UDACITY

‹ Return to "Blockchain Developer" in the classroom

DISCUSS ON STUDENT HUB

Decentralized Star Notary

REVIEW

CODE REVIEW 6

HISTORY

Meets Specifications

You did an excellent job on this project.👏👏

It was a great project submission.👏
I hope you keep up your spirits, motivation, dedication and hard work like this throughout the entire program and as well as your career and life.⭐⭐⭐⭐⭐

Congratulations on completing this project.👍

Add Smart Contract Functions

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'
```

✅ Add a name and a symbol to the starNotary tokens.

Add a function `lookUptokenIdToStarInfo`, that looks up the stars using the Token ID, and then returns the name of the star.

✅ Add a function lookUptokenIdToStarInfo, that looks up the stars using the Token ID, and then returns the name of the star.

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.

✅ 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.

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.

✅ 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.

Add supporting Unit Tests

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() => {

});
```

✅ All the required tests are added and all tests pass.

```
  ✓ can Create a Star (110ms)
  ✓ lets user1 put up their star for sale (120ms)
  ✓ lets user1 get the funds after the sale (154ms)
  ✓ lets user2 buy a star, if it is put up for sale (267ms)
  ✓ lets user2 buy a star and decreases its balance in ether (225ms)
  ✓ can add the star name and star symbol properly (126ms)
  ✓ lets 2 users exchange stars (248ms)
  ✓ lets a user transfer a star (114ms)
  ✓ lookUptokenIdToStarInfo test (68ms)

  9 passing (1s)
```

Deploy your Contract to Rinkeby

`truffle-config.js` file should have settings to deploy the contract to the Rinkeby Public Network.

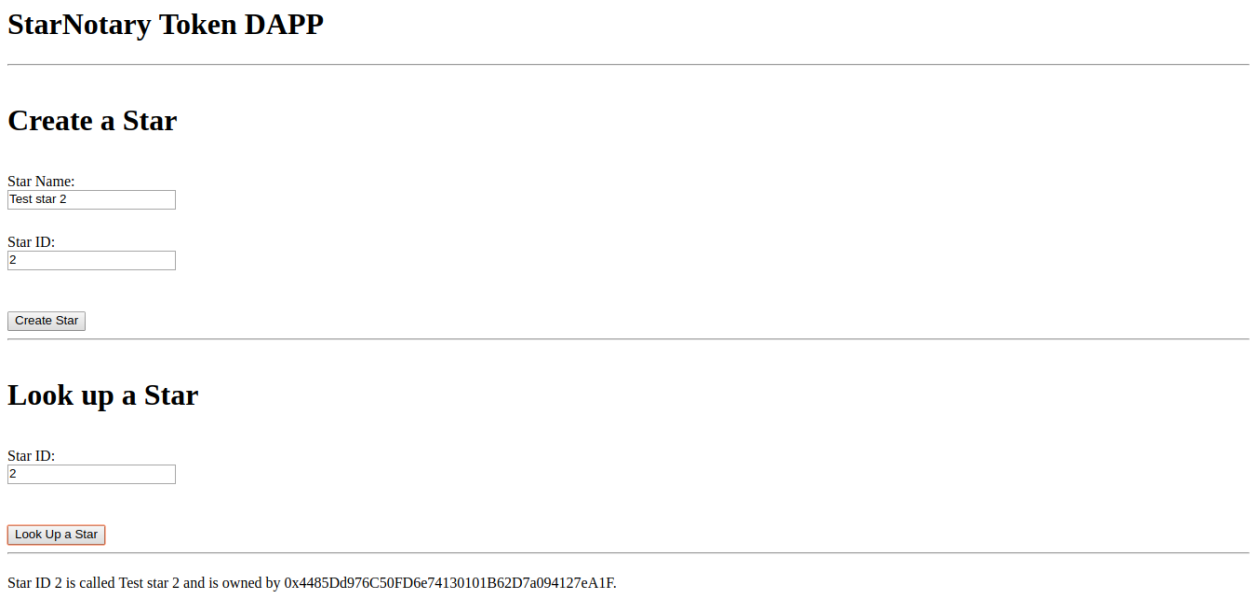
Infura should be used in the truffle-config.js file for deployment to Rinkeby.

✅ 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

When you click on the button "Look Up a Star" the application shows in the status the Star information.

✅ The front-end of the DAPP works as expected 💯



Add 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

✅ The `README` file contains all the required information.

DOWNLOAD PROJECT

6 CODE REVIEW COMMENTS ›

RETURN TO PATH