# FITE 1010 Introduction to financial technologies

## Lab 3: Blockchain: Smart Contract and NFT

**Deadline: March 13 23:59**

**Overview**

Non-fungible token, NFT, is one of the most popular application of smart contract last year. At a very high level, most NFTs are part of the Ethereum blockchain. Ethereum’s smart contracts store extra information that makes them work differently from, say, an ETH coin. NFTs can really be anything digital (such as drawings, music), but a lot of the current excitement is around using the technology to sell digital art.

In this lab, we will build our own smart contract and mint an NFT. You will be able to see the NFT in an NFT marketplace called OpenSea, and you can transfer it to your friend.

**Objectives**

After completing this lab, you will be able to:

* Understand what is a crypto wallet.
* Learn how to write, compile and deploy a smart contract.
* Understand what is NFT and ERC-721 contract
* Create an NFT and trade it in OpenSea.

# Exercise 1 Install Digital Wallet for Crypto – MetaMask

A crypto wallet is a place where you can securely keep your crypto. There are many different types of crypto wallets, but the most popular ones are hosted wallets, non-custodial wallets, and hardware wallets. In this lab, we will use a “non-custodial wallet” which doesn’t rely on a third party — or a “custodian” — to keep your crypto safe. We will use a popular wallet called “MetaMask”.

## Metamask

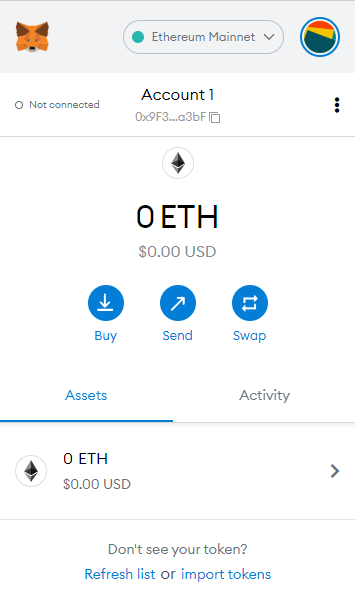
In your virtual machine (used in previous labs), open the browser Firefox.

Download the digital wallet app called “MetaMask” at <https://metamask.io>

Click the download button and select “Install MetaMask for Firefox”.

After it is installed, create a wallet in MetaMask. You can select your own password. Choose “Remind me later” for secret recovery phrase.

After installation, you should see it in your browser (next to the address bar):



The number below “Account 1” is your Ethereum mainnet address generated by the wallet.

**Question 1**. (5 mark) Capture Screenshot 1 for your MetaMask wallet and write down your address.

**Question 2**. (10 mark) Your private key is securely stored in the wallet and it will not be displayed on the screen. How is your private key related to your address?

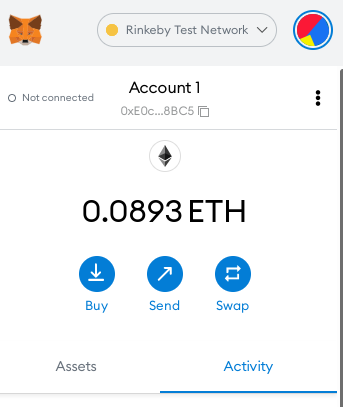
**Question 3**. (15 mark) Currently you have no money in your Ethereum mainnet address. Suggest three possible ways to get some ETH.

## Rinkeby Test Network

Press the “Ethereum Mainnet” button 🡪 Show/hide test network 🡪 Turn on the “Show test network”.

Switch to “Rinkeby Test Network” and copy your address.

Get some ETH for your testnet address from <https://faucet.rinkeby.io/> or <https://faucets.chain.link/rinkeby> or any other website that you can find. You will need at least 0.02ETH to complete the lab. You should follow the instructions from those websites to get your ETH. You will be able to see it in your MetaMask afterward.



**Question 4.** (10 mark) Why do we need to use a test network to test our smart contract in this lab, instead of using the Ethereum mainnet?

## Exercise 2 Writing a Smart Contract for NFT

A Non-Fungible Token (NFT) is used to identify something or someone in a unique way. This type of Token is perfect to be used on platforms that offer collectible items, access keys, lottery tickets, numbered seats for concerts and sports matches, etc.

The ERC-721 introduces a standard for NFT, in other words, this type of Token is unique and can have different value than another Token from the same Smart Contract, maybe due to its age, rarity or even something else like its visual. All NFTs have a uint256 variable called *tokenId*, so for any ERC-721 Contract, the pair *contract* *address*, *uint256 tokenId* must be globally unique.

In this excercise, we will create an NFT token of our own using the ERC-721 smart contract.

### Remix IDE

Remix is an IDE (Integrated Development Environment) officially provided by Ethereum. It contains the full functionality of the compiler, executing smart contracts, publishing contracts, and more. No installation is required, you can just open <https://remix.ethereum.org/> with your browser.

## Pre-requisites for installation

Install NodeJS and npm if they are not installed. Open the terminal and run:

sudo apt install nodejs

sudo apt install npm

## OpenZeppelin

OpenZeppelin is an open-source smart contract repository with many smart contract templates, including ERC-721 smart contracts. By using the OpenZeppelin's ERC-721 template, we just need to extend our own specific functions.

Install OpenZeppelin by:

cd /home/(your VM name)/

npm init -y

sudo npm install @openzeppelin/contracts

## Remixd

Since Remix is an online IDE, we need to make some changes to link it to local storage (in case you need to save the file and update it later). We need to install an add-on called “remixd”.

Install Remixd by:

sudo npm install -g @remix-project/remixd

You can verify if you have installed it successfully.

remixd -v

Create a folder “mynft” and link it.

cd /home/(your VM name)/

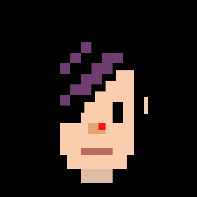
mkdir mynft

sudo remixd -s /home/(your VM name)/ -u https://remix.ethereum.org/

## Creating metadata file with IPFS

We will upload an image and the metadata for our NFT token in the IPFS.

Draw/download your own small icon for your future NFT.



Create a folder in the IPFS user interface and put the jpg there. Copy the ipfs address for the image.

Create a JSON file called “0” in this folder, with the content:

{

"attributes": [

{

"trait\_type": "Hair color ",

"value": "Black"

},

{

"trait\_type": "Eye color",

"value": "Mocha"

}

],

"description": " <describe yourself here>",

"image": "[https://ipfs.io/ipfs/<the ipfs hash of the photo is put here>](https://ipfs.io/ipfs/%3cthe%20ipfs%20hash%20of%20the%20photo%20is%20put%20here%3e)",

"name": "<your name>"

}

You can create the file by opening a terminal in desktop and call “nano 0” (Refer to Lab 3 for creating a file). Feel free to change/add the attributes to match the description your icon. This file will be the Metadata file of your future NFT token.

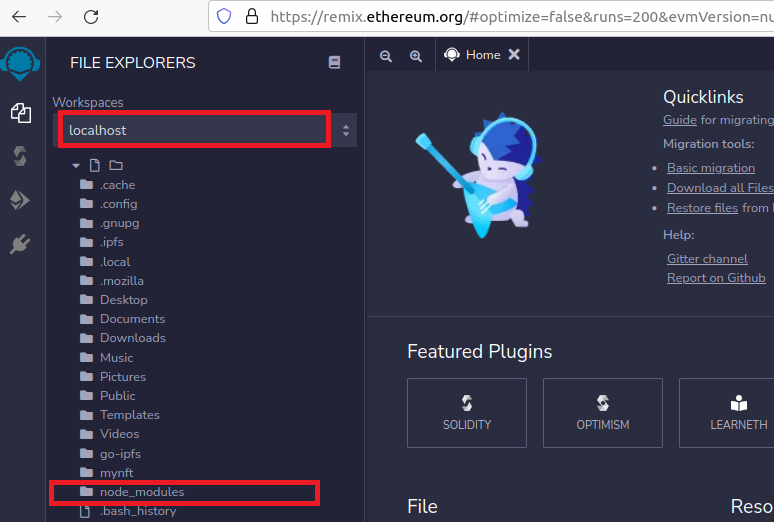
Save the file “0”, and upload it in your ipfs folder.

Copy the ipfs hash of the folder (not the file “0” or the image).

## Writing Smart Contract

Open Remix IDE in Firefox: <https://remix.ethereum.org/>

In the button “Workspaces”, switch to “localhost”. The folder should contain node\_modules installed by OpenZeppelin.



Right click “mynft” and create “FITE1010.sol”. Then open the “FITE1010.sol”.

Now you can create your first ERC721 contract by using the template before:

// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.4;

import "../node\_modules/@openzeppelin/contracts/token/ERC721/presets/ERC721PresetMinterPauserAutoId.sol";

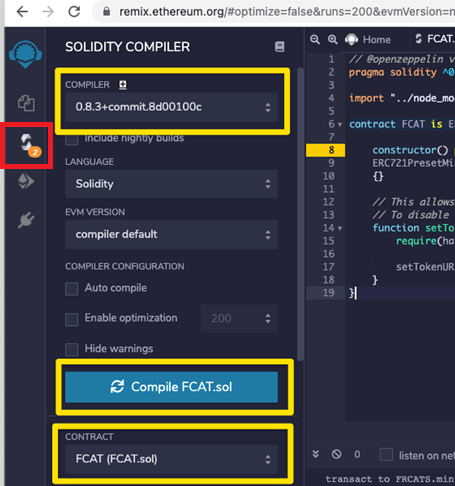
contract FITE1010 is ERC721PresetMinterPauserAutoId {  
   
 constructor()   
 ERC721PresetMinterPauserAutoId("<describe yourself here>", "<your name>", "[https://ipfs.io/ipfs/<your ipfs folder address>/](https://ipfs.io/ipfs/%3cyour%20ipfs%20folder%20address%3e/)")   
 {}  
   
 // This allows the minter to update the tokenURI after it's been minted.  
 // To disable this, delete this function.  
 function setTokenURI(uint256 tokenId, string memory tokenURI) public {  
 require(hasRole(MINTER\_ROLE, \_msgSender()), "web3 CLI: must have minter role to update tokenURI");  
   
 setTokenURI(tokenId, tokenURI);  
 }  
}

The above code is built using OpenZeppelin's ERC-721 contract. We simply set our NFT name, token code and token MetaData URI by the function input to ERC721PresetMinterPauserAutoId.

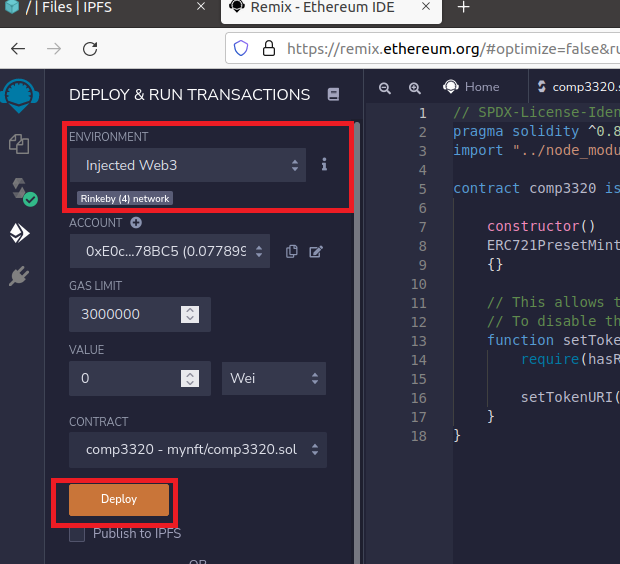
**Question 5.** (10 mark) What is the main difference between the ERC-721 contract and the standard ERC-20 contract?

## Exercise 3 Compiling and Deploying a Smart Contract for NFT

After editing, you can switch to the Compile panel to compile. Be careful to choose the correct compiler version, the compiler version must be the same as the OpenZeppelin contract you are using, otherwise a compilation error will occur.



Once the compilation is complete, it is ready to be deployed. Switch to the Deploy panel. At this point, we need to select [Injected Web 3] in the Environment in the upper left corner, and MetaMask will pop up to link.



There are many testnets for Ethereum, such as Ropsten, Rinkeby, Goerli, etc. To be able to test on OpenSea (an NFT trading platform) in the next excercise, we chose the Rinkeby testnet. To use the test network, remember to select Rinkeby Test Network in MetaMask.

After everything is ready, you can press [Deploy] to start publishing your smart contract, and MetaMask will pop up for payment confirmation.

After the deployment is completed, Remix will display the corresponding Transaction Hash. The corresponding Transaction can be seen on https://rinkeby.etherscan.io, and the ERC721 smart contract address we created can also be seen in it.

**Question 6.** (10 mark) Capture thescreenshots of your deployed contract:

(1) in remix,

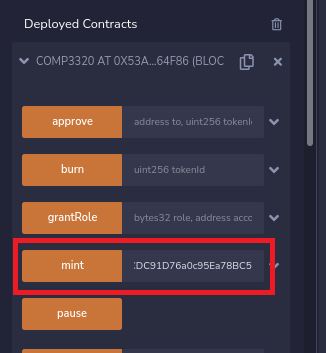
(2) in<https://rinkeby.etherscan.io>.

**Question 7.** (10 mark) Is there any change in your MetaMask? Why?

## Exercise 4 Minting an NFT

After the contract is deployed, the next step is to Mint NFT.

Back to Remix, you can see that the smart contract we just released already exists in Deployed Contracts. After clicking on it, you can see that there is a function of mint in it. Now we can call this function to Mint NFT.



Enter your Rinkeby address in “address to” after the “mint” function, and this address will become the owner of the newly created NFT.

**Question 8.** (10 mark) Is there any change in your MetaMask? Why?

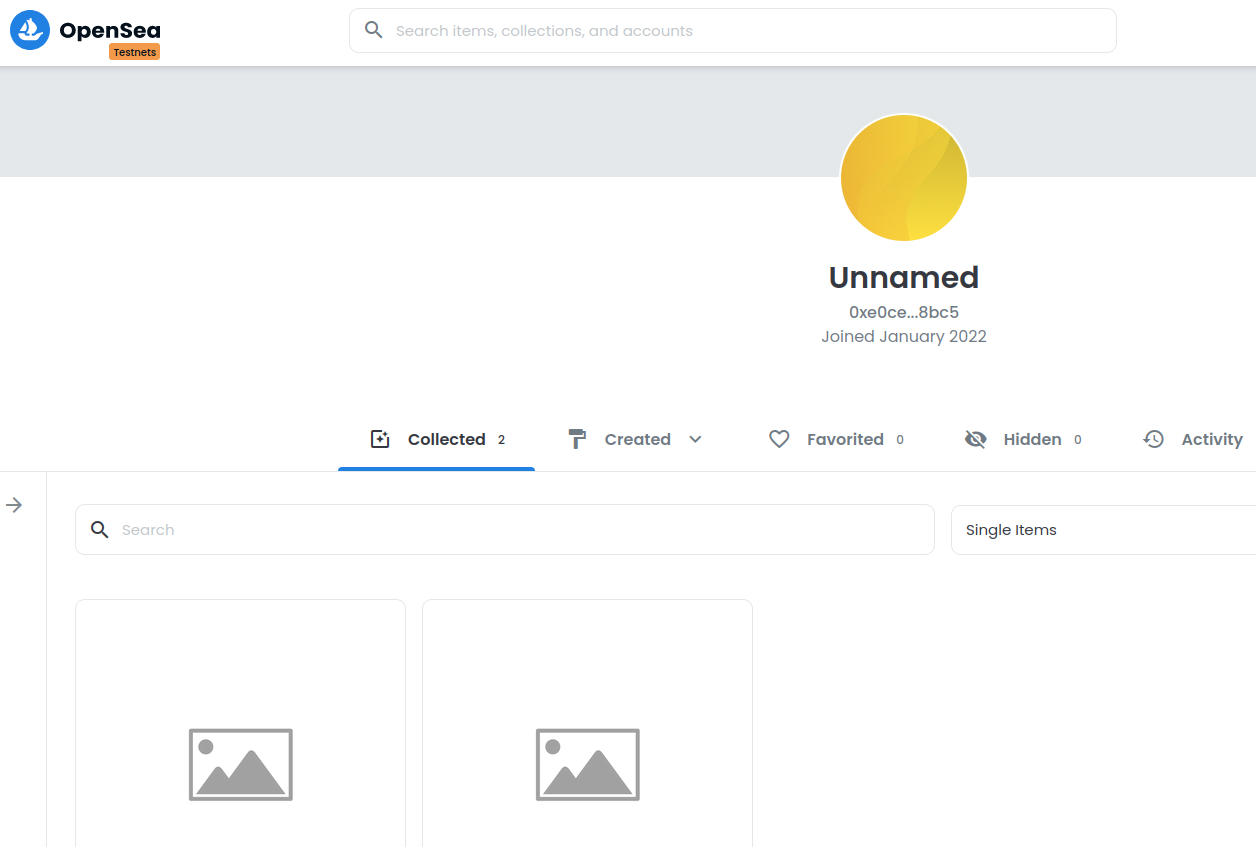
## Exercise 5 Viewing your NFT in OpenSea

OpenSea (https://opensea.io) is a popular marketplace for NFT. But now we are issuing NFTs on the testnet, so we use the following URL instead:

<https://testnets.opensea.io/>

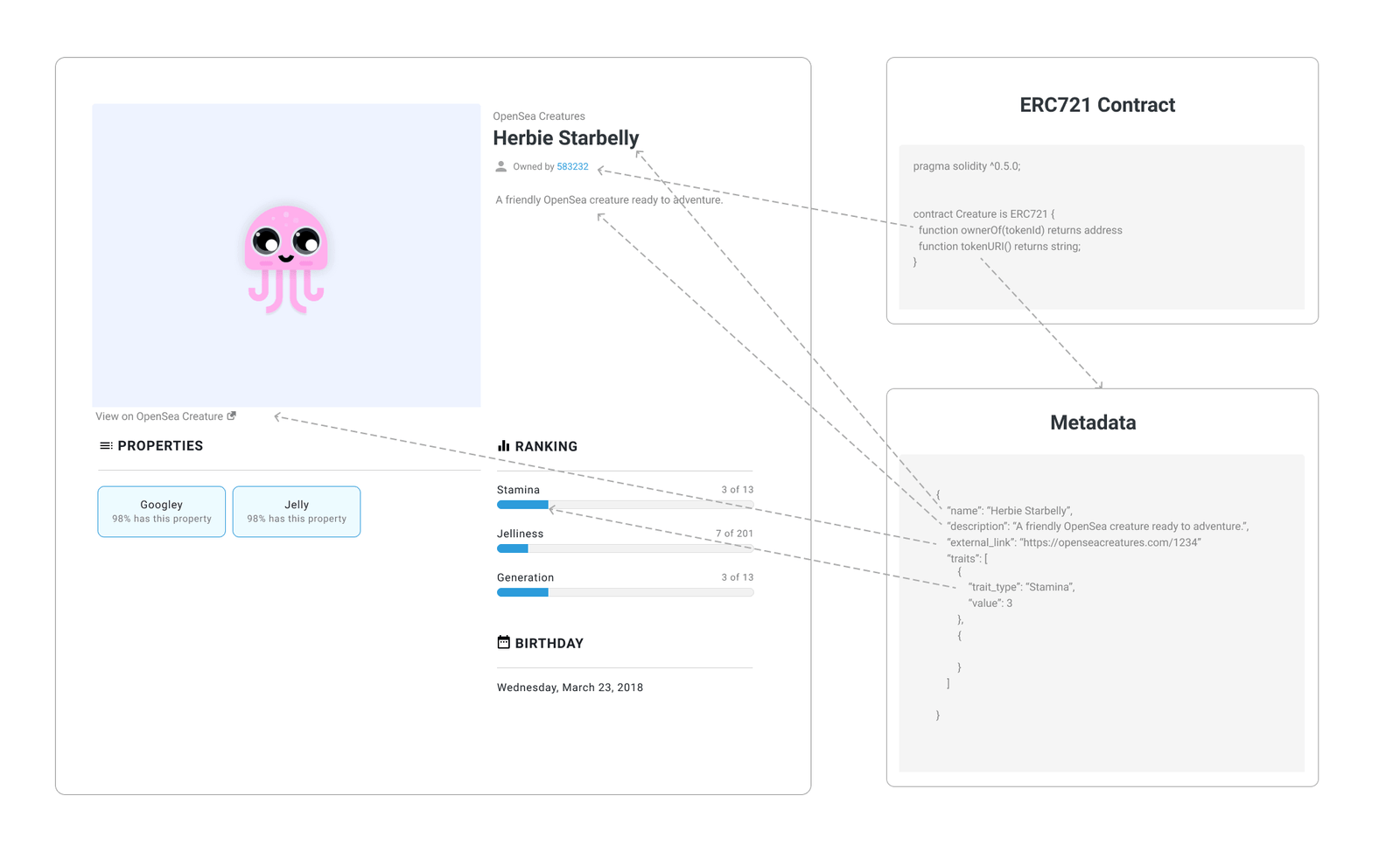
After entering, register an account with your own MetaMask address and you can see the NFT you just minted. MetaMask will automatically pop up when you register an account.

When you click your profile, you will see your NFT there.



The image and other attributes corresponding to the NFT are determined by the Metadata of the NFT. In order for the NFT to display the image and other attributes in OpenSea, the Metadata of the NFT needs to be specified.

According to the ERC-721 Metadata standard on OpenSea, OpenSea will call the tokenURI function in our smart contract and pass in the tokenID, and this function needs to return an HTTP or IPFS URL, which must return data in JSON format, such as in the JSON file in IPFS in Exercise 2, and this JSON defines various properties of our NFT.

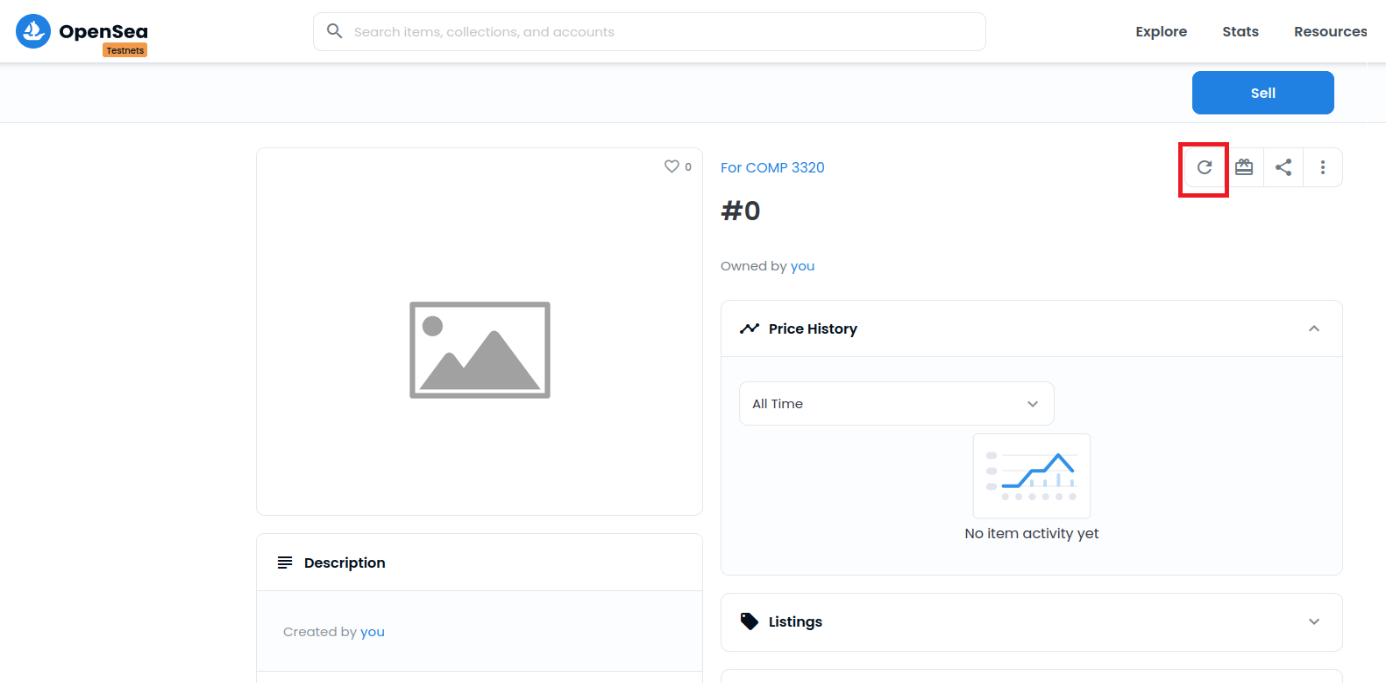
[](https://www.frank.hk/static/427f6be0a2bc59eb682dfd05dc3cfbc1/ee9e5/meta.png)

Taking the smart contract we wrote above as an example, notice that we have the following in the constructor:

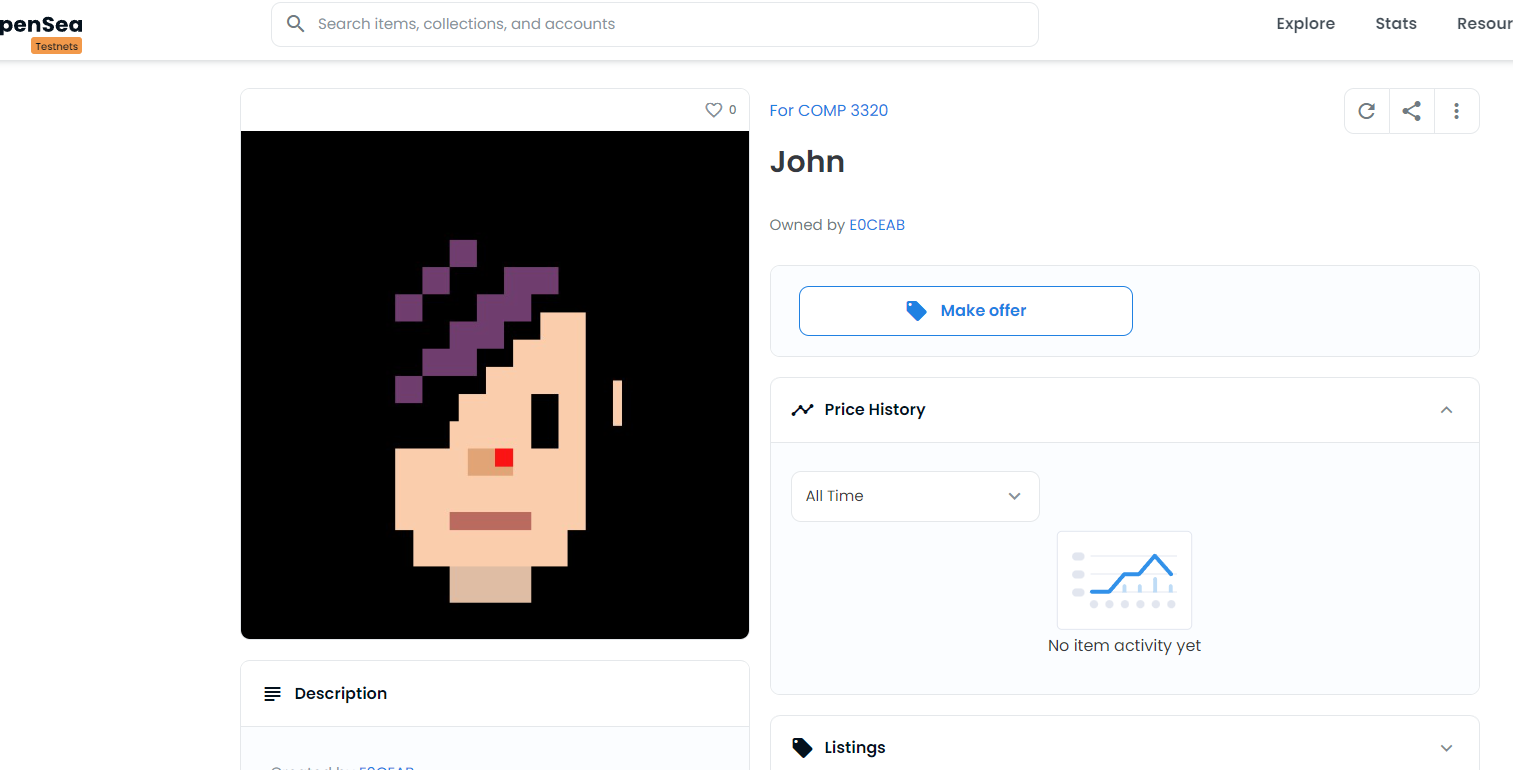
ERC721PresetMinterPauserAutoId("<describe yourself here>", "<your name>", "https://ipfs.io/ipfs/<your ipfs folder address>/")

The base URI of our NFT is https://ipfs.io/ipfs/<your ipfs folder address>/. When you call the tokenURI function, our contract will return https://ipfs.io/ipfs/<your ipfs folder address>/<tokenID>. If this Metadata is set, the mint NFT will have attributes such as images.

Click on the NFT you just created. You should see something like this:

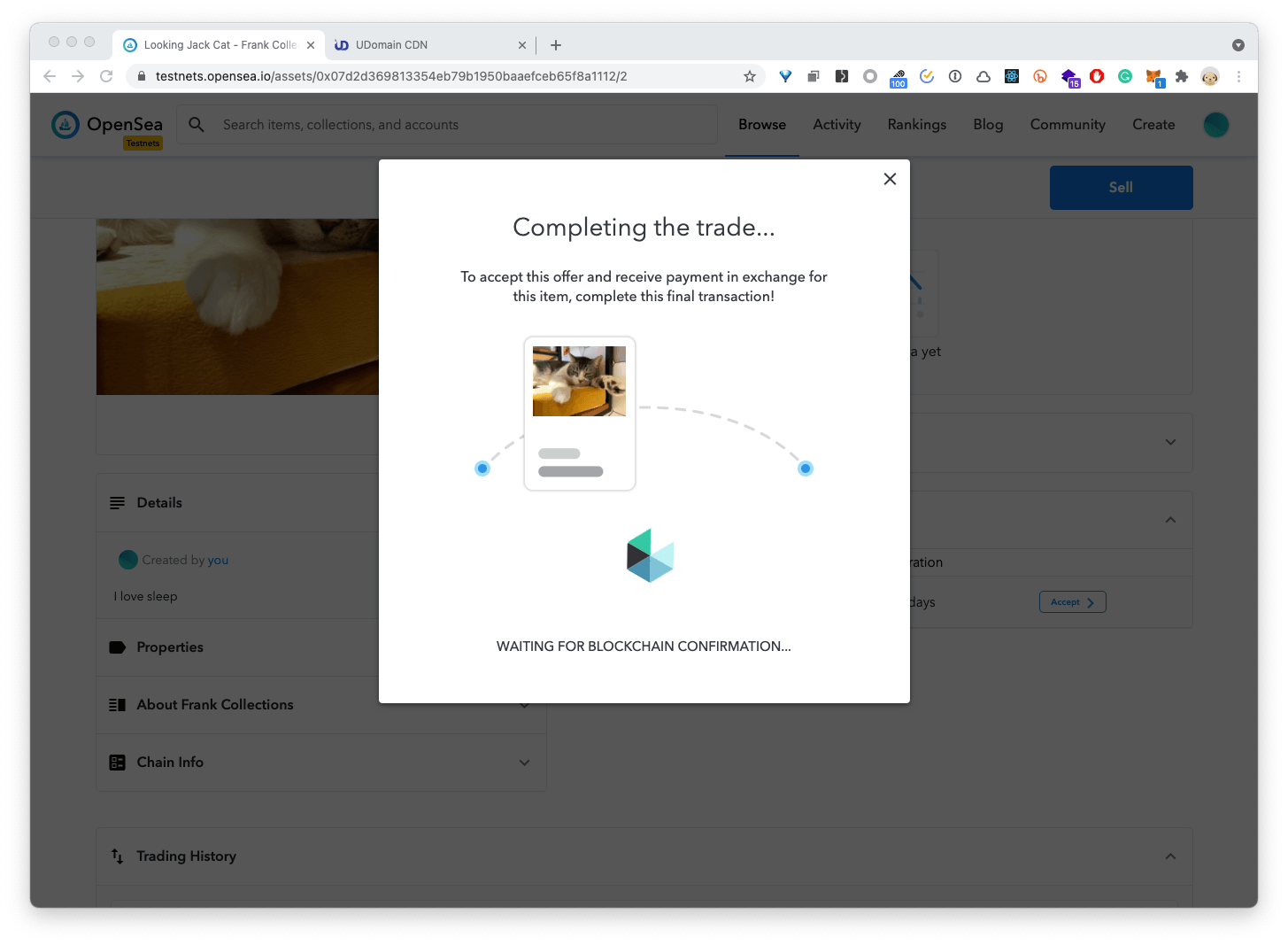


Click on the reload button. You should be able to see your icon later (You may need to wait for 5-10 minutes).



**Question 9.** (10 mark) Capture thescreenshot of your NFT icon and attributes in OpenSea.

Congratulations! You have created your own NFT. Of course, the NFT you built can also be traded on OpenSea.

[](https://www.frank.hk/static/2800aac7ea14fdb0b0b10238b076b683/e5a51/screen-20210325155621.png)

**Question 10.** (10 mark) Create another address in MetaMask (using the create account button in MetaMask) or ask your classmate for an address. Send the NFT token to him/her in OpenSea. Capture thescreenshot of the transaction.

**End of lab 4**