**Blockchain Lab Experiment 4**

**Shashwat Tripathi**

**D20A Roll No: 64**

**Aim:** Implement the Blockchain platform Ganache.

**Practical: Case Study on Central Bank Digital Currency(CBDC)**

**Code:**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract CBDC {

string public name = "Central Bank Digital Currency";

string public symbol = "CBDC";

uint8 public decimals = 18;

uint256 public totalSupply;

mapping(address => uint256) public balanceOf;

mapping(address => mapping(address => uint256)) public allowance;

address public centralBank;

event Transfer(address indexed from, address indexed to, uint256 value);

event Approval(address indexed owner, address indexed spender, uint256 value);

event Mint(address indexed to, uint256 value);

event Burn(address indexed from, uint256 value);

modifier onlyCentralBank() {

require(msg.sender == centralBank, "Only central bank can execute this");

\_;

}

constructor(uint256 \_initialSupply) {

centralBank = msg.sender;

mint(centralBank, \_initialSupply);

}

function mint(address \_to, uint256 \_amount) public onlyCentralBank {

totalSupply += \_amount;

balanceOf[\_to] += \_amount;

emit Mint(\_to, \_amount);

emit Transfer(address(0), \_to, \_amount);

}

function burn(uint256 \_amount) public onlyCentralBank {

require(balanceOf[centralBank] >= \_amount, "Insufficient balance");

balanceOf[centralBank] -= \_amount;

totalSupply -= \_amount;

emit Burn(centralBank, \_amount);

emit Transfer(centralBank, address(0), \_amount);

}

function transfer(address \_to, uint256 \_amount) public returns (bool success) {

require(balanceOf[msg.sender] >= \_amount, "Insufficient balance");

balanceOf[msg.sender] -= \_amount;

balanceOf[\_to] += \_amount;

emit Transfer(msg.sender, \_to, \_amount);

return true;

}

function approve(address \_spender, uint256 \_amount) public returns (bool success) {

allowance[msg.sender][\_spender] = \_amount;

emit Approval(msg.sender, \_spender, \_amount);

return true;

}

function transferFrom(address \_from, address \_to, uint256 \_amount) public returns (bool success) {

require(balanceOf[\_from] >= \_amount, "Insufficient balance");

require(allowance[\_from][msg.sender] >= \_amount, "Allowance exceeded");

balanceOf[\_from] -= \_amount;

balanceOf[\_to] += \_amount;

allowance[\_from][msg.sender] -= \_amount;

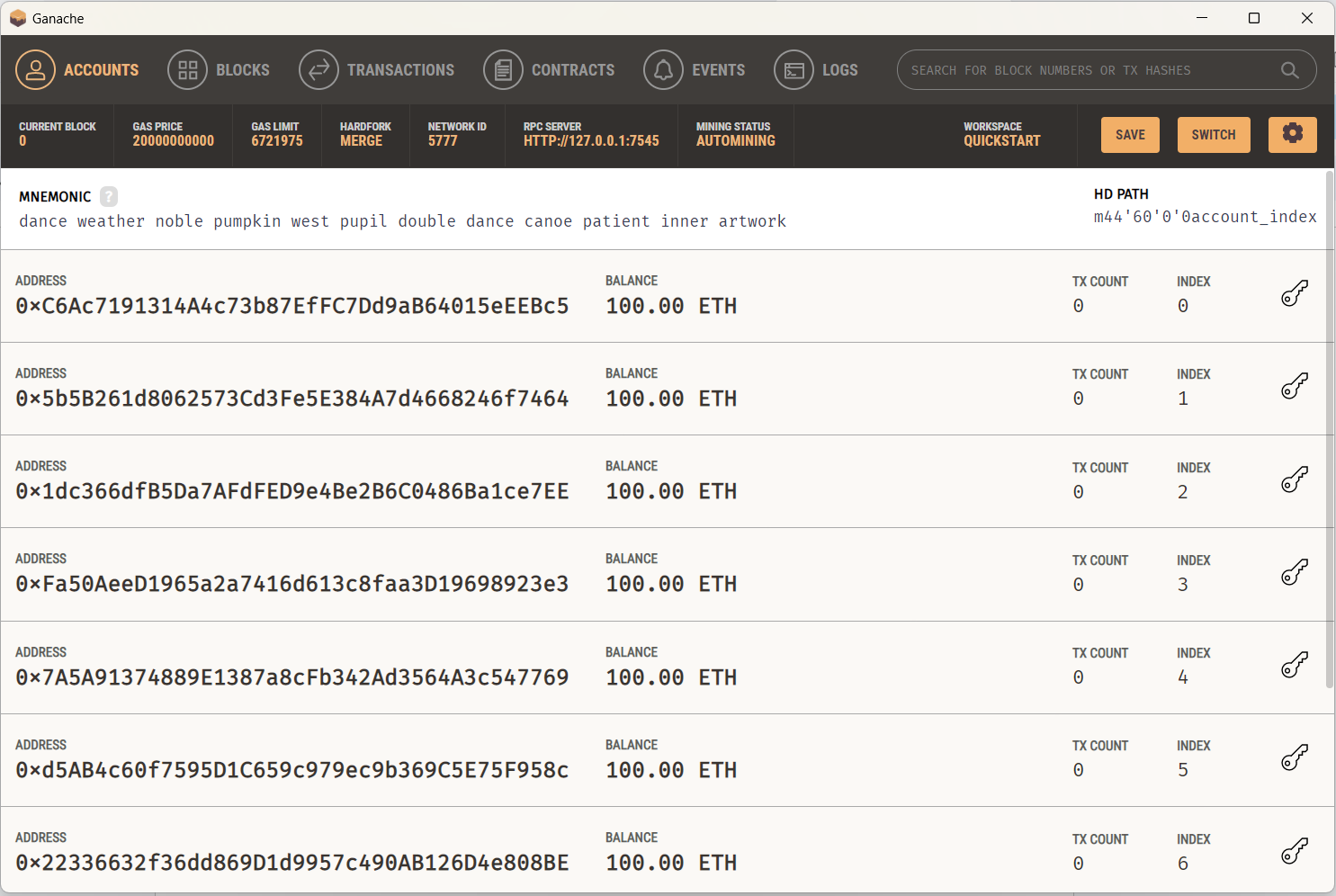
emit Transfer(\_from, \_to, \_amount);

return true;

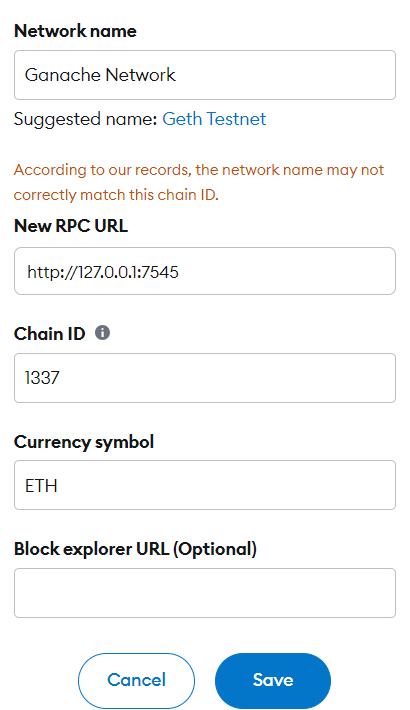
}

}

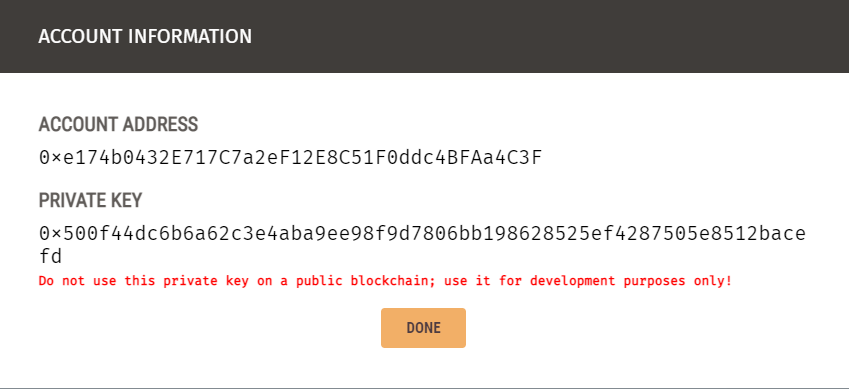
Installing Ganache on Windows



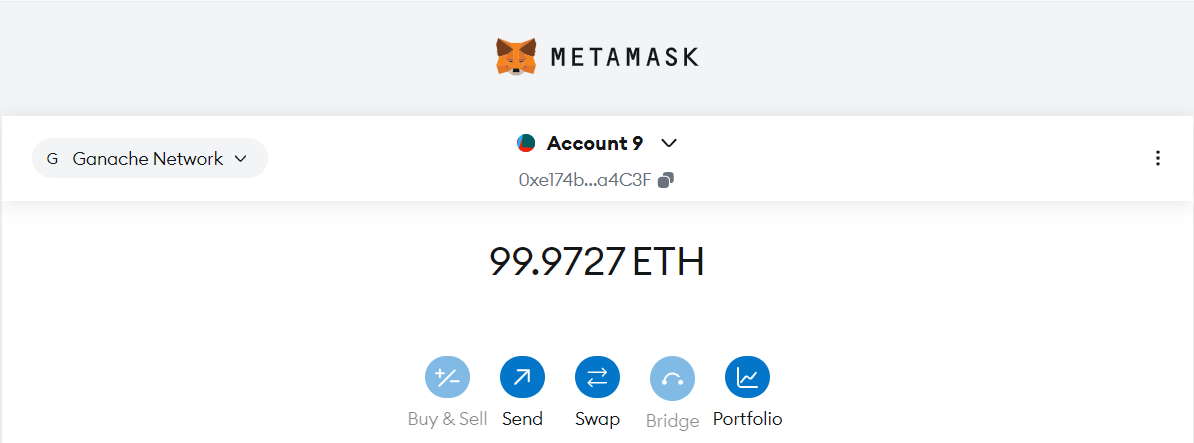
Setting up the Ganache network on Metamask



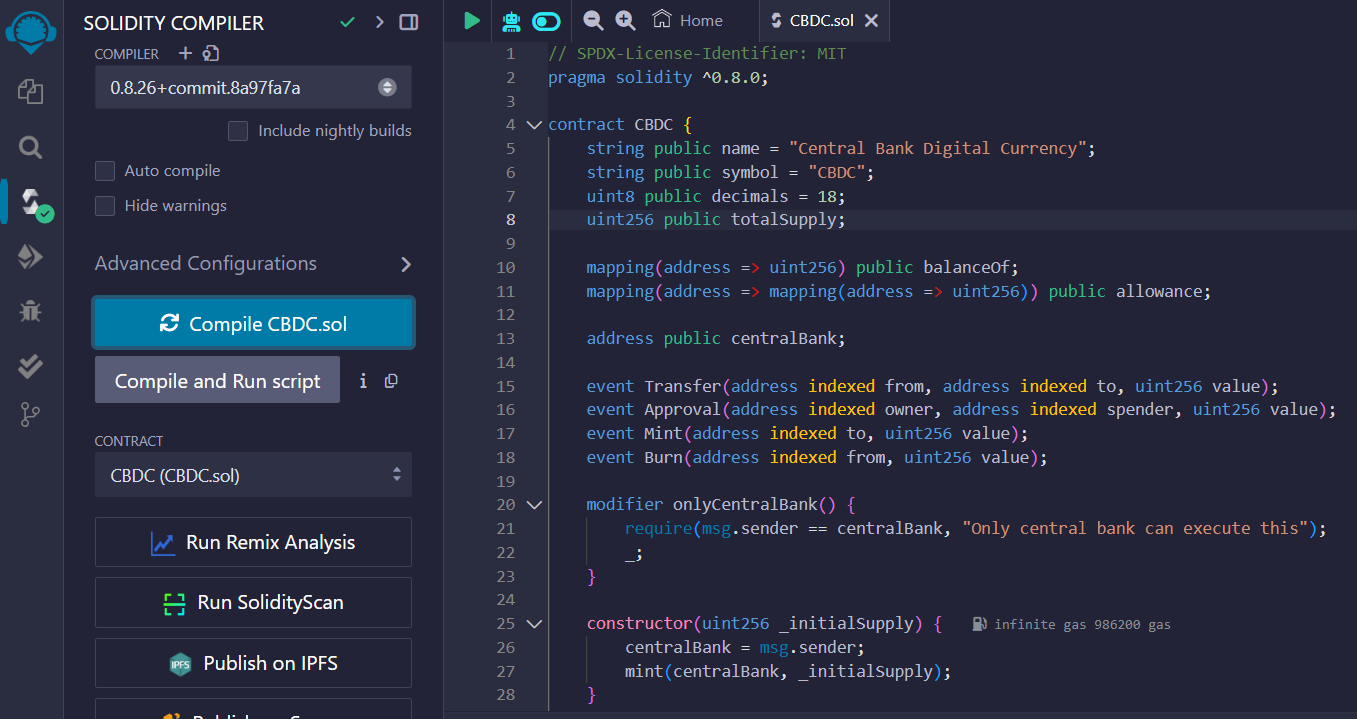
Copy the private key from the Ganache account and paste it into the Metamask.



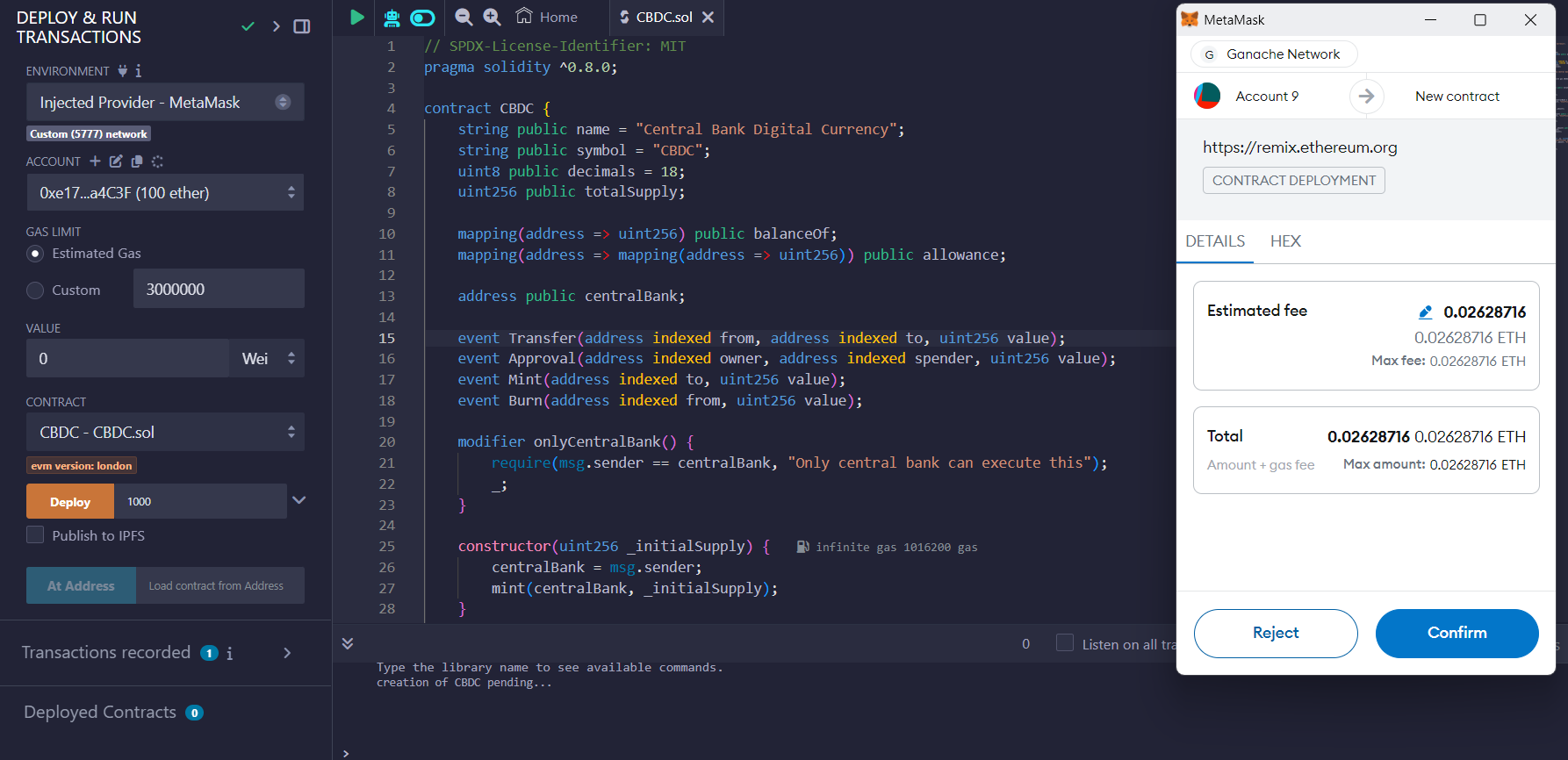
Importing the Ganache account on the Metamask by entering the private key.



Compile the solidity contract

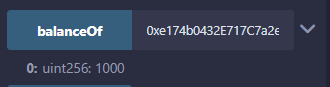


Deployed the contract with central bank account.

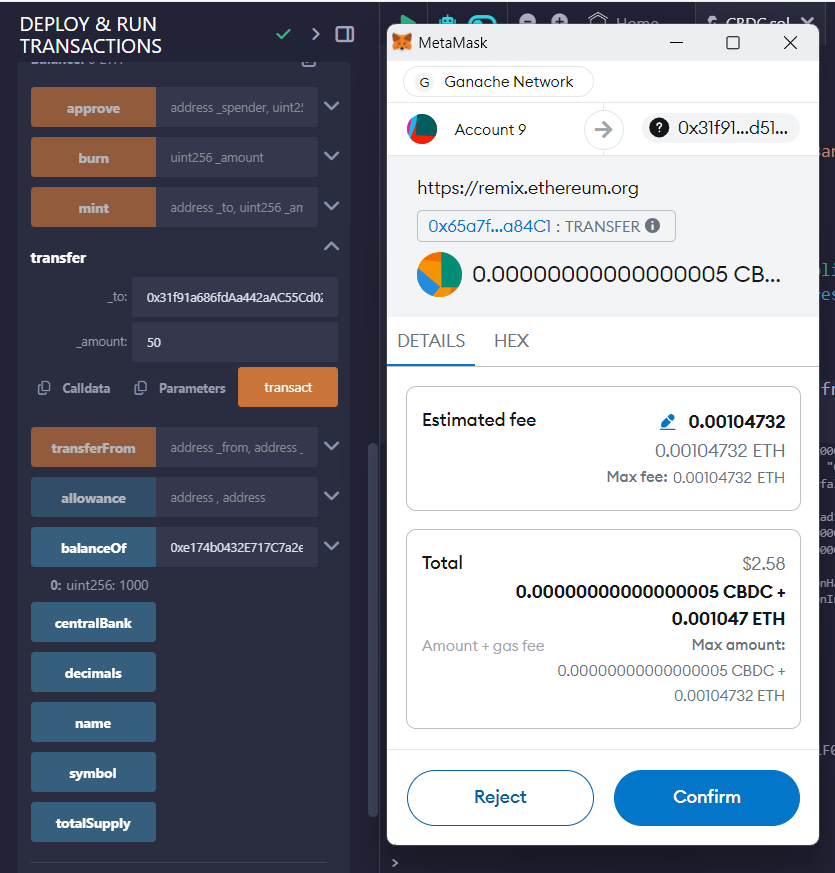


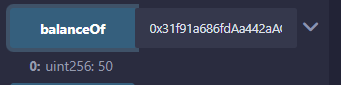


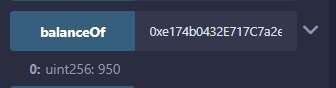
Owner now has 1000 coins



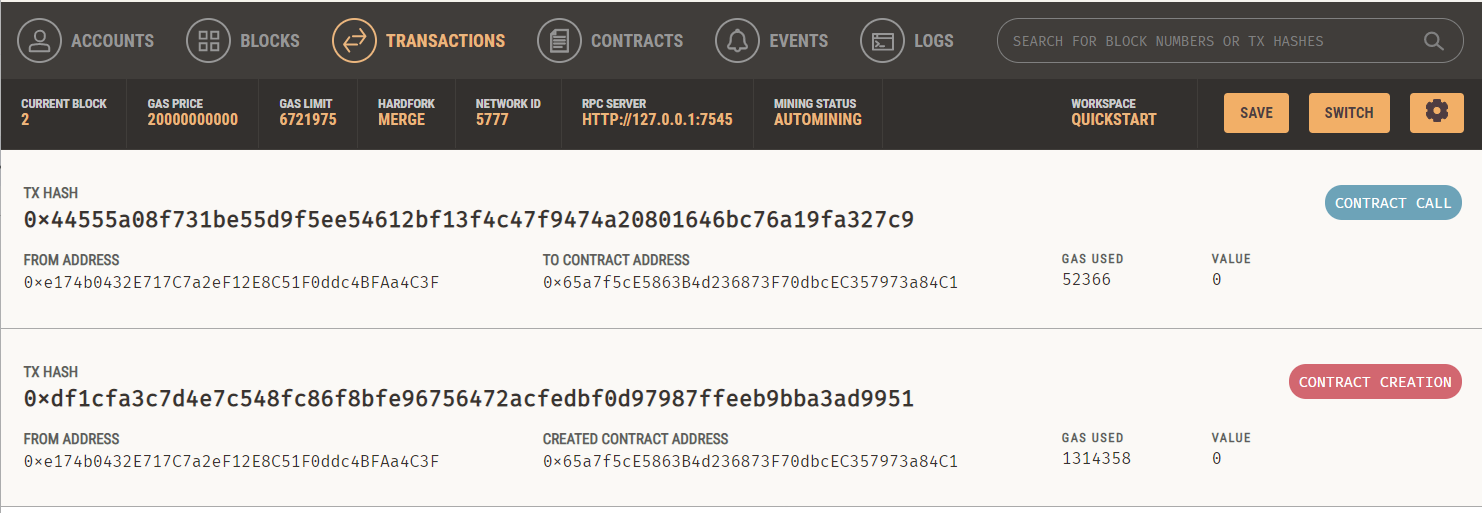
Sending 50 coins to other address

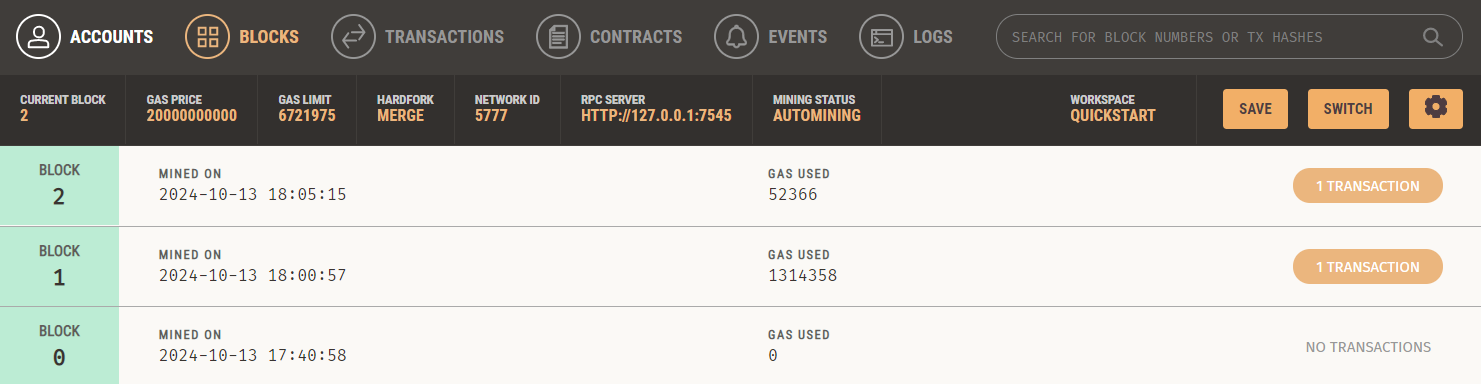






Verifying the transactions in the Ganache





Conclusion: Thus we have set up the Ganache platform and executed the contract for Central Bank Digital Currenct(CBDC) and successfully verified.