## Department of Computer Engineering

Academic Term: Jan-May 23-24

**Class:** B.E Computer Sem -VII **Subject:** Blockchain Technology Lab **Subject Code :** CSDL7022

|  |  |
| --- | --- |
| **Practical No:** | **4** |
| **Title:** | **Creating Tokens/Coins on Ethereum Network** |
| **Date of Performance:** | **18/08/2023** |
| **Date of Submission:** | **18 08/2023** |
| **Roll No:** | **9427** |
| **Name of the Student:** | **Atharva Prashant Pawar** |

Evaluation:

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Rubric** | **Grade** |
| **1** | **Time Line (2)** |  |
| **2** | **Output (3)** |  |
| **3** | **Code optimization (2)** |  |
| **4** | **Post lab (3)** |  |

**Signature of the Teacher :**

## Experiment No. 4

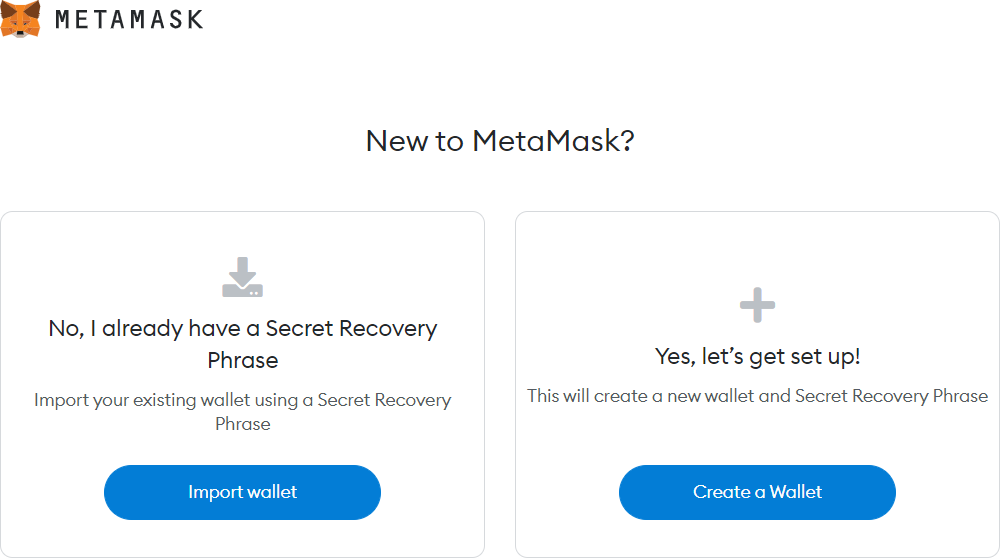
**Creating Tokens/Coins on Ethereum Network**

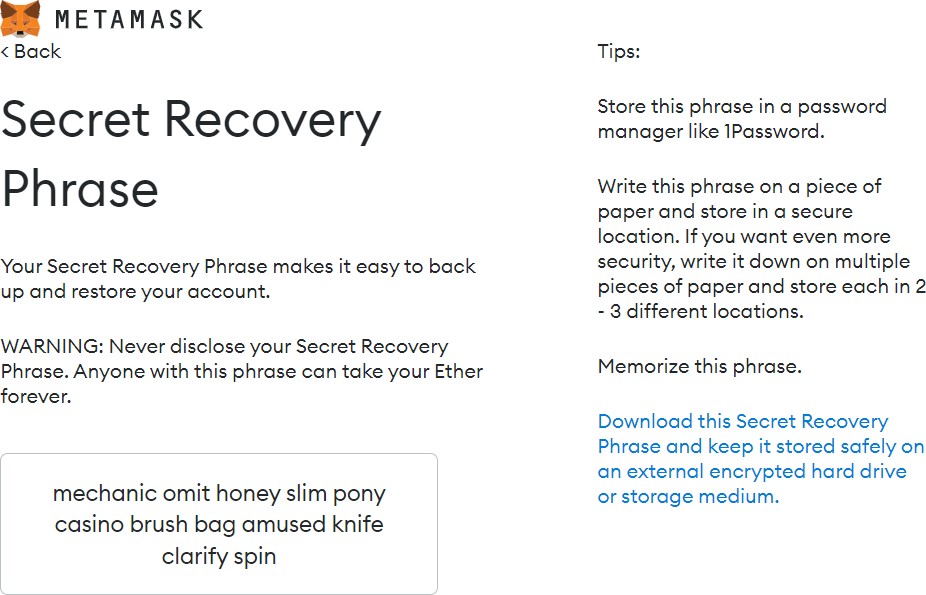
**Aim:** Creating a Metamask wallet and performing the transactions to add Ethers and Tokens.

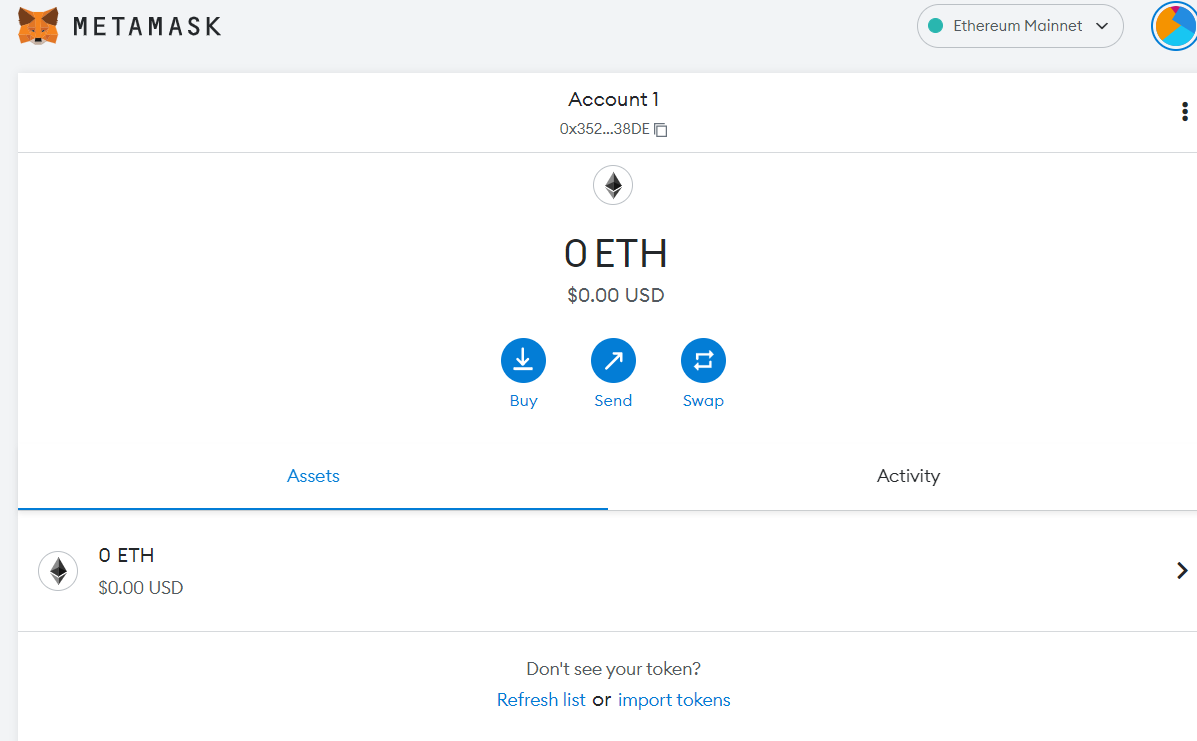
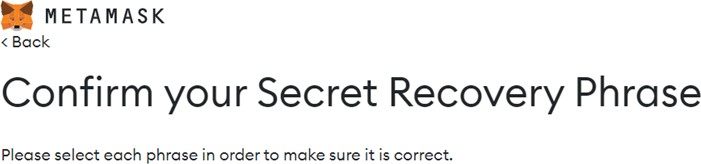
## Theory:

### Step 1: Create a wallet at meta-mask

* Install MetaMask in Chrome browser and enable it. After installation, click on its icon on the top right of the browser page. It will open in a new tab of the browser. Click on “Create Wallet” and click “I agree” to proceed further. Now you are agreeing to the terms and conditions. Create a password and then it will send you a secret backup phrase used for backing up and restoring the account.
* It should not be disclosed or shared with anyone, as this phrase can steal your Ethers. Ensure that you are in the “Main Ethereum Network.” If you find a checkmark next to “Main Ethereum Network”, you are in the right place. You can see following pages.



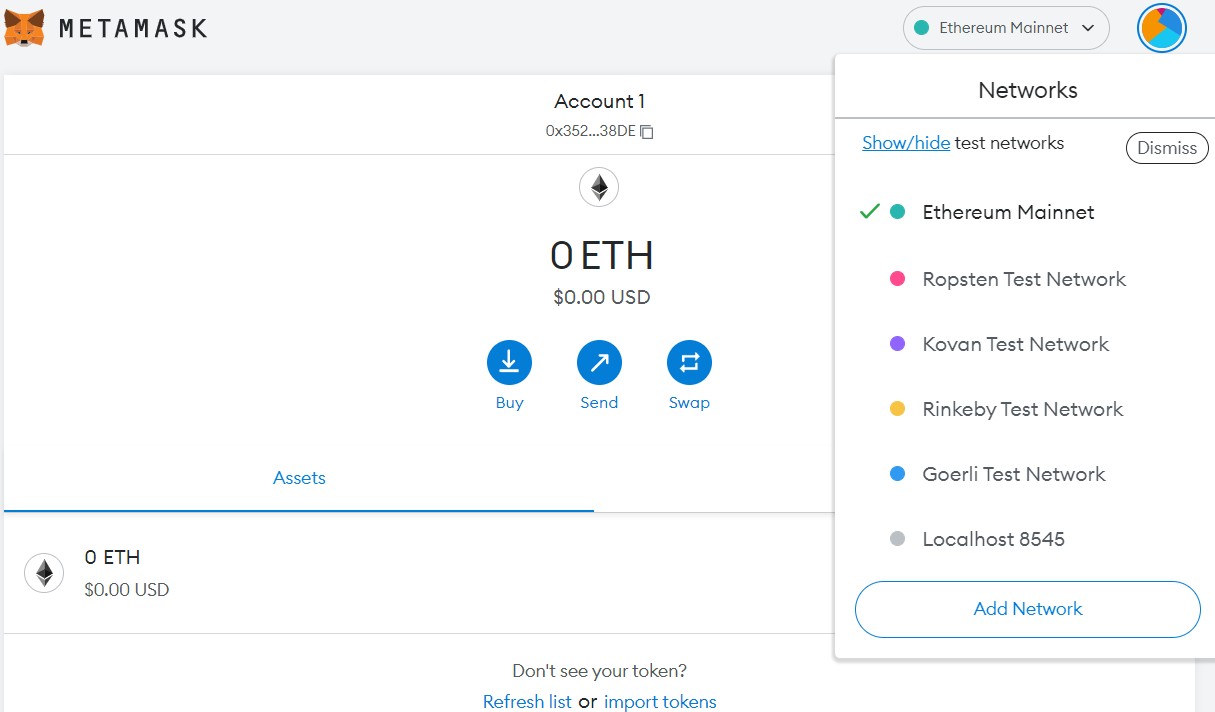




### Step 2: Select any one test network

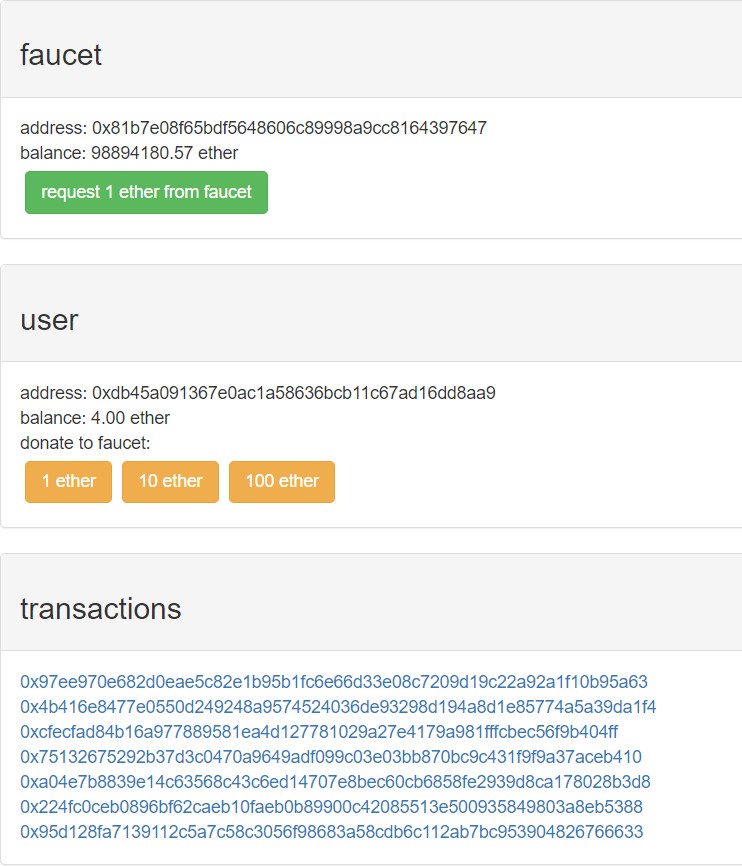
Click on show networks. Enable show networks and other information. Following test networks can be seen in your MetaMask wallet. These networks are only for the purpose of testing and ethers involved in it have no value.

* Ropsten Test Network
* Kovan Test Network
* Rinkeby Test Network
* Goerli Test Network Select Ropsten Test Network.



**Step 3:** Add some dummy Ethers to your wallet

* To test the smart contract, your MetaMask wallet should contain some dummy ethers. For example, if you test a contract using the Ropsten test network, select it and you will find 0 ETH as the initial balance in your account. Click on the “Deposit” and “Get Ether” buttons under
* Test Faucet in order to add dummy ethers. Open site <https://faucet.metamask.io/>. Link your wallet account and add one dummy ether. You will receive one ether in account as below. Click Request 1 Ether from faucet.
* Click on Buy button. Buy test ethers.



**Step 4:** Compile and Deploy following contract in Ethereum Remix IDE. Give name to file token.sol. Deploy by selecting **QKCToken–contract/token.sol**. **Select Environment as Injected Provider-Metamask.** Replace **YOUR\_METAMASK\_WALLET\_ADDRESS** in code by your wallet address which you have created.

pragma solidity ^0.4.24;.

//Safe Math Interface contract SafeMath {

function safeAdd(uint a, uint b) public pure returns (uint c) { c = a + b;

require(c >= a);

}

function safeSub(uint a, uint b) public pure returns (uint c) { require(b <= a);

c = a - b;

}

function safeMul(uint a, uint b) public pure returns (uint c) { c = a \* b;

require(a == 0 || c / a == b);

}

function safeDiv(uint a, uint b) public pure returns (uint c) { require(b > 0);

c = a / b;

}

}

//ERC Token Standard #20 Interface contract ERC20Interface {

function totalSupply() public constant returns (uint);

function balanceOf(address tokenOwner) public constant returns (uint balance);

function allowance(address tokenOwner, address spender) public constant returns (uint remaining);

function transfer(address to, uint tokens) public returns (bool success); function approve(address spender, uint tokens) public returns (bool success);

function transferFrom(address from, address to, uint tokens) public returns (bool success);

event Transfer(address indexed from, address indexed to, uint tokens);

event Approval(address indexed tokenOwner, address indexed spender, uint tokens);

}

//Contract function to receive approval and execute function in one call contract ApproveAndCallFallBack {

function receiveApproval(address from, uint256 tokens, address token, bytes data) public;

}

//Actual token contract

contract QKCToken is ERC20Interface, SafeMath { string public symbol;

string public name; uint8 public decimals; uint public \_totalSupply;

mapping(address => uint) balances;

mapping(address => mapping(address => uint)) allowed;

constructor() public { symbol = "QKC";

name = "QuikNode Coin"; decimals = 2;

\_totalSupply = 100000; balances[YOUR\_METAMASK\_WALLET\_ADDRESS] = \_totalSupply;

emit Transfer(address(0), YOUR\_METAMASK\_WALLET\_ADDRESS, \_totalSupply);

}

function totalSupply() public constant returns (uint) { return \_totalSupply - balances[address(0)];

}

function balanceOf(address tokenOwner) public constant returns (uint balance) { return balances[tokenOwner];

}

function transfer(address to, uint tokens) public returns (bool success) { balances[msg.sender] = safeSub(balances[msg.sender], tokens); balances[to] = safeAdd(balances[to], tokens);

emit Transfer(msg.sender, to, tokens); return true;

}

function approve(address spender, uint tokens) public returns (bool success) { allowed[msg.sender][spender] = tokens;

emit Approval(msg.sender, spender, tokens); return true;

}

function transferFrom(address from, address to, uint tokens) public returns (bool success) { balances[from] = safeSub(balances[from], tokens);

allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens); balances[to] = safeAdd(balances[to], tokens);

emit Transfer(from, to, tokens); return true;

}

function allowance(address tokenOwner, address spender) public constant returns (uint remaining) {

return allowed[tokenOwner][spender];

}

function approveAndCall(address spender, uint tokens, bytes data) public returns (bool success) {

allowed[msg.sender][spender] = tokens; emit Approval(msg.sender, spender, tokens);

ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data);

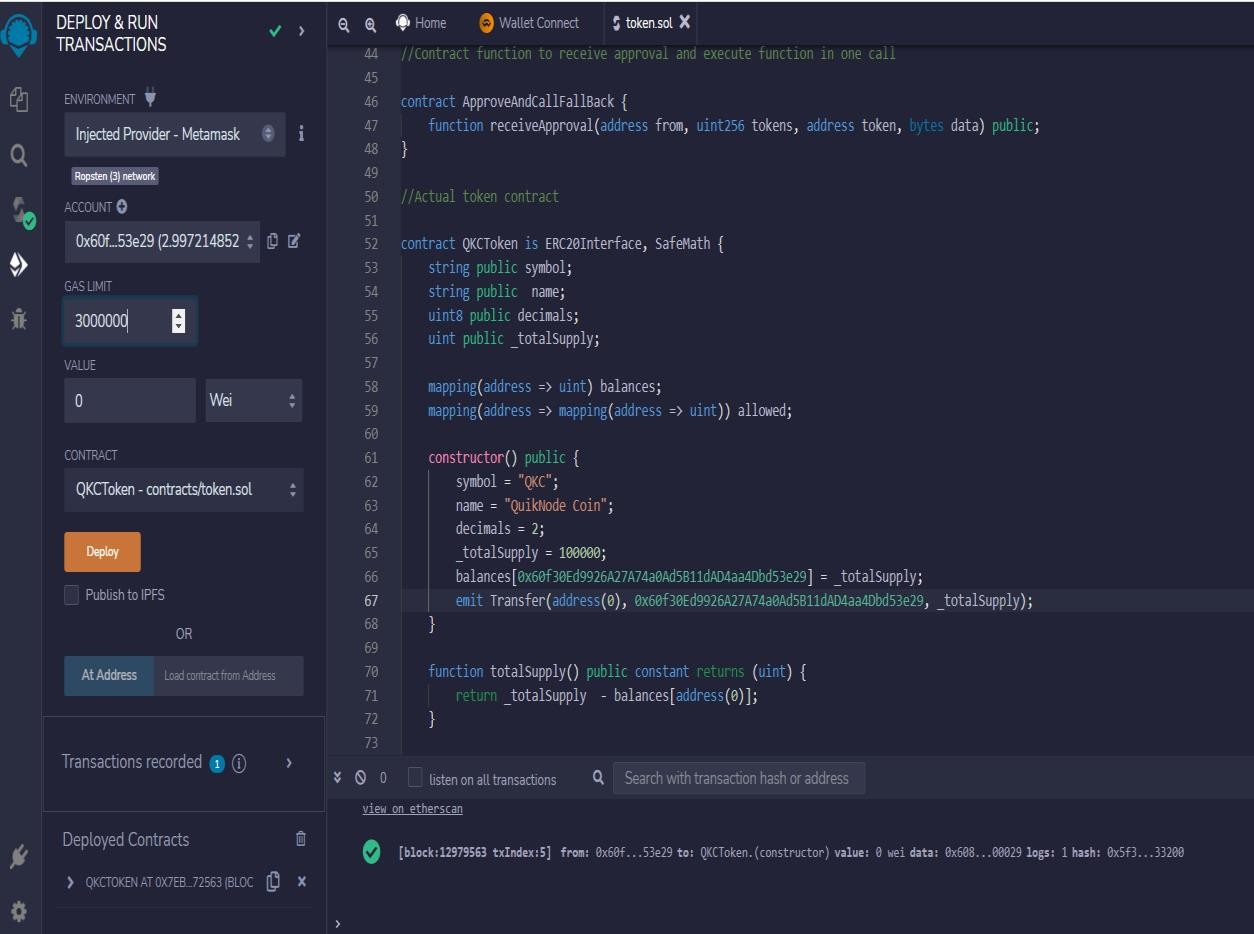
return true;

}

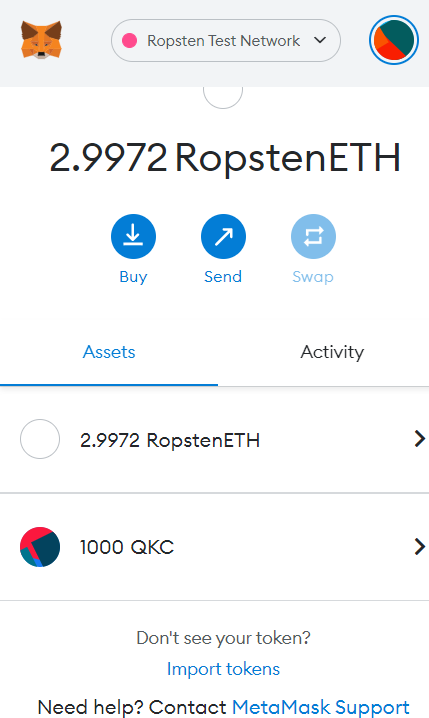
function () public payable { revert();

}

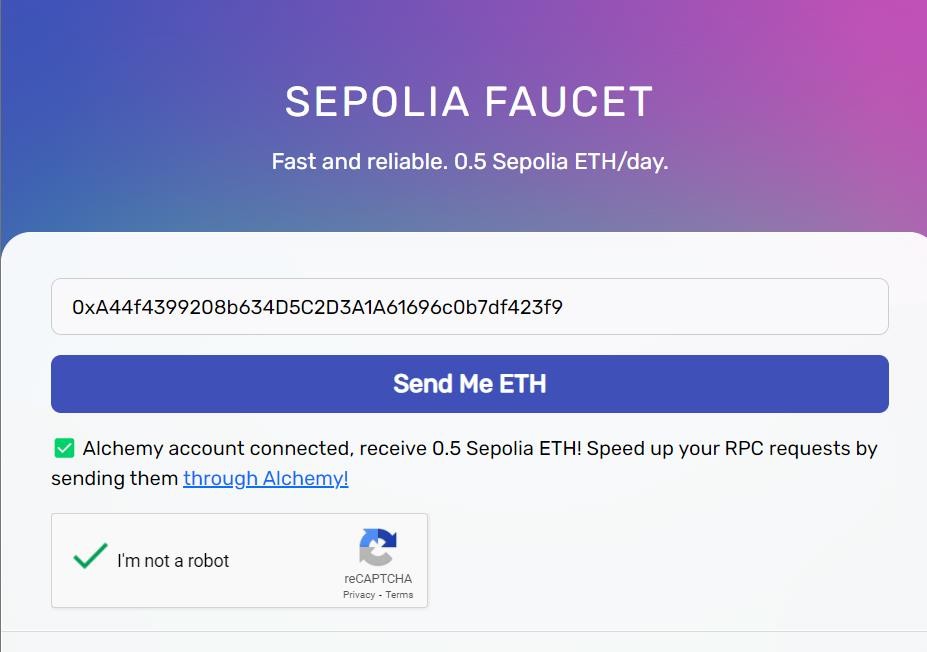
}



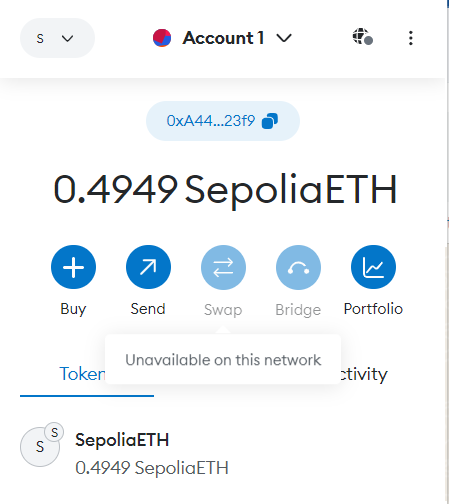
**Step 5:** Copy deployed contract address. Open Metamask. Click on Assest. Goto import token. Paste Token contract address, Select Token Symbol as QKC and Token decimal as 2 and add tokens. You will get 1000 QKC added as below.



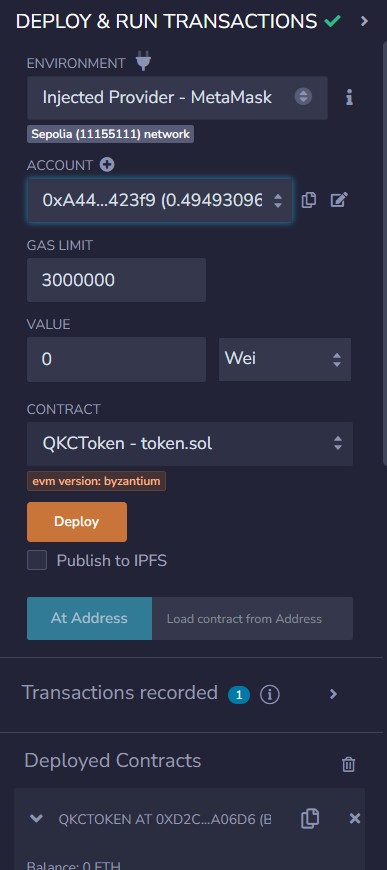
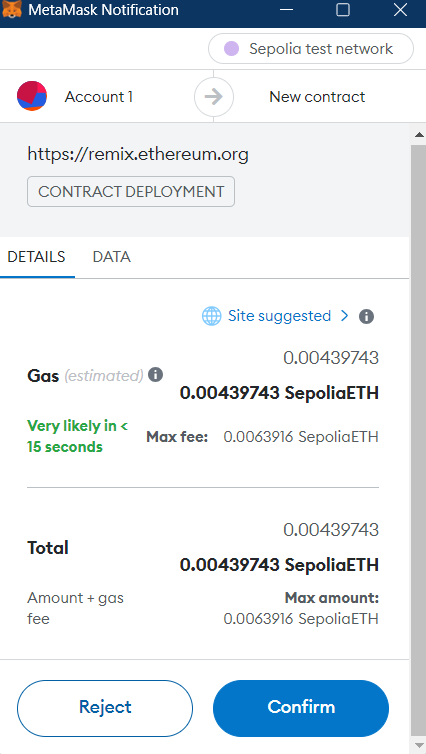
# A screenshot of a login form Description automatically generatedOutput :



1. **Creating new metamask account 4 and getting test eth from sepolia faucet**



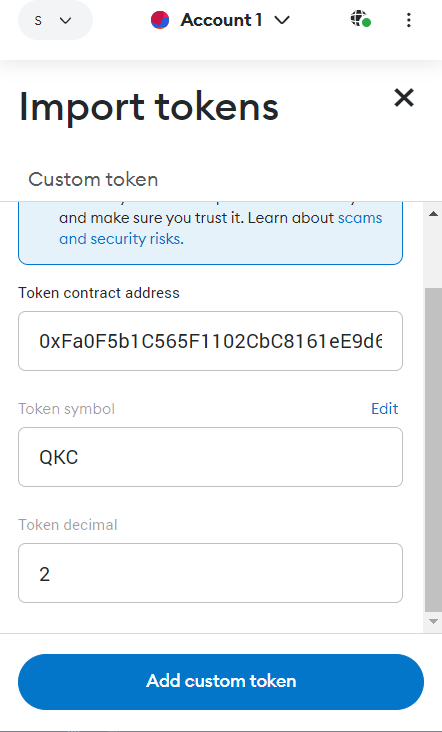
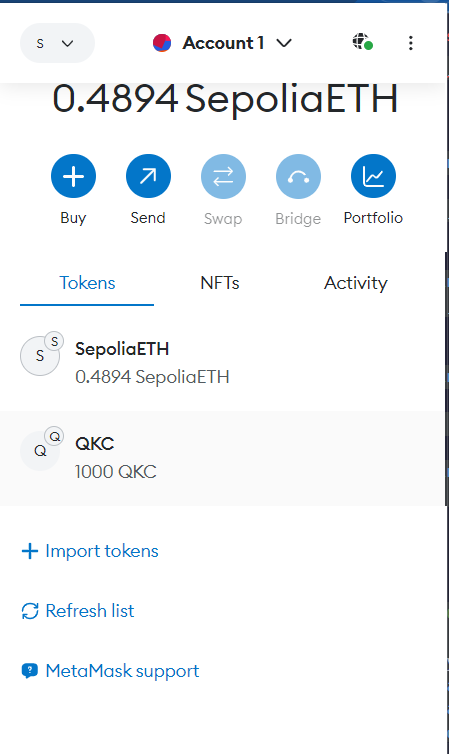
# Sepolia test network



# Deployed Contract -

### Address - 0xA44f4399208b634D5C2D3A1A61696c0b7df423f9

# 4 )Adding custom QKC tokens - 1000 QKC

**Conclusion:** We have succesfully created Metamask Wallet and Added dummy Ethers and Tokens.