National College of Ireland

MSC_PGDCLOUD Blockchain Concepts Project Assignment CA1

Weighting: 50%

Due Date: 29th of November 2022

Examiners: Mr Eoin Connolly

Overview

For this assignment, each individual student will issue an Ethereum ERC20 token on an Ethereum testnet (new proof of stake testnet Goerli) and provide a code-based mechanism to perform basic token distribution.

Requirements

1. Create an Ethereum account

Create an Ethereum account (public/private secp256k1 keypair with associated address) on Ethereum Goerli network with at least 100-bits of entropy.

2. Create an ERC20-compliant, fixed-supply token with the following parameters:

- Total, fixed supply of **1 million** tokens.
- **18** decimal places per token.
- Token owner to be the Ethereum account created above.
- Token contract to be **Verified** by contract creator.
- Name and description to be “SHB_XXXXX”, where XXXXX is the five rightmost digits of the submitting student’s student number.

3. Deploy the token contract to the Ethereum Testnet (Goerli).

Using the Ethereum account created in Requirement 1 above, deploy the ERC20 contract to an Ethereum Testnet (Goerli). We assume Goerli will be used, but in the event of Goerli unavailability, the contract can be deployed to

another testnet, providing the application provided in Requirement 4 interfaces with the contract deployed to that test network.

4. Build an application.

The application (Node.JS or Python) must be capable of accepting an ETH address as an input parameter, either via REST API POST request, or static input configuration file.

The application must be capable of performing the following activities:

- Checking the balance of the provided account.
- If the balance is less than 1000 tokens, distribute 1000 tokens to the account.
- If the balance is greater than 1000 tokens, distribute 1 token to the account.

The application will interact with Infura, a cloud-based blockchain infrastructure provider for node access and transaction propagation.

Required Assignment Output:

- Address of contract on Ethereum Test Net
- Contract Solidity Code (including interfaces)
- Ethereum testnet ETH Address of Token Owner
- Application interacting with Ethereum testnet to perform token distribution
- Separate .ENV file containing private key or seed phrase details.
- Source Code via zip file on Moodle (excluding npm_modules folder if NodeJS)
- Dockerfile containing code to dockerize Node.JS/Python application.
- Docker Hub URL of dockerised project
- Tools Report Submission
- Analysis Report Submission

All assignment outputs to be provided in a single zip file in Moodle

5%	Setting up Ethereum account via keypair generation
15%	Deploying ERC20 contract to Ethereum Test Net
10%	Building Node.js or Python application to process address input and distribute tokens
5%	Providing complete code from accessible repo in Github
10%	Providing **complete and clear** readme.md instructions for application *execution* in Github repository and/or Moodle .zip
5%	Providing application in Docker container
5%	Making Docker container available on Docker Hub
10%	Providing **no more than 3-page** report on process of development of contract and setting up of accounts across the platform. • Introduction explaining the tools being used and their purpose. • Document the Setup and configuration of said tools. • Describe the process of operations between each of the components at use throughout the project • Conclusions • Bibliography/References
10%	**No more than** 5 minute pre-recorded Demonstration displaying the operation of the contract deployment to testnet.
25%	At least **500-word** report on the impact and utility of token generation via ERC20 contracts. Provide your thoughts and insight into the impact of the ability to create tokens in this manner, from technical implications, positive and negative funding implications to teams and investors, including any ethical concerns around the generation of tokens for funding or incentivisation purposes.

NCI policies on plagiarism apply.

Papers will be submitted through Turnitin with similarity scores provided to establish originality of respective works