**Crowd Funding Smart Contract Release Three Deployment Document**

Soumil Vavikar

Purdue University Global

IN532 – Blockchain Application Development (dApps)

Dr. David Ostrowski

November 13, 2024

**Crowd Funding Smart Contract Release Three Deployment Document**

This document contains the development, testing, and deployment-related documentation for the enhancements to the “crowdfunding” smart contract developed as part of the Unit 6 and Unit 7 assignments.

**Developed the Crowd Funding Factory Contract**

A crowdfunding project has been upgraded, and a new crowdfunding factory contract has been added to the project to support having multiple crowdfunding created, maintained, and executed:

|  |  |
| --- | --- |
| **File Name** | **Details on the File Content** |
| CrowdFundingFactory.sol | The Solidity contract file has various functions that enable opening, handling, and managing multiple crowd-fundings. |
| CrowdFundingFactoryModule.js | This module class has the code to deploy the factory contract to the local chain. |
| CrowdFundingFactory.test.js | This test class has the unit tests and assertions developed/written to test the factory contract. |

**Smart Contract and Supporting Files**

This smart contract is written in Solidity, and the "Hardhat" framework is used to develop, test, and deploy the smart contract.

A GitHub repository (<https://github.com/soumilvavikar/hardhat-crowd-funding>) has been created, and the smart contract, supporting files (test files, interaction files, and library files), test evidence, and commands required to start the local chain, deploy the smart contract, and test the smart contract have been pushed to the repository.

The entire workspace for the crowdfunding contract has also been submitted with this deployment document.

**Crowd Funding Smart Contract Code**

Two libraries have been created to make crowdfunding smart contracts readable and maintainable. The sol files for the libraries are attached to the smart contract code.

 

**Crowd Funding Factory Smart Contract Code**



**Deployment and Execution Commands**

The deployment and execution commands remain the same, as mentioned in the deployment document submitted as part of the Unit 6 and Unit 7 assignments.

**Test Evidence(s)**

Extensive testing (including unit testing and end-to-end functional testing) of the developed smart contract has been done. The project's README.md contains quick links to the “.md” files containing the test evidence (logs and screenshots) captured within the workspace under the “test evidences” folder. The test evidence can also be found below.

**Unit Test Evidence**

Five tests have been added to assert that the factory smart contract and its functions work as intended. The unit test file and the unit testing evidence text file, which contains the command and the logs generated from the unit test run, are attached below.

 

The unit test readme file “README\_UNIT\_TESTING.md” is in the workspace's “testevidences” folder.

A screenshot of a computer

Description automatically generated

**Deployment Test Evidence**

No changes were made to the deployment process/steps from the process setup during the smart contract development work done during the Unit 6 and Unit 7 assignments. The crowdfunding smart contract has been deployed using the ignition module. The file containing the module code required for deployment and the test evidence of successful local chain startup and deployment are attached below.



A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

A new “CrowdFundingFactoryModule.js” file has been created to deploy the “CrowdFundingFactory” contract to the local chain. PFB, the module file, and the test evidence of successful deployment to it.

 

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The deployment test evidence readme file “README\_E2E\_TEST\_EVIDENCES.md” is in the workspace's “testevidences" folder.

**End-to-End Test Evidence**

The end-to-end interactions have been updated to test the newly added functions. The end-to-end test interactions file is attached below, along with the end-to-end test evidence text file containing the logs generated from the end-to-end test run.

 

The end-to-end test readme file “README\_E2E\_TEST\_EVIDENCES.md” is in the workspace's “testevidences” folder.

A screenshot of a computer

Description automatically generated

The end-to-end testing of the factory contract would be completed and added to the next unit’s assignment.