

Implementation of Stable coin - nUSD

Test Cases and Results

Connection to Tesnet

- Connect to Sepolia testnet via Metamask.
- Ensure minimum balance of 1 Ether in account.

The screenshot shows the Ethereum IDE (Remix) interface. The left sidebar contains various tools and status indicators. The main area is divided into sections: "DEPLOY & RUN TRANSACTIONS" on the left and the "nUSD_code_v3_Working_Final.sol" code editor on the right.

DEPLOY & RUN TRANSACTIONS (Left Side):

- ENVIRONMENT:** Set to "Injected Provider - MetaMask".
- ACCOUNT:** Set to "Sepolia (11155111) network" with address "0x087...78C69 (1.3032)".
- GAS LIMIT:** Set to 3000000.
- VALUE:** Set to 0 Wei.
- CONTRACT:** Compiled by Remix, named "nUSD - priyanka/Hiring Task".
- evm version:** paris.
- Deploy:** A prominent orange button.
- PUBLISH TO IPFS:** An unchecked checkbox.
- At Address:** A button to load a contract from an address.

nUSD_code_v3_Working_Final.sol (Right Side):

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

/*
 * ***** Notes *****
 *
 * - We will be using the Sepolia Testnet
 * - Total supply of nUSD tokens will be held by the contract address
 * - All transactions will be done in wei. The contract will handle wei-ether conversion internally.
 * - 1 ether = 1000000000000000000 wei (18 zeroes)
 * - User should deposit amount in Wei, in positive integers only.
 * - For all divisions and conversions, solidity by default generates only integer quotient, rounds down and ignores remainder.
 * - Floating point values will not be generated or accepted as input.
 * - Little loss of value in tokens or ethers is possible due to this limitation.
 */

contract nUSD {
    AggregatorV3Interface internal priceFeed;

    uint256 totalSupply = 1000;
    string internal constant name = "nUSD";
    mapping(address => uint256) balances;

    // event Transfer(address indexed From, address indexed To, uint256 Value);
    event Deposit(uint256 Wei_Deposited, uint256 Current_Exchange_Rate, uint256 Tokens_Generated, address indexed User);
    event Redeem(uint256 Collateral_Tokens, uint256 Tokens_Redeemed, uint256 Value_in_Wei, uint256 Value_in_Eth, address indexed User);

    // All transactions will be done in wei. The contract will handle wei-ether conversion internally.
    // 1 ether = 1000000000000000000 wei (18 zeroes)

    constructor() payable { }
}
```

Bottom Panel:

- 0 transactions recorded.
- Search bar: "Search with transaction hash or ...".
- Library list: "webs version 1.5.2", "ethers.js", "remix".
- Text: "Type the library name to see available commands."

Deploy contract

- Compile and deploy the contract
- Approve on Metamask when it prompts for approval.

The screenshot shows the Remix Ethereum IDE interface. The top navigation bar includes tabs for "Remix - Ethereum IDE", "Sepolia PoW Faucet", and "Ethereum Unit Converter | Ethe". The main workspace displays the Solidity code for the nUSD contract, which imports the AggregatorV3Interface from Chainlink. The code includes notes about using the Sepolia Testnet and handling Wei-ether conversion. The "DEPLOY & RUN TRANSACTIONS" sidebar is open, showing deployment options like "At Address" and a list of recorded transactions. A deployed contract named "nUSD AT 0xC25..." is visible, along with its balance of 0 ETH and various interaction buttons like deposit, redeem, and BalanceOf. The bottom section shows low-level interactions and a transaction history entry for the constructor call.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

/*
 * ***** Notes *****
 *
 * - We will be using the Sepolia Testnet
 * - Total supply of nUSD tokens will be held by the contract address
 * - All transactions will be done in wei. The contract will handle wei-ether conversion internally.
 * - 1 ether = 1000000000000000000 wei (18 zeroes)
 * - User should deposit amount in Wei, in positive integers only.
 * - For all divisions and conversions, solidity by default generates only integer quotient, rounds down and ignores remainder.
 * - Floating point values will not be generated or accepted as input.
 * - Little loss of value in tokens or ethers is possible due to this limitation.
 */

contract nUSD {
    AggregatorV3Interface internal priceFeed;

    function deposit(uint256 amount) public;
    function redeem(uint256 numTokens) public;
    function BalanceOf() external view returns (address);

    function getExchRate() external view returns (uint256);
    function TotalSupply() external view returns (uint256);
    function Unminted() external view returns (uint256);

    function LowLevelInteraction(string memory _functionName, bytes memory _calldata) external;
}
```

Transactions recorded: 1

Deployed Contracts: NUSD AT 0xC25...E6DC4 (BLOCK)

Balance: 0 ETH

deposit uint256 amount

redeem uint256 numTokens

BalanceOf 0xC255780fd92159355d9EB620

getExchRa...

TotalSupply

Unminted

Low level interactions

CALLDATA

Transact

Debug

[block:3795355 txIndex:32] from: 0x087...78C69 to: nUSD.(constructor) value: 0 wei data: 0x608...20033 logs: 0 hash: 0x741...4ed3b

call to nUSD.BalanceOf

CALL [call] from: 0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69 to: nUSD.BalanceOf(address) data: 0xd99...e6dc4

Debug

Initial State

Initial state after contract deployment will be like this :

The screenshot shows the Remix Ethereum IDE interface. On the left, there's a sidebar with icons for deploying contracts, viewing transactions, and interacting with deployed contracts. The main area has tabs for "DEPLOY & RUN TRANSACTIONS" and "nUSD_code_v3_Working_Final.sol". The code editor contains the Solidity source code for the nUSD contract, which includes imports from Chainlink and comments explaining its behavior. Below the code, the "Deployed Contracts" section shows a single deployed contract named "NUSD AT 0xC25...E6DC4 (BLOCK)". The "Balance" is listed as "0 ETH". Underneath, there are four buttons: "deposit", "redeem", "BalanceOf", and "getExchRate". The "BalanceOf" button is highlighted with an orange border. To the right of the buttons, a transaction log shows a call to "nUSD.BalanceOf(address)" with a result of "0: uint256: 0". Below this, the "getExchRate" button is also highlighted with an orange border, and its corresponding transaction log shows a call to "nUSD.getExchRate()" with a result of "0: int256: 1889". Further down, the "TotalSupply" and "Unminted" buttons are highlighted with orange borders, and their transaction logs show calls to "nUSD.TotalSupply()" and "nUSD.Unminted()" respectively, both resulting in "0: uint256: 1000".

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

/*
 * ***** Notes *****
 *
 * - We will be using the Sepolia Testnet
 * - Total supply of nUSD tokens will be held by the contract address
 * - All transactions will be done in wei. The contract will handle wei-ether conversion internally.
 * - 1 ether = 1000000000000000000 wei (18 zeroes)
 * - User should deposit amount in Wei, in positive integers only.
 * - For all divisions and conversions, solidity by default generates only integer quotient, rounds down and ignores remainder.
 * - Floating point values will not be generated or accepted as input.
 * - Little loss of value in tokens or ethers is possible due to this limitation.
 */

contract nUSD {
    AggregatorV3Interface internal priceFeed;
}
```

Transactions recorded: 1

Deployed Contracts:

- NUSD AT 0xC25...E6DC4 (BLOCK)
Balance: 0 ETH
 - deposit: uint256 amount
 - redeem: uint256 numTokens
 - BalanceOf: 0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69
 - 0: uint256: 0
 - getExchRate:
 - 0: int256: 1889
 - TotalSupply:
 - 0: uint256: 1000
 - Unminted:
 - 0: uint256: 1000

Low level interactions

CALldata

Deposit

We can deposit ethers to get tokens in return. Ensure to convert ethers into Wei and then make deposits.

Deposit

Deposit operation is successful.

The screenshot shows the Remix Ethereum IDE interface with the following details:

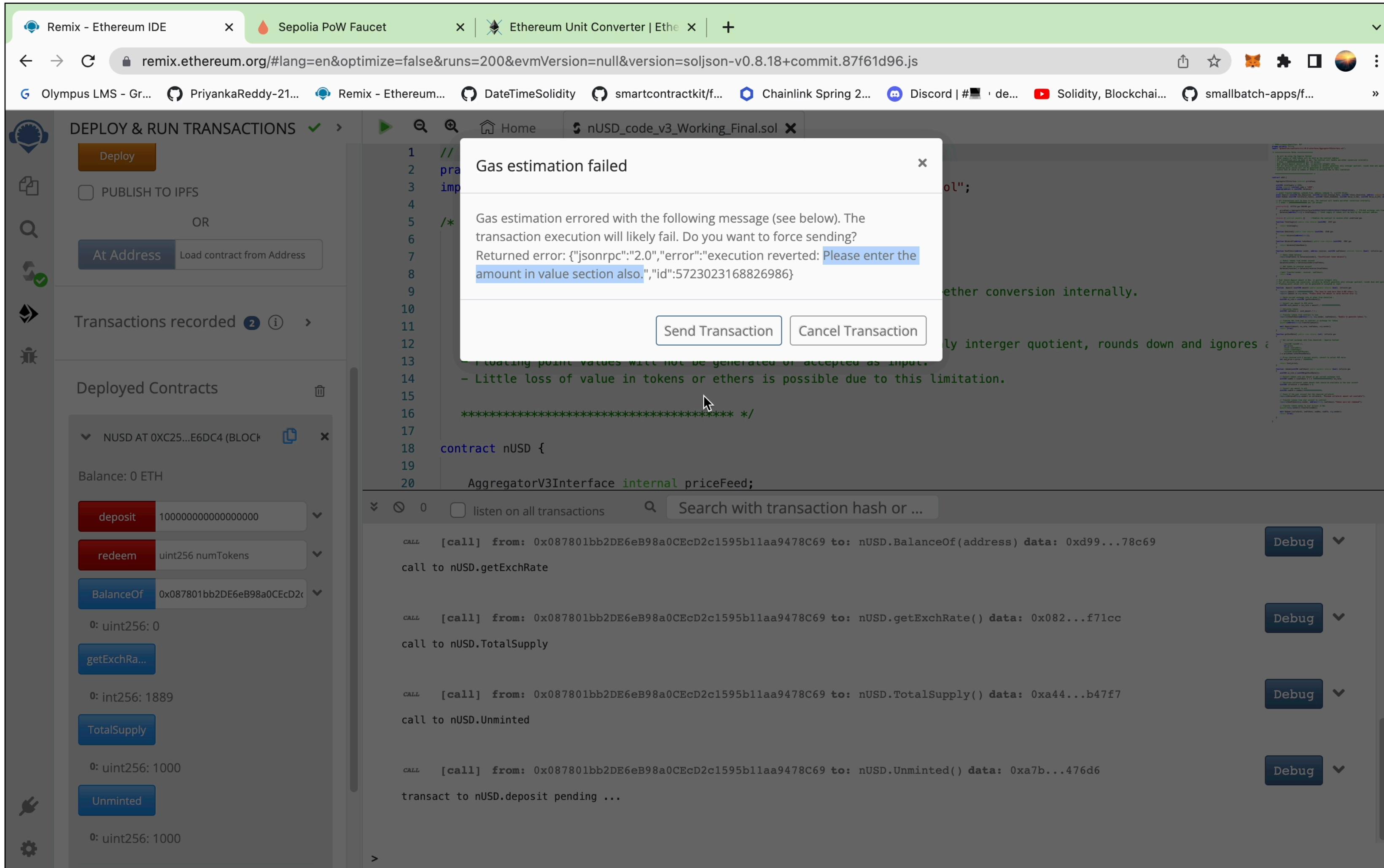
- Deploy & Run Transactions:** Shows a transaction record for a deposit of 0.1 ETH to the contract address 0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69.
- Deployed Contracts:** The NUSD contract at address 0xC25...E6DC4 (BLOCK) is selected. It shows:
 - deposit:** 1000000000000000000 Wei (0.1 ETH)
 - redeem:** uint256 numTokens
 - BalanceOf:** 0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69 (0: uint256: 94)
 - getExchRa...**
 - TotalSupply:** 0: uint256: 1000
 - Unminted:** 0: uint256: 906
- Event Log:** An event log entry for the Deposit event is shown, containing the following data:

```
"event": "Deposit",
"args": {
    "0": "1000000000000000000",
    "1": "1889",
    "2": "94",
    "3": "0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69",
    "Wei_Deposited": "1000000000000000000",
    "Current_Exhange_Rate": "1889",
    "Tokens_Generated": "94",
    "User_Address": "0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69"
}
```
- Low level interactions:** Shows two calls:
 - CALL [call] from: 0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69 to: nUSD.BalanceOf(address) data: 0xd99...78c69
 - CALL [call] from: 0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69 to: nUSD.Unminted() data: 0xa7b...476d6

Deposit

Error when Wei amount is not entered in "value" section.

Solution - Enter the Wei amount in "value" section as well as the "deposit" section before clicking on deposit button.



Redeem

Before redeeming tokens, we need to ensure a minimum balance of (No. of ETH to be redeemed* ETH value in USD * 2) amount of tokens in the user account. To achieve this, deposit more Wei and accordingly the tokens will be sent to user account for collateral. Now, we can proceed with redeem operation.

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible, showing a deployed contract named 'NUSD AT 0xC25...E6DC4 (BLOCK)' with a balance of 0.4 ETH. In the center, the code editor displays the Solidity contract code for 'nUSD_code_v3_Working_Final.sol'. The 'redeem' function is highlighted with a red box and an arrow points from it to a callout box containing instructions. The callout box contains the following text:

Enter the amount of tokens to be redeemed here. Ensure that user account has atleast 4 times tokens as the number of tokens entered here. Then click on redeem. Also approve the transaction on metamask when prompted.

The contract code includes comments explaining the use of the Sepolia Testnet and the handling of wei-ether conversion.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

/*
***** Notes *****
- We will be using the Sepolia Testnet
- Total supply of nUSD tokens will be held by the contract address
- All transactions will be done in wei. The contract will handle wei-ether conversion internally.
- 1 ether = 1000000000000000000 wei (18 zeroes)
- User should deposit amount in Wei, in positive integers only.
- For all divisions and conversions, solidity by default generates only interger quotient, rounds down and ignores remainder.
- Floating point values will not be generated or accepted as input.
- Little loss of value in tokens or ethers is possible due to this limitation.

```

Redeem

Redeem operation was successful.

The screenshot shows the Ethereum IDE interface with the following details:

- Deploy & Run Transactions:** A sidebar on the left with a "Deploy" button and "At Address" selected.
- Transactions recorded:** 7 transactions recorded.
- Deployed Contracts:** A list of deployed contracts, with "NUSD AT 0xC25...E6DC4 (BLOCK)" expanded. It shows:
 - Balance:** 0.300476442562202224 ETH
 - deposit:** 1000000000000000000
 - redeem:** 94
 - BalanceOf:** 0x087801bb2DE6eB98a0CEcD2c
 - getExchRa...**
 - TotalSupply:** 0: int256: 1889
 - Unminted:** 0: uint256: 1000
- Contract Source Code:** The code for `nUSD_code_v3_Working_Final.sol` is displayed, containing comments about the Sepolia Testnet and the conversion of Wei to Ether.
- Event Log:** An event log entry for the "Redeem" event is shown, with transaction details:

```
"from": "0xC255780fd92159355d9EB620b1CF07730a0e6DC4",
"topic": "0xf2861d2b32e2bf043df33cb44cb4d197154fcf0092c19fdc3610df8ebbf1394",
"event": "Redeem",
"args": {
    "0": "376",
    "1": "94",
    "2": "99523557437797776",
    "3": "0",
    "4": "0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69",
    "Collateral_Tokens": "376",
    "Tokens_Redeemed": "94",
    "Value_in_Wei": "99523557437797776",
    "User_Address": "0x087801bb2DE6eB98a0CEcD2c1595b11aa9478C69"
}
```

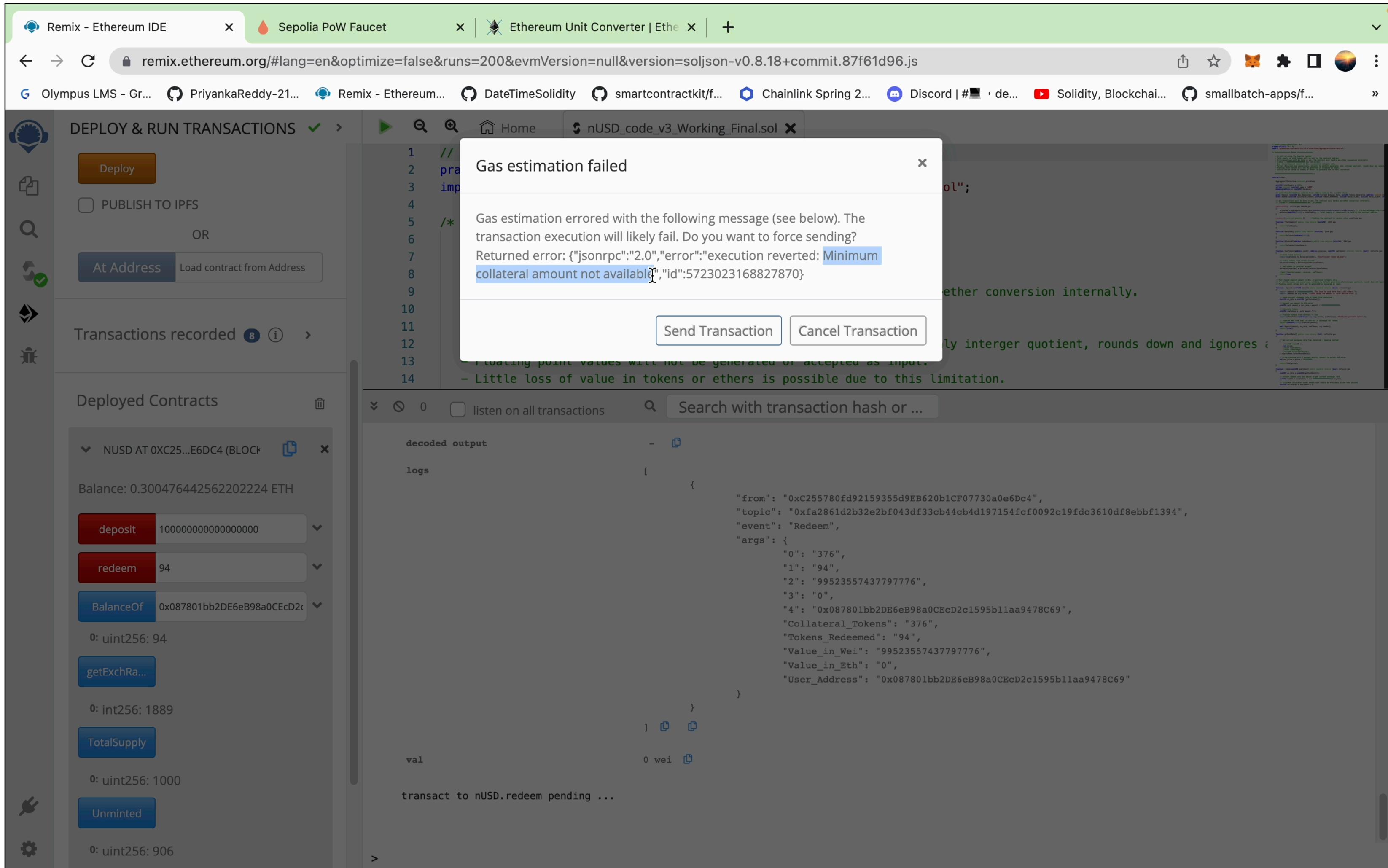
Annotations highlight the following:

- An orange box surrounds the "Balance" field in the contract interface, with a callout pointing to it: "Wei amount for 94 tokens, as per current exchange rate is sent back to user account."
- An orange box surrounds the event log entry, with a callout pointing to it: "Event is generated with transaction details".

Redeem

Error when minimum collateral token amount is not available in user account.

Solution - Either add more tokens to user account, or reduce the number of tokens to be redeemed.



Thank You !