toc81502660)

[3.2.1 Development Environment 16](#_Toc81502661)

[3.2.2 OpenZeppelin Library 16](#_Toc81502662)

[3.2.3 Sequence Diagram 17](#_Toc81502663)

[3.2.4 Test Driven Development 18](#_Toc81502664)

[3.2.5 Open Zeppelin ERC-721 19](#_Toc81502665)

[3.2.6 Smart Contract API 20](#_Toc81502666)

[3.3 Front-End Design and Development 22](#_Toc81502667)

[3.3.1 JavaScript 22](#_Toc81502668)

[3.3.2 React 22](#_Toc81502669)

[3.3.3 Material-UI 22](#_Toc81502670)

[3.3.4 Meta Mask 22](#_Toc81502671)

[3.3.5 Use Cases 22](#_Toc81502672)

[3.3.6 Front-End Requirements 23](#_Toc81502673)

[3.3.7 Wireframes 24](#_Toc81502674)

[3.3.8 Developed Web Pages 24](#_Toc81502675)

[3.4 Front-end and Back-end Integration 27](#_Toc81502676)

[4.0 Conclusion 29](#_Toc81502677)

[4.1 Evaluation w.r.t Proposal 29](#_Toc81502678)

[4.2 Critique 29](#_Toc81502679)

[4.3 Personal Growth 29](#_Toc81502680)

[5.0 References 31](#_Toc81502681)

[6.0 Appendix A 33](#_Toc81502682)

[7.0 Appendix B 37](#_Toc81502683)

[8.0 Appendix C 61](#_Toc81502684)

[9.0 Appendix D 65](#_Toc81502685)

[10.0 Appendix E 88](#_Toc81502686)

Figure 1 Overview (Bizzaro & Garcia, 2019) 9

Figure 2 State Update (Takenobu, 2018) 12

Figure 3 Blockchain based Voting Systems (Kshetri & Voas, 2018) 14

Figure 4 Sequence Diagram Smart Contract Deployment and Interaction 17

Figure 5 (ethers.io, 2021) 18

Figure 6 (OpenZeppelin, 2021) ERC-721 Interface 19

Figure 7 Voting Component Wireframe 24

Figure 8 Conclude Component Wireframe 24

Figure 9 Developed Voting Component 24

Figure 10 Developed Conclude Component 24

Figure 11 Minting Component Wireframe 25

Figure 12 Candidate Component Wireframe 25

Figure 13 Developed Minting Component 25

Figure 14 Developed Candidate Component 25

Figure 15 Conclude Component Wireframe 26

Figure 16 Developed Conclude Component 26

Figure 17 Voting Data Flow 27

# Project Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Task** | | **Begin date** | **End date** |
| **Technology Overview** | Literature Review | 02/06/2021 | | 16/06/2021 |
| Solidity Overview | 19/06/2021 | | 28/06/2021 |
| Overview HTML/CSS/JavaScript | 29/06/2021 | | 06/07/2021 |
| Overview React | 07/07/2021 | | 11/07/2021 |
| Overview ethers.js | 12/07/2021 | | 13/07/2021 |
| Solidity Revision | 14/07/2021 | | 14/07/2021 |
| **Milestone** | **Technology Overview Complete** | | | |
| **Back-End Design, Implementation and Testing** | Smart Contract Module Design | 15/07/2021 | | 15/07/2021 |
| Module Development | 16/07/2021 | | 22/07/2021 |
| Module Testing | 23/07/2021 | | 24/07/2021 |
| Module Integration | 25/07/2021 | | 27/07/2021 |
| **Milestone** | **Smart Contract Complete** | | | |
| **Font-End Design Implementation and Testing** | Front-End Design | 28/07/2021 | | 28/07/2021 |
| Font-End Development | 29/07/2021 | | 02/08/2021 |
| Front-End Integration | 03/08/2021 | | 03/08/2021 |
| **Milestone** | **Artefact Complete** | | | |
| **Final Testing and Technical Report Completion** | Application Testing | 04/08/2021 | | 06/08/2021 |
| Technical Report | 01/07/2021 | | 16/08/2021 |
| **Milestone** | **Project Completion** | | | |

## Resources and Skills Required

|  |  |
| --- | --- |
| Planning | The project is headed by a single developer therefore a detailed project plan focusing on daily documentation and progress tracking is critical to the timely delivery of the artefact. A Gantt will be created a daily log of all ongoing and completed tasks will be maintained |
| Documentation | Detailed research and documentation are a major part of the project and therefore there will be a daily task to document all progress. |
| Development | The development will be in solidity using in Visual Studio Code. Ethers.js will be used to read data from the blockchain and data will be displayed on a website created in React. An intermediate understanding of the mentioned technologies is crucial to the success of the project and these skills must be learned before any major design decisions are made and development begins. |
| Version Control | Version control aids a developer by providing a cloud storage solution for storage on any ongoing development as well as the ability to undo or roll back to a previously uploaded version. Version control also allows for affective progress monitoring which is very crucial to meeting deadlines. Version control service used will be GitHub. |
| Hardware | No extra hardware is required for this project, a desktop or laptop for software development and report writing is required. |
| Research Resources | Access to research papers that is provided by the University of Hertfordshire’s online student library as well as free resources offered by Google Scholar. Other free resources accessible on the internet will be referenced and researched. |
| Costs | Everything is already paid for or free, the only running cost being electricity which is minor. No paid software licenses are required. |

## Risks

#### Time Management and Planning

There is adequate time to complete the project and there is a set deadline, improper allocation of time can jeopardize the timely completion of the project. Improper planning and a lack of understanding regarding the required skills can also increase the risk of a missed deadline. To mitigate this, an initial and detailed technology overview will be undertaken and there will be an added focus on documentation and task logging to ensure that the project plan is followed.

#### Data Loss

Data loss due to hardware failure is a risk that has been completely avoided by utilizing a cloud storage solution in addition to version control and local storage.

#### Scope Creep

Scope creep must be minimized, there will be minor additions to the project’s requirements but there will be an extra effort to limit that to only the requirement gathering phase, no further functionalities will be added in later stages unless they are critical to basic functionality.

Pandemic  
There is a single developer and covid-19 exposure could cause delays, although necessary precautions are being taken on a daily basis, a timely proposal for extension will be submitted in case of any symptoms.

## Legal, Social, Ethical, and Professional Issues

#### Legal

In adherence to the Equality Act, there is no discrimination based on any criteria, the software can be used by any individual. A public blockchain provides a system of accountability, all interactions between users and operators with the software will be documented on the blockchain and therefore auditable, accountability in adherence to the Computer Misuse Act is a part of the software’s functional requirements. The body that is distributing ballots is responsible for protecting user data and is outside the scope of this project, the dApp does not save any user’s personal data.

#### Social

Individuals who are not familiar with blockchain technology are at a disadvantage while using the application, familiarizing users with associated risks before casting votes is crucial to an accurate polling result.

#### Ethical

The university requires students to complete the EC1 form and submit it along with supporting evidence, there is no human participation in primary research conducted during this project therefore no associated ethical approval is required. There are no surveys conducted or any other research that required obtaining personal information from human participants. Any literature reviewed will be adequately referenced. The project is solely for academic purposes.

#### Professional

Any contribution to this project will be credited within the final technical report. In adherence to the Intellectual Property Rights, any work part of the literature review will be adequately referenced. Within the source code multiple libraries have been used, in each case, they are allowed for reuse under the MIT license and have been adequately referenced.

## Analysis

#### Essential Functional Requirements (Core):

1. The app is fully functional and tested on an Ethereum test-net.

Testing on the public blockchain is extremely expensive and inefficient therefore a test-net is used instead, it duplicates the functionality of the Ethereum blockchain but without any of the associated costs

1. Ballots will be in the form of Non-Fungible Tokens

Non-fungible tokens cannot be broken down, therefore the ERC-721 interface is ideal for this project

1. The smart contract governs the task of counting votes

Once the smart contract is deployed it is publicly viewable and cannot be altered/removed – even if such functionality is part of the software, it cannot be done discretely since every transaction is public

1. The smart contract computes the winner of the election.

A winner calculating algorithm is part of the smart contract

1. After voting ends the smart contract declares a publicly viewable winner.

The smart contract executes the voting algorithm and declares a winner

1. The conducting address can generate non-fungible voting tokens and send them to voters.

The deployer (owner) of the contract can mint non-fungible voting tokens

#### Advanced Functional Requirements:

1. The conducting body can signal the start and end of the election.

The owner of the contract can start the election by deploying the contract and conclude the election by accessing the conclude function in the smart contract API

1. A website to read and display data from the blockchain as the election proceeds.

A front end application aids users in voting and also displays the progress of the ongoing election

1. The web app has an admin panel to aid the conducting contract in distributing ballots.

#### Non-Functional Requirements

1. Optimized solidity code to reduce gas fee
2. Responsive website
3. Low-loading time and intuitive design

# Background

## Introduction

Elections require transparency and a reliable method to count and authenticate ballots. A public Blockchain provides an immutable data store. Therefore, the validity of a vote cannot be questioned, nor can there be an undetected error in vote calculation. Blockchain technology is well suited to address the problems associated with conducting an election (Hjálmarsson, et al., 2018).  
This research will aim to justify the use of the Ethereum blockchain to conduct an election utilizing the blockchain as a ballot box by investigating how users send and receive transactions, how the transactions are verified and how business logic is implemented in smart contracts running on the Ethereum Virtual Machine.

## Bitcoin and Blockchain Technology

Although the concept of decentralized applications existed, it was far from a workable model until 2008 when Satoshi Nakamoto released the whitepaper for bitcoin. addressing the double-spending problem. He came up with a model that incorporated Byzantine fault tolerance for the consensus algorithm creating the first trust-less system for electronic transactions (Nakamoto, 2008). Figure 1 shows how a transaction is added to the blockchain, in the first step the transaction’s authenticity is verified, in step 2 it is added to the candidate blocked (explained in mining in 2.4) and in step 3 the state of the blockchain is updated on every connected node – recording the transaction.

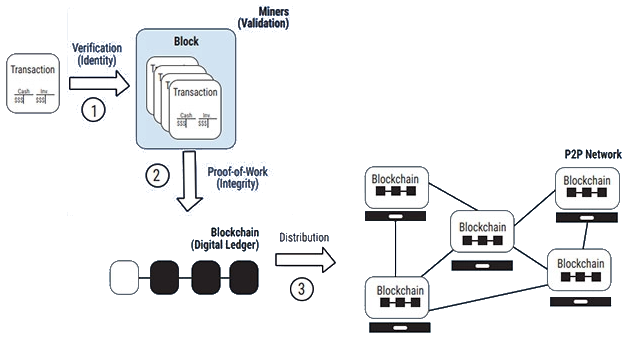
.

Figure 1 Overview (Bizzaro & Garcia, 2019)

Blockchain technology provides a secure and immutable platform using asymmetric cryptography to transfer virtual currency. Users transfer coins signed by their private key to (a hashed version of) the public key of a recipient (Pilkterton, 2016). Blockchain removes the need for a trusted intermediary to do electronic transactions.

## Wallets, Transactions & Keys

Wallets are applications that aid users in sending and receiving cryptocurrency. The user’s private key is saved in the wallet and provides access to any coins associated with it. If the private key is lost, the wallet and its contents are not recoverable. Derived from that private key is a public key that is shared to receive bitcoins – the same model is followed within other cryptocurrencies. Wallets are not running on the blockchain, instead, they simply help users by signing their requests with their private keys and sending transactions to the blockchain on their behalf (Antonopoulos, 2017). When a transaction has initiated the system first verifies its validity and as blocks (section 2.3) are mined consensus is achieved and it is saved on the public ledger or blockchain.

## Blocks

Blocks are unlocked by miners on an interval, each block containing a timestamp, some data, and a cryptographic hash of the previous block – creating a blockchain. Data can only be added to the blockchain, not removed, making it an immutable and append-only data structure (Hjálmarsson, et al., 2018).  
When a user sends some bitcoin to a recipient, after validating the legitimacy of the transaction, it is recorded in a candidate block (Antonopoulos, 2017). As soon as the block is mined the transaction receives one confirmation, as more blocks are mined the number of confirmations is incremented. 6 confirmations are considered enough to ensure the transaction is valid.

## Mining

A trusted intermediary is not necessary because of mining. A decentralized network of computers exists that verifies every transaction across the bitcoin blockchain.

Miners validate new transactions and record them on the global ledger. A new block, containing transactions that occurred since the last block, is mined every 10 minutes on average, thereby adding those transactions to the blockchain. (Antonopoulos, 2017, p. 236)

Mining allows for a trust-less system if a centralized intermediary is used to maintain the ledger mining or miners are not required (Buterin, 2014).

### Proof of Work

Miners compete to find the solution to a block, the first to crack the code broadcasts it to the entire network, then other nodes use that solution to reach consensus on the state of the blockchain. “The proof of work algorithm contains Byzantine fault tolerance meaning that it can handle failures and bad actors that participate in the network.” (Musan, 2020)  
Proof of Work or PoW is an algorithm that incentivizes miners to provide processing power to facilitate transactions on the blockchain in exchange for bitcoin – or some other cryptocurrency to miners a part of its distributed consensus system – and is the heart of the decentralized system. As more and more miners join the mining pool, the PoW algorithm increases the difficulty in-turn the time required to mine a block stays approximately the same.

## Ethereum

Ethereum is built on a similar model as bitcoin, a proof of work algorithm to achieve consensus between nodes on any change of state, with one added benefit: smart contracts (refer to 2.5.1) running on the Ethereum Virtual Machine or EVM (refer to 2.5.2). Ethereum is defined by Vitalik Buterin, its creator as “The general-purpose blockchain.”  
While bitcoin is considered a commodity, Ethereum, a global singleton; it runs as if it is a global, single instance computer (Antonopoulos & Wood, 2018). This functionality allows for the creation of decentralized applications operating on the blockchain.

### Smart Contracts

A smart contract is defined by Nick Szabo (1994) as “A computerized transaction protocol that executes the terms of a contract”. There exist two types of accounts in Ethereum, externally owned accounts (EOAs) with private keys and an account that is controlled entirely by a smart contract with no private keys. The latter can have associated code and data storage while the former cannot.  
A transaction sent from an EOA to a smart contract account triggers its execution using the Ethereum transferred, and any data included in the transaction as input (Antonopoulos & Wood, 2018). The data can contain function calls as well as define parameters to pass to those functions.  
To create smart contracts, first solidity must be compiled to bytecode and then that bytecode is deployed on the Ethereum blockchain by sending it in a transaction to a 0x0 address also known as the zero address.

### Ethereum Virtual Machine

The EVM is a global computer, its state is updated everywhere as each block is mined.

The EVM is a quasi–Turing-complete state machine; “quasi” because all execution processes are limited to a finite number of computational steps by the amount of gas available for any given smart contract execution. (Antonopoulos & Wood, 2018, p. 320)

For every computation on the EVM, its state must be updated globally. This is inefficient and is being improved upon in Ethereum 2.0 but for now, executing any code on the blockchain is resource-intensive and therefore Gas (refer to 2.5.3) – the computational price for running that code – is paid in exchange for executing any computations on the blockchain to prevent overuse.

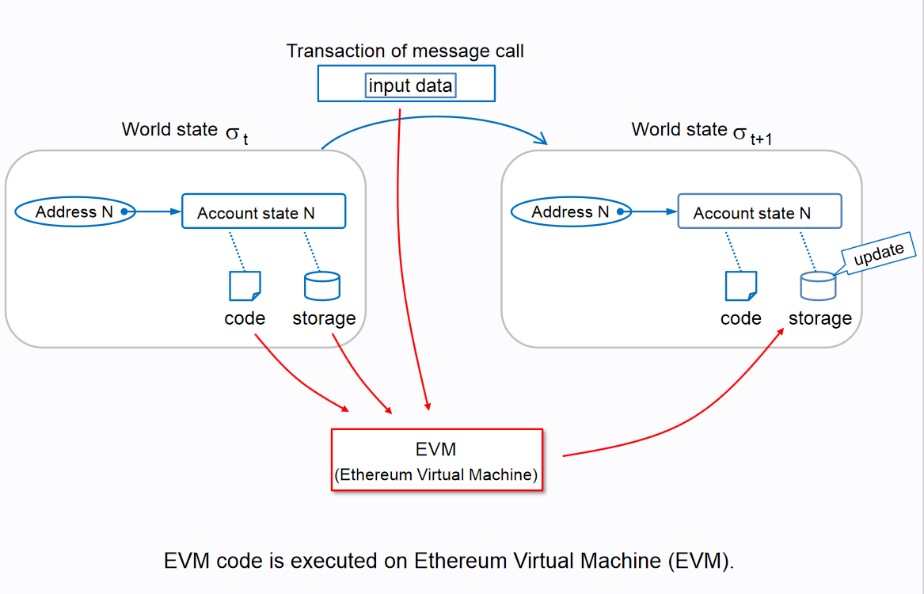


Figure 2 State Update (Takenobu, 2018)

A smart contract deployed on the EVM has a volatile memory and a permanent data store, both initialized to zero. As well as its bytecode, saved on the blockchain.  
Smart contracts can write data on the blockchain in exchange for gas, once that data is saved it is immutable and there are no rents associated with it. Reading the data is free since it is available on the blockchain but updating it or performing any on-chain computation through a smart contract requires gas.

### Gas

Gas is Ethereum’s method of calculating the cost of processing a computation on the blockchain, it is a separate crypto-currency and a crucial part of the system. In Ethereum’s yellow paper Wood (2014) defines the cost of adding two numbers on the blockchain as 3 gas and sending a transaction as 21,000 gas. Gas is priced in terms of Ethereum, determined by demand at a given time.

### ERC-20 Standard Tokens

ERC-20 is an interface for fungible tokens, these fungible tokens can represent a currency or theoretically, anything (e.g. gas), they are transferable between addresses and can contain APIs to interact with smart contracts. Like Ethereum, ERC-20 tokens are transferable in pieces (<1) and require a gas fee. ERC-20 tokens are separate crypto-currencies operating on the Ethereum blockchain.

### ERC-721 Standard – Non-Fungible Token (NFT)

Unlike ERC-20 tokens, Non-Fungible Tokens or NFTs are not interchangeable, every NFT is unique, and cannot be broken down into smaller units (<1). As ERC-20 tokens are transferable on the Ethereum blockchain.

### Decentralized Applications

Utilizing smart contracts, a decentralized application is an interface that exists on the blockchain which users can interact with through transactions.

### Oracles

An oracle is an external data source that impacts the operations of a smart contract on the blockchain, used to create decentralized applications that utilize inputs coming from an off-chain source (Musan, 2020).

## Blockchain Based Election Systems

In a case study utilizing the private Go-Ethereum blockchain operated by a government the authors (Hjálmarsson, et al., 2018) concluded that conducting the election through a private blockchain would be transparent and secure. The system reduces costs, increases efficiency, and reduces risks such as voter fraud.

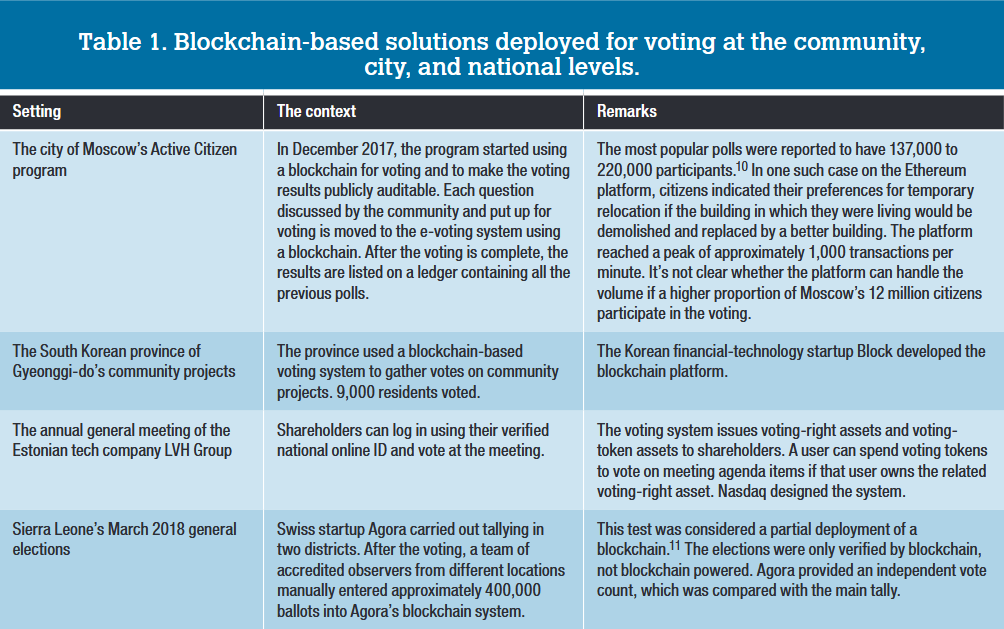


Figure 3 Blockchain based Voting Systems (Kshetri & Voas, 2018)

Blockchain Enabled Voting or BEV promotes voter participation and provides a datastore that cannot be altered while also being publicly auditable. Normally, a voter has no assurance that their vote has been cast, in the case of BEV a voter can search for their transaction on the blockchain and verify that their vote was counted. Larger elections pose many challenges, figure 3 highlights some successful BEV operations all over the world albeit on a small scale. BEV allows for a much quicker vote count as well as removes any chance that a ballot may be wasted due to ambiguities (Kshetri & Voas, 2018).

There are hazards that come with an internet-based voting system, where nation-state attackers could attack these systems and possibly affect the outcome of the election without ever being detected (Park, et al., 2021). Park and his team also discussed how electronic systems do not allow for full transparency; where independent observers can verify paper ballots, that task is instead handed to software creating the possibility of undetected bugs or bad actors. In recent years, many users have lost crypto currency by sending it to invalid addresses or losing access to their private keys, BEV comes with its risks for users with a limited understanding of the technology.

# Design and Development

The smart contract that has all the business logic for the application and the Ethereum blockchain is used as a data store. The smart contract contains an API that can be used to read and write data. The front-end is a react application that aids the owner of the contract in managing the election, aids users in voting and displays the progress of the election in real-time.

The smart contract is a single use contract, it will be deployed to a 0x0 address every time an election must be conducted. The web application must be assigned the address of the contract in order to connect with the API. Users interact with the smart contract through any web3 supported ethereum wallet such as Metamask. Once the election is concluded the smart contract will not accept more votes, the winner of the election will be computed and saved to the blockchain. Making the process immutable, auditable and fully transparent.

### Methodology

he Waterfall model of software development will be used in this project, primarily because there is a fixed timeline and there is a single developer. Utilizing the water-fall model will allow for strict deadlines and added focus on documentation. During the software development cycle there will likely be some unforeseen challenges therefore while setting deadlines and goals, managing risk will be of high priority by leaving as much excess time to address any obstacles and focusing on completing each task at the earliest.

### Version Control

A git repository is linked with Visual Studio Code to allow for swift and easy version control.

### Libraries Used (Package.JSON)

*"@material-ui/core"*: "^4.12.3",

*"@nomiclabs/hardhat-ethers"*: "^2.0.2",

*"@nomiclabs/hardhat-waffle"*: "^2.0.1",

*"@openzeppelin/contracts"*: "^4.2.0",

*"@testing-library/jest-dom"*: "^5.14.1",

*"@testing-library/react"*: "^11.2.7",

*"@testing-library/user-event"*: "^12.8.3",

*"chai"*: "^4.3.4",

*"ethereum-waffle"*: "^3.4.0",

*"ethers"*: "^5.4.1",

*"hardhat"*: "^2.4.3",

*"react"*: "^17.0.2",

*"react-dom"*: "^17.0.2",

*"react-scripts"*: "4.0.3",

*"web-vitals"*: "^1.1.2"

*@material-ui* or user interface provides access to an extensive library of reusable react components.

*@nomiclabs/hardhat-ethers / @ethers* is a JavaScript library that makes it much easier to read data from the blockchain and write data to the blockchain.

*@nomiclabs/hardhat-waffle* is a library that helps with smart contract testing

*@openzeppelin/contracts* provides access to the Open Zeppelin ERC721 library, this allows for importing the ERC721 interface which is used to implement the balloting system

*@chai* an assertion library that allows for testing smart contracts

*@ethereum-waffle* a library for testing as well as writing smart contracts

*@hardhat* is a local development environment for Ethereum

*@react / @react-dom / @react-scripts* React.js library for

## Smart Contract Design

### Development Environment

IDE used is Visual Studio Code or VS Code, the technologies used in the project include JavaScript, React (JavaScript library) and Solidity. VS Code is light, quick, allows compiling JavaScript and running a local development server to display the application during development. Solidity is not compiled in the IDE instead, a development environment for smart contracts called Hardhat is used.

Hardhat is the local testing environment used, it allows developers to run solidity code locally (Hardhat, 2021). A major benefit of using hardhat is the unit testing that comes with it via an assertion library called Chai, coupled with hardhat, asynchronous unit tests can be run while locally emulating a blockchain on a hardhat node.

During unit tests hardhat automatically creates a local blockchain, developers can also separately create hardhat nodes to emulate the Ethereum blockchain for manual testing.

### OpenZeppelin Library

The OpenZeppelin library is an open-source framework that aids developers in writing complex decentralized applications. The library is audited and tested, allowing users to create robust and secure applications (OpenZepplin, 2021). Within this project, OpenZepplin’s implementation of the ERC721 standard will be utilized to create the Non-Fungible Ballot.

The ERC721 standard includes implementations of the ERC721 interface and provides a detailed and secured API to build upon. This includes: Ethers.js

“The ethers.js library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem. It was originally designed for use with [ethers.io](https://ethers.io/) and has since expanded into a more general-purpose library” (docs.ethers.io, 2021).

Ethers.js is a JavaScript library that contains abstractions that aid developers in writing data to the blockchain, reading data from the blockchain as well as signing transactions, compatible with web3 wallets such as metamask. Ethers is used extensively within the front-end of the application, in regards to the back-end it is used within unit tests.

### Sequence Diagram

Figure 4 Sequence Diagram Smart Contract Deployment and Interaction

### Test Driven Development

11 unit tests were written to complete all the functional requirements of the project (within the scope of the smart contract, not the front-end).

Using the hardhat development environment along with chai, unit tests are written in asynchronous JavaScript and tested by deploying Voting.sol –the application’s smart contract - to a local Ethereum node.

Testing is done through Waffle and written via Chai (JavaScript testing library). Within the unit tests ethers.js is used in order to communicate with the smart contract.

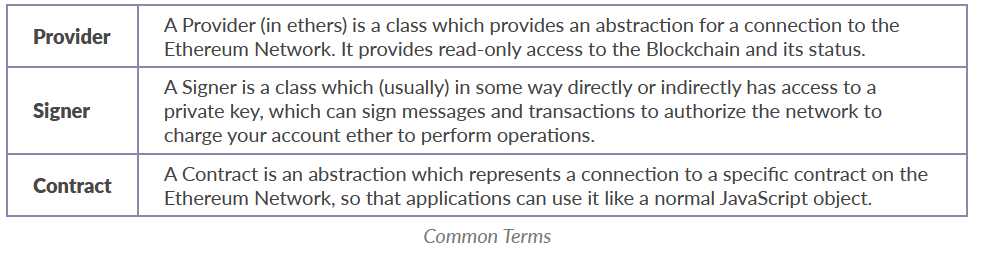


Figure 5 (ethers.io, 2021)

Figure 6 is taken from the Ethers.js documentation, ethers makes it much easier to sign requests with a private key and send transactions. Attempting to do so without the ether.js abstraction would require extensive effort. Signer and Contract are used numerous times within the unit tests (while interacting with the smart contract API), an example below:

1. beforeEach(async function () {
2. //linking the contract ABI
3. Voting = await ethers.getContractFactory("MyToken");
4. //deconstructing array into owner, candidates and voters
5. //signers returns an array of 20 signers on the hardhat testing node
6. //the address at index 0 is the owner's address
7. [owner, cand1, cand2,  cand3, voter1, voter2, voter3, ...voters] = await ethers.getSigners();
8. NFTVoting = await Voting.deploy();
9. //adding three candidates
10. await NFTVoting.addCandidates(cand1.address);
11. await NFTVoting.addCandidates(cand2.address);
12. await NFTVoting.addCandidates(cand3.address);
13. //minting three ballots
14. await NFTVoting.safeMint(voter1.address);
15. await NFTVoting.safeMint(voter2.address);
16. await NFTVoting.safeMint(voter3.address);
17. });

Albeit not a unit test on its own, the above code is part of a beforeEach function which is run before each unit test (part of tests/Voting.js). This function initializes the smart contract and declares some global variables. In line 7 await.ethers.getSigners(); returns an array of 20 signers that exist on the local testnet which is decomposed into multiple variables and the …voters array

The chai library provides access to assert (Chai, 2021), the following is a Unit test utilizing the assertion library and the Ethers.js Contract abstraction:

 describe("Testing deployment, safeMint and balanceOf ", function () {

 it("Minting a ballot and transfering it to an address on the network", async function () {

           expect(await NFTVoting.balanceOf(voter1.address)).to.equal(1);

        });

On the Ethereum main-net block time varies and is approximately 10-20 seconds, on the hardhat node transactions are mined instantly to allow for quick testing.

### Open Zeppelin ERC-721

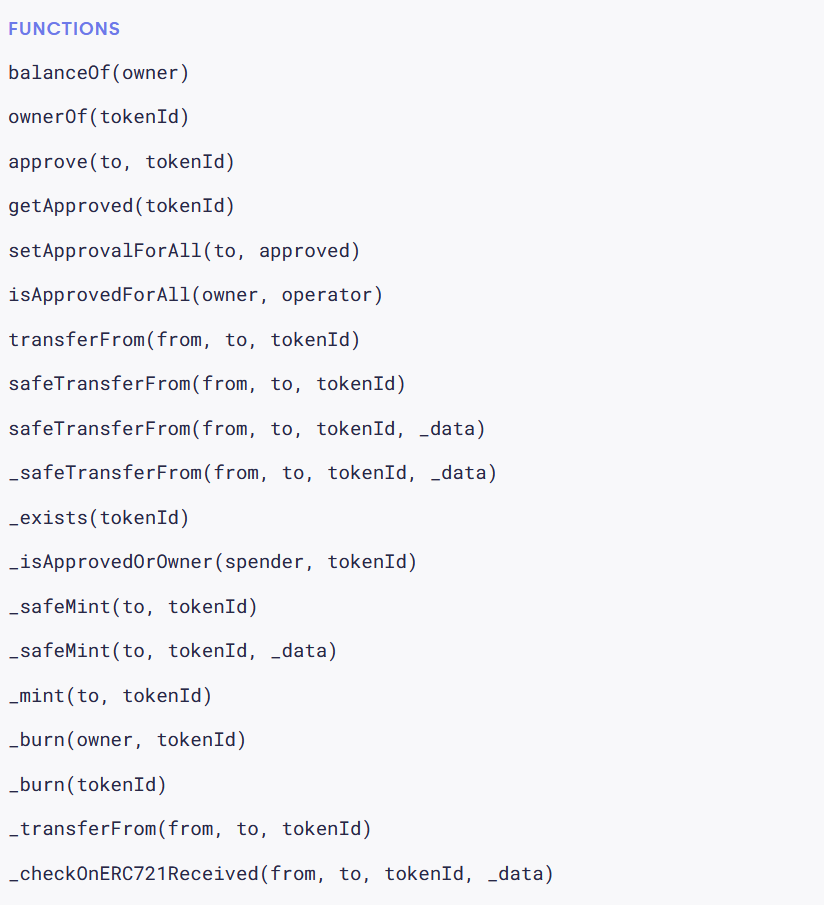


Figure 6 (OpenZeppelin, 2021) ERC-721 Interface

Function names that are preceded by an “\_” are internal or private, meaning they are not accessible externally. Since this application’s primary use case is voting, users should not be allowed to transfer ballots among each other stopping users from potentially selling ballots to others. To implement this safeTransferFrom and Transfer were removed from the interface and a vote function is created that will internally call \_transfer in order to move the ballot from the voter’s address to the candidate’s address.

With the ERC-721 interface, two more libraries are used namely ownable and counters. Ownable allows role-based access control through its function modifier “onlyOwner” this prevents that public function from being executed by anybody other than the owner of the contract while also creating functionality to transfer ownership and set new owners. Counters adds a unique UUID to each token, incrementing automatically when a new token is minted.

### Smart Contract API

The smart contract is responsible for the business logic of the application, deployed on the blockchain. The owner of the contract – the address that deployed it on the blockchain – can execute the following public functions:

safeMint(*address* *to*) *public* onlyOwner

safeMint accepts an Ethereum address as an argument, mints a token and transfers it to the address. The function adds the address that has received a ballot to local storage (saved on the blockchain) and cannot be altered, once an address has received a ballot, it cannot receive another.

*function* safeMintMany(*address*[] *memory* *\_addresses*) *public* onlyOwner {

safeMintMany allows minting multiple ballots, it accepts an array of addresses as an argument and iterates through it, calling safeMint for each.

*function* addCandidates(*address* *\_candidate*) *external* onlyOwner

addCandidates accepts an Ethereum address and adds it to the candidates list, if the address is already a candidate it cannot be added to the candidate list.

*function* conclude() *external* onlyOwner

conclude() simply concludes the election and runs highestVotes() to compute the winner of the election and save it to local storage. If there is a draw, the 0x0 address is returned as the winner followed by the vote count

Although the aforementioned functions are public, if the transaction is sent by a non-owner address it is rejected and no action is taken due to the onlyOwner function modifier.

Users (and the front-end application) can read data from within the contract by calling the following:

*function* votesForCandidate(*address* *\_candidateAddress*) *public* *view* returns (*uint256*)

accepts an address and returns the number of votes for the candidate

*function* totalVotesCast() *public* *view* returns (*uint256*)

returns the total votes cast in the election

*function* allCandidates() *public* *view* returns (*address*[] *memory*)

returns an array consisting of all candidate addresses

*function* vote(*address* *\_candidateAddress*) *external*

the vote function is used to vote, it automatically detects the sending address using msg.sender, it is a solidity variable that stores the address that sends the transaction and accepts the candidate address as an argument . If the voter does not own a ballot, is a candidate, or attempts to vote for an address that is not a candidate, the transaction is rejected.

## Front-End Design and Development

### JavaScript

One of the most popular scripting languages out there, JavaScript is a light-weight single threaded programming language (MDN, 2021). JavaScript standards include the ECMAScript Language Specification which exists to allow interoperability of different websites across browsers.

### React

“React is a JavaScript library for creating user interfaces” (Facebook, 2021). React is open source and free. It is maintained by Facebook primarily as well as other companies and individuals. React allows developers to create complex single-page applications. The single more crucial use-case of react (and JavaScript) is that it allows the creation of applications that can change data without having to re-load the page while also being quick and scalable.

### Material-UI

Material-UI is a component library for React that provides a wide range of components and styling, developed by Google in 2014, Material-UI provides grid based layouts making it much easier to create responsive and fast web applications.

### Meta Mask

MetaMask is a chrome extension that allows users to import their Ethereum wallets through their private keys, MetaMask aids users in sending transactions by signing them with their private keys on their behalf, it is a wallet for Ethereum. On web3 supported websites, developers can create functionality to request transactions through button clicks for example, that end-users can process through MetaMask.

MetaMask is the recommended Ethereum wallet to be used to interact with the voting dApp, but there are many alternatives that can also be used.

### Use Cases

Actor: Owner

Start: Mint Ballot  
Task: Owner can mint a ballot to an address that does not already own a ballot

Start: Add Candidate  
Task: Owner can add a candidate that does not already exist

Start: Conclude Election  
Task: Owner can conclude the election, stop voting and declare a winner

Actor: Voter

Start: Vote and View Progress  
Task: Voter can vote for a valid candidate if they have a ballot in their wallet, they can also view the progress of the election on the landing page of the website

### Front-End Requirements

The front-end of the application has the following requirements:

1. Allow the owner of the contract to add candidates
2. Allow the owner of the contract to mint tokens
3. Allow the owner of the contract to conclude the election
4. Allow voters to vote for their candidates
5. Allow anybody to view the progress of the application

In order to complete the aforementioned tasks, the extension metamask must be installed in the users’s browser

Based on these requirements the follow wireframes and web pages were developed:

### Wireframes

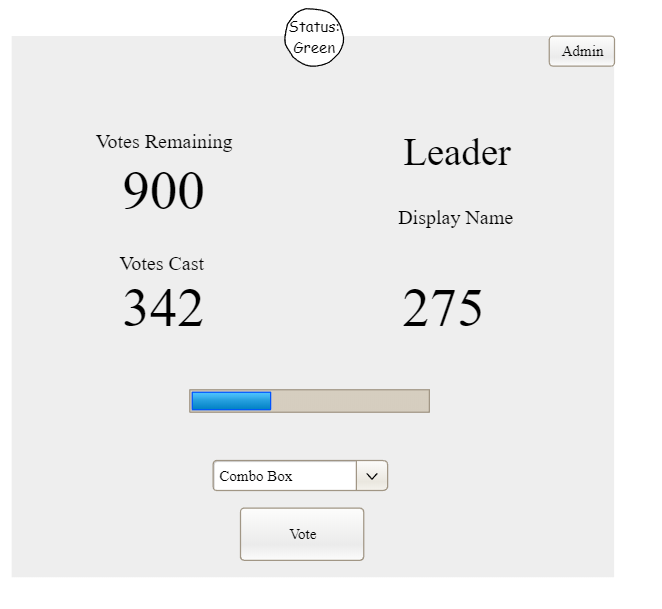


Figure 7 Voting Component Wireframe

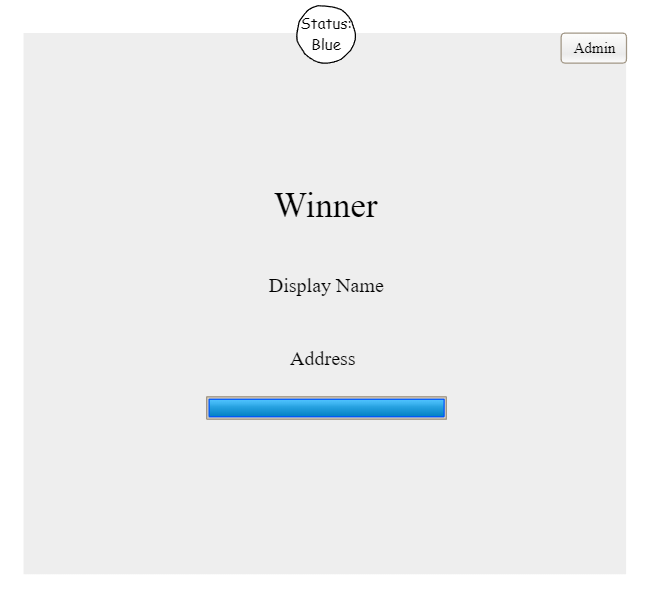


Figure 8 Conclude Component Wireframe

### Developed Web Pages

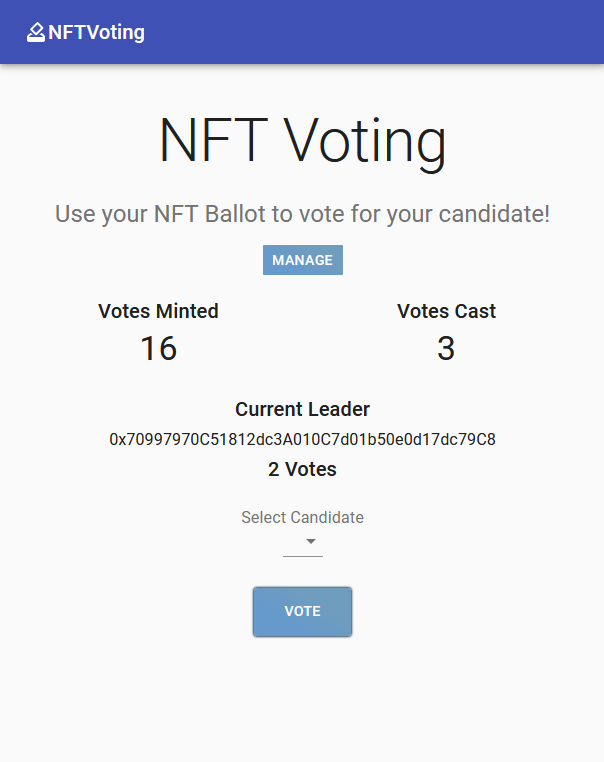


Figure 9 Developed Voting Component

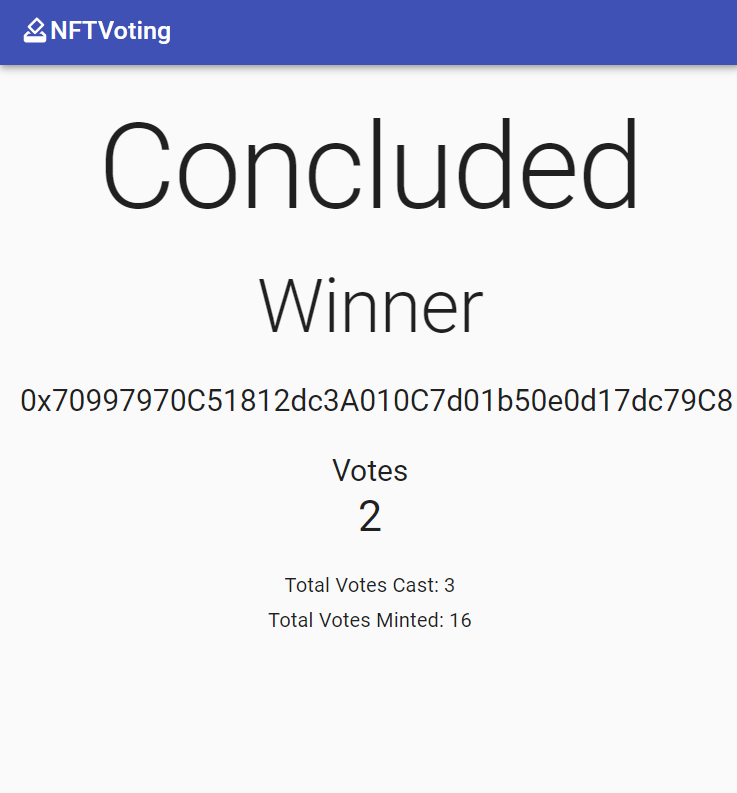


Figure 10 Developed Conclude Component

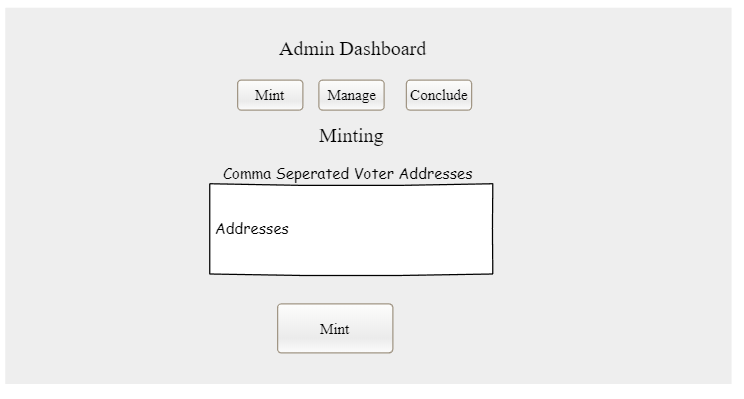


Figure 11 Minting Component Wireframe

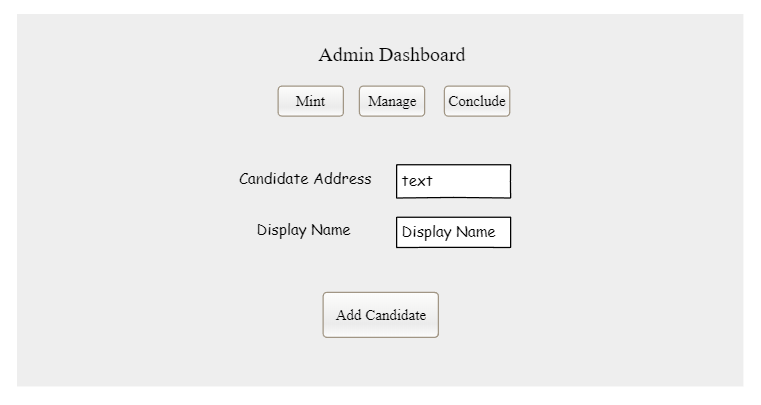


Figure 12 Candidate Component Wireframe

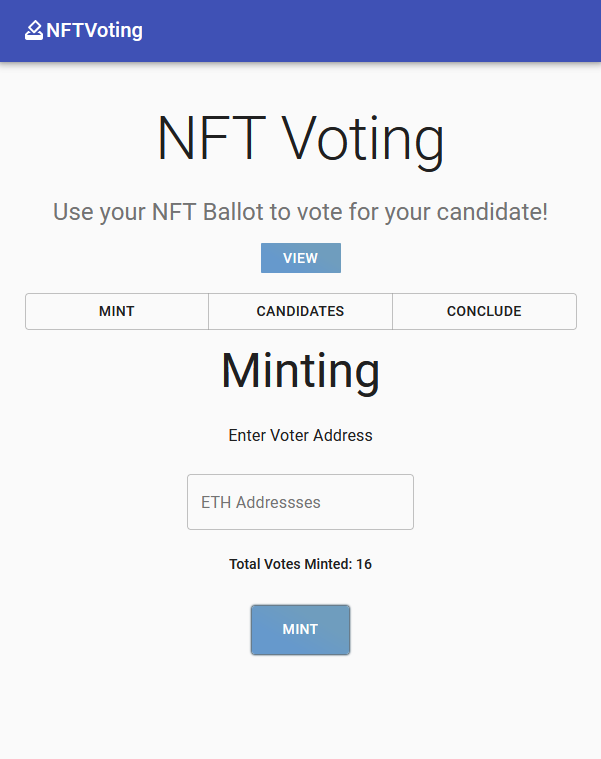


Figure 13 Developed Minting Component

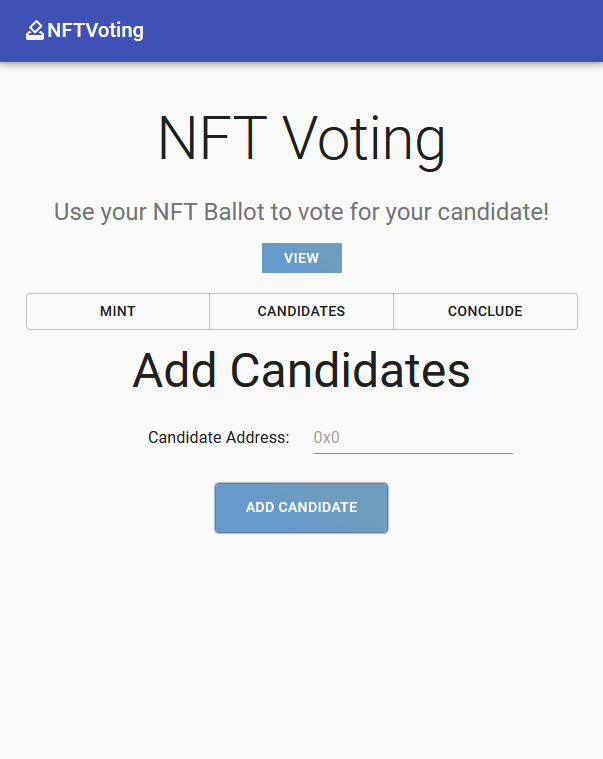


Figure 14 Developed Candidate Component

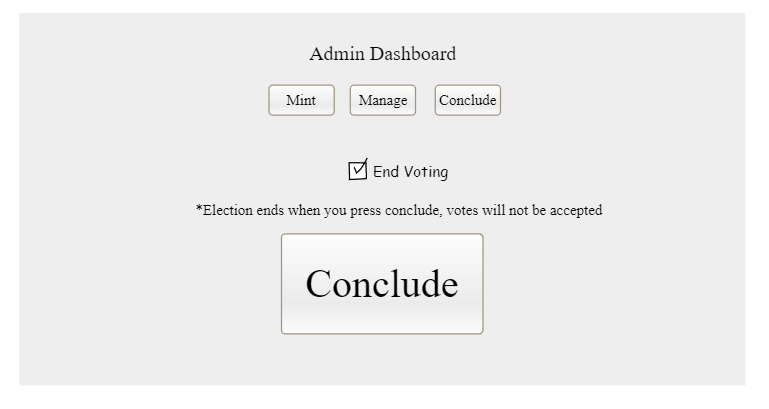


Figure 15 Conclude Component Wireframe

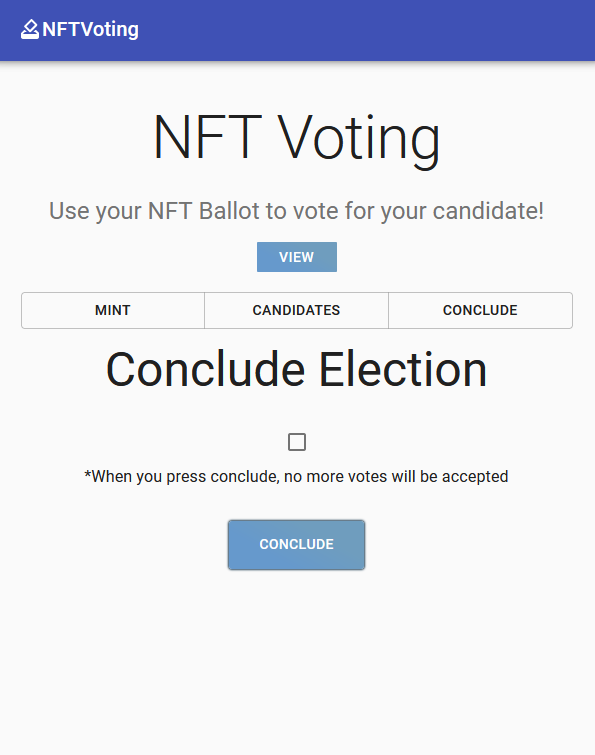


Figure 16 Developed Conclude Component

Each react component behaves as intended, the website is a single page layout, using react to dynamically change the data on the screen when relevant buttons are clicked by the user.

## Front-end and Back-end Integration

React dynamically renders each component depending on state.

Each React component that requires user input has text fields and buttons. Each textfield is a controlled component and all data that is entered is saved to state. When a user clicks the subsequent button the data is fetched from the text fields and a transaction is initiated. The user is prompted to login to their MetaMask wallet and confirm the transaction. Once the transaction is confirmed data is written to the blockchain, this process is followed for any onchain operation that is managed by the event handler “onChainHandler” in Main.js.

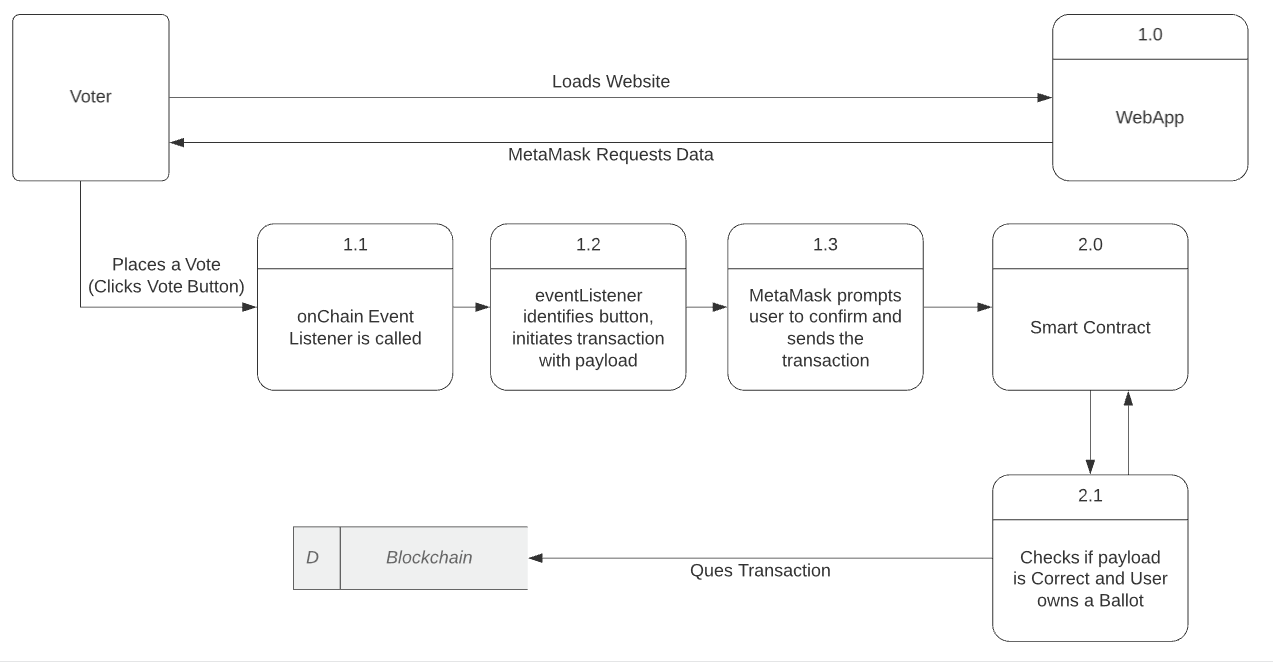


Figure 17 Voting Data Flow

The following is an example of a single function that writes data to the blockchain, the explanation for each line is beneath it in white text:

async sendVote(*candidateToVote*) {

Async function to gain access to await

        if (!*candidateToVote*) return

Return if function parameter is null

        if (typeof window.ethereum !== 'undefined') {

Proceed if metamask exists

            this.setState({ voteWorking: true })

Displaying text “Working…” beneath the react component

            await this.requestAccount() <- request the users metamask account

*const* provider = new ethers.providers.Web3Provider(window.ethereum)

*const* signer = provider.getSigner()

Get and save details for the user, signer allows metamask to send transactions on behalf of the user

*const* contract = new ethers.Contract(votingAddress, NFTVoting.abi, signer)

Create an instance of the smart contract, any interaction will be from the user using signer

            try {

*const* transaction = await contract.vote(*candidateToVote*)

                await transaction.wait()

waiting for the transaction to process

                alert("Success!")

            }

            catch (err) {

                console.log('Error:', err.message)

            }

            this.fetchData()

Updating state with new data

            this.setState({ voteWorking: false })

}

Hiding the “Working…” text beneath the component

# Conclusion

## Evaluation w.r.t Proposal

Requirements Specified in the Proposal:

Essential Functional Requirements:

1. App fully functional and tested on an Ethereum test-net. (Completed)
2. Ballots will be in the form of Non-Fungible Tokens. (Completed)
3. Smart contract governs the task of counting votes. (Completed)
4. Smart contract computes the winner of the election. (Completed)
5. After voting ends the smart contract declares a publicly viewable winner. (Completed)
6. The conducting address can generate non-fungible voting tokens and send them to voters. (Completed)

Advanced Functional Requirements:

1. The conducting body can signal the start and end of the election. (Completed)
2. A website to read and display data from the blockchain as the election proceeds. (Completed)
3. Web app has an admin panel to aid the conducting contract in distributing ballots. (Completed)

Additional Feature Added:

* The owner can mint multiple ballots at a time, in a single transaction.

Non-Functional Requirements

1. Optimized solidity code to reduce gas fee. (Completed)
2. Responsive website. (Completed)
3. Low-loading time and intuitive design. (Completed)

## Critique

After completing the project, I believe that had I been better versed with solidity and web technologies this application could have many more features. Namely:

Maintain a list of ballot addresses that own ballots and adding a check in input control  
Functionality to burn ballots  
A detailed dashboard that that includes the ability to generate reports  
Gas fees should be paid by the contract owner and not the voter

While planning this project, I had no prior experience with web technologies or solidity, therefore I was not able to make design decisions to include these features.

## Personal Growth

I undertook this project to better understand web-programming and the Ethereum cryptocurrency.

Before starting this project I had no knowledge of web development and a limited understanding of how large applications on Ethereum existed and remained secure. In order to create this application, I learned solidity, HTML, CSS, JavaScript and React as well as reading academic papers on blockchain technology.

Now, I have an intermediate understanding of the web technologies mentioned, how Node.js functions and how technologies like React allow for dynamically changing the code within webpage as a developer intends. I now also have a basic-intermediate understanding of solidity and a comprehensive understanding of why the Ethereum ecosystem exists and remains operational.

# References

Antonopoulos, A. M., 2017. *Mastering Bitcoin: Programming the Open Blockchain.* 2nd ed. s.l.:O'Reilly Media.

Antonopoulos, A. M. & Wood, G., 2018. Mastering Ethereum. In: s.l.:O'Reilly.

Bizzaro, P. A. & Garcia, A., 2019. Blockchain Explained and Implications for Accountancy. *ISACA JOURNAL ,* Volume 1, pp. 1-9.

Buterin, V., 2014. *Ethereum\_white\_paper.pdf.* [Online]   
Available at: https://cryptorating.eu/whitepapers/Ethereum/Ethereum\_white\_paper.pdf  
[Accessed 16 03 2021].

Chai, 2021. *Assert - Chai.* [Online]   
Available at: https://www.chaijs.com/api/assert/  
[Accessed 24 July 2021].

docs.ethers.io, 2021. *Documentation.* [Online]   
Available at: https://docs.ethers.io/v5/  
[Accessed 1 August 2021].

ethers.io, 2021. *Getting Started.* [Online]   
Available at: https://docs.ethers.io/v5/getting-started/  
[Accessed 12 June 2021].

Facebook, 2021. *Getting Started - React.* [Online]   
Available at: https://reactjs.org/docs/getting-started.html  
[Accessed 18 August 2021].

Hardhat, 2021. *Hardhat - Ethereum Development Environment for Professionals by Nomic Labs.* [Online]   
Available at: https://hardhat.org/  
[Accessed 6 July 2021].

Hjálmarsson, F. Þ., Hreiðarsson, G. K., M. H. & Hjálmtýsson, G., 2018. *Blockchain-Based E-Voting System.* San Francisco, IEEE, pp. 983-986.

Kshetri, N. & Voas, J., 2018. Blockchain-Enabled E-Voting. *IEEE Software, 35(4),* 35(4), pp. 95-99.

MDN, 2021. *JavaScript | MDN.* [Online]   
Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript  
[Accessed 14 August 2021].

Musan, D. I., 2020. *NFT.financeLeveraging Non-Fungible Tokens.* [Online]   
Available at: https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1920-ug-projects/distinguished-projects/NFT.-finance-Leveraging-Non-Fungible-Tokens.pdf  
[Accessed 16 March 2021].

Nakamoto, S., 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System.* [Online]   
Available at: https://bitcoin.org/bitcoin.pdf  
[Accessed 19 March 2021].

OpenZeppelin, 2021. *ERC 721 - OpenZeppelin Docs.* [Online]   
Available at: https://docs.openzeppelin.com/contracts/2.x/api/token/erc721  
[Accessed 4 August 2021].

OpenZepplin, 2021. *OpenZepplin.* [Online]   
Available at: https://openzeppelin.com/  
[Accessed 8 August 2021].

Park, S., Specter, M., N. N. & Rivest, R. L., 2021. Going from bad to worse: from Internet voting to blockchain voting. *Journal of Cybersecurity,* 7(1), pp. 1-15.

Pilkterton, M., 2016. Blockchain technology: principles and applications. In: *Research Handbook on Digital Transformations.* s.l.:s.n., pp. 252-253.

Szabo, N., 1994. *Smart Contracts.* [Online]   
Available at: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html  
[Accessed 16 March 2021].

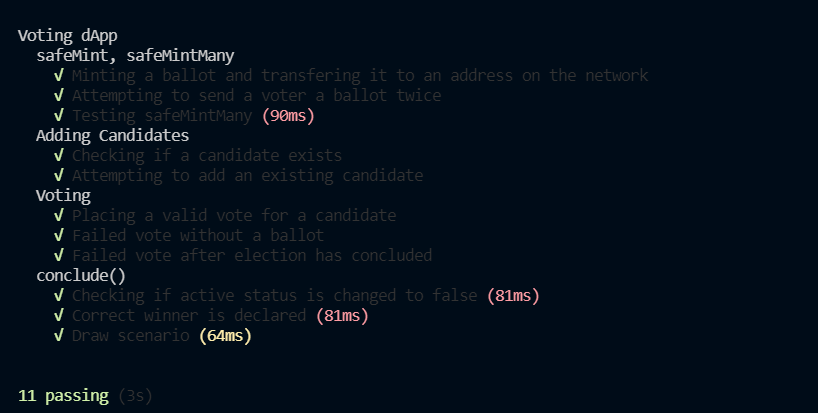
Takenobu, T., 2018. *Ethereum EVM Illustrated.* [Online]   
Available at: https://takenobu-hs.github.io/downloads/ethereum\_evm\_illustrated.pdf  
[Accessed 19 March 2021].

web3, 2021. *web3.js - Ethereum JavaScript API - web3.js 1.0.0 documentation.* [Online]   
Available at: https://web3js.readthedocs.io/en/v1.4.0/  
[Accessed 18 August 2021].

Wood, G., 2014. [Online]   
Available at: https://gavwood.com/paper.pdf  
[Accessed 19 March 2021].

# Appendix A

Unit Tests



*const* { expect } = require("chai")

*const* { ethers } = require("hardhat")

describe("Voting dApp", *function* () {

    //assigning global variables to be used within the unit tests

*let* owner;

*let* cand1;

*let* cand2;

*let* cand3;

*let* voter1;

*let* voter2;

*let* voter3;

*let* voters;

*let* Voting;

*let* NFTVoting;

    //beforeEach will be executed before every unit test

    beforeEach(async *function* () {

        //linking the contract ABI

        Voting = await ethers.getContractFactory("NFTVoting");

        NFTVoting = await Voting.deploy();

        //deconstructing array into owner, candidates and voters

        //signers returns an array of 20 signers on the hardhat testing node

        //the address at index 0 is the owner's address

        [owner, cand1, cand2, cand3, voter1, voter2, voter3, ...voters] = await ethers.getSigners();

        //adding three candidates

        await NFTVoting.addCandidates(cand1.address);

        await NFTVoting.addCandidates(cand2.address);

        await NFTVoting.addCandidates(cand3.address);

        //minting three ballots

        await NFTVoting.safeMint(voter1.address);

        await NFTVoting.safeMint(voter2.address);

        await NFTVoting.safeMint(voter3.address);

    }

    );

    describe("safeMint, safeMintMany", *function* () {

        it("Minting a ballot and transfering it to an address on the network", async *function* () {

            expect(await NFTVoting.balanceOf(voter1.address)).to.equal(1);

        });

        it("Attempting to send a voter a ballot twice", async *function* () {

            await expect(NFTVoting.safeMint(voter1.address)).to.be.revertedWith("Ballot Found");

        });

        it("Testing safeMintMany", async *function* () {

            //mapping voters to an array of addresses

*let* addresses = [];

            for (*let* i = 0; i < voters.length; i++) {

                addresses[i] = voters[i].address

            }

            NFTVoting.safeMintMany(addresses)

            for (*let* i = 0; i < voters.length; i++) {

                expect(await NFTVoting.balanceOf(addresses[0])).to.equal(1);

            }

        });

    });

    describe("Adding Candidates", *function* () {

        it("Checking if a candidate exists", async *function* () {

            expect(await NFTVoting.candidates(0)).to.equal(cand1.address);

        });

        it("Attempting to add an existing candidate", async *function* () {

            await expect(NFTVoting.addCandidates(cand1.address)).to.be.revertedWith("Candidate Exists");

        });

    });

    describe("Voting", *function* () {

        it("Placing a valid vote for a candidate", async *function* () {

            await NFTVoting.connect(voter1).vote(cand1.address);

            expect(await NFTVoting.votesForCandidate(cand1.address)).to.equal(1);

        });

        -

            it("Failed vote without a ballot", async *function* () {

                await NFTVoting.connect(voter1).vote(cand1.address);

                await expect(NFTVoting.connect(voter1).vote(cand1.address)).to.be.revertedWith("No Ballots");

            });

        it("Failed vote after election has concluded", async *function* () {

            await NFTVoting.conclude()

            await expect(NFTVoting.connect(voter1).vote(cand1.address)).to.be.revertedWith("Concluded");

        });

    });

    //Testing the conlude function

    describe("conclude()", *function* () {

        it("Checking if active status is changed to false", async *function* () {

            await NFTVoting.connect(voter1).vote(cand1.address);

            await NFTVoting.connect(voter2).vote(cand2.address);

            await NFTVoting.connect(voter3).vote(cand2.address);

            await NFTVoting.conclude();

            expect(await NFTVoting.active()).to.equal(false);

        });

        it("Correct winner is declared", async *function* () {

            await NFTVoting.connect(voter1).vote(cand1.address);

            await NFTVoting.connect(voter2).vote(cand2.address);

            await NFTVoting.connect(voter3).vote(cand2.address);

            await NFTVoting.conclude();

            expect(await NFTVoting.winner()).to.equal(cand2.address);

        });

        it("Draw scenario", async *function* () {

            await NFTVoting.connect(voter1).vote(cand1.address);

            await NFTVoting.connect(voter2).vote(cand2.address);

            await NFTVoting.conclude();

            expect(await NFTVoting.winner()).to.equal("0x0000000000000000000000000000000000000000");

        });

    });

});

# Appendix B

Code from the Open Zeppellin Library

All code within this Appendix is part of the OpenZeppelin Library, my sourcecode is an addition to this, to be added within the same file for deployment and included in Appendix C

/ SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

/\*\*

 \* *@dev* Interface of the ERC165 standard, as defined in the

 \* https://eips.ethereum.org/EIPS/eip-165[EIP].

 \*

 \* Implementers can declare support of contract interfaces, which can then be

 \* queried by others ({ERC165Checker}).

 \*

 \* For an implementation, see {ERC165}.

 \*/

*interface* IERC165 {

    /\*\*

     \* *@dev* Returns true if this contract implements the interface defined by

     \* `interfaceId`. See the corresponding

     \* https://eips.ethereum.org/EIPS/eip-165#how-interfaces-are-identified[EIP section]

     \* to learn more about how these ids are created.

     \*

     \* This function call must use less than 30 000 gas.

     \*/

*function* supportsInterface(*bytes4* *interfaceId*) *external* *view* returns (*bool*);

}

pragma solidity ^0.8.0;

/\*\*

 \* *@dev* String operations.

 \*/

*library* Strings {

*bytes16* *private* *constant* \_HEX\_SYMBOLS = "0123456789abcdef";

    /\*\*

     \* *@dev* Converts a `uint256` to its ASCII `string` decimal representation.

     \*/

*function* toString(*uint256* *value*) *internal* *pure* returns (*string* *memory*) {

        // Inspired by OraclizeAPI's implementation - MIT licence

        // https://github.com/oraclize/ethereum-api/blob/b42146b063c7d6ee1358846c198246239e9360e8/oraclizeAPI\_0.4.25.sol

        if (value == 0) {

            return "0";

        }

*uint256* temp = value;

*uint256* digits;

        while (temp != 0) {

            digits++;

            temp /= 10;

        }

*bytes* *memory* buffer = new *bytes*(digits);

        while (value != 0) {

            digits -= 1;

            buffer[digits] = *bytes1*(*uint8*(48 + *uint256*(value % 10)));

            value /= 10;

        }

        return *string*(buffer);

    }

    /\*\*

     \* *@dev* Converts a `uint256` to its ASCII `string` hexadecimal representation.

     \*/

*function* toHexString(*uint256* *value*) *internal* *pure* returns (*string* *memory*) {

        if (value == 0) {

            return "0x00";

        }

*uint256* temp = value;

*uint256* length = 0;

        while (temp != 0) {

            length++;

            temp >>= 8;

        }

        return toHexString(value, length);

    }

    /\*\*

     \* *@dev* Converts a `uint256` to its ASCII `string` hexadecimal representation with fixed length.

     \*/

*function* toHexString(*uint256* *value*, *uint256* *length*)

*internal*

*pure*

        returns (*string* *memory*)

    {

*bytes* *memory* buffer = new *bytes*(2 \* length + 2);

        buffer[0] = "0";

        buffer[1] = "x";

        for (*uint256* i = 2 \* length + 1; i > 1; --i) {

            buffer[i] = \_HEX\_SYMBOLS[value & 0xf];

            value >>= 4;

        }

        require(value == 0, "Strings: hex length insufficient");

        return *string*(buffer);

    }

}

pragma solidity ^0.8.0;

pragma solidity ^0.8.0;

/\*\*

 \* *@dev* Required interface of an ERC721 compliant contract.

 \*/

*interface* IERC721 is IERC165 {

    /\*\*

     \* *@dev* Emitted when `tokenId` token is transferred from `from` to `to`.

     \*/

*event* Transfer(

*address* *indexed* *from*,

*address* *indexed* *to*,

*uint256* *indexed* *tokenId*

    );

    /\*\*

     \* *@dev* Emitted when `owner` enables `approved` to manage the `tokenId` token.

     \*/

*event* Approval(

*address* *indexed* *owner*,

*address* *indexed* *approved*,

*uint256* *indexed* *tokenId*

    );

    /\*\*

     \* *@dev* Emitted when `owner` enables or disables (`approved`) `operator` to manage all of its assets.

     \*/

*event* ApprovalForAll(

*address* *indexed* *owner*,

*address* *indexed* *operator*,

*bool* *approved*

    );

    /\*\*

     \* *@dev* Returns the number of tokens in ``owner``'s account.

     \*/

*function* balanceOf(*address* *owner*) *external* *view* returns (*uint256* *balance*);

    /\*\*

     \* *@dev* Returns the owner of the `tokenId` token.

     \*

     \* Requirements:

     \*

     \* - `tokenId` must exist.

     \*/

*function* ownerOf(*uint256* *tokenId*) *external* *view* returns (*address* *owner*);

    /\*\*

     \* *@dev* Gives permission to `to` to transfer `tokenId` token to another account.

     \* The approval is cleared when the token is transferred.

     \*

     \* Only a single account can be approved at a time, so approving the zero address clears previous approvals.

     \*

     \* Requirements:

     \*

     \* - The caller must own the token or be an approved operator.

     \* - `tokenId` must exist.

     \*

     \* Emits an {Approval} event.

     \*/

*function* approve(*address* *to*, *uint256* *tokenId*) *external*;

    /\*\*

     \* *@dev* Returns the account approved for `tokenId` token.

     \*

     \* Requirements:

     \*

     \* - `tokenId` must exist.

     \*/

*function* getApproved(*uint256* *tokenId*)

*external*

*view*

        returns (*address* *operator*);

    /\*\*

     \* *@dev* Approve or remove `operator` as an operator for the caller.

     \* Operators can call {transferFrom} or {safeTransferFrom} for any token owned by the caller.

     \*

     \* Requirements:

     \*

     \* - The `operator` cannot be the caller.

     \*

     \* Emits an {ApprovalForAll} event.

     \*/

*function* setApprovalForAll(*address* *operator*, *bool* *\_approved*) *external*;

    /\*\*

     \* *@dev* Returns if the `operator` is allowed to manage all of the assets of `owner`.

     \*

     \* See {setApprovalForAll}

     \*/

*function* isApprovedForAll(*address* *owner*, *address* *operator*)

*external*

*view*

        returns (*bool*);

}

pragma solidity ^0.8.0;

/\*\*

 \* *@title* ERC721 token receiver interface

 \* *@dev* Interface for any contract that wants to support safeTransfers

 \* from ERC721 asset contracts.

 \*/

*interface* IERC721Receiver {

    /\*\*

     \* *@dev* Whenever an {IERC721} `tokenId` token is transferred to this contract via {IERC721-safeTransferFrom}

     \* by `operator` from `from`, this function is called.

     \*

     \* It must return its Solidity selector to confirm the token transfer.

     \* If any other value is returned or the interface is not implemented by the recipient, the transfer will be reverted.

     \*

     \* The selector can be obtained in Solidity with `IERC721.onERC721Received.selector`.

     \*/

*function* onERC721Received(

*address* *operator*,

*address* *from*,

*uint256* *tokenId*,

*bytes* *calldata* *data*

    ) *external* returns (*bytes4*);

}

pragma solidity ^0.8.0;

/\*\*

 \* *@title* ERC-721 Non-Fungible Token Standard, optional metadata extension

 \* *@dev* See https://eips.ethereum.org/EIPS/eip-721

 \*/

*interface* IERC721Metadata is IERC721 {

    /\*\*

     \* *@dev* Returns the token collection name.

     \*/

*function* name() *external* *view* returns (*string* *memory*);

    /\*\*

     \* *@dev* Returns the token collection symbol.

     \*/

*function* symbol() *external* *view* returns (*string* *memory*);

    /\*\*

     \* *@dev* Returns the Uniform Resource Identifier (URI) for `tokenId` token.

     \*/

*function* tokenURI(*uint256* *tokenId*) *external* *view* returns (*string* *memory*);

}

pragma solidity ^0.8.0;

/\*\*

 \* *@dev* Collection of functions related to the address type

 \*/

*library* Address {

    /\*\*

     \* *@dev* Returns true if `account` is a contract.

     \*

     \* [IMPORTANT]

     \* ====

     \* It is unsafe to assume that an address for which this function returns

     \* false is an externally-owned account (EOA) and not a contract.

     \*

     \* Among others, `isContract` will return false for the following

     \* types of addresses:

     \*

     \*  - an externally-owned account

     \*  - a contract in construction

     \*  - an address where a contract will be created

     \*  - an address where a contract lived, but was destroyed

     \* ====

     \*/

*function* isContract(*address* *account*) *internal* *view* returns (*bool*) {

        // This method relies on extcodesize, which returns 0 for contracts in

        // construction, since the code is only stored at the end of the

        // constructor execution.

*uint256* size;

        assembly {

            size := extcodesize(account)

        }

        return size > 0;

    }

    /\*\*

     \* *@dev* Replacement for Solidity's `transfer`: sends `amount` wei to

     \* `recipient`, forwarding all available gas and reverting on errors.

     \*

     \* https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost

     \* of certain opcodes, possibly making contracts go over the 2300 gas limit

     \* imposed by `transfer`, making them unable to receive funds via

     \* `transfer`. {sendValue} removes this limitation.

     \*

     \* https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].

     \*

     \* IMPORTANT: because control is transferred to `recipient`, care must be

     \* taken to not create reentrancy vulnerabilities. Consider using

     \* {ReentrancyGuard} or the

     \* https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-interactions-pattern[checks-effects-interactions pattern].

     \*/

*function* sendValue(*address* *payable* *recipient*, *uint256* *amount*) *internal* {

        require(

*address*(this).balance >= amount,

            "Address: insufficient balance"

        );

        (*bool* success, ) = recipient.call{value: amount}("");

        require(

            success,

            "Address: unable to send value, recipient may have reverted"

        );

    }

    /\*\*

     \* *@dev* Performs a Solidity function call using a low level `call`. A

     \* plain `call` is an unsafe replacement for a function call: use this

     \* function instead.

     \*

     \* If `target` reverts with a revert reason, it is bubbled up by this

     \* function (like regular Solidity function calls).

     \*

     \* Returns the raw returned data. To convert to the expected return value,

     \* use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].

     \*

     \* Requirements:

     \*

     \* - `target` must be a contract.

     \* - calling `target` with `data` must not revert.

     \*

     \* \_Available since v3.1.\_

     \*/

*function* functionCall(*address* *target*, *bytes* *memory* *data*)

*internal*

        returns (*bytes* *memory*)

    {

        return functionCall(target, data, "Address: low-level call failed");

    }

    /\*\*

     \* *@dev* Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with

     \* `errorMessage` as a fallback revert reason when `target` reverts.

     \*

     \* \_Available since v3.1.\_

     \*/

*function* functionCall(

*address* *target*,

*bytes* *memory* *data*,

*string* *memory* *errorMessage*

    ) *internal* returns (*bytes* *memory*) {

        return functionCallWithValue(target, data, 0, errorMessage);

    }

    /\*\*

     \* *@dev* Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],

     \* but also transferring `value` wei to `target`.

     \*

     \* Requirements:

     \*

     \* - the calling contract must have an ETH balance of at least `value`.

     \* - the called Solidity function must be `payable`.

     \*

     \* \_Available since v3.1.\_

     \*/

*function* functionCallWithValue(

*address* *target*,

*bytes* *memory* *data*,

*uint256* *value*

    ) *internal* returns (*bytes* *memory*) {

        return

            functionCallWithValue(

                target,

                data,

                value,

                "Address: low-level call with value failed"

            );

    }

    /\*\*

     \* *@dev* Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but

     \* with `errorMessage` as a fallback revert reason when `target` reverts.

     \*

     \* \_Available since v3.1.\_

     \*/

*function* functionCallWithValue(

*address* *target*,

*bytes* *memory* *data*,

*uint256* *value*,

*string* *memory* *errorMessage*

    ) *internal* returns (*bytes* *memory*) {

        require(

*address*(this).balance >= value,

            "Address: insufficient balance for call"

        );

        require(isContract(target), "Address: call to non-contract");

        (*bool* success, *bytes* *memory* returndata) = target.call{value: value}(

            data

        );

        return \_verifyCallResult(success, returndata, errorMessage);

    }

    /\*\*

     \* *@dev* Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],

     \* but performing a static call.

     \*

     \* \_Available since v3.3.\_

     \*/

*function* functionStaticCall(*address* *target*, *bytes* *memory* *data*)

*internal*

*view*

        returns (*bytes* *memory*)

    {

        return

            functionStaticCall(

                target,

                data,

                "Address: low-level static call failed"

            );

    }

    /\*\*

     \* *@dev* Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],

     \* but performing a static call.

     \*

     \* \_Available since v3.3.\_

     \*/

*function* functionStaticCall(

*address* *target*,

*bytes* *memory* *data*,

*string* *memory* *errorMessage*

    ) *internal* *view* returns (*bytes* *memory*) {

        require(isContract(target), "Address: static call to non-contract");

        (*bool* success, *bytes* *memory* returndata) = target.staticcall(data);

        return \_verifyCallResult(success, returndata, errorMessage);

    }

    /\*\*

     \* *@dev* Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],

     \* but performing a delegate call.

     \*

     \* \_Available since v3.4.\_

     \*/

*function* functionDelegateCall(*address* *target*, *bytes* *memory* *data*)

*internal*

        returns (*bytes* *memory*)

    {

        return

            functionDelegateCall(

                target,

                data,

                "Address: low-level delegate call failed"

            );

    }

    /\*\*

     \* *@dev* Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],

     \* but performing a delegate call.

     \*

     \* \_Available since v3.4.\_

     \*/

*function* functionDelegateCall(

*address* *target*,

*bytes* *memory* *data*,

*string* *memory* *errorMessage*

    ) *internal* returns (*bytes* *memory*) {

        require(isContract(target), "Address: delegate call to non-contract");

        (*bool* success, *bytes* *memory* returndata) = target.delegatecall(data);

        return \_verifyCallResult(success, returndata, errorMessage);

    }

*function* \_verifyCallResult(

*bool* *success*,

*bytes* *memory* *returndata*,

*string* *memory* *errorMessage*

    ) *private* *pure* returns (*bytes* *memory*) {

        if (success) {

            return returndata;

        } else {

            // Look for revert reason and bubble it up if present

            if (returndata.length > 0) {

                // The easiest way to bubble the revert reason is using memory via assembly

                assembly {

*let* returndata\_size := mload(returndata)

                    revert(add(32, returndata), returndata\_size)

                }

            } else {

                revert(errorMessage);

            }

        }

    }

}

pragma solidity ^0.8.0;

/\*

 \* @dev Provides information about the current execution context, including the

 \* sender of the transaction and its data. While these are generally available

 \* via msg.sender and msg.data, they should not be accessed in such a direct

 \* manner, since when dealing with meta-transactions the account sending and

 \* paying for execution may not be the actual sender (as far as an application

 \* is concerned).

 \*

 \* This contract is only required for intermediate, library-like contracts.

 \*/

*abstract* *contract* Context {

*function* \_msgSender() *internal* *view* *virtual* returns (*address*) {

        return msg.sender;

    }

*function* \_msgData() *internal* *view* *virtual* returns (*bytes* *calldata*) {

        return msg.data;

    }

}

pragma solidity ^0.8.0;

/\*\*

 \* *@dev* Implementation of the {IERC165} interface.

 \*

 \* Contracts that want to implement ERC165 should inherit from this contract and override {supportsInterface} to check

 \* for the additional interface id that will be supported. For example:

 \*

 \* ```solidity

 \* function supportsInterface(bytes4 interfaceId) public view virtual override returns (bool) {

 \*     return interfaceId == type(MyInterface).interfaceId || super.supportsInterface(interfaceId);

 \* }

 \* ```

 \*

 \* Alternatively, {ERC165Storage} provides an easier to use but more expensive implementation.

 \*/

*abstract* *contract* ERC165 is IERC165 {

    /\*\*

     \* *@dev* See {IERC165-supportsInterface}.

     \*/

*function* supportsInterface(*bytes4* *interfaceId*)

*public*

*view*

*virtual*

*override*

        returns (*bool*)

    {

        return interfaceId == type(IERC165).interfaceId;

    }

}

/\*\*

 \* *@dev* Implementation of https://eips.ethereum.org/EIPS/eip-721[ERC721] Non-Fungible Token Standard, including

 \* the Metadata extension, but not including the Enumerable extension, which is available separately as

 \* {ERC721Enumerable}.

 \*/

*contract* ERC721 is Context, ERC165, IERC721, IERC721Metadata {

    using Address for address;

    using Strings for uint256;

    // Token name

*string* *private* \_name;

    // Token symbol

*string* *private* \_symbol;

    // Mapping from token ID to owner address

*mapping*(*uint256* => *address*) *private* \_owners;

    // Mapping owner address to token count

*mapping*(*address* => *uint256*) *private* \_balances;

    // Mapping from token ID to approved address

*mapping*(*uint256* => *address*) *private* \_tokenApprovals;

    // Mapping from owner to operator approvals

*mapping*(*address* => *mapping*(*address* => *bool*)) *private* \_operatorApprovals;

    /\*\*

     \* *@dev* Initializes the contract by setting a `name` and a `symbol` to the token collection.

     \*/

*constructor*(*string* *memory* *name\_*, *string* *memory* *symbol\_*) {

        \_name = name\_;

        \_symbol = symbol\_;

    }

    /\*\*

     \* *@dev* See {IERC165-supportsInterface}.

     \*/

*function* supportsInterface(*bytes4* *interfaceId*)

*public*

*view*

*virtual*

*override*(ERC165, IERC165)

        returns (*bool*)

    {

        return

            interfaceId == type(IERC721).interfaceId ||

            interfaceId == type(IERC721Metadata).interfaceId ||

            super.supportsInterface(interfaceId);

    }

    /\*\*

     \* *@dev* See {IERC721-balanceOf}.

     \*/

*function* balanceOf(*address* *owner*)

*public*

*view*

*virtual*

*override*

        returns (*uint256*)

    {

        require(

            owner != *address*(0),

            "ERC721: balance query for the zero address"

        );

        return \_balances[owner];

    }

    /\*\*

     \* *@dev* See {IERC721-ownerOf}.

     \*/

*function* ownerOf(*uint256* *tokenId*)

*public*

*view*

*virtual*

*override*

        returns (*address*)

    {

*address* owner = \_owners[tokenId];

        require(

            owner != *address*(0),

            "ERC721: owner query for nonexistent token"

        );

        return owner;

    }

    /\*\*

     \* *@dev* See {IERC721Metadata-name}.

     \*/

*function* name() *public* *view* *virtual* *override* returns (*string* *memory*) {

        return \_name;

    }

    /\*\*

     \* *@dev* See {IERC721Metadata-symbol}.

     \*/

*function* symbol() *public* *view* *virtual* *override* returns (*string* *memory*) {

        return \_symbol;

    }

    /\*\*

     \* *@dev* See {IERC721Metadata-tokenURI}.

     \*/

*function* tokenURI(*uint256* *tokenId*)

*public*

*view*

*virtual*

*override*

        returns (*string* *memory*)

    {

        require(

            \_exists(tokenId),

            "ERC721Metadata: URI query for nonexistent token"

        );

*string* *memory* baseURI = \_baseURI();

        return

*bytes*(baseURI).length > 0

                ? *string*(abi.encodePacked(baseURI, tokenId.toString()))

                : "";

    }

    /\*\*

     \* *@dev* Base URI for computing {tokenURI}. If set, the resulting URI for each

     \* token will be the concatenation of the `baseURI` and the `tokenId`. Empty

     \* by default, can be overriden in child contracts.

     \*/

*function* \_baseURI() *internal* *view* *virtual* returns (*string* *memory*) {

        return "";

    }

    /\*\*

     \* *@dev* See {IERC721-approve}.

     \*/

*function* approve(*address* *to*, *uint256* *tokenId*) *public* *virtual* *override* {

*address* owner = ERC721.ownerOf(tokenId);

        require(to != owner, "ERC721: approval to current owner");

        require(

            \_msgSender() == owner || isApprovedForAll(owner, \_msgSender()),

            "ERC721: approve caller is not owner nor approved for all"

        );

        \_approve(to, tokenId);

    }

    /\*\*

     \* *@dev* See {IERC721-getApproved}.

     \*/

*function* getApproved(*uint256* *tokenId*)

*public*

*view*

*virtual*

*override*

        returns (*address*)

    {

        require(

            \_exists(tokenId),

            "ERC721: approved query for nonexistent token"

        );

        return \_tokenApprovals[tokenId];

    }

    /\*\*

     \* *@dev* See {IERC721-setApprovalForAll}.

     \*/

*function* setApprovalForAll(*address* *operator*, *bool* *approved*)

*public*

*virtual*

*override*

    {

        require(operator != \_msgSender(), "ERC721: approve to caller");

        \_operatorApprovals[\_msgSender()][operator] = approved;

        emit ApprovalForAll(\_msgSender(), operator, approved);

    }

    /\*\*

     \* *@dev* See {IERC721-isApprovedForAll}.

     \*/

*function* isApprovedForAll(*address* *owner*, *address* *operator*)

*public*

*view*

*virtual*

*override*

        returns (*bool*)

    {

        return \_operatorApprovals[owner][operator];

    }

    /\*\*

     \* *@dev* Safely transfers `tokenId` token from `from` to `to`, checking first that contract recipients

     \* are aware of the ERC721 protocol to prevent tokens from being forever locked.

     \*

     \* `\_data` is additional data, it has no specified format and it is sent in call to `to`.

     \*

     \* This internal function is equivalent to {safeTransferFrom}, and can be used to e.g.

     \* implement alternative mechanisms to perform token transfer, such as signature-based.

     \*

     \* Requirements:

     \*

     \* - `from` cannot be the zero address.

     \* - `to` cannot be the zero address.

     \* - `tokenId` token must exist and be owned by `from`.

     \* - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.

     \*

     \* Emits a {Transfer} event.

     \*/

*function* \_safeTransfer(

*address* *from*,

*address* *to*,

*uint256* *tokenId*,

*bytes* *memory* *\_data*

    ) *internal* *virtual* {

        \_transfer(from, to, tokenId);

        require(

            \_checkOnERC721Received(from, to, tokenId, \_data),

            "ERC721: transfer to non ERC721Receiver implementer"

        );

    }

    /\*\*

     \* *@dev* Returns whether `tokenId` exists.

     \*

     \* Tokens can be managed by their owner or approved accounts via {approve} or {setApprovalForAll}.

     \*

     \* Tokens start existing when they are minted (`\_mint`),

     \* and stop existing when they are burned (`\_burn`).

     \*/

*function* \_exists(*uint256* *tokenId*) *internal* *view* *virtual* returns (*bool*) {

        return \_owners[tokenId] != *address*(0);

    }

    /\*\*

     \* *@dev* Returns whether `spender` is allowed to manage `tokenId`.

     \*

     \* Requirements:

     \*

     \* - `tokenId` must exist.

     \*/

*function* \_isApprovedOrOwner(*address* *spender*, *uint256* *tokenId*)

*internal*

*view*

*virtual*

        returns (*bool*)

    {

        require(

            \_exists(tokenId),

            "ERC721: operator query for nonexistent token"

        );

*address* owner = ERC721.ownerOf(tokenId);

        return (spender == owner ||

            getApproved(tokenId) == spender ||

            isApprovedForAll(owner, spender));

    }

    /\*\*

     \* *@dev* Safely mints `tokenId` and transfers it to `to`.

     \*

     \* Requirements:

     \*

     \* - `tokenId` must not exist.

     \* - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.

     \*

     \* Emits a {Transfer} event.

     \*/

*function* \_safeMint(*address* *to*, *uint256* *tokenId*) *internal* *virtual* {

        \_safeMint(to, tokenId, "");

    }

    /\*\*

     \* *@dev* Same as {xref-ERC721-\_safeMint-address-uint256-}[`\_safeMint`], with an additional `data` parameter which is

     \* forwarded in {IERC721Receiver-onERC721Received} to contract recipients.

     \*/

*function* \_safeMint(

*address* *to*,

*uint256* *tokenId*,

*bytes* *memory* *\_data*

    ) *internal* *virtual* {

        \_mint(to, tokenId);

        require(

            \_checkOnERC721Received(*address*(0), to, tokenId, \_data),

            "ERC721: transfer to non ERC721Receiver implementer"

        );

    }

    /\*\*

     \* *@dev* Mints `tokenId` and transfers it to `to`.

     \*

     \* WARNING: Usage of this method is discouraged, use {\_safeMint} whenever possible

     \*

     \* Requirements:

     \*

     \* - `tokenId` must not exist.

     \* - `to` cannot be the zero address.

     \*

     \* Emits a {Transfer} event.

     \*/

*function* \_mint(*address* *to*, *uint256* *tokenId*) *internal* *virtual* {

        require(to != *address*(0), "ERC721: mint to the zero address");

        require(!\_exists(tokenId), "ERC721: token already minted");

        \_beforeTokenTransfer(*address*(0), to, tokenId);

        \_balances[to] += 1;

        \_owners[tokenId] = to;

        emit Transfer(*address*(0), to, tokenId);

    }

    /\*\*

     \* *@dev* Destroys `tokenId`.

     \* The approval is cleared when the token is burned.

     \*

     \* Requirements:

     \*

     \* - `tokenId` must exist.

     \*

     \* Emits a {Transfer} event.

     \*/

*function* \_burn(*uint256* *tokenId*) *internal* *virtual* {

*address* owner = ERC721.ownerOf(tokenId);

        \_beforeTokenTransfer(owner, *address*(0), tokenId);

        // Clear approvals

        \_approve(*address*(0), tokenId);

        \_balances[owner] -= 1;

        delete \_owners[tokenId];

        emit Transfer(owner, *address*(0), tokenId);

    }

    /\*\*

     \* *@dev* Transfers `tokenId` from `from` to `to`.

     \*  As opposed to {transferFrom}, this imposes no restrictions on msg.sender.

     \*

     \* Requirements:

     \*

     \* - `to` cannot be the zero address.

     \* - `tokenId` token must be owned by `from`.

     \*

     \* Emits a {Transfer} event.

     \*/

*function* \_transfer(

*address* *from*,

*address* *to*,

*uint256* *tokenId*

    ) *internal* *virtual* {

        require(

            ERC721.ownerOf(tokenId) == from,

            "ERC721: transfer of token that is not own"

        );

        require(to != *address*(0), "ERC721: transfer to the zero address");

        \_beforeTokenTransfer(from, to, tokenId);

        // Clear approvals from the previous owner

        \_approve(*address*(0), tokenId);

        \_balances[from] -= 1;

        \_balances[to] += 1;

        \_owners[tokenId] = to;

        emit Transfer(from, to, tokenId);

    }

    /\*\*

     \* *@dev* Approve `to` to operate on `tokenId`

     \*

     \* Emits a {Approval} event.

     \*/

*function* \_approve(*address* *to*, *uint256* *tokenId*) *internal* *virtual* {

        \_tokenApprovals[tokenId] = to;

        emit Approval(ERC721.ownerOf(tokenId), to, tokenId);

    }

    /\*\*

     \* *@dev* Internal function to invoke {IERC721Receiver-onERC721Received} on a target address.

     \* The call is not executed if the target address is not a contract.

     \*

     \* *@param* from address representing the previous owner of the given token ID

     \* *@param* to target address that will receive the tokens

     \* *@param* tokenId uint256 ID of the token to be transferred

     \* *@param* \_data bytes optional data to send along with the call

     \* *@return* bool whether the call correctly returned the expected magic value

     \*/

*function* \_checkOnERC721Received(

*address* *from*,

*address* *to*,

*uint256* *tokenId*,

*bytes* *memory* *\_data*

    ) *private* returns (*bool*) {

        if (to.isContract()) {

            try

                IERC721Receiver(to).onERC721Received(

                    \_msgSender(),

                    from,

                    tokenId,

                    \_data

                )

            returns (*bytes4* *retval*) {

                return retval == IERC721Receiver(to).onERC721Received.selector;

            } catch (*bytes* *memory* reason) {

                if (reason.length == 0) {

                    revert(

                        "ERC721: transfer to non ERC721Receiver implementer"

                    );

                } else {

                    assembly {

                        revert(add(32, reason), mload(reason))

                    }

                }

            }

        } else {

            return true;

        }

    }

    /\*\*

     \* *@dev* Hook that is called before any token transfer. This includes minting

     \* and burning.

     \*

     \* Calling conditions:

     \*

     \* - When `from` and `to` are both non-zero, ``from``'s `tokenId` will be

     \* transferred to `to`.

     \* - When `from` is zero, `tokenId` will be minted for `to`.

     \* - When `to` is zero, ``from``'s `tokenId` will be burned.

     \* - `from` and `to` are never both zero.

     \*

     \* To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].

     \*/

*function* \_beforeTokenTransfer(

*address* *from*,

*address* *to*,

*uint256* *tokenId*

    ) *internal* *virtual* {}

}

/\*\*

 \* *@title* Counters

 \* *@author* Matt Condon (@shrugs)

 \* *@dev* Provides counters that can only be incremented, decremented or reset. This can be used e.g. to track the number

 \* of elements in a mapping, issuing ERC721 ids, or counting request ids.

 \*

 \* Include with `using Counters for Counters.Counter;`

 \*/

*library* Counters {

*struct* Counter {

        // This variable should never be directly accessed by users of the library: interactions must be restricted to

        // the library's function. As of Solidity v0.5.2, this cannot be enforced, though there is a proposal to add

        // this feature: see https://github.com/ethereum/solidity/issues/4637

*uint256* \_value; // default: 0

    }

*function* current(*Counter* *storage* *counter*) *internal* *view* returns (*uint256*) {

        return counter.\_value;

    }

*function* increment(*Counter* *storage* *counter*) *internal* {

        unchecked {

            counter.\_value += 1;

        }

    }

*function* decrement(*Counter* *storage* *counter*) *internal* {

*uint256* value = counter.\_value;

        require(value > 0, "Counter: decrement overflow");

        unchecked {

            counter.\_value = value - 1;

        }

    }

*function* reset(*Counter* *storage* *counter*) *internal* {

        counter.\_value = 0;

    }

}

/\*

 \* @dev Provides information about the current execution context, including the

 \* sender of the transaction and its data. While these are generally available

 \* via msg.sender and msg.data, they should not be accessed in such a direct

 \* manner, since when dealing with meta-transactions the account sending and

 \* paying for execution may not be the actual sender (as far as an application

 \* is concerned).

 \*

 \* This contract is only required for intermediate, library-like contracts.

 \*/

/\*\*

 \* *@dev* Contract module which provides a basic access control mechanism, where

 \* there is an account (an owner) that can be granted exclusive access to

 \* specific functions.

 \*

 \* By default, the owner account will be the one that deploys the contract. This

 \* can later be changed with {transferOwnership}.

 \*

 \* This module is used through inheritance. It will make available the modifier

 \* `onlyOwner`, which can be applied to your functions to restrict their use to

 \* the owner.

 \*/

*abstract* *contract* Ownable is Context {

*address* *private* \_owner;

*event* OwnershipTransferred(

*address* *indexed* *previousOwner*,

*address* *indexed* *newOwner*

    );

    /\*\*

     \* *@dev* Initializes the contract setting the deployer as the initial owner.

     \*/

*constructor*() {

        \_setOwner(\_msgSender());

    }

    /\*\*

     \* *@dev* Returns the address of the current owner.

     \*/

*function* owner() *public* *view* *virtual* returns (*address*) {

        return \_owner;

    }

    /\*\*

     \* *@dev* Throws if called by any account other than the owner.

     \*/

*modifier* onlyOwner() {

        require(owner() == \_msgSender(), "Ownable: caller is not the owner");

        \_;

    }

    /\*\*

     \* *@dev* Leaves the contract without owner. It will not be possible to call

     \* `onlyOwner` functions anymore. Can only be called by the current owner.

     \*

     \* NOTE: Renouncing ownership will leave the contract without an owner,

     \* thereby removing any functionality that is only available to the owner.

     \*/

*function* renounceOwnership() *public* *virtual* onlyOwner {

        \_setOwner(*address*(0));

    }

    /\*\*

     \* *@dev* Transfers ownership of the contract to a new account (`newOwner`).

     \* Can only be called by the current owner.

     \*/

*function* transferOwnership(*address* *newOwner*) *public* *virtual* onlyOwner {

        require(

            newOwner != *address*(0),

            "Ownable: new owner is the zero address"

        );

        \_setOwner(newOwner);

    }

*function* \_setOwner(*address* *newOwner*) *private* {

*address* oldOwner = \_owner;

        \_owner = newOwner;

        emit OwnershipTransferred(oldOwner, newOwner);

    }

}

# Appendix C

/// *@author* Muhammad Hamza Saquib Malik

/// *@title* a simple voting application using NFTs for ballots

*contract* NFTVoting is ERC721, Ownable {

    //counter allows for unique UUIDs on each ballot

    using Counters for Counters.Counter;

    Counters.Counter *private* \_tokenIdCounter;

    //total votes minted global variable

*uint256* *public* total;

    //election conclusion event

*event* concluded(*address* *winner*, *uint256* *votes*);

    //election state

*bool* *public* active = true;

    //mappings for ownership

*mapping*(*address* => *uint256*) *internal* \_ballotId;

*address*[] *public* candidates;

*address* *public* winner;

    //constructor

*constructor*() ERC721("Ballot", "BAL") {}

    /// safeMint function is part of the openZeppelin library

    /// *@dev* Altered safeMint to maintain a global mapping, addresses=>UUIDs to conduct checks later

    /// *@notice* Added a check to make sure no user gets two ballots

*function* safeMint(*address* *to*) *public* onlyOwner {

        require(balanceOf(to) == 0, "Ballot Found");

*uint256* tokenId = \_tokenIdCounter.current();

        \_safeMint(to, tokenId);

        \_ballotId[to] = tokenId;

        \_tokenIdCounter.increment();

        total = tokenId;

    }

    //@dev Minting multiple tokens

    //@notice calls safeMint multiple times

*function* safeMintMany(*address*[] *memory* *\_addresses*) *public* onlyOwner {

*uint256* \_addressLength = \_addresses.length;

        for (*uint8* i = 0; i < \_addressLength; i++) {

            safeMint(\_addresses[i]);

        }

    }

    /// *@notice* adding candidates to the candidates array

*function* addCandidates(*address* *\_candidate*) *external* onlyOwner {

        require(\_isCandidate(\_candidate) == false, "Candidate Exists");

        candidates.push(\_candidate);

    }

    /// *@notice* Internal function to check whether an address is a candidate

*function* \_isCandidate(*address* *\_candidateAddress*)

*internal*

*view*

        returns (*bool*)

    {

*uint8* \_candidatesLength = *uint8*(candidates.length);

        for (*uint8* i = 0; i < \_candidatesLength; i++) {

            if (\_candidateAddress == candidates[i]) {

                return true;

            }

        }

        return false;

    }

    // /@notice concluding the election and determining the winner

*function* conclude() *external* onlyOwner {

        require(active = true, "Concluded");

        active = false;

*uint256* \_votes;

        (winner, \_votes) = highestVotes();

        emit concluded(winner, \_votes);

    }

    /// *@notice* calculating the highest votes

    /// *@return* candidate address with highest votes and its balance

*function* highestVotes() *public* *view* returns (*address*, *uint256*) {

*uint8* candidatesLength = *uint8*(candidates.length);

*uint256* highest = balanceOf(candidates[0]);

*uint8* pointer = 0;

        //iterating through candidates

        for (*uint8* i = 1; i < candidatesLength; i++) {

*uint256* voteCount = balanceOf(candidates[i]);

            if (voteCount > highest) {

                pointer = i;

                highest = voteCount;

            }

        }

        //checking for a draw

        //counting occurance of highest votes

*uint8* occurance = 0;

        for (*uint8* i = 0; i < candidatesLength; i++) {

*uint256* voteCount = balanceOf(candidates[i]);

            if (voteCount == highest) {

                occurance++;

            }

            if (occurance > 1) {

                ///*@dev* there wil be no winner and function returns a zero address

                return (0x0000000000000000000000000000000000000000, highest);

            }

        }

        return (candidates[pointer], highest);

    }

    /// *@param* \_candidateAddress must be a valid candidate address

    /// *@return* votes for candidate

*function* votesForCandidate(*address* *\_candidateAddress*)

*public*

*view*

        returns (*uint256*)

    {

        require(

            \_isCandidate(\_candidateAddress) == true,

            "Candidate Doesn't Exist"

        );

        return balanceOf(\_candidateAddress);

    }

    /// *@return* votes for candidate

*function* totalVotesCast() *public* *view* returns (*uint256*) {

*uint256* \_totalVotes = 0;

*uint8* \_candidatesLength = *uint8*(candidates.length);

        for (*uint8* i = 0; i < \_candidatesLength; i++) {

            \_totalVotes += votesForCandidate(candidates[i]);

        }

        return (\_totalVotes);

    }

    /// *@return* candidates

*function* allCandidates() *public* *view* returns (*address*[] *memory*) {

        return (candidates);

    }

    /// *@notice* The Vote function aids voters in sending Ballots

    /// *@dev* checks for a valid candidate address, voter account balance and whether the voter is a candidate

    /// *@param* \_candidateAddress must be a valid candidate address

*function* vote(*address* *\_candidateAddress*) *external* {

        //checking if election is still ongoing

        require(active == true, "Concluded");

        //checking if msg.sender has any ballots

        require(balanceOf(msg.sender) > 0, "No Ballots");

        //checking if address in argument is a candidate address

        require(\_isCandidate(\_candidateAddress) == true, "Invalid Candidate");

        //checking if msg.sender is a candidate

        require(\_isCandidate(msg.sender) == false, "Candidates Cannot Vote");

*uint256* tokenId = \_ballotId[msg.sender];

        \_transfer(msg.sender, \_candidateAddress, tokenId);

    }

}

# Appendix D

Front End Code

import React from 'react'

import Main from "./Main"

*function* App() {

    return (

        <div>

            <*Main* />

        </div>

    )

}

export default App

import React, { Component } from 'react'

import Grid from '@material-ui/core/Grid';

// import Box from '@material-ui/core/Box';

import Container from '@material-ui/core/Container'

import Typography from '@material-ui/core/Typography'

*class* Conclude extends Component {

    render(*props*) {

        return (

            <div>

                <*Container*

                    maxWidth='sm'

                >

                    <*Grid*

                        container

                        spacing={3}

                        justifyContent="center"

                        alignItems="center"

                    >

                        <br/>

                        <*Grid* item xs={12} sm={12}>

                            <*Typography* align="center" variant="h1" style={{paddingTop:'25px'}}>Concluded</*Typography*>

                            <br/>

                            <*Typography* align="center" variant="h2">Winner</*Typography*>

                        </*Grid*>

                        <*Grid* item xs={12} sm={12}>

                            <*Typography* align="center" variant="h5">{this.props.currentState.leader}</*Typography*>

                        </*Grid*>

                        <*Grid* item xs={12} sm = {12}>

                            <*Typography* align="center" variant="h5">Votes</*Typography*>

                            <*Typography* align="center" variant="h4">{this.props.currentState.leaderVotes}</*Typography*>

                                <br/>

                            <*Typography* align="center" variant="subtitle1">Total Votes Cast: {this.props.currentState.cast}</*Typography*>

                            <*Typography* align="center" variant="subtitle1">Total Votes Minted: {this.props.currentState.minted}</*Typography*>

                        </*Grid*>

                    </*Grid*>

                </*Container*>

            </div >

        )

    }

}

export default Conclude

import React, { Component } from 'react'

import Container from '@material-ui/core/Container'

import Typography from '@material-ui/core/Typography'

*class* Description extends *Component* {

    render() {

        return (

            <div>

                <div>

                    <*Container* maxWidth="sm">

                        <br/>

                        <br/>

                        <*Typography*

                            variant='h2'

                            align='center'

                            color='textPrimary'

                            gutterBottom >

                            NFT Voting

                        </*Typography*>

                        <*Typography*

                            variant='h5'

                            align='center'

                            color='textSecondary'

                            paragraph>

                            Use your NFT Ballot to vote for your candidate!

                        </*Typography*>

                    </*Container*>

                </div>

            </div>

        )

    }

}

export default Description

import React from 'react'

import AppBar from '@material-ui/core/AppBar'

import Toolbar from '@material-ui/core/Toolbar'

import Typography from '@material-ui/core/Typography'

import { CssBaseline } from '@material-ui/core'

import HowToVoteIcon from '@material-ui/icons/HowToVote';

*function* Header() {

    return (

        <div>

            <*CssBaseline* />

            <*AppBar* position='relative'>

                <*Toolbar*>

                    <*HowToVoteIcon* />

                    <*Typography* variant='h6'>

                     NFTVoting

                    </*Typography*>

                </*Toolbar*>

            </*AppBar*>

        </div>

    )

}

export default Header;

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import './index.css';

ReactDOM.render(

    <*App* />,document.getElementById('root')

);

import React, { Component } from 'react'

import Description from "./Description"

import Header from "./Header"

import Conclude from "./Conclude"

import Vote from "./Vote"

import Manage from "./Manage"

import { ethers } from 'ethers'

//importing smart contract ABI

import NFTVoting from "./artifacts/contracts/Voting.sol/NFTVoting.json"

//address of smart contract

*const* votingAddress = "0x5FbDB2315678afecb367f032d93F642f64180aa3"

*class* Main extends Component {

*constructor*() {

        super()

        this.state = {

            //controls rendering of manage and its children components

            view: 'user',

            candidates: true,

            conclude: false,

            //variables for controlled form components within the manage section

            //conclude

            concludeCheck: false,

            //candidates

            candidateAddress: "",

            candidateWorking: false,

            //minting

            ethAddresses: "",

            mintWorking: false,

            //vote selection

            selectedCandidate: "",

            voteWorking: false,

            //data from smart contract API

            active: true,

            electionCandidates: [],

            minted: '',

            cast: '',

            leader: '',

            leaderVotes: '',

        }

        this.handleClickDisplay = this.handleClickDisplay.bind(this)

        this.onChainHandler = this.onChainHandler.bind(this)

        this.formHandler = this.formHandler.bind(this)

        this.fetchData = this.fetchData.bind(this)

    }

    async requestAccount() {

        //requests the client to give access to their meta mask account's data

        await window.ethereum.request({ method: 'eth\_requestAccounts' });

    }

    async fetchData() {

        if (typeof window.ethereum !== 'undefined') {

*const* provider = new ethers.providers.Web3Provider(window.ethereum)

*const* contract = new ethers.Contract(votingAddress, NFTVoting.abi, provider)

            try {

                //adding 1 to count since counter is initialized with 0

*const* votes = (await contract.total()).toNumber() + 1

*let* [candidate, count] = await contract.highestVotes()

*const* allCandidates = await contract.allCandidates()

*const* totalVotes = (await contract.totalVotesCast()).toString()

*const* status = await contract.active()

                this.setState({

                    minted: votes,

                    leader: candidate,

                    leaderVotes: count.toString(),

                    electionCandidates: allCandidates,

                    cast: totalVotes,

                    active: status

                })

            }

            catch (err) {

                console.log('Error:', err.message)

            }

        }

    }

    async addCandidate(*addressToAdd*) {

        if (!*addressToAdd*) return

        if (typeof window.ethereum !== 'undefined') {

            this.setState({ candidateWorking: true })

            await this.requestAccount()

*const* provider = new ethers.providers.Web3Provider(window.ethereum)

*const* signer = provider.getSigner()

*const* contract = new ethers.Contract(votingAddress, NFTVoting.abi, signer)

            try {

*const* transaction = await contract.addCandidates(*addressToAdd*)

                await transaction.wait()

            }

            catch (err) {

                console.log('Error:', err.message)

            }

        }

        this.fetchData()

        this.setState({ candidateWorking: false })

    }

    async mintTokens(*addressToMint*) {

        if (!*addressToMint*) return

        if (typeof window.ethereum !== 'undefined') {

            this.setState({ mintWorking: true })

            await this.requestAccount()

*const* provider = new ethers.providers.Web3Provider(window.ethereum)

*const* signer = provider.getSigner()

*const* contract = new ethers.Contract(votingAddress, NFTVoting.abi, signer)

            if (*addressToMint*.length > 43) {

                contract.once("minted", (*minted*) *=>* {

                    console.log(*minted*);

                })

                try {

*const* arrayOfAddresses = *addressToMint*.split(",")

                    console.log(arrayOfAddresses)

*const* transaction = await contract.safeMintMany(arrayOfAddresses)

                    await transaction.wait()

                }

                catch (err) {

                    console.log('Error:', err.message)

                }

            }

            else {

                try {

*const* transaction = await contract.safeMint(*addressToMint*)

                    await transaction.wait()

                }

                catch (err) {

                    console.log('Error:', err.message)

                }

            }

        }

        this.fetchData()

        this.setState({ mintWorking: false })

    }

    async conclude() {

        if (!this.state.concludeCheck) return

        if (typeof window.ethereum !== 'undefined') {

            await this.requestAccount()

*const* provider = new ethers.providers.Web3Provider(window.ethereum)

*const* signer = provider.getSigner()

*const* contract = new ethers.Contract(votingAddress, NFTVoting.abi, signer)

            try {

*const* transaction = await contract.conclude()

                await transaction.wait()

            }

            catch (err) {

                console.log('Error:', err.message)

            }

        }

        this.fetchData()

    }

    async sendVote(*candidateToVote*) {

        if (!*candidateToVote*) return

        if (typeof window.ethereum !== 'undefined') {

            this.setState({ voteWorking: true })

            await this.requestAccount()

*const* provider = new ethers.providers.Web3Provider(window.ethereum)

*const* signer = provider.getSigner()

*const* contract = new ethers.Contract(votingAddress, NFTVoting.abi, signer)

            try {

*const* transaction = await contract.vote(*candidateToVote*)

                await transaction.wait()

                alert("Success!")

            }

            catch (err) {

                console.log('Error:', err.message)

            }

            this.fetchData()

            this.setState({ voteWorking: false })

        }

    }

    //function to handle writing data to the blockchain

    onChainHandler(*event*) {

*const* { parentElement } = *event*.target

        if (parentElement.id === "voteButton") {

            this.sendVote(this.state.selectedCandidate)

        }

        if (parentElement.id === "mintButton") {

            this.mintTokens(this.state.ethAddresses)

        }

        else if (parentElement.id === "addCandidateButton") {

            this.addCandidate(this.state.candidateAddress)

        }

        else if (parentElement.id === "concludeButton") {

            this.conclude()

        }

    }

    handleClickDisplay(*event*) {

*const* { parentElement } = *event*.target

        parentElement.id === 'mint' ?

            this.setState({

                mint: true,

                candidates: false,

                conclude: false

            })

            :

            parentElement.id === 'candidates' ? this.setState({

                mint: false,

                candidates: true,

                conclude: false

            }

            )

                :

                parentElement.id === 'conclude' ? this.setState({

                    mint: false,

                    candidates: false,

                    conclude: true

                })

                    :

                    this.setState({

                        view: parentElement.id

                    })

    }

    formHandler(*event*) {

*const* { name, value, checked } = *event*.target

        name === "concludeCheck" ?

            this.setState({ [name]: checked })

            :

            this.setState({ [name]: value })

    }

    componentDidMount() {

        this.requestAccount()

        this.fetchData()

    }

    render() {

        return (

            <div>

                <*Header* />

                {this.state.active && <*Description* />}

                {!this.state.active ?

                    <*Conclude*

                        currentState={this.state} /> :

                    this.state.view === "user" ? <*Vote*

                        currentState={this.state}

                        handler={this.handleClickDisplay}

                        formHandler={this.formHandler}

                        onChainHandler={this.onChainHandler}

                    />

                        :

                        <*Manage* onChainHandler={this.onChainHandler} formHandler={this.formHandler} displayHandler={this.handleClickDisplay} currentState={this.state} />

                }

            </div>

        )

    }

}

export default Main

import React, { Component } from 'react'

import Grid from '@material-ui/core/Grid';

import Box from '@material-ui/core/Box';

import { styled } from '@material-ui/core/styles';

import Button from '@material-ui/core/Button';

import Container from '@material-ui/core/Container'

import ButtonGroup from '@material-ui/core/ButtonGroup';

import ManageMint from "./ManageMint"

import ManageCandidates from './ManageCandidates';

import ManageConclude from './ManageConclude';

*class* Manage extends Component {

    render(*props*) {

*const* AdminButton = styled(Button)({

            background: 'linear-gradient(30deg, #6699cc 30%, #6f9dbe 70%)',

            border: 0,

            borderRadius: 1,

            color: 'white',

            height: 30,

            minWidth: '80px'

        })

        return (

            <div>

                <*Container*

                    maxWidth='sm'

                >

                    <*Box* textAlign='center'>

                        <*AdminButton* id="user" onClick={this.props.displayHandler}>View</*AdminButton*>

                    </*Box*>

                    <br />

                    <*Grid*

                        container

                        spacing={3}

                        justifyContent="center"

                        alignItems="center"

                    >

                        <*Grid* item xs={12} sm={12}>

                            <*ButtonGroup*

                                color="default"

                                fullWidth={true}

                            >

                                <*Button* id="mint" onClick={this.props.displayHandler}>

                                    Mint

                                </*Button*>

                                <*Button* id="candidates" onClick={this.props.displayHandler}>

                                    Candidates

                                </*Button*>

                                <*Button* id="conclude" onClick={this.props.displayHandler}>

                                    Conclude

                                </*Button*>

                            </*ButtonGroup*>

                        </*Grid*>

                    </*Grid*>

                    {this.props.currentState.mint && <*ManageMint* currentState={this.props.currentState} formHandler={this.props.formHandler} onChainHandler={this.props.onChainHandler} />}

                    {this.props.currentState.candidates && <*ManageCandidates* currentState={this.props.currentState} formHandler={this.props.formHandler} onChainHandler={this.props.onChainHandler} />}

                    {this.props.currentState.conclude && <*ManageConclude* currentState={this.props.currentState} formHandler={this.props.formHandler} onChainHandler={this.props.onChainHandler} />}

                </*Container*>

            </div >

        )

    }

}

export default Manage

import React, { Component } from 'react'

import Box from '@material-ui/core/Box';

import TextField from '@material-ui/core/TextField'

import Grid from '@material-ui/core/Grid';

import Button from '@material-ui/core/Button';

import Typography from '@material-ui/core/Typography'

import { styled } from '@material-ui/core/styles';

*class* ManageCandidates extends Component {

    render() {

*const* MyButton = styled(Button)({

            background: 'linear-gradient(30deg, #6699cc 30%, #6f9dbe 70%)',

            border: 0,

            borderRadius: 3,

            boxShadow: '0 0 1px 1px  #29465b ',

            color: 'white',

            height: 48,

            marginTop: '30px',

            padding: '0 30px',

        })

        return (

            <div>

                <*Grid*

                    container

                    spacing={3}

                    justifyContent="center"

                    alignItems="center"

                >

                    <*Grid* item xs={12} sm={12}>

                        <*Typography* align="center" variant="h3">Add Candidates</*Typography*>

                    </*Grid*>

                    <*Grid* item xs={6} sm={6}>

                        <*Typography* align="right" variant="subtitle1">Candidate Address:</*Typography*>

                    </*Grid*>

                    <*Grid* item xs={6} sm={6}>

                        <*Box* align="left">

                            <*TextField*

                                name="candidateAddress"

                                value={this.props.currentState.candidateAddress}

                                placeholder="0x0"

                                onChange={this.props.formHandler}

                            /> </*Box*>

                    </*Grid*>

                </*Grid*>

                <*Box* textAlign='center'>

                    <*MyButton* id="addCandidateButton" onClick = {this.props.onChainHandler}>Add Candidate</*MyButton*>

                    <br /><br />

                    <*Typography* variant="subtitle2" >{this.props.currentState.candidateWorking && "Working..."}</*Typography*>

                </*Box*>

            </div>

        )

    }

}

export default ManageCandidates

import Button from '@material-ui/core/Button';

import Typography from '@material-ui/core/Typography'

import { styled } from '@material-ui/core/styles';

import React, { Component } from 'react'

import Box from '@material-ui/core/Box';

import Grid from '@material-ui/core/Grid';

import CheckBox from '@material-ui/core/Checkbox'

*class* ManageConclude extends Component {

    render() {

*const* MyButton = styled(Button)({

            background: 'linear-gradient(30deg, #6699cc 30%, #6f9dbe 70%)',

            border: 0,

            borderRadius: 3,

            boxShadow: '0 0 1px 1px  #29465b ',

            color: 'white',

            height: 48,

            marginTop: '30px',

            padding: '0 30px',

        })

        return (

            <div>

                <*Grid*

                    container

                    spacing={3}

                    justifyContent="center"

                    alignItems="center"

                >

                    <*Grid* item xs={12} sm={12}>

                        <*Typography* align="center" variant="h3">Conclude Election</*Typography*>

                    </*Grid*>

                    <*Grid* item xs={12} sm={12} align="center">

                        <*CheckBox*

                            checked={this.props.currentState.concludeCheck}

                            onChange={this.props.formHandler}

                            name="concludeCheck"

                            color="primary"

                            label="When you press conclude, no more votes will be accepted"

                        />

                        <*Typography* align="center" variant="subtitle1">\*When you press conclude, no more votes will be accepted</*Typography*>

                    </*Grid*>

                </*Grid*>

                <*Box* textAlign='center'>

                    <*MyButton* id="concludeButton" size="medium" onClick = {this.props.onChainHandler}>Conclude</*MyButton*>

                </*Box*>

            </div>

        )

    }

}

export default ManageConclude

import React, { Component } from 'react'

import Box from '@material-ui/core/Box';

import TextField from '@material-ui/core/TextField'

import Grid from '@material-ui/core/Grid';

import Button from '@material-ui/core/Button';

import Typography from '@material-ui/core/Typography'

import { styled } from '@material-ui/core/styles';

*class* ManageMint extends Component {

    render(*props*) {

*const* MyButton = styled(Button)({

            background: 'linear-gradient(30deg, #6699cc 30%, #6f9dbe 70%)',

            border: 0,

            borderRadius: 3,

            boxShadow: '0 0 1px 1px  #29465b ',

            color: 'white',

            height: 48,

            marginTop: '30px',

            padding: '0 30px',

        })

        return (

            <div>

                <*Grid*

                    container

                    spacing={3}

                    justifyContent="center"

                    alignItems="center"

                >

                    <*Grid* item xs={12} sm={12}>

                        <*Typography* align="center" variant="h3">Minting</*Typography*>

                    </*Grid*>

                    <*Grid* item xs={12} sm={12}>

                        <*Typography* align="center" variant="subtitle1">Enter Voter Address</*Typography*>

                    </*Grid*>

                    <*Grid* item xs={12} sm={12}>

                        <*Box* textAlign='center'>

                            <*TextField* onChange={this.props.formHandler}

                                name="ethAddresses"

                                label="ETH Addressses"

                                multiline

                                placeholder="0x0,0x0"

                                variant="outlined"

                                align="center"

                                value={this.props.currentState.ethAddresses}

                            />

                        </*Box*>

                    </*Grid*>

                    <*Grid* item xs={12} sm={12}>

                        {/\* {this.props.item.leaderVotes} \*/}

                        <*Typography* align="center" variant="subtitle2">Total Votes Minted: {this.props.currentState.minted}</*Typography*>

                        <*Box* textAlign='center'>

                            <*MyButton* id="mintButton" onClick={this.props.onChainHandler}>MINT</*MyButton*>

                        </*Box*>

                    </*Grid*>

                    <*Grid* item xs={12} sm={12}>

                        <*Typography* align="center" variant="subtitle2">{this.props.currentState.mintWorking && "Minting..."}</*Typography*>

                    </*Grid*>

                </*Grid*>

            </div>

        )

    }

}

export default ManageMint

import React, { Component } from 'react'

import Grid from '@material-ui/core/Grid';

import Box from '@material-ui/core/Box';

import { styled } from '@material-ui/core/styles';

import Button from '@material-ui/core/Button';

import Container from '@material-ui/core/Container'

import Typography from '@material-ui/core/Typography'

import Select from '@material-ui/core/Select';

import MenuItem from '@material-ui/core/MenuItem';

import InputLabel from '@material-ui/core/InputLabel';

*class* Vote extends *Component* {

    render(*props*) {

*const* candidates = this.props.currentState.electionCandidates.map(*candidate* *=>* {

            return (

                <*MenuItem* value={candidate} key={candidate}>{candidate}</*MenuItem*>)

        })

*const* MyButton = styled(Button)({

            background: 'linear-gradient(30deg, #6699cc 30%, #6f9dbe 70%)',

            border: 0,

            borderRadius: 3,

            boxShadow: '0 0 1px 1px  #29465b ',

            color: 'white',

            height: 48,

            marginTop: '30px',

            padding: '0 30px',

        })

*const* AdminButton = styled(Button)({

            background: 'linear-gradient(30deg, #6699cc 30%, #6f9dbe 70%)',

            border: 0,

            borderRadius: 1,

            color: 'white',

            height: 30,

            minWidth: '80px'

        })

        return (

            <div>

                <*Container*

                    maxWidth='sm'

                >

                    <*Box* textAlign='center'>

                        <*AdminButton* id="admin" onClick={this.props.handler}>Manage</*AdminButton*>

                    </*Box*>

                    <br />

                    <*Grid*

                        container

                        spacing={3}

                        justifyContent="center"

                        alignItems="center"

                    >

                        <*Grid* item xs={12} sm={6}>

                            <*Typography* align="center" variant="h6">Votes Minted</*Typography*>

                            <*Typography* align="center" variant="h4">{this.props.currentState.minted}</*Typography*>

                        </*Grid*>

                        <*Grid* item xs={12} sm={6}>

                            <*Typography* align="center" variant="h6">Votes Cast</*Typography*>

                            <*Typography* align="center" variant="h4">{this.props.currentState.cast}</*Typography*>

                        </*Grid*>

                        <*Grid* item xs={12} >

                            <*Typography* align="center" variant="h6">Current Leader</*Typography*>

                            <*Typography* align="center" variant="subtitle1">{this.props.currentState.leader}</*Typography*>

                            <*Typography* align="center" variant="h6">{this.props.currentState.leaderVotes} Votes</*Typography*>

                        </*Grid*>

                        <*Grid* item xs={12} >

                            <*Box* textAlign='center'>

                                <*InputLabel*>Select Candidate</*InputLabel*>

                                <*Select*

                                    id="voterSelect"

                                    name="selectedCandidate"

                                    value={this.props.currentState.selectedCandidate}

                                    onChange={this.props.formHandler}

                                >

                                    {candidates}

                                </*Select*>

                            </*Box*>

                        </*Grid*>

                    </*Grid*>

                    <*Box* textAlign='center'>

                        <*MyButton* id="voteButton" onClick={this.props.onChainHandler}>Vote</*MyButton*>

                        <br /><br />

                        <*Typography* variant="subtitle2" >{this.props.currentState.voteWorking && "Working..."}</*Typography*>

                    </*Box*>

                </*Container*>

            </div >

        )

    }

}

export default Vote

# Appendix E

Commits on Aug 26, 2021

1. added mintVoteMany function and improved UI
2. flatenned Voting.sol, added imports in project file

Commits on Aug 19, 2021

1. minor bug fix

Commits on Aug 12, 2021

1. some UI changes, app is complete

Commits on Aug 10, 2021

1. added conclude functionality
2. added voting, addCandidate functionality

Commits on Aug 9, 2021

1. removed unused imports
2. removed unused imports
3. deleted some files, renamed some files
4. made all form controlled elements
5. began work on controlled forms in manage and its children

Commits on Aug 8, 2021

1. added mint, candidates and conclude components
2. added an admin menu and a voting menu
3. added a view page, manage page, app bar
4. began work on user interface

Commits on Jul 24, 2021

1. improved code readability
2. improved code readability

Commits on Jul 23, 2021

1. began front end development
2. improved general code readability
3. added unit tests for conclude function
4. fixed comments in accordance with NatSpec format

Commits on Jul 20, 2021

1. [Added another unit test for vote function, improved descriptions](https://github.com/hamzasaquib/election-dapp/commit/1bdd4d2c1ae1bbe9638844ea4e4beab62a2f1264)
2. [Added a vote function and unit tests](https://github.com/hamzasaquib/election-dapp/commit/45beb5084037e782e09fc285a144b31fd56da759)
3. [Fixed come code formatting and added a unit test for adding candidates](https://github.com/hamzasaquib/election-dapp/commit/2148f1b13fb7b7eb49e757c52c355bde3d6ba6ae)

Commits on Jul 18, 2021

1. [Created a smart Contract voting.sol, imported erc721 interface librar…](https://github.com/hamzasaquib/election-dapp/commit/0c9bf55e35677ed985751d5c69777dcd879b66ad)

**Total Commits: 24**