ERC20Token.sol:

```
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";


contract ERC20Token is ERC20 {
    constructor(
        string memory _name,
        string memory _symbol
    ) ERC20(_name, _symbol) {}

    function mint(address _to, uint256 _amount) external {
        _mint(_to, _amount);
    }
}
```

ERC721Token.sol:

```
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract ERC721Token is ERC721 {
    constructor(
        string memory _name,
        string memory _symbol
    ) ERC721(_name, _symbol) {}

    function mint(address _to, uint256 _tokenId) external {
        _safeMint(_to, _tokenId);
    }
}
```

combined.sol:

```
pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";

contract ERC721Token is ERC721 {

    mapping(uint256 => uint256) private _tokenIdToAmount;

    mapping(uint256 => address) private _tokenIdToERC20;


    constructor(string memory _name, string memory _symbol) ERC721(_name, _symbol) {}

    function mint(address _to, uint256 _tokenId, address _erc20Token) external {
        require(_erc20Token != address(0), "wrong ERC20 address");

         _safeMint(_to, _tokenId);
        _tokenIdToAmount[_tokenId] = 1000;
        _tokenIdToERC20[_tokenId] = _erc20Token;
    }

    function transferERC721(uint256 _tokenId,  address _to) external {

        require(_exists(_tokenId), "Token does not exist");
        require(ownerOf(_tokenId) == msg.sender,  "for the owner");


        _transfer(msg.sender, _to, _tokenId);
```

```
    }

    function transferERC20(uint256 _tokenId, address _to) external  {
        require(_exists(_tokenId), "Token does not exist");
        require(ownerOf(_tokenId) == msg.sender, "for the owner");

        address erc20Token = _tokenIdToERC20[_tokenId];

        uint256 amount = _tokenIdToAmount[_tokenId];

        require(erc20Token != address(0), "wrong ERC20 address");

         IERC20(erc20Token).transferFrom(msg.sender, _to, amount);
    }
}
```

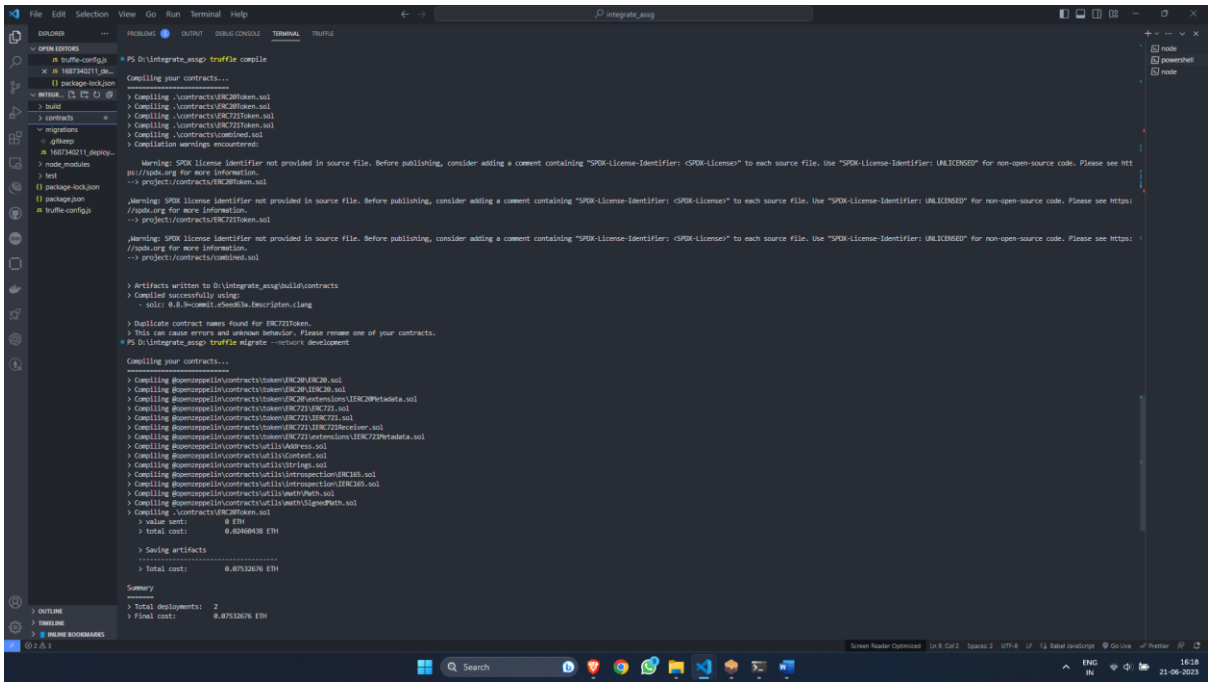
Deployment(js):

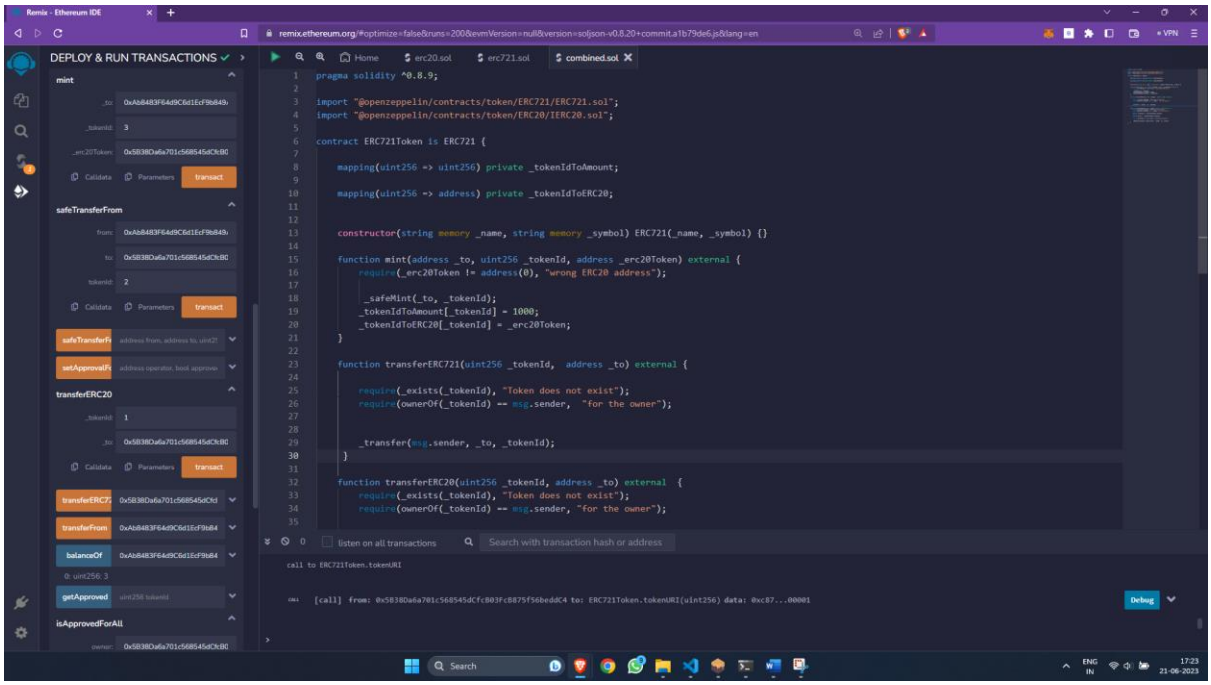
```
const ERC721Token = artifacts.require('ERC721Token');
const ERC20Token = artifacts.require('ERC20Token');

module.exports = function  (deployer)
{
   deployer.deploy(ERC721Token, 'MyERC721', 'ERC721');
  deployer.deploy(ERC20Token, 'MyERC20','ERC20');

};
```

Deployment:





Working:

firstly imported libraries in smart contracts and created deployment file then installed all the dependencies , compiled & deployed the same.

The resulting hash generated, block usage and total cost used is showed up.

Now, the functional components of the combined contract fulfilling the requirements:
ERC721Token.sol is an ERC721 token contract that inherits from OpenZeppelin's ERC721 implementation. The contract includes a constructor that takes in the token name and symbol as parameters, and I also implemented a mint function that allows the contract owner to mint new tokens and assign them to a specific address.

combined.sol is a contract that combines the functionality of ERC20Token.sol and ERC721Token.sol. The contract inherits from OpenZeppelin's ERC721 implementation and includes two mappings: _tokenIdToAmount and _tokenIdToERC20.

This contract also includes two functions for transferring tokens: transferERC721 and transferERC20, which allow the owner of a token to transfer either the ERC721 token or the associated ERC20 tokens to another address.

Also, in future using I'll be using web3.js for interaction from frontend for good user experience on it.

If anything more required from my part,You can mention that as well.

Thankyou.