**Crowd Funding Smart Contract Deployment Document**

Soumil Vavikar

Purdue University Global

IN532 – Blockchain Application Development (dApps)

Dr. David Ostrowski

October 24, 2024

**Crowd Funding Smart Contract Deployment Document**

This document contains the development, testing, and deployment related documentation for the “crowd funding” smart contract developed based of the design document from the Unit 4 assignment.

**Business Use-Case**

A crowdfunding contract will swap Ether (ETH) for non-fungible tokens (NFTs), representing a creative approach to decentralized fundraising on the Ethereum blockchain. This smart contract lets crowdfunding creators raise funds by offering custom NFTs to supporters in exchange for their ETH contributions. Participants can contribute by transmitting ETH to the contract address and receiving one project-specific token per successful contribution.

**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.

**Smart Contract Code**

To make crowdfunding smart contract readable and maintainable, two libraries have been created. The sol files for the libraries are attached with the smart contract code.



**Deployment and Execution Commands**

|  |
| --- |
| ## Initial Repository Setup  *# Initialize NPM*  npm init  # Install hardhat if not done already  npm install --save-dev hardhat  # Initialize the hardhat project (select a valid option)  npx hardhat init  # Install the openzeppelin contract to use ERC721 interface  npm install --save-dev @openzeppelin/contracts  ## Compiling and Testing the Contract  # Compile the project  npx hardhat compile  # Run the tests  npx hardhat test  ## Starting the local chain and Deploying the Contract  # Spin the local chain  npx hardhat node  # Deploy the contract / hardhat project  npx hardhat ignition deploy ignition/modules/CrowdFundingModule.js --network localhost  ### Command for End-to-End Testing via Interactions  # Run all the functions to test the flow end to end.  npx hardhat run interactions/end-to-end/InteractingWithCrowdFundingContract.js --network localhost |

**Test Evidence(s)**

Extensive testing (which includes unit testing as well as end to end functional testing) of the developed smart contract has been done. The README.md for the project contains quick links to the “.md” files containing the test evidence (logs and screenshots) captured within the workspace under the “testevidences” folder. The test evidence can also be found below.

**Unit Test Evidence**

The unit test file is attached below along with the unit testing evidence text file which contains the command, and the logs generated from the unit test run.



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

A screen shot of a computer code

Description automatically generated

**Deployment Test Evidence**

The deployment of the crowdfunding smart contract has been done using “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.



The deployment test evidence readme file “README\_E2E\_TEST\_EVIDENCES.md” is present in the “testevidences” folder of the workspace.A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

**End to End Test Evidence**

The end-to-end test interactions file is attached below along with the end-to-end test evidence text file which contains the logs generated from the end-to-end test run.

 

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

A screen shot of a computer code

Description automatically generated