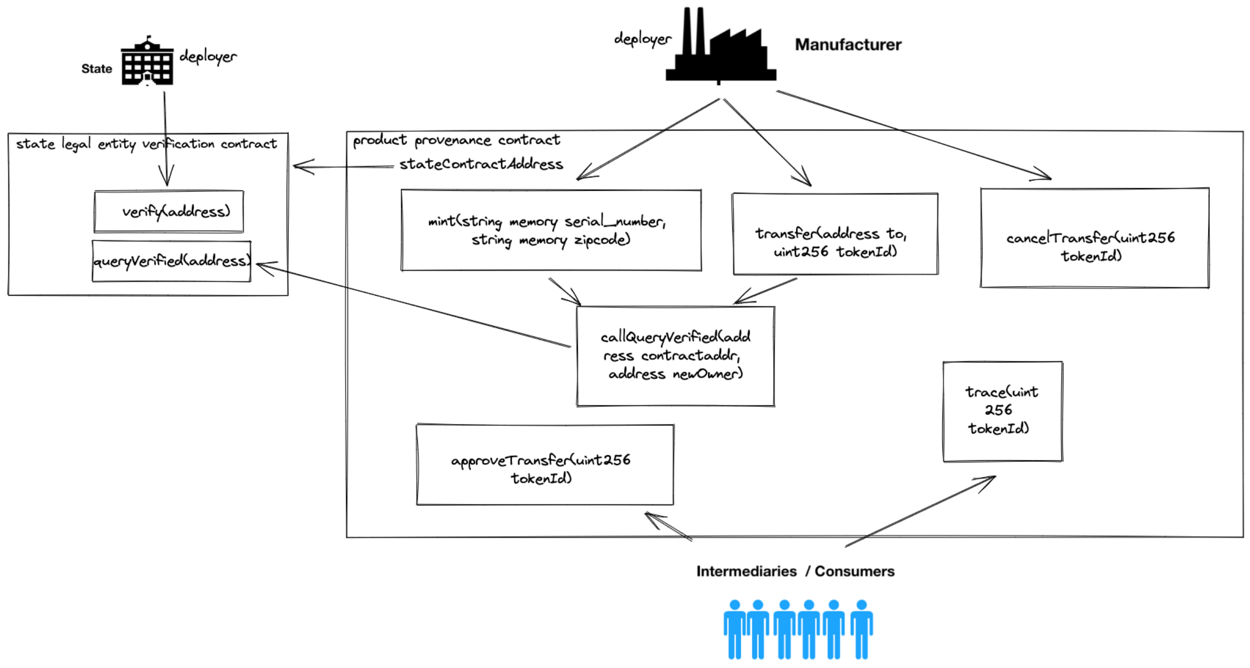
CMPE 483 ASSIGNMENT 1 REPORT

Gözde Ünver 2018400309
Emin Deniz 2017400186
Berkant Koç 2020400342

state legal entity verification contract:
https://testnet.bscscan.com/address/0x8700f4d2eaf25c35c590fcbad7a6c40ea98e131f
product provenance contract:
https://testnet.bscscan.com/address/0x161507931f6e2f945d8c1cdc2d4f3e2af5a2c624

WORKING DIAGRAM OF THE CONTRACTS


State
deployer
deployer
Manufacturer
state legal entity verification contract
verify(address)
queryVerified(address)
product provenance contract
stateContractAddress
mint(string memory serial_number, string memory zipcode)
transfer(address to, uint256 tokenId)
cancelTransfer(uint256 tokenId)
callQueryVerified(address contractaddr, address newOwner)
trace(uint 256 tokenId)
approveTransfer(uint256 tokenId)
Intermediaries / Consumers

EXPLANATIONS OF THE FUNCTIONS

state.sol

1- constructor()

When this contract is deployed this function is executed and the address of the owner of this contract (msg.sender) is stored in the "owner" address variable of the contract.

2- verify(address a)
This function is called to store the address which is registered in the state as verified. The address is saved in the “verified” mapping with the value true.

3- queryVerified(address a)
Returns true if the address is previously verified and stored present in the “verified” mapping. If it is not present in this mapping then returns false.

manufacturer.sol

1- transfer(address to, uint256 tokenId)
Takes send address and tokenId as parameters. If there is no waiting address in the “waitingtransfer” mapping for the tokenId and if the caller of the function is the last owner of this tokenId then this tokenId is stored in the waitingtransfer mapping with the value of the “to” address. In other words, until the other address accepts this transfer, the tokenId is not simply assigned to them, instead the tokenId waits in the waitingtransfer mapping.

2- approveTransfer(uint256 tokenId)
If the caller of this function is the waited address of the token then this function continues to its execution. By calling this function, the address approves the transaction and the transfer operation actually occurs via _transfer() function of the ERC721.sol library. The new owner is stored in the “prev_owners” mapping as an Owner object. This object stores the address of the owner and the status field of its verification at the moment. Lastly, the tokenId is deleted from the waitingtransfers mapping because the transfer has been accomplished.

3- constructor(address _stateContractAddress)
When this contract is deployed this function is executed with the “state” contract address as a parameter. The address of the owner of this contract (msg.sender) is stored in the “manufacturer” address variable of the contract and the state address is stored in “stateContractAddress” variable in order to query verified status of the owners.

4- function mint(string memory serial_number, string memory zipcode) public
This function is called with serial_number and zipcode parameters. They represent a unique product. The function checks if the caller of the function is the manufacturer(owner of the contract) or not. Also it checks the uniqueness of the serial number. It uses Counters Library for increment of tokenIDs. Hash of serial number and zipcode is stored in a map for tokens. Also, it adds the manufacturer as first owner of the token with verified status of the new owner. This verification is done by calling the callQueryVerified function.

5- function callQueryVerified(address contractaddr, address newOwner) internal
This function calls querVerify function of the state contract. The state contract address is given as a parameter for this function. It returns a boolean value.

6- function cancelTransfer(uint256 tokenId) public virtual
When a transfer is started but the receiver does not approve it, the sender can cancel the transfer by removing the token from waitingtransfers mapping. We added this function because we do not want any token to be locked when it is sent to the wrong address etc. Also, we thought the reject function but reject needs gas for execution, if someone wants to reject a transaction, they do not want to pay any gas for rejection.

7- function trace(uint256 tokenId) public view returns (Owner[] memory)
It returns the owner history of the token that has the ID given as a parameter for the function. Owner history is an array that consists of Owner records. Owner record is {add: address , verified:bool}.

*After the initialization of truffle, Migrations.sol contract is automatically created. The purpose of this contract is to keep the list of migrations that are done on the current network.

GAS USAGE

We calculated gas usage by executing the functions and checking the gas usage in the BSCScan.

State Contract Deployment Gas Usage:
https://testnet.bscscan.com/tx/0xdc992983b69bd019eb584b20e91c0a8b02d24ab14e4b2d54c45fa77c56f54eb3
Gas Used by Transaction:
2,179,190

State Contract "verify" Gas Usage:
https://testnet.bscscan.com/tx/0x020e2cf68d5ba1d1cc84c260b5048901fa32cf676b9d976ed21786269a425792
Gas Used by Transaction:
43,759

State Contract "queryVerified" Gas Usage:
View function, no gas usage

Product Contract Deployment Gas Usage:
https://testnet.bscscan.com/tx/0x4d0bc148f3d3f4a6128128945ce30c9970372aa4745482ac89a3a5a27a5e7fcb
Gas Used by Transaction:
3,869,651

Product Contract "mint" Gas Usage:
https://testnet.bscscan.com/tx/0x152c4017b8f92b2b419cbef5c4cbd83cfcfe662ff5e79b46b2590e777dd15030
Gas Used by Transaction:
343,697

Product Contract "transfer" Gas Usage:
https://testnet.bscscan.com/tx/0x5430d391fce3d5d8c51279ffbaef9f3dbd58b235743ae44ef5531d1e8e735f48
Gas Used by Transaction:
47,140

Product Contract "cancelTransfer" Gas Usage:
https://testnet.bscscan.com/tx/0x9646ff61163f40a5af90c2aaa7d74f629e1ada9eaca8b5c99d8a3fd04e8a285d
Gas Used by Transaction:
16,465

Product Contract "approveTransfer" Gas Usage:
https://testnet.bscscan.com/tx/0x5299bba0fe57bef3452009507454a7dddca156a911104986ec3c1b767c270935
Gas Used by Transaction:
77,189

Product Contract "trace" Gas Usage:
View function, no gas usage

Product Contract "lastOwner" Gas Usage:
View function, no gas usage

testScripts.js

This file consists of two main parts to check the functionality of both state and manufacturer contracts. We have used truffle.js framework to test these contracts. What we have done in unit testing is explained in a detailed way in the documentation. Conditions that are tested are as follows:

For state contract

1- Should deploy the state contract properly
2- Should verify an address
3- Another address other than state should NOT verify an address

For manufacturer contract

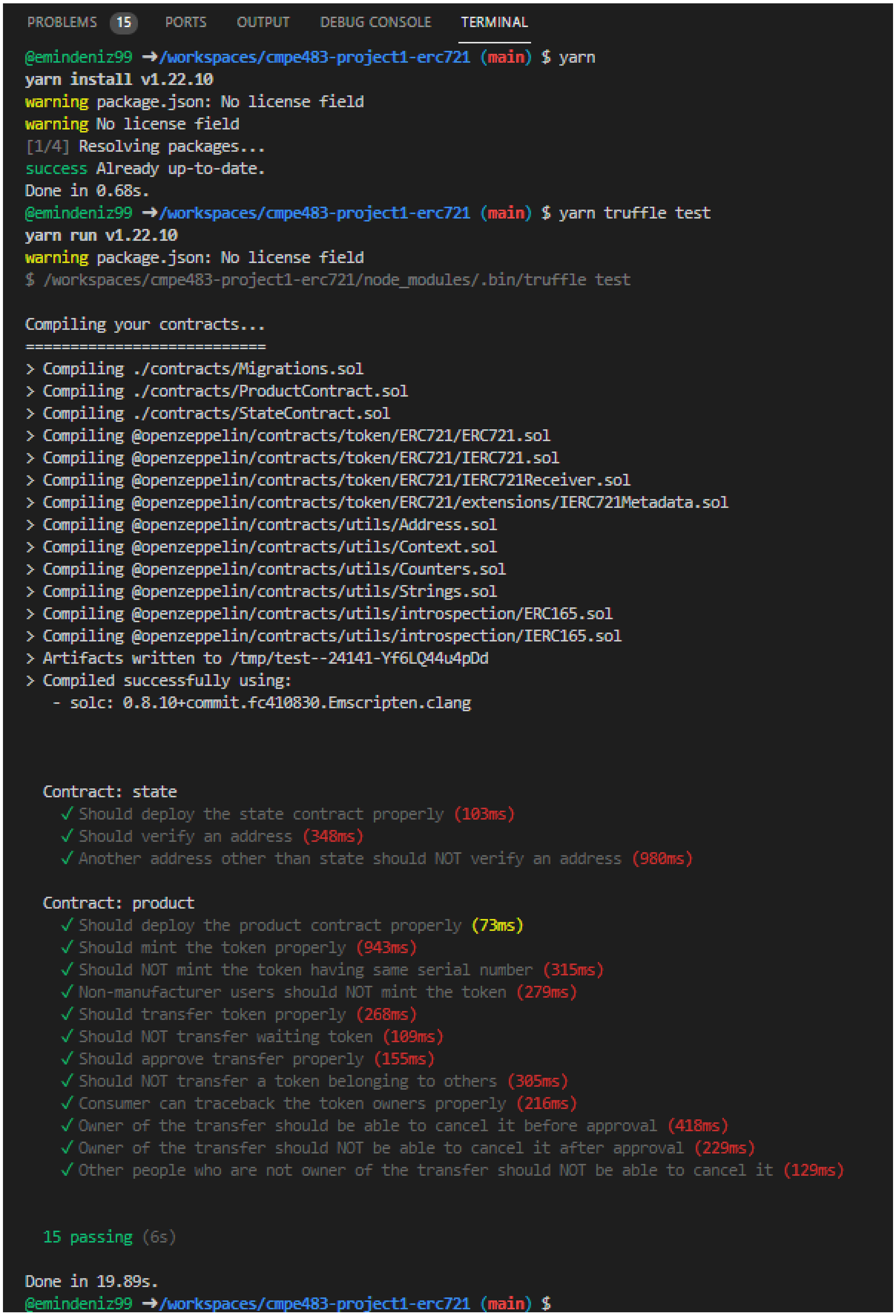
1- Should deploy the product contract properly
2- Should mint the token properly
3- Should NOT mint the token having same serial number
4- Non-manufacturer users should NOT mint the token
5- Should transfer token properly
6- Should NOT transfer waiting token
7- Should approve transfer properly
8- Should NOT transfer a token belonging to others
9- Consumer can trace back the token owners properly
10- Owner of the transfer should be able to cancel it before approval
11- Owner of the transfer should NOT be able to cancel it after approval
12- Other people who are not owner of the transfer should NOT be able to cancel it

Installing Truffle and dependencies
- Go to the folder to install the truffle and dependencies
- Run: yarn ("Yarn" for installing npm packages chai truffle-assertions (For revert handling))

Running Test
-"Yarn truffle test" for starting tests

The important thing is that when applying unit tests, we need errors at some tests. For example, when non-manufacturer users try to mint a token there should be an error since only manufacturers are eligible to mint a token. For these testing cases, We have installed an error handling library truffle-assertions.
And imported them below on testScripts.js

const assert = require(“chai”).assert
const truffleAssert = require(“truffle-assertions”)
By using these, when I have an error, the test case is passed.
Other test cases are handled by assert(condition) style as expected.



PROBLEMS 15   PORTS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
@emindeniz99 ➜/workspaces/cmpe483-project1-erc721 (main) $ yarn
yarn install v1.22.10
warning package.json: No license field
warning No license field
[1/4] Resolving packages...
success Already up-to-date.
Done in 0.68s.
@emindeniz99 ➜/workspaces/cmpe483-project1-erc721 (main) $ yarn truffle test
yarn run v1.22.10
warning package.json: No license field
$ /workspaces/cmpe483-project1-erc721/node_modules/.bin/truffle test

Compiling your contracts...
===========================
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/ProductContract.sol
> Compiling ./contracts/StateContract.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/token/ERC721/extensions/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling @openzeppelin/contracts/utils/Context.sol
> Compiling @openzeppelin/contracts/utils/Counters.sol
> Compiling @openzeppelin/contracts/utils/Strings.sol
> Compiling @openzeppelin/contracts/utils/introspection/ERC165.sol
> Compiling @openzeppelin/contracts/utils/introspection/IERC165.sol
> Artifacts written to /tmp/test--24141-Yf6LQ44u4pDd
> Compiled successfully using:
   - solc: 0.8.10+commit.fc410830.Emscripten.clang



  Contract: state
    ✓ Should deploy the state contract properly (103ms)
    ✓ Should verify an address (348ms)
    ✓ Another address other than state should NOT verify an address (980ms)

  Contract: product
    ✓ Should deploy the product contract properly (73ms)
    ✓ Should mint the token properly (943ms)
    ✓ Should NOT mint the token having same serial number (315ms)
    ✓ Non-manufacturer users should NOT mint the token (279ms)
    ✓ Should transfer token properly (268ms)
    ✓ Should NOT transfer waiting token (109ms)
    ✓ Should approve transfer properly (155ms)
    ✓ Should NOT transfer a token belonging to others (305ms)
    ✓ Consumer can traceback the token owners properly (216ms)
    ✓ Owner of the transfer should be able to cancel it before approval (418ms)
    ✓ Owner of the transfer should NOT be able to cancel it after approval (229ms)
    ✓ Other people who are not owner of the transfer should NOT be able to cancel it (129ms)


  15 passing (6s)

Done in 19.89s.
@emindeniz99 ➜/workspaces/cmpe483-project1-erc721 (main) $
```