Internship Report

Blockchain & Crypto Internship

Task 02: Create and Deploye an ERC-20 Token on Polygon Amoy

Submitted by: Vaibhav pratap singh

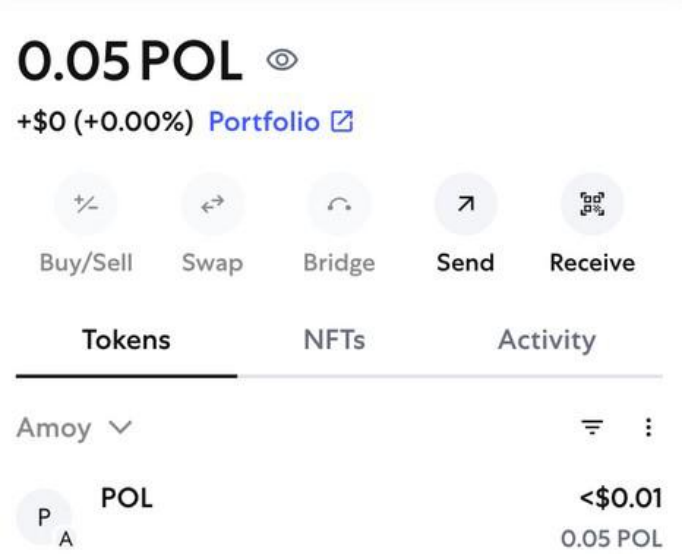
Intern at Future Interns

Date: July 26, 2025

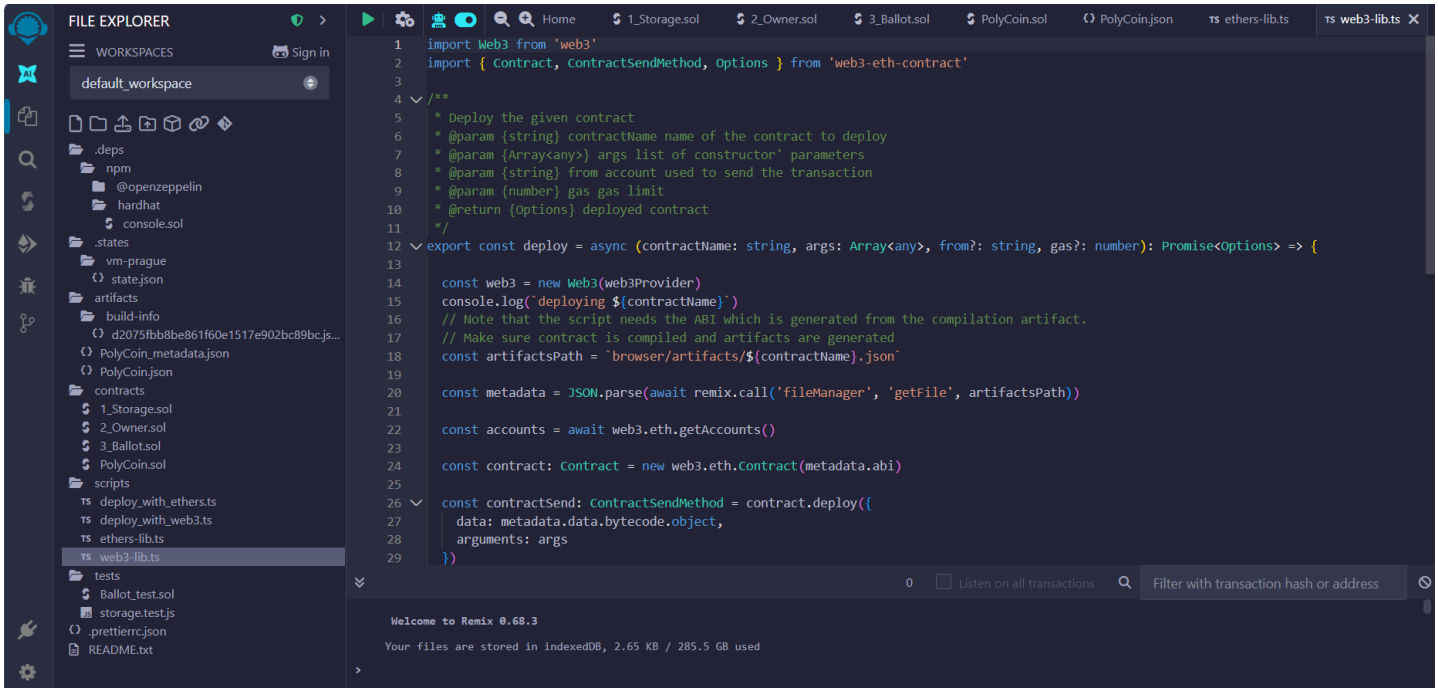
Task Overview:

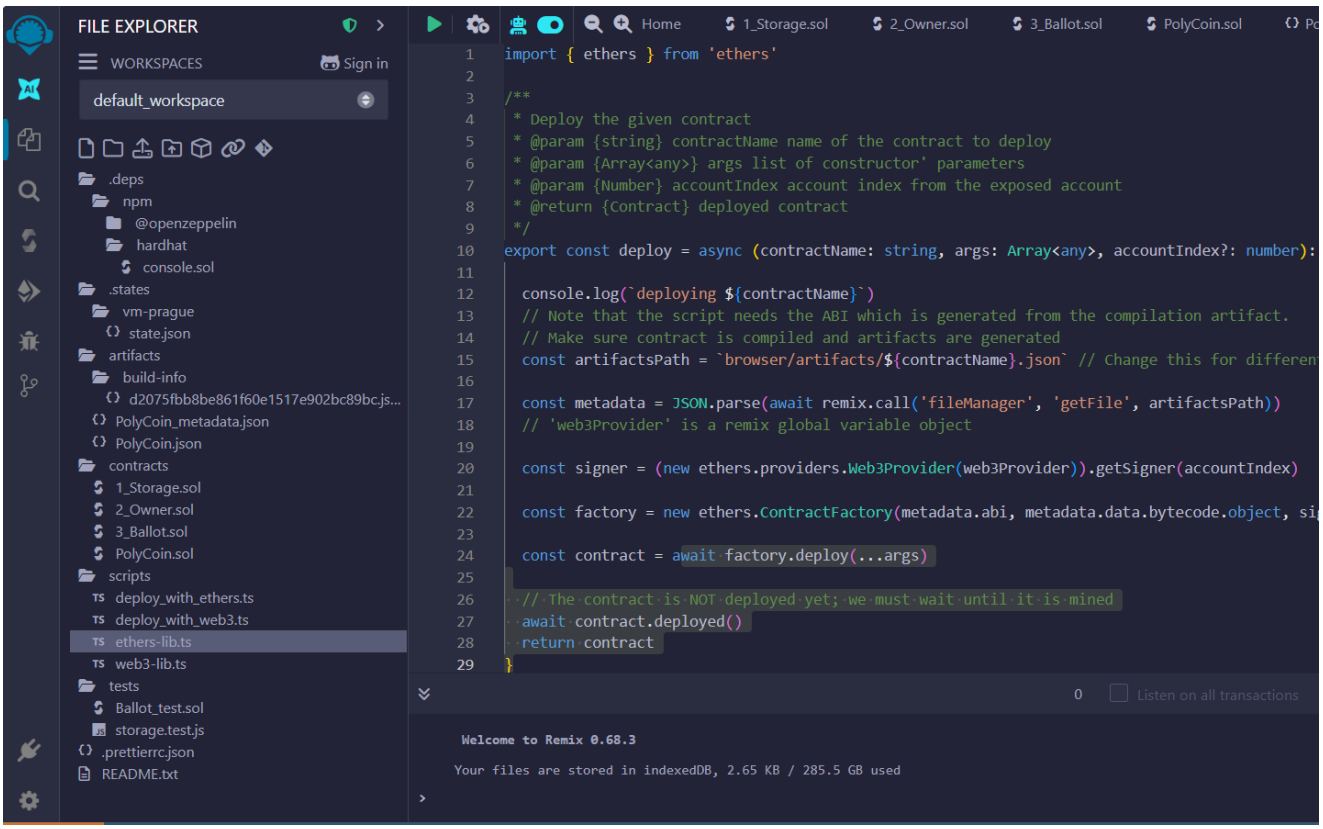
The goal of Task 02 was to create a custom ERC-20 token using Solidity and deploy it on a public blockchain testnet. Due to the deprecation of the Polygon Mumbai Testnet, this task was completed using the upgraded and actively supported Polygon Amoy Testnet. The token was written in Solidity, compiled and deployed using Remix IDE, and verified via MetaMask and the Amoy explorer.

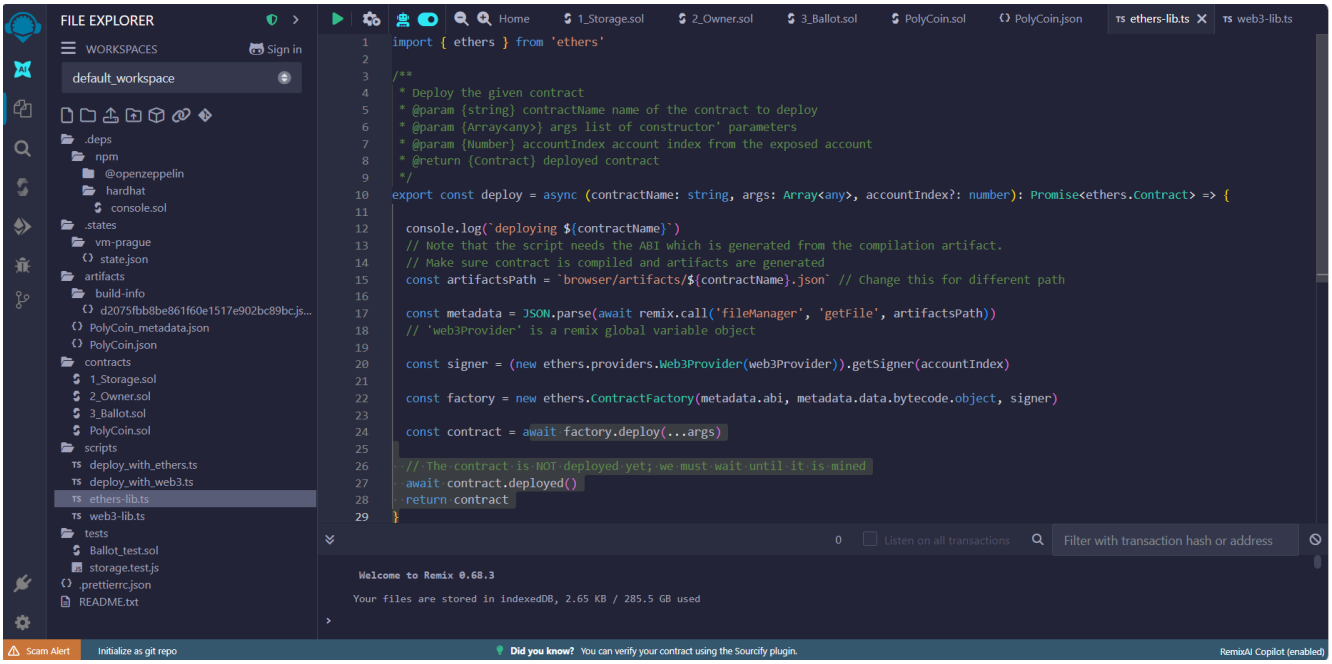
Step 1: Setup MetaMask with Amoy Testnet and Fund with 0.05 POL



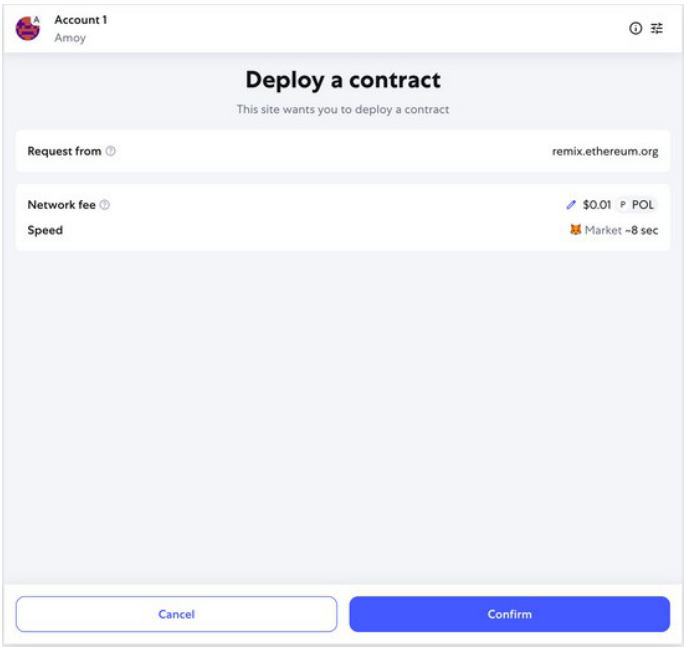
Step 2: Write the ERC-20 Token Smart Contract in Remix



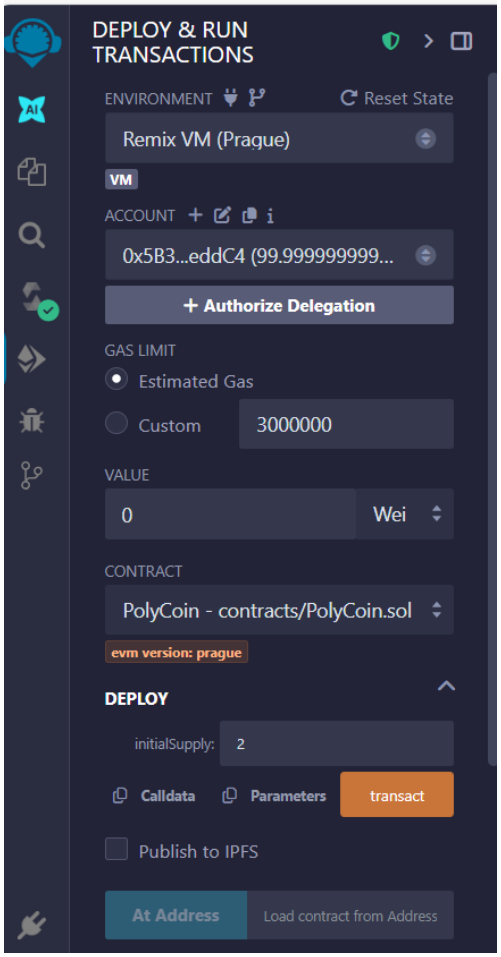

FILE EXPLORER
WORKSPACES
Sign in
default_workspace
.deps
npm
@openzeppelin
hardhat
console.sol
.states
vm-prague
state.json
artifacts
build-info
d2075fbb8be861f60e1517e902bc89bc.js...
PolyCoin_metadata.json
PolyCoin.json
contracts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
PolyCoin.sol
scripts
deploy_with_ethers.ts
deploy_with_web3.ts
ethers-lib.ts
web3-lib.ts
tests
Ballot_test.sol
storage.test.js
.prettierrc.json
README.txt
Home
1_Storage.sol
2_Owner.sol
3_Ballot.sol
PolyCoin.sol
import { ethers } from 'ethers'
/**
* Deploy the given contract
* @param {string} contractName name of the contract to deploy
* @param {Array<any>} args list of constructor' parameters
* @param {Number} accountIndex account index from the exposed account
* @return {Contract} deployed contract
*/
export const deploy = async (contractName: string, args: Array<any>, accountIndex?: number):
console.log(`deploying ${contractName}`)
// Note that the script needs the ABI which is generated from the compilation artifact.
// Make sure contract is compiled and artifacts are generated
const artifactsPath = `browser/artifacts/${contractName}.json` // Change this for differen
const metadata = JSON.parse(await remix.call('fileManager', 'getFile', artifactsPath))
// 'web3Provider' is a remix global variable object
const signer = (new ethers.providers.Web3Provider(web3Provider)).getSigner(accountIndex)
const factory = new ethers.ContractFactory(metadata.abi, metadata.data.bytecode.object, si
const contract = await factory.deploy(...args)
// The contract is NOT deployed yet; we must wait until it is mined
await contract.deployed()
return contract
}
0
Listen on all transactions
Welcome to Remix 0.68.3
Your files are stored in indexedDB, 2.65 KB / 285.5 GB used

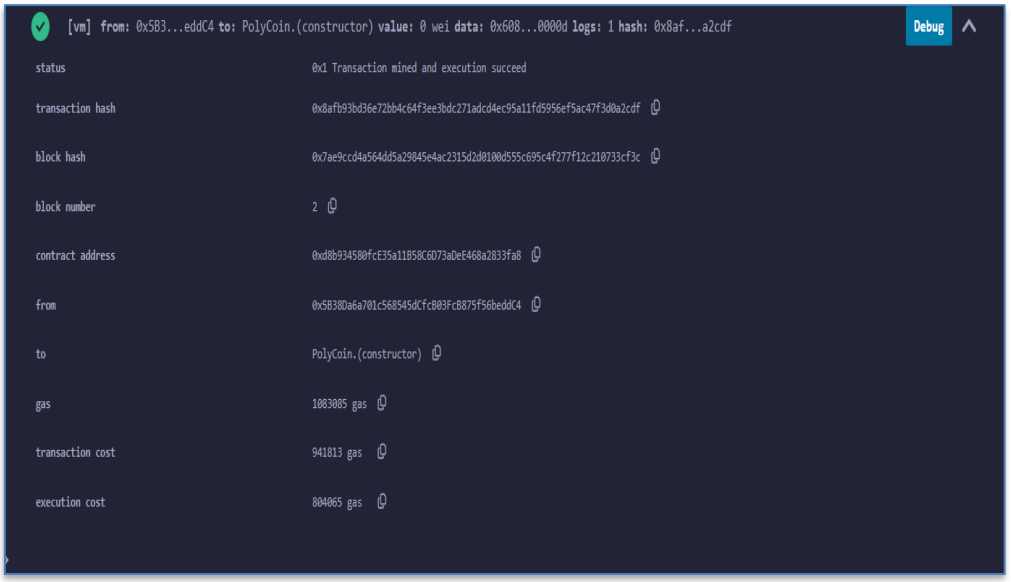

FILE EXPLORER
WORKSPACES
Sign in
default_workspace
.deps
npm
@openzeppelin
hardhat
console.sol
.states
vm-prague
state.json
artifacts
build-info
d2075fbb8be861f60e1517e902bc89bc.js...
PolyCoin_metadata.json
PolyCoin.json
contracts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
PolyCoin.sol
scripts
deploy_with_ethers.ts
deploy_with_web3.ts
ethers-lib.ts
web3-lib.ts
tests
Ballot_test.sol
storage.test.js
.prettierrc.json
README.txt
Home
1_Storage.sol
2_Owner.sol
3_Ballot.sol
PolyCoin.sol
PolyCoin.json
ethers-lib.ts
web3-lib.ts
import { ethers } from 'ethers'
/**
* Deploy the given contract
* @param {string} contractName name of the contract to deploy
* @param {Array<any>} args list of constructor' parameters
* @param {Number} accountIndex account index from the exposed account
* @return {Contract} deployed contract
*/
export const deploy = async (contractName: string, args: Array<any>, accountIndex?: number): Promise<ethers.Contract> => {
console.log(`deploying ${contractName}`)
// Note that the script needs the ABI which is generated from the compilation artifact.
// Make sure contract is compiled and artifacts are generated
const artifactsPath = `browser/artifacts/${contractName}.json` // Change this for different path
const metadata = JSON.parse(await remix.call('fileManager', 'getFile', artifactsPath))
// 'web3Provider' is a remix global variable object
const signer = (new ethers.providers.Web3Provider(web3Provider)).getSigner(accountIndex)
const factory = new ethers.ContractFactory(metadata.abi, metadata.data.bytecode.object, signer)
const contract = await factory.deploy(...args)
// The contract is NOT deployed yet; we must wait until it is mined
await contract.deployed()
return contract
}
0
Listen on all transactions
Filter with transaction hash or address
Welcome to Remix 0.68.3
Your files are stored in indexedDB, 2.65 KB / 285.5 GB used
Scam Alert
Initialize as git repo
Did you know? You can verify your contract using the Sourcify plugin.
RemixAI Copilot (enabled)

Step 3: Confirm Contract Deployment via MetaMask on Polygon Amoy

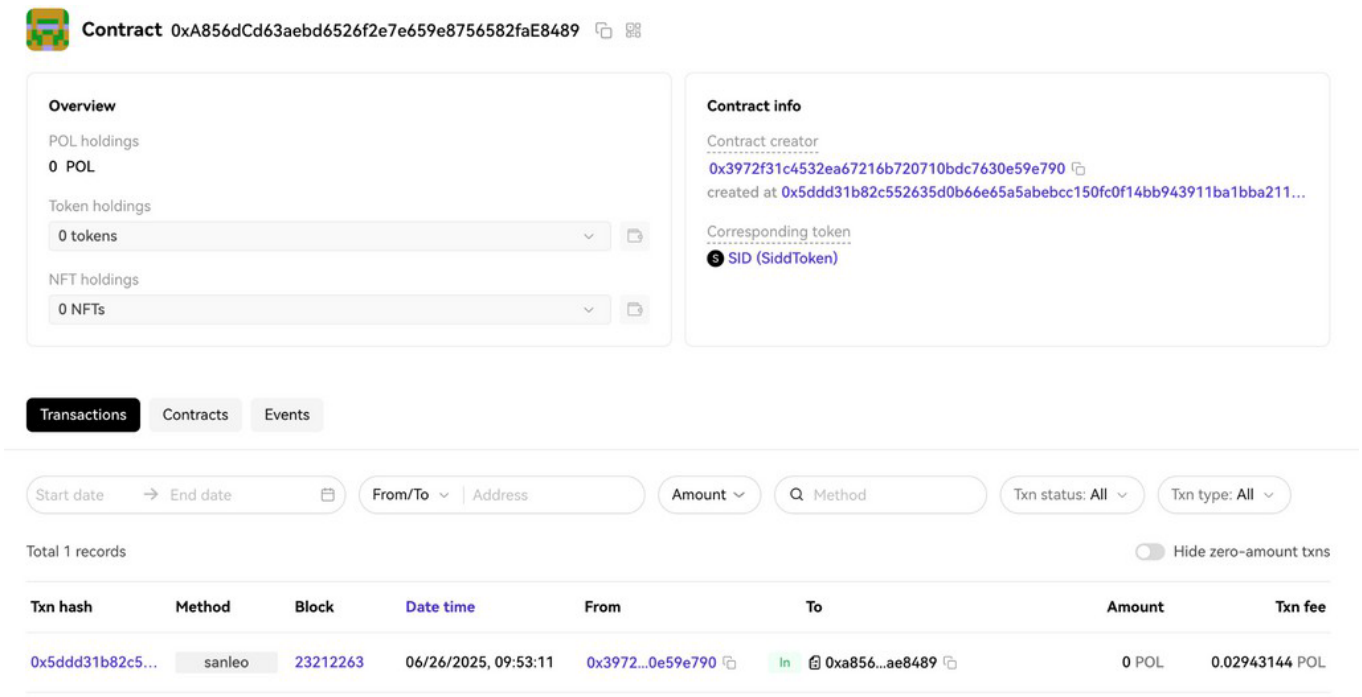


Step 4: Use Remix's Injected Web3 to Deploy Contract to Amoy

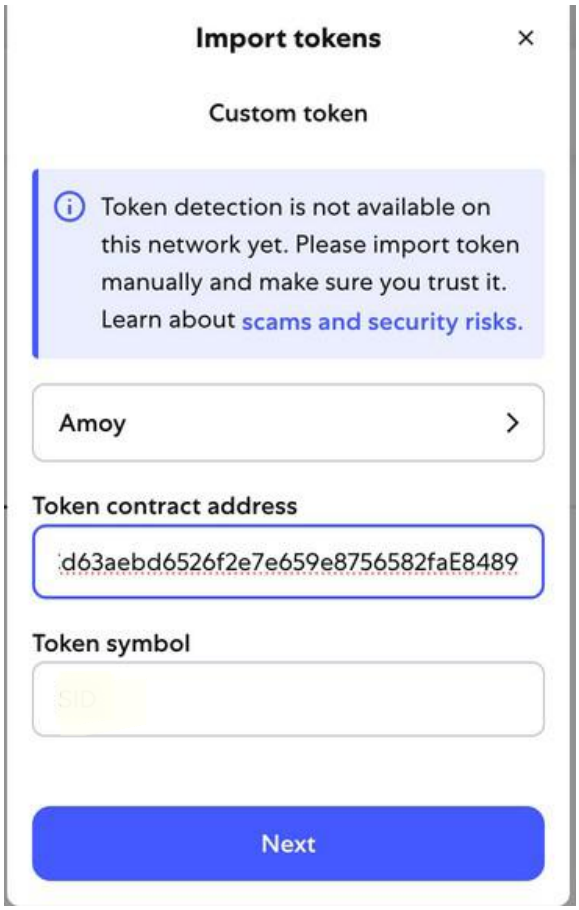




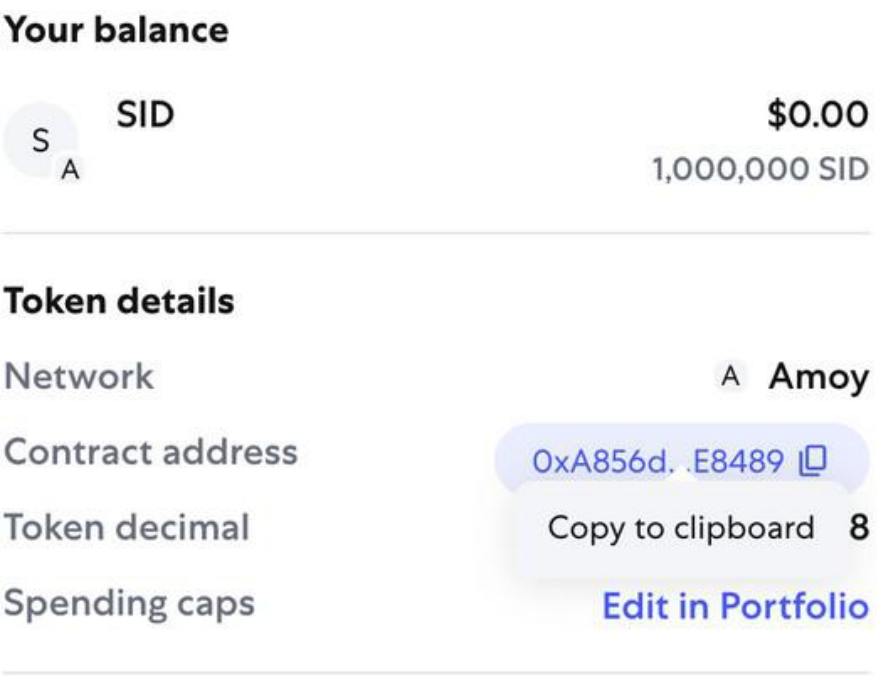
Step 5: View Deployed Contract on Amoy Block Explorer



Step 6: Manually Import the Token into MetaMask



Step 7: Verify Token is Added with Correct Balance



SolidityCode:

```
// SPDX-License-Identifier: MIT pragma
solidity ^0.8.0;
import "@openzeppelin/contracts/token/ERC20/ERC20.sol"; contract
MyToken is ERC20 {
   constructor() ERC20("VaiCoin", "VAION") {
      _mint(msg.sender, 1000000 * 10 ** decimals());
   }
}
```

Learnings

- ✅ Mastered ERC-20 token deployment via OpenZeppelin and Remix IDE.
- ✅ Understood MetaMask interaction with custom networks.
- ✅ Became aware of Polygon Mumbai deprecation and Amoy migration.
- ✅ Gained confidence working with smart contracts and testnets.