Blockchain & Solidity Lab3 – Crowdfunding dApp Development

S2BC

Lab 3: Integrate Web App with Smart Contracts

- BUILD / TEST / **INTEGRATE** / RUN

Objective: The aim of this Lab3 is to integrate the smart contracts you developed in Lab1 and Lab2 with a Crowdfunding dApp for users to access the dApp using the web browser.

Deploy Compiled Smart Contract with Hardhat

To deploy the compiled contract to the Ethereum blockchain network, follow these steps:

Step 1: Configure a dotenv (.env) file

First, install the `dotenv` package using the following command:

```
npm install dotenv
```

Next, create a `.env` file in the root folder of your HardHat project.(hardhat/.env) This file will contain sensitive information that should be kept secure. Add the following variables to the `.env` file:

```
# This is the URL of the Ethereum RPC provider
RPC_URL="https://example.com/rpc" (optain from morpheus)

# This is a private key for signing transactions (private key of the
deployer account)
PRIVATE_KEY="your_private_key_here"

# This is the chain ID for the Ethereum network
CHAIN_ID=12345
```

Make sure to replace the placeholder values with your actual credentials.

Step 2: Configure hardhat.config.js

Modify your `hardhat.config.js` file as follows:

```
require("@nomicfoundation/hardhat-toolbox");
require("dotenv").config();

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.22",
  networks: {
    // Add your network configuration here
    poa: {
      url: process.env.RPC_URL, // RPC URL of your network
      chainId: parseInt(process.env.CHAIN_ID), // Chain ID of your network
      accounts: [process.env.PRIVATE_KEY], // Array of private keys to use
with this network
    },
  },
};
```

Step 3: Create a Deployment Script

Create a new file named `deploy.js` inside the `hardhat/scripts` directory. Add the following content to the file:

```
const { ethers } = require("hardhat");
const fs = require("fs");

async function deployCampaignCreator() {
  // Get the deployer's address
  const [deployer] = await ethers.getSigners();
  console.log(
    "Deploying CampaignCreator contract with the account:",
    deployer.address
  );

  // Get the CampaignCreator contract factory
  const CampaignCreator = await
ethers.getContractFactory("CampaignCreator");

  // Deploy the CampaignCreator contract
  const campaignCreator = await CampaignCreator.deploy();
  // console.log(campaignCreator.target);

  // Save deployment information to a text file
  const deploymentInfo = `Deployer Address:
${deployer.address}\nCampaignCreator Contract Address:
${campaignCreator.target}`;
  console.log(
```

```
      `CampaignCreator Contract Address deployed: ${campaignCreator.target}`
    );
    fs.writeFileSync("deploymentInfoCampaignCreator.txt", deploymentInfo);

    // Return the deployed CampaignCreator contract instance
    return campaignCreator;
  }

  async function main() {
    try {
      // Deploy the CampaignCreator contract
      const campaignCreator = await deployCampaignCreator();

      console.log("Deployment completed successfully!");
    } catch (error) {
      console.error("Error deploying contracts:", error);
      process.exitCode = 1;
    }
  }

  main();
```

To deploy the contracts, use the following command in your terminal:

```
npx hardhat run scripts/deploy.js --network poa
```

The result output from the terminal will provide the contract addresses.

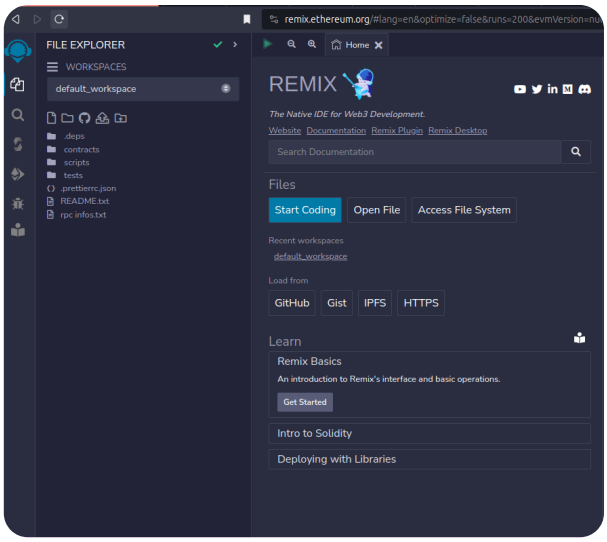
A "deploymentInfoCampaignCreator.txt" file will be created with the CampaignCreator contract address.

That is what you will need to add to the ".env.local" file in the front-end later on.

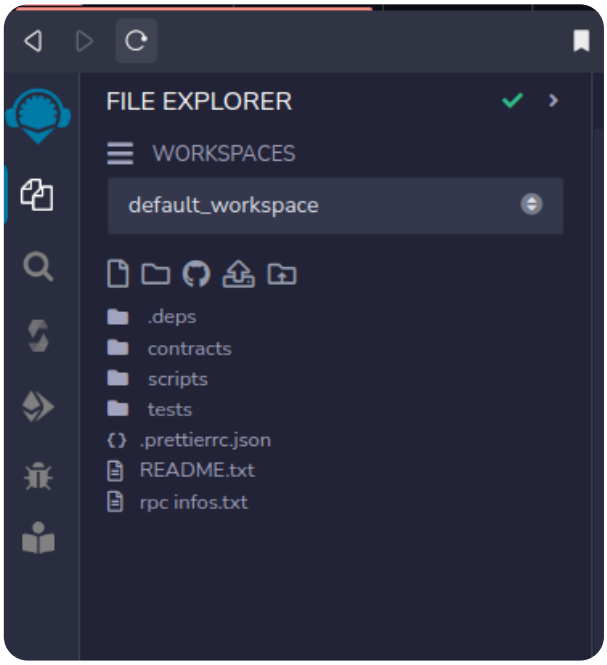
Try Your Contracts on Remix IDE

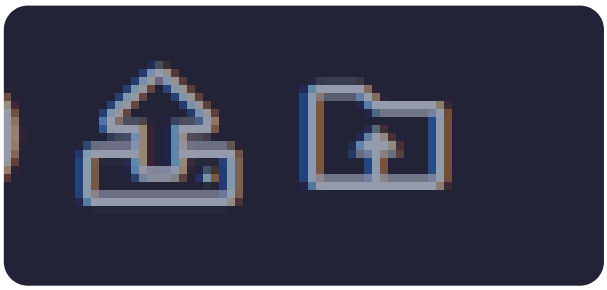
Remix IDE provides a visual way to interact with your contracts before implementing your frontend. Follow these steps to test your contracts:

1. Visit the Remix website: Remix IDE.



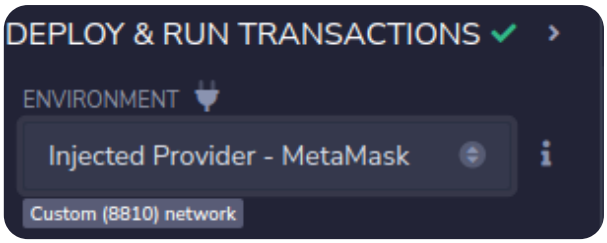
2. Upload your contracts CampaignCreator.sol and CrowdCollab.sol:
    - Navigate to the contract folder.
    - Click on one contract and press the compile green arrow.



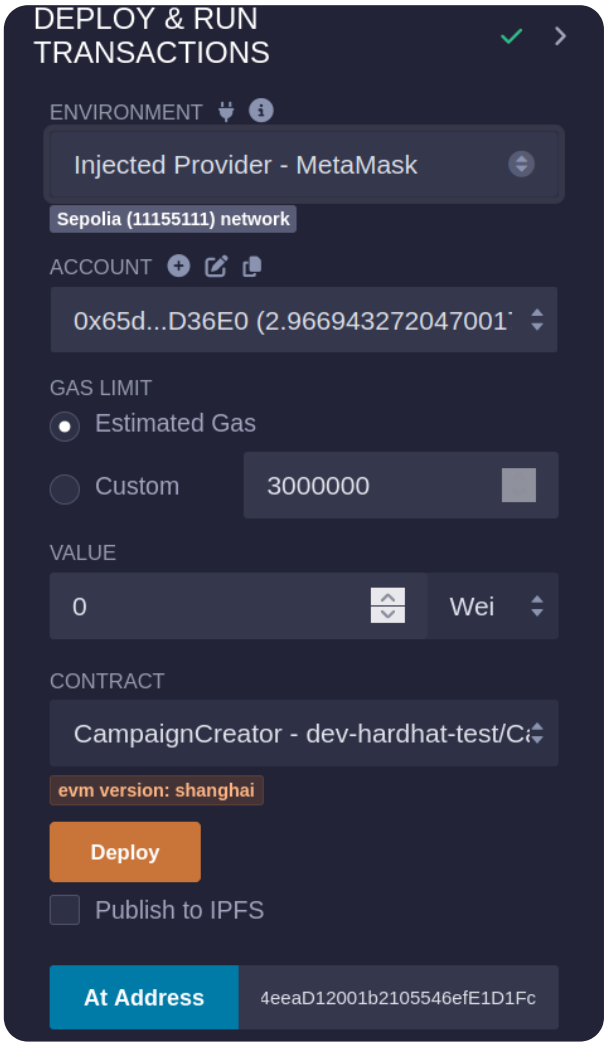


3. Ensure that the compiler version is set to 0.8.22:
    - Select the "Compiler" tab.
    - Confirm that version 0.8.22 is checked.

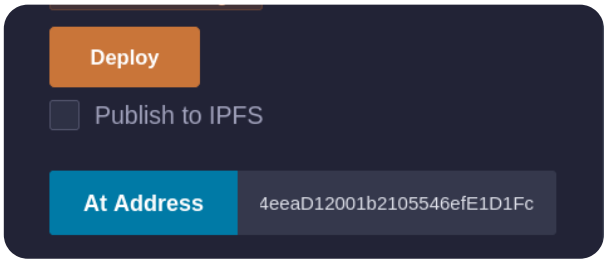
4. Go to the "Deploy" tab:
    - In the deploy tab, select "Wallet Injected Provider."
    - Connect your MetaMask account to Remix IDE.



Depending on your wallet network, poa, hardhat node or sepolia.

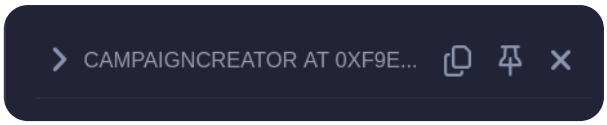


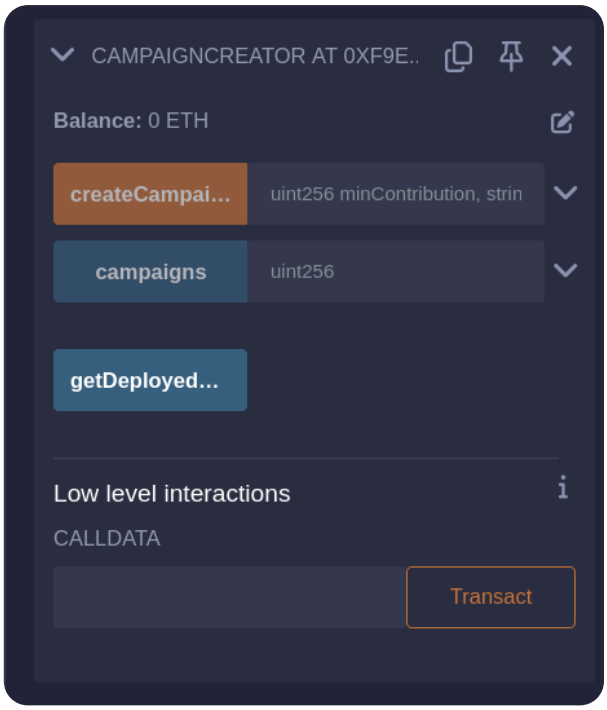
5. Paste the address of your deployed CampaignCreator.sol contract at the bottom of the deploy tab. (contract need to compiled at that point)

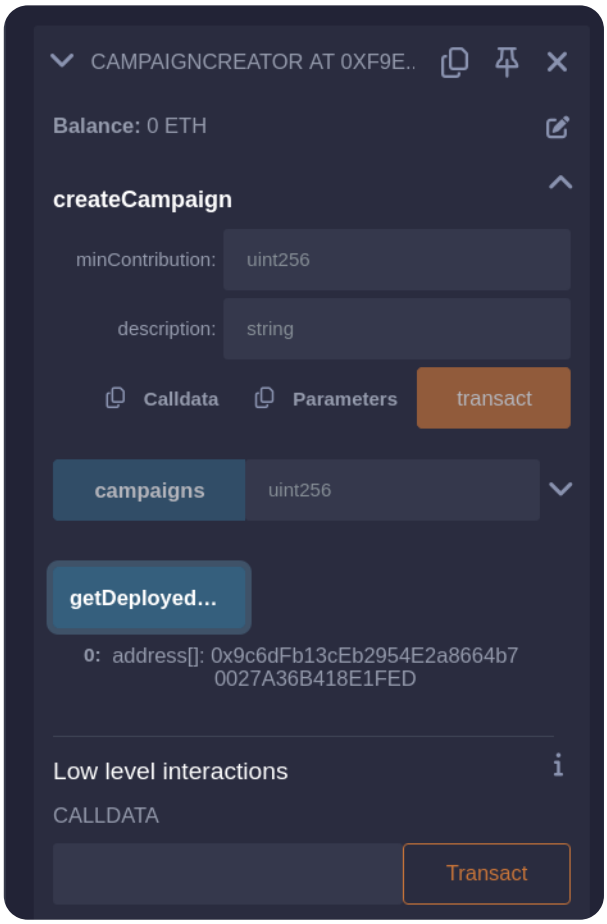


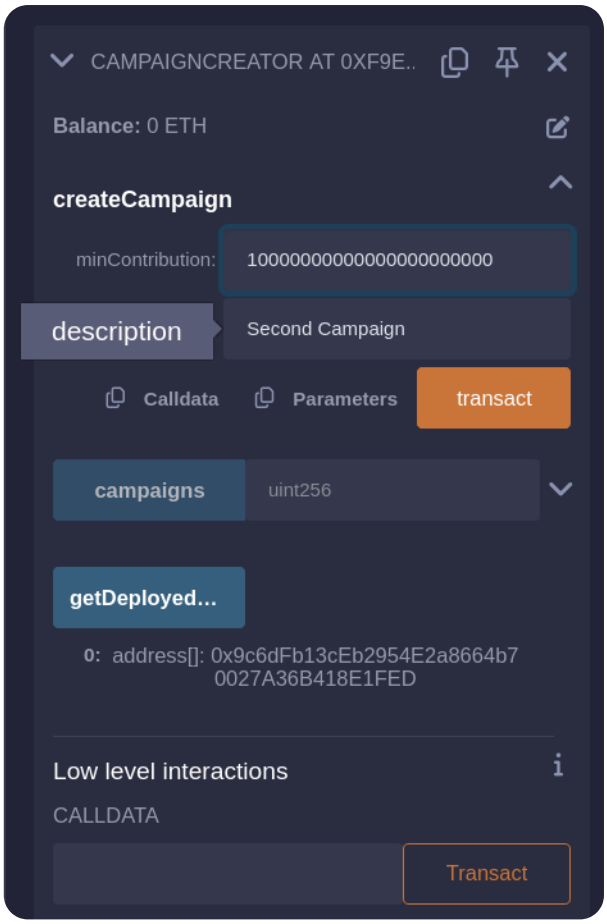
and click on "address" button

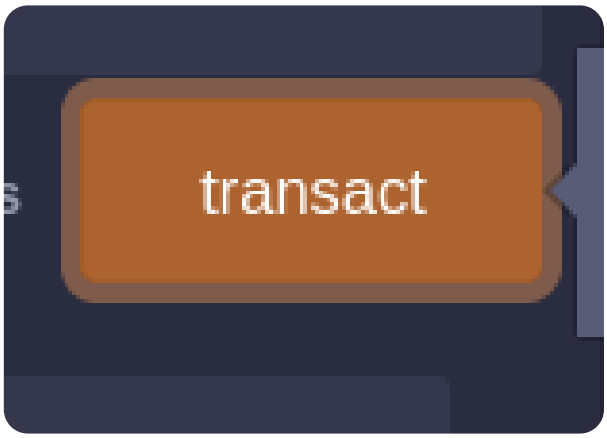
6. Load your already deployed contract:
    - This action allows you to interact with your contract in the newly appeared menu.

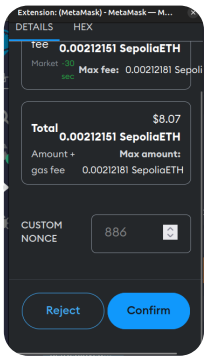

CAMPAIGNCREATOR AT 0XF9E...

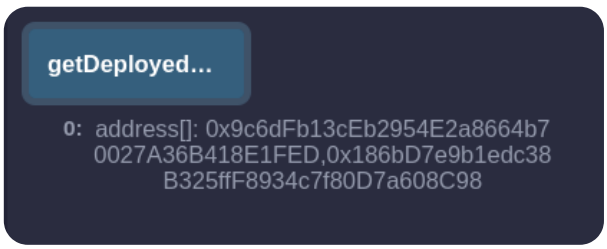

CAMPAIGNCREATOR AT 0XF9E..
Balance: 0 ETH
createCampai...
uint256 minContribution, strin
campaigns
uint256
getDeployed...
Low level interactions
CALLDATA
Transact

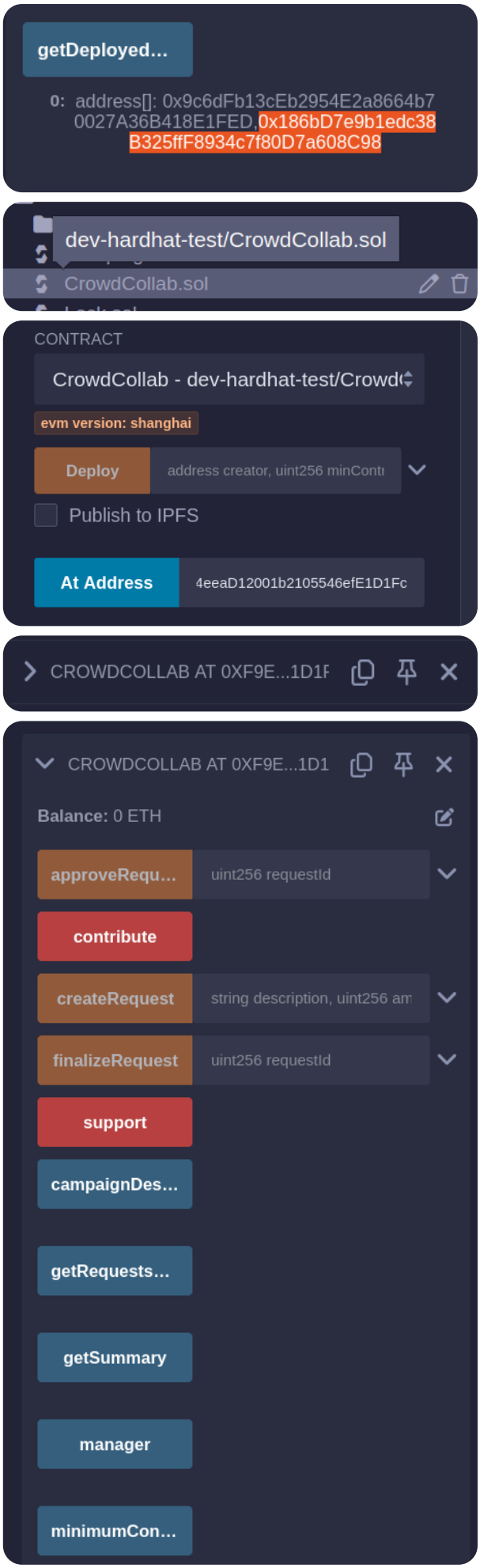

CAMPAIGNCREATOR AT 0XF9E..
Balance: 0 ETH
createCampaign
minContribution:
uint256
description:
string
Calldata
Parameters
transact
campaigns
uint256
getDeployed...
0: address[]: 0x9c6dFb13cEb2954E2a8664b7
0027A36B418E1FED
Low level interactions
CALLDATA
Transact

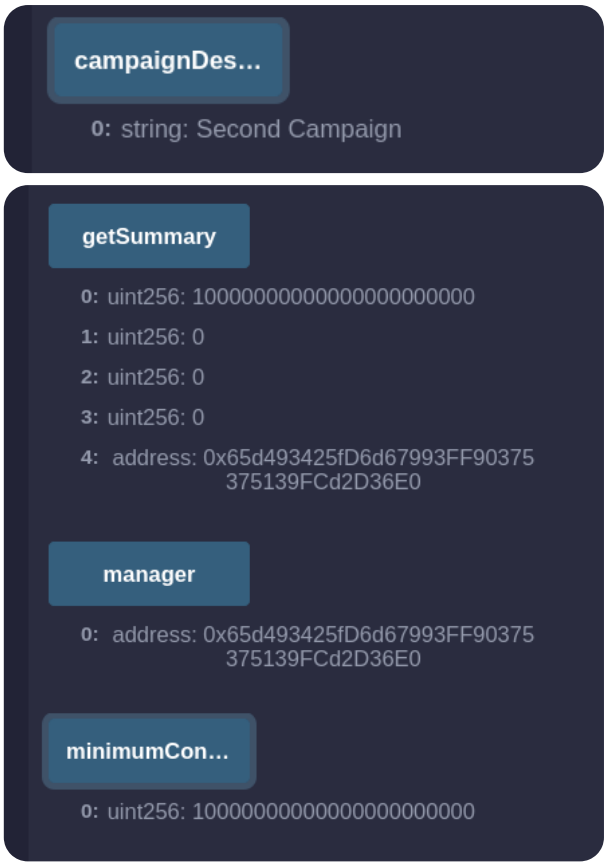

CAMPAIGNCREATOR AT 0XF9E..
Balance: 0 ETH
createCampaign
minContribution:
100000000000000000000
description
Second Campaign
Calldata
Parameters
transact
campaigns
uint256
getDeployed...
0: address[]: 0x9c6dFb13cEb2954E2a8664b7
0027A36B418E1FED
Low level interactions
CALLDATA
Transact


transact


Extension: (MetaMask) - MetaMask — M...
DETAILS
HEX
fee
0.00212151 SepoliaETH
Max fee: 0.00212181 Sepoli
$8.07
Total
0.00212151 SepoliaETH
Amount +
gas fee
Max amount:
0.00212181 SepoliaETH
CUSTOM NONCE
886
Reject
Confirm


getDeployed...
0: address[]: 0x9c6dFb13cEb2954E2a8664b7
0027A36B418E1FED,0x186bD7e9b1edc38
B325ffF8934c7f80D7a608C98


getDeployed...
0: address[]: 0x9c6dFb13cEb2954E2a8664b7
0027A36B418E1FED,0x186bD7e9b1edc38
B325ffF8934c7f80D7a608C98
dev-hardhat-test/CrowdCollab.sol
CrowdCollab.sol
CONTRACT
CrowdCollab - dev-hardhat-test/Crowd
evm version: shanghai
Deploy
address creator, uint256 minCont
Publish to IPFS
At Address
4eeaD12001b2105546efE1D1Fc
CROWDCOLLAB AT 0XF9E...1D1F
CROWDCOLLAB AT 0XF9E...1D1
Balance: 0 ETH
approveRequ...
uint256 requestId
contribute
createRequest
string description, uint256 am
finalizeRequest
uint256 requestId
support
campaignDes...
getRequests...
getSummary
manager
minimumCon...



By adhering to these guidelines, you can efficiently verify and engage with your contracts through Remix IDE before advancing to frontend development.

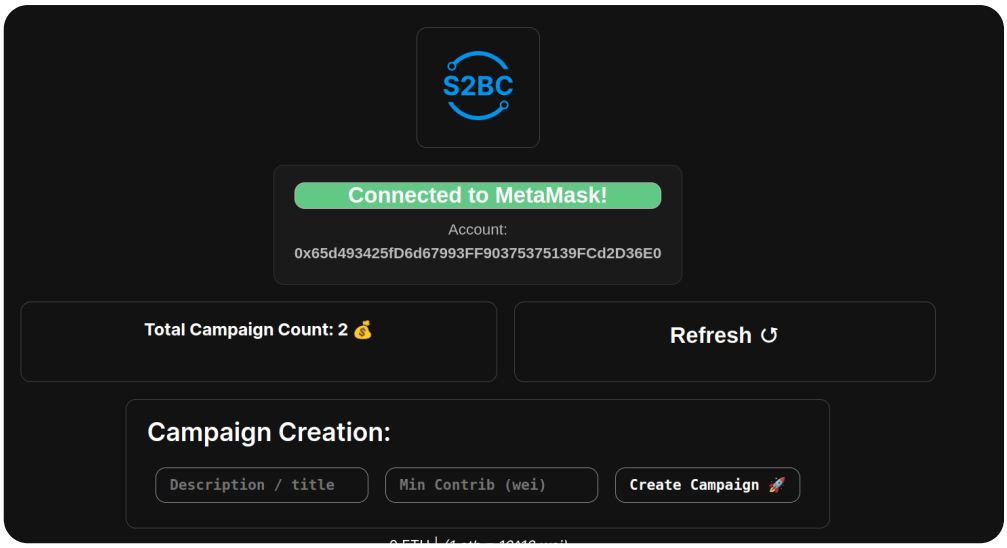
Once you've established your initial campaign, you may access the CrowdCollab instance address by repeating the earlier procedure, this time selecting the CrowdCollab contract and ensuring it's compiled before invocation.

You can test your contract like this before front-end integration.

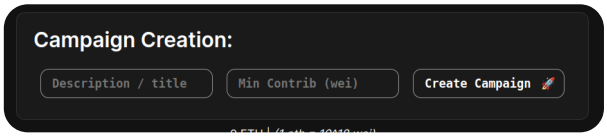
Frontend integration

UI-Screenshoots :

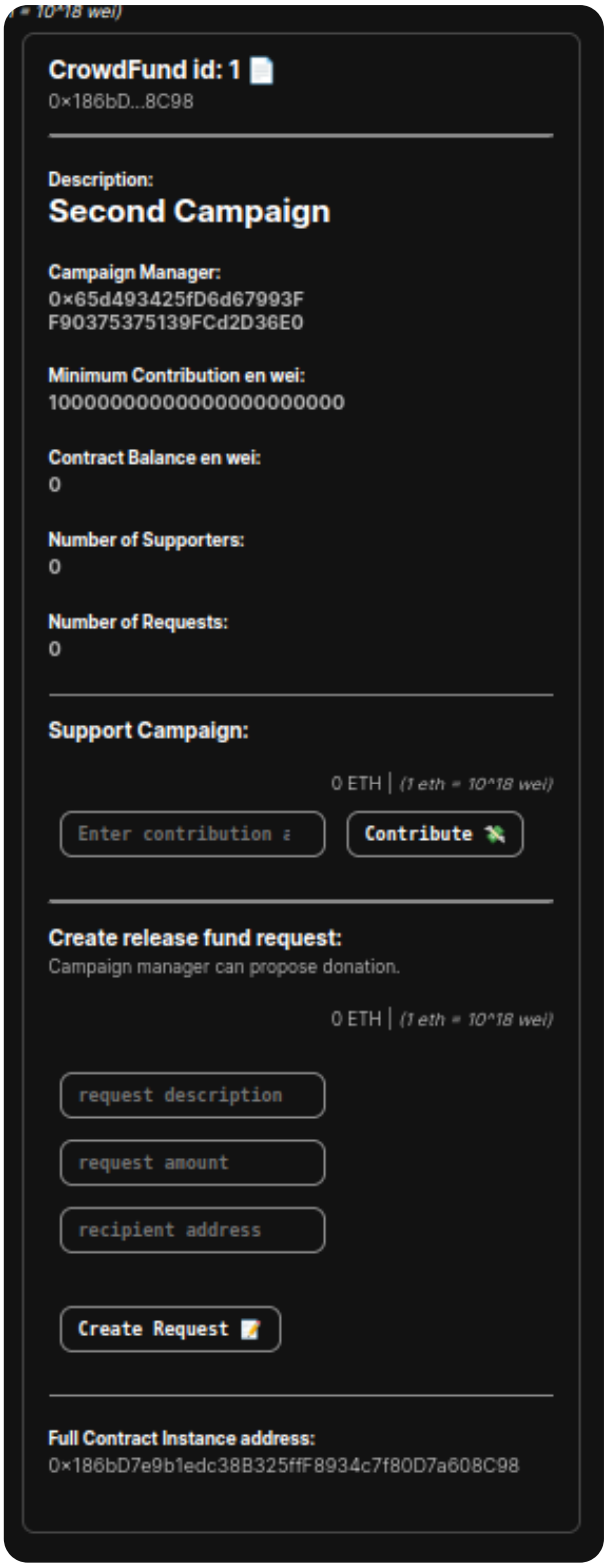
Crowdfund panel :



Create panel :



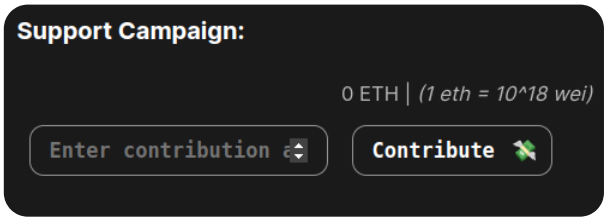
Campaign panel :



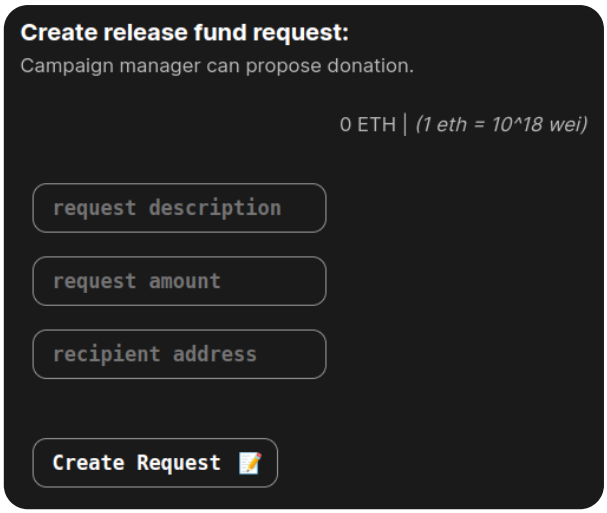
Description section:

Description section:

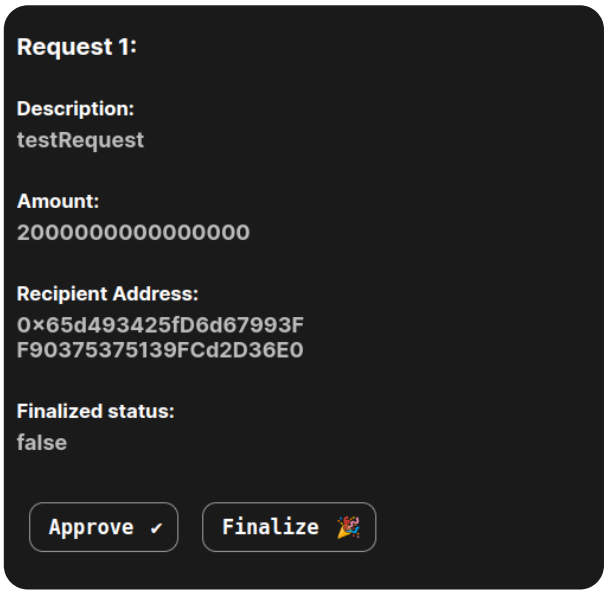
Support / Contribute section:



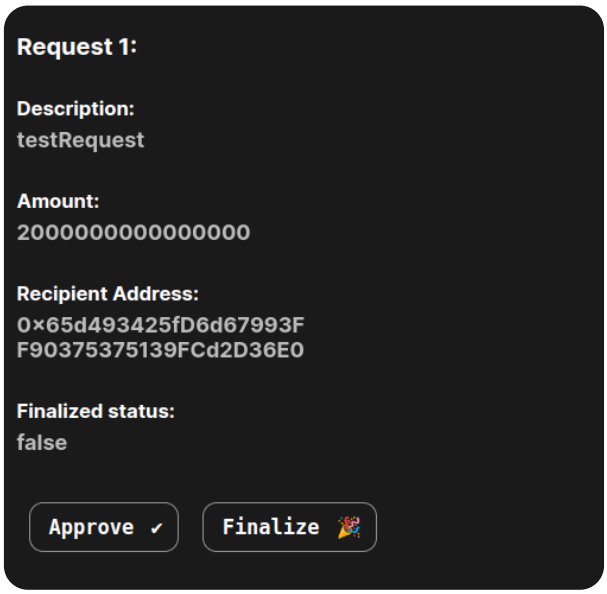
Create request:



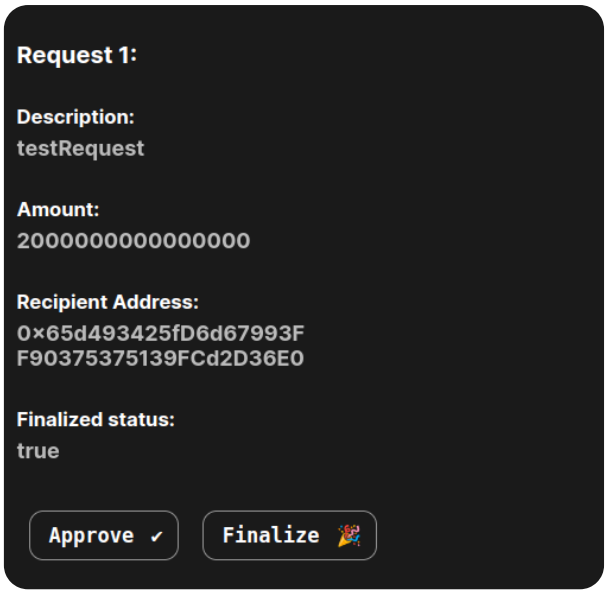
Request description:



Aproove:



Finalize:



Setting Up the Frontend

In this section, we will guide you through setting up the frontend of your Crowdfunding dApp. Follow these steps to create the necessary folders and files:

1. Create a Frontend Folder

Begin by creating a folder named `frontend` within your project directory. This folder will house all the files related to the frontend of your dApp.

Your tree folder should be like:

```
- crowdfunding-dapp-2024
   - hardhat
   - frontend
```

So if you were in hardhat folder, come back to your root folder:

```
cd ..
```

then create the frontend folder

```
mkdir frontend
cd frontend
```

2. Let's initiate Nextjs

```
npx create-next-app@latest .
```

Choose:

```
✔ Would you like to use TypeScript? … No / Yes  (NO)
✔ Would you like to use ESLint? … No / Yes  (NO)
✔ Would you like to use Tailwind CSS? … No / Yes  (NO)
✔ Would you like to use `src/` directory? … No / Yes  (YES)
✔ Would you like to use App Router? (recommended) … No / Yes  (YES)
✔ Would you like to customize the default import alias (@/*)? … No / Yes
(NO)
```

then you can install this dependencies we will need for use Web3 library and read .env.local files

```
npm install web3 dotenv
```