Course Project1 - Spring B 2023

CSE 598: Engineering Blockchain Applications

Project 1: Creating an ERC-721 Token Standard Smart Contract

Welcome to project 1 of the Engineering Blockchain Applications course. This project will help you familiarize yourself with the Ethereum (EVM- Ethereum Virtual Machine) ecosystem by writing, testing, and deploying an ERC-721 Token Standard smart contract on the Polygon Mumbai Testnet in the Solidity programming language. Through this project, you will learn foundational knowledge to begin your journey creating EVM-based Web3 projects that utilize token mechanisms including Decentralized Autonomous Organizations (DAOs) and Non-Fungible Tokens (NFTs).

Background

We will be deploying the code on Polygon Mumbai TEstnet which is a part of EVM Ecosystem. All the development is similar to Ethereal based environments but only the deployment part changes as you will be deploying in Polygon Testnet. The currency used here is MATIC instead of Ether.

Tokens are widely used in the Ethereum ecosystem to represent many things including governance power, in-game currencies, reputation points, and much more. The Ethereum ecosystem uses Ethereum Improvement Proposals (EIPs) for the community to propose standards specifying potential new features or processes for the network.

EIPs play a central role in how changes happen and are documented on Ethereum. They are the way for people to propose, debate and adopt changes. There are different types of EIPs including core EIPs for low-level protocol changes that affect consensus and require a network upgrade as well as Ethereum Request for Comments (ERCs) for application standards. One such type of ERC standards are token standards, like the ERC-721 Token Standard, which allow applications interacting with the specific ERC standard token to treat all other tokens using the same rules, which makes it easier to create interoperable applications.

The ERC standard this project will use is the ERC-721 Token Standard, the most widely used non-fungible token standard currently on the Ethereum network. Fungibility refers to the ability for each token to be treated exactly the same, in contrast to non-fungible tokens in which the token has properties that make it distinct from other sets of tokens.

A smart contract on the Ethereum Network can be considered an ERC-721 Token Contract if it implements the following methods and events (for further reading on the standard, visit https://ethereum.org/en/developers/docs/standards/tokens/ERC-721/):

For

Methods

function balanceOf(address _owner) external view returns (uint256);
function ownerOf(uint256 _tokenId) external view returns (address);
function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable;
function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;
function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
function approve(address _approved, uint256 _tokenId) external payable;
function setApprovalForAll(address _operator, bool _approved) external;
function getApproved(uint256 _tokenId) external view returns (address);
function isApprovedForAll(address _owner, address _operator) external view returns (bool);

Events

event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId);
event Approval(address indexed _owner, address indexed _approved, uint256 indexed _tokenId);
event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);

The rest of this project documentation will walk you through the steps you will need to take to get acquainted with the tools and resources that will allow you to write, test, and deploy your own ERC-721 Token Standard Smart Contract on the Polygon Mumbai Testnet.

You will be graded by submitting

1) your smart contract address,
2) the account you used to deploy the contract, and
3) the .sol code file in the project submission (so that we can call the methods and query past events to grade you).

The tasks below are separated into two categories: Set-Up Tasks and Graded Tasks. Set-Up Tasks guide you to the resources, tools, and documentation that will help you successfully complete the project. Graded Tasks define how we expect the smart contract you submit to function for us to grade. The tools and resources recommended in the Set-Up tasks will allow you to test that your smart contract is working properly so that you know what grade to expect upon submission (please ask us questions along the way, we want all smart contracts submitted to be properly functioning ERC-721 Token Standard Smart Contracts that you can be proud of)! Happy web3 development!

Set-Up Tasks

Set-Up Task 1 - MetaMask

For us to deploy a smart contract on an Polygon Mumbai Network, we will first need a way to connect to an Polygon Mumbai Network and pay the gas fees associated with deploying your smart contract (it costs the network computational power and storage to execute transitions, those engaging in transactions pay for these costs using "gas").

We recommend using the MetaMask browser extension (supported browsers include Chrome, Firefox, Brave, and Edge), as it gives us an easy-to-use interface into Ethereum Networks that can be used for this project, as well as providing you a wallet and wide-ranging interoperability with other smart contracts and ecosystems you may be interested in exploring in the future.
You can download the browser extension by going to this link: https://metamask.io/
Once you've downloaded the browser extension and gone through the set-up process, you are ready for the next task.

Set-Up Task 2 - Polygon Mumbai Testnet

We will be deploying our smart contracts to the Polygon Mumbai Testnet. The Polygon Mumbai Testnet allows us to deploy our smart contract in a live setting, without the need for real currency and other mainnet security considerations as we are engaging in development. Testnets are widely used and it is highly recommended to test and deploy smart contracts on a testnet before you deploy on the mainnet.

You can learn more about Ethereum Networks by going to this link (We will be using Polygon network):
https://ethereum.org/en/developers/docs/networks/
And Polygon Network here -
https://wiki.polygon.technology/docs/integrate/network/

To set up the Polygon Mumbai Testnet on MetaMask, open this link - https://mumbai.polygonscan.com/ . At the bottom right of this webpage, click on button saying - ***'Add Mumbai Network'*** It will also have metamask Icon at the start of the button. Then follow the pop ups to add the network.
On the main pop-up of Metamask , if you click the Ethereum Mainnet option you can now switch to the Polygon Mumbai Test Network. Once switched, you have now completed the steps necessary in this task to be able to connect to the Polygon Mumbai Test Network via your browser.

Set-Up Task 3 - Polygon Mumbai Testnet Faucet

The on-chain currency for the Polygon Mumbai Testnet is MATIC. You will need MATIC to pay gas fees when deploying your smart contract and transacting on the Polygon Mumbai network. Similar to many developer testnets, the Polygon Mumbai network has faucets that give you MATIC for free (usually up to a daily limit per address).

Working faucets as of writing this project documentation can be found at these links: https://mumbaifaucet.com/,

Once on the website, go to your MetaMask extension and copy your account address by clicking address (will prompt you to copy to clipboard). Note: you need to do this when the network you are pointing to on MetaMask is the Polygon Mumbai Test Network (not the Ethereum Mainnet). Once copied, paste your address on the faucet's website and follow the website instructions to receive MATIC. Once the faucet sends the transaction you will be able to see your balance update on your MetaMask extension.
Note: these faucets are public and many developers globally are using it to request MATIC. If the MATIC does not show up in your account immediately, do not panic. You will be able to internally write and test your smart contract using the tools that come to follow. If you successfully requested MATIC and your balance doesn't update in 24 hours, please reach out to the course team and we will resolve the issue.

Set-Up Task 4 - Remix IDE and Solidity

We recommend using the Remix IDE for this project (https://remix.ethereum.org). Remix is a browser-based IDE that easily connects to MetaMask, allowing you to develop, test, and deploy your Solidity smart contract, all directly in your browser.

The best way to get familiar with any new development platform is to explore and try it out for yourself! Remix provides you with example contracts for you to explore in the default workspace (found on the left in your File Explorers tab) and detailed documentation here: https://remix-ide.readthedocs.io/en/latest/#

We will be helping you navigate and use Remix through our Solidity development live sessions. If you would like to explore Solidity development on your own, the Solidity documentation is well detailed and can be found here: https://docs.soliditylang.org/en/v0.8.12/

Set-Up Task 5 - IPFS

The project requires you to have an understanding of IPFS and how it stores the files and images. The Interplanetary File System (IPFS) is a distributed file storage protocol that allows computers all over the globe to store and serve files as part of a giant peer-to-peer network.

Graded Tasks

You will be graded by submitting your deployed ERC-721 Smart Contract to the project submission link in Coursera/Canvas. We will be testing your smart contract by calling the required ERC-721 methods and querying the required ERC-721 events over the public Polygon Mumbai Testnet.

In week 1 of the course, we will be releasing a spreadsheet file that will give you the

parameters we expect you to use so that each method returns to us our desired output (the spreadsheet file will have a row for each student). For example, we will give you the strings we want your smart contract to return to us when we call the name() method.

Also in week 1, we will begin introducing Solidity in the live sessions to help you get set-up with your development environment, understand the Solidity programming language and how to complete the tasks below, and answer any questions related to Project 1.

The following graded tasks define what we expect the output to be when you submit your deployed ERC-721 smart contract address and we test the methods/events.

Graded Task 1 - Token Name

When called, the contract should return the string we provided you in the spreadsheet file for the name value in task 1.

Graded Task 2 - Token Symbol

This returns the symbol used to denote your token. When called, the contract should return the string we provided you in the spreadsheet file for the token symbol in task 2.

How We Are Grading task 1 and 2 :

We will fetch the NFT information by interacting with the smart contract address provided by you and parse through the key values and verify if the value obtained matches the value we provided in the sheet

Graded Task 3 - function mint(address to, string memory tokenURI) public returns (uint256)

This method returns the tokenId of the minted NFT (ERC721 Token). When called, the contract should mint a new NFT by passing the address to and token URI.

You can mint multiple NFTs to different addresses.
tokenURI - This tokenURI is discussed in the next Task

Graded Task 4-6 – tokenURI

The tokenURI is URL of the JSON file which contains metadata for your NFT. You need to use a IPFS service (there are multiple options available and you can use any to get the URL for the file)
This task includes comprises of three graded tasks present in the JSON file as a key value pair-

- *Task 4 - name*
  - The key - ‘name’ should have the value we provided you in the

spreadsheet file for the name in task 4.

- *Task 5 – description*
  - The key - 'description' should have the value we provided you in the spreadsheet file for the 'description' in task 5.

- *Task 6 - image*
  - The key - 'image' should have the value we provided you in the spreadsheet file for the 'image' in task6.

How We Are Grading :
We will fetch the NFT metadata and parse through the key values and verify if the value obtained matches the value we provided in the sheet

Graded Task 7 – transfer NFT

This task requires you to transfer one of the NFT to the provided wallet address from the sheet under task 7. You need to use the safeTransferFrom function which is provided by deploying the smart contract.
* Also, you need to transfer at least 0.2 MATIC to this address provided to you, for the autogravder to work and call few functions.

How We Are Grading :
We will check if the provided address holds the NFT from your developed contract or not

Submission Steps

The autograder should not be used to debug or test function-by-function as you develop your erc-721 token. All testing of erc-721 functionality can happen through remix, and does not require deployment to the Polygon Mumbai test network (which costs gas). The autograder should be used when you believe you have a working erc-721 contract and have gone through all the necessary steps to be graded.

You will be graded by going to the autograder link in week 4 and doing the following:

1) submitting your smart contract address,and
2) submitting the account address you used to deploy the contract.

3) In a separate link, we ask for uploading a PDF file containing all the code of your file.

We will be grading by directly calling the methods and events on your live contract that is deployed on the testnet as specified in the instructions above. The output will be your grade, and which methods/events passed or failed. The grader automatically saves your highest score on our backend. The grader is not linked to Canvas's grading backend, so we will upload the grades all at once. Submissions are unlimited, once you are happy

with the highest score you have received from the autograder your work is done.