UDACITY

Logout

PROJECT SPECIFICATION

Decentralized Star Notary

Add Smart Contract Functions

CRITERIA	MEETS SPECIFICATIONS

The smart contract tokens should have a name and a symbol.	• Add a name and a symbol to the starNotary tokens.
	• In the Starter Code (StarNotary.sol file) you implement:
	`// Implement Task 1 Add a name and symbol properties`
	`// name: Is a short name to your token`
	`// symbol: Is a short string like 'USD' -> 'American Dollar'`
	
	Note: The project starter codes use solidity v0.4.21. Please ensure to write the constructors accordingly.

CRITERIA	MEETS SPECIFICATIONS
Implement the function: **lookUptokenIdToStarInfo** in StarNotary.sol file	Add a function `lookUptokenIdToStarInfo`, that looks up the stars using the Token ID, and then returns the name of the star.

```
// Implement Task 1 lookUptokenIdToStarInfo
function lookUptokenIdToStarInfo (uint _tokenId) public view returns (string memory) {

 }
```

CRITERIA	MEETS SPECIFICATIONS

Implement the function: **exchangeStars** in StarNotary.sol file.
 	
`// Implement Task 1 Exchange Stars function`
`function exchangeStars(uint256 _tokenId1, uint256 _tokenId2) public {`
 	
`}`
	• Add a function called `exchangeStars`, so 2 users can exchange their star tokens. Do not worry about the price, just write code to exchange stars between users.
	• Check if the owner of `_tokenId1` or `_tokenId2` is in fact the sender in order to pass the star `tokenId`.

Implement the function **transferStar** in StarNotary.sol file.
 	
`function transferStar(address _to1, uint256 _tokenId) public {`
 	
`}`
	• Write a function to Transfer a Star. The function should transfer a star from the address of the caller. The function should accept 2 arguments, the address to transfer the star to, and the token ID of the star.
	• Check if the owner of `_tokenId1` or `_tokenId2` is in fact the sender in order to pass the star `tokenId`.

Add supporting Unit Tests

CRITERIA	MEETS SPECIFICATIONS

CRITERIA	MEETS SPECIFICATIONS
Add supporting Unit Tests in **TestStarNotary.js** file.	Tests for: 1) The token name and token symbol are added properly.

	`it('can add the star name and star symbol properly', async() => {`
	
	`});`

	2) 2 users can exchange their stars.

	`it('lets 2 users exchange stars', async() => {`
	
	`});`

	3) Stars Tokens can be transferred from one address to another.

	`it('lets a user transfer a star', async() => {`
	
	`});`

Deploy your Contract to Rinkeby

CRITERIA	MEETS SPECIFICATIONS
Deploy Your Contract to Public Test Network	Students must successfully deploy the contract to Rinkeby: • `truffle-config.js` file should have settings to deploy the contract to the Rinkeby Public Network. • Infura should be used in the truffle-config.js file for deployment to Rinkeby.

Modify the front end of the DAPP

CRITERIA	MEETS SPECIFICATIONS
Implement the front-end function **lookUp** in the index.js file.	When you click on the button "Look Up a Star" the application shows in the status the Star information.
`lookUp: async function (){ }`	

Add a Readme.md file

CRITERIA	MEETS SPECIFICATIONS
Inside your project folder, create a `readme.md` file.	The readme.md file should include the following: 1) Your ERC-721 Token Name 2) Your ERC-721 Token Symbol 3) Version of the Truffle and OpenZeppelin used 4) Your Token Address on the Rinkeby Network