

Objective

After completion of this assignment you will be able to :

- How to write ERC 20 token smart contract
- Understand inheritance
- Compile the Smart Contract using Remix Web IDE
- Deploy the Smart Contract to Rinkeyby Test Network
- Test the token functionalities

ERC20 Token - brief introduction

- Read the blog on [ERC20 Token](#) for brief introduction.

Write a Smart Contract for Decentralized Election

Note before you start with the solidity file :

1. Select the version of solidity compiler for this example as 0.4.19
2. Everything in the code written below with "//" is a comment and is provided for your understanding
3. Copy complete code given in multiple slides (ERC20-1 to ERC20-5) into the remix editor and then perform the compilation and deployment steps

ERC20 - 1

```
pragma solidity ^0.4.24;

contract Token {

    function totalSupply() constant returns (uint256 supply) {}

    function balanceOf(address _owner) constant returns (uint256 balance) {}

    function transfer(address _to, uint256 _value) returns (bool success) {}

    function transferFrom(address _from, address _to, uint256 _value) returns
(bool success) {}

    function approve(address _spender, uint256 _value) returns (bool success){}

    function allowance(address _owner, address _spender) constant returns
(uint256 remaining) {}

    event Transfer(address indexed _from, address indexed _to, uint256 _value);

    event Approval(address indexed _owner, address indexed _spender, uint256
_value); }
```

ERC20 - 2

```
contract XYZToken is Token {  
    string public name;  
    uint8 public decimals;  
    string public symbol;  
    uint256 public  
unitsOneEthCanBuy;  
    uint256 public totalEthInWei;  
    address public fundsWallet;
```

```
    constructor() public {  
        balances[msg.sender] =  
1000000000000000000000000;  
        totalSupply =  
1000000000000000000000000;  
        name = "XYZToken";  
        decimals = 18;  
        symbol = "XYZ";  
        unitsOneEthCanBuy = 10;  
        fundsWallet = msg.sender;  
    }
```

ERC20 - 3

```
function transfer(address _to,
uint256 _value) returns (bool
success) {
    if (balances[msg.sender] >= _value
    && balances[_to] + _value >
    balances[_to]) {
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        emit Transfer(msg.sender, _to,
        _value);
        return true;
    } else { return false; } }
```

```
function transferFrom(address _from,
address _to, uint256 _value) returns
(bool success) {
    if (balances[_from] >= _value &&
    allowed[_from][msg.sender] >= _value
    && balances[_to] + _value >
    balances[_to]) {
        balances[_to] += _value;
        balances[_from] -= _value;
        allowed[_from][msg.sender] -=
        _value;
```

ERC20- 4

```
emit Transfer(_from, _to, _value);
    return true;
} else { return false; }
}

function balanceOf(address _owner)
constant returns (uint256 balance) {
    return balances[_owner];
}

function approve(address _spender,
uint256 _value) returns (bool
success) {
```

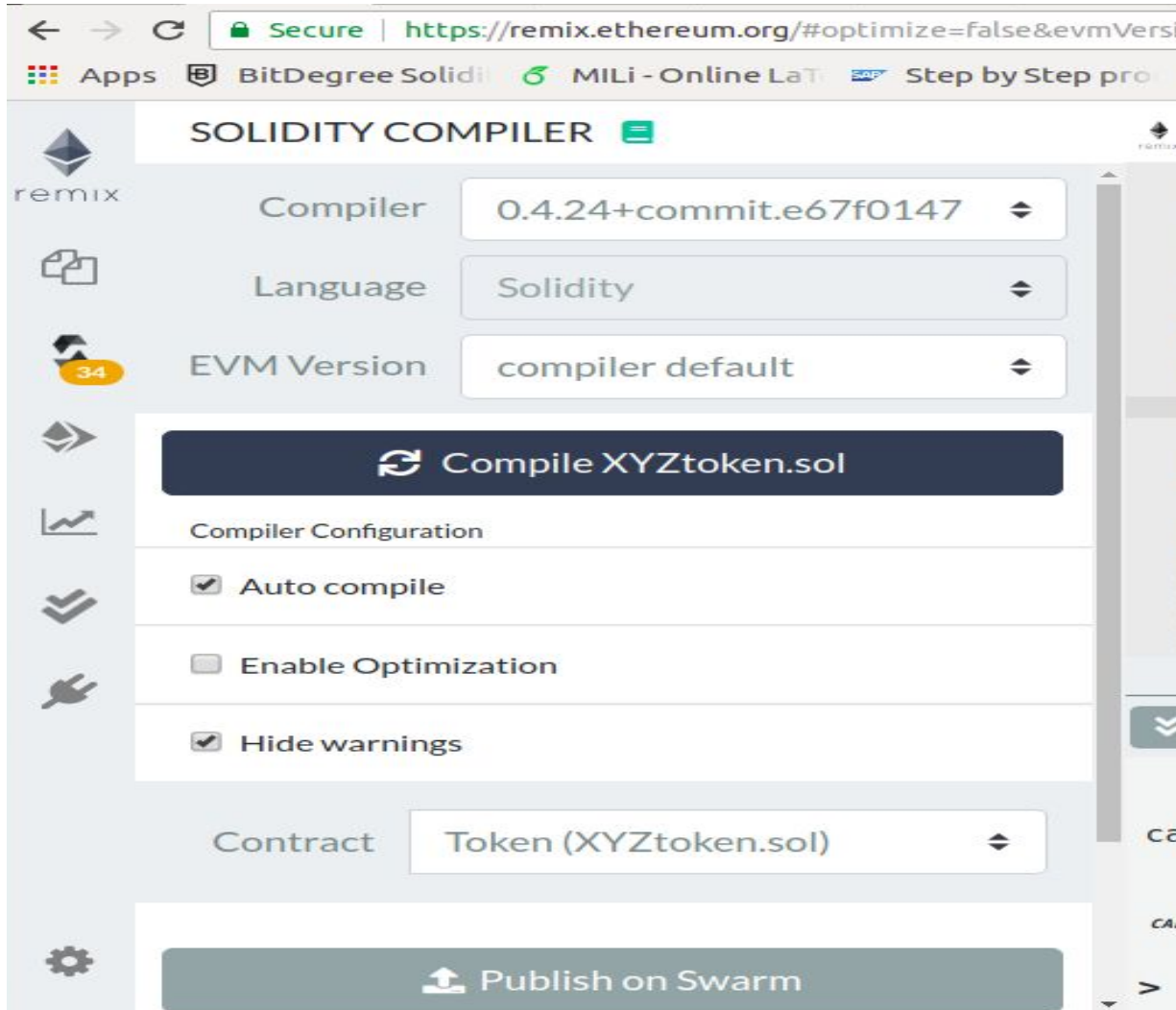
```
allowed[msg.sender][_spender] =
_value;
    emit Approval(msg.sender, _spender,
_value);
    return true;
}

function allowance(address _owner,
address _spender) constant returns
(uint256 remaining) {
    return
allowed[_owner][_spender];
}
```


ERC20 - 5

```
mapping (address => uint256) balances;  
mapping (address => mapping (address => uint256)) allowed;  
uint256 public totalSupply;  
function() payable{ // This function is a fallback function  
    totalEthInWei = totalEthInWei + msg.value;  
    uint256 amount = msg.value * unitsOneEthCanBuy;  
    require(balances[fundsWallet] >= amount);  
    balances[fundsWallet] = balances[fundsWallet] - amount;  
    balances[msg.sender] = balances[msg.sender] + amount;  
    emit Transfer(fundsWallet, msg.sender, amount);  
    fundsWallet.transfer(msg.value);  
}
```

Compile the .sol file using Remix



Follow the below steps to compile the code:

1. Go to Solidity Compiler
2. Select the correct compiler version
3. Select the Auto compile option
4. Check for the “Compilation Successful” message
5. If compiled successfully, Bytecode and ABI will be generated.
6. Ignore warnings

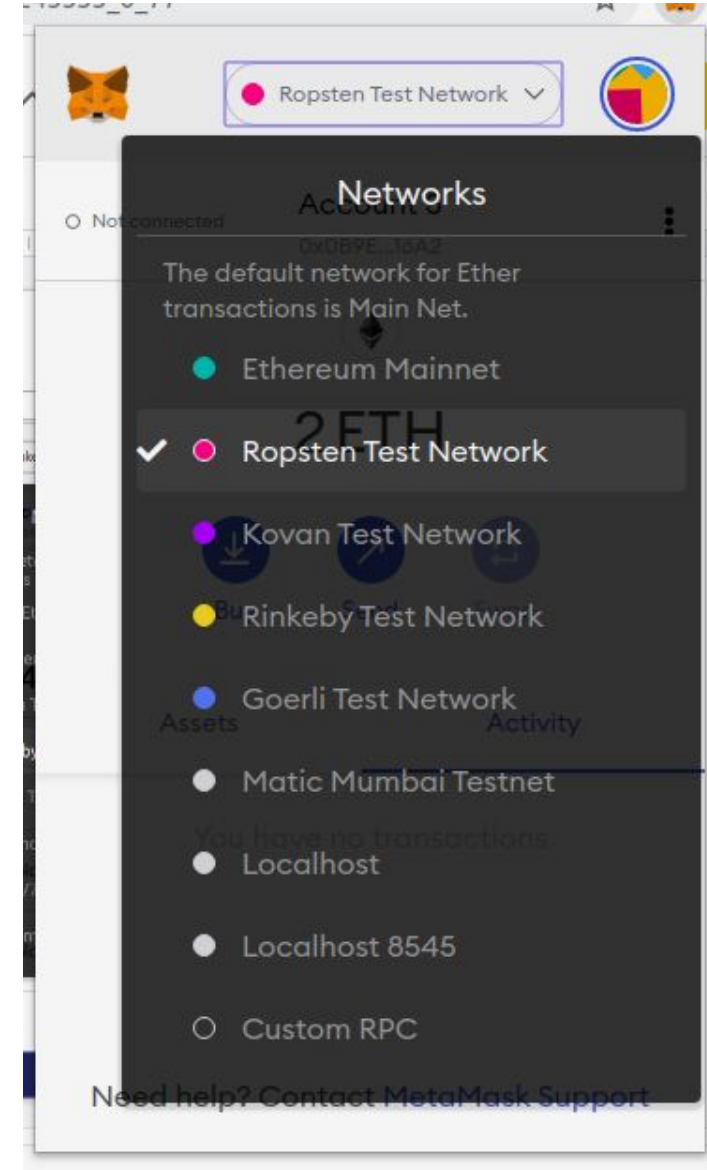
Install Metamask for Chrome

This step is important before we select an environment for deployment of the smart contract.

1. Go to [Google Chrome Webstore](#)
2. Click on Add to Chrome
3. Please follow the [blog to install metamask](#)

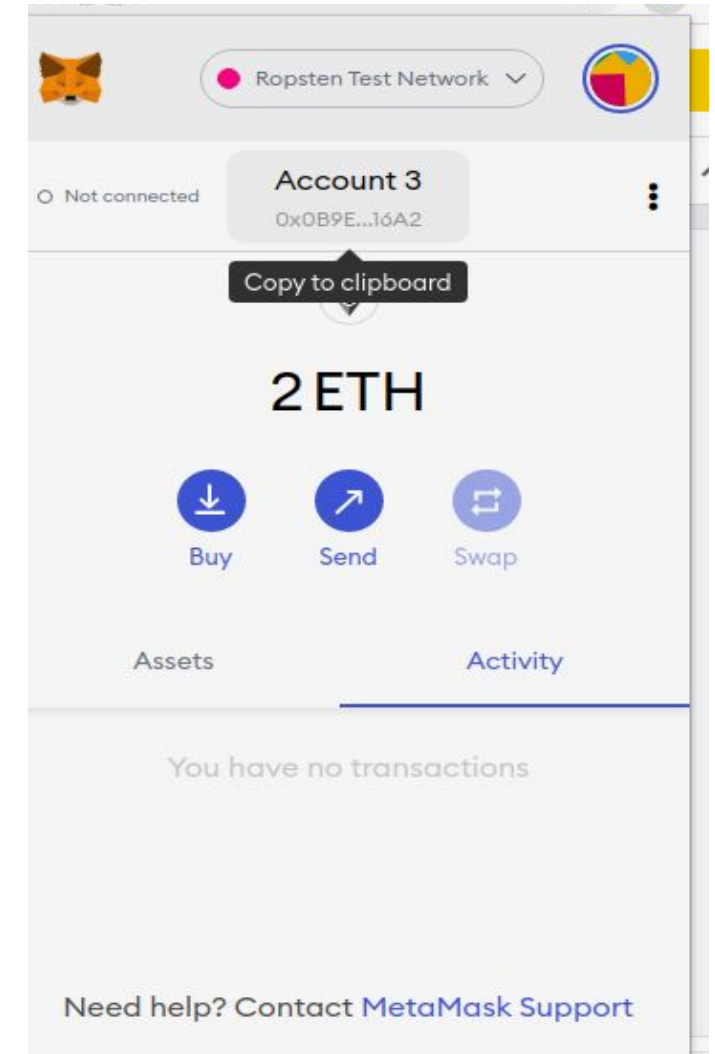
Metamask Setup

After installation
select a Ropsten Network



Add test ethers to ropsten account

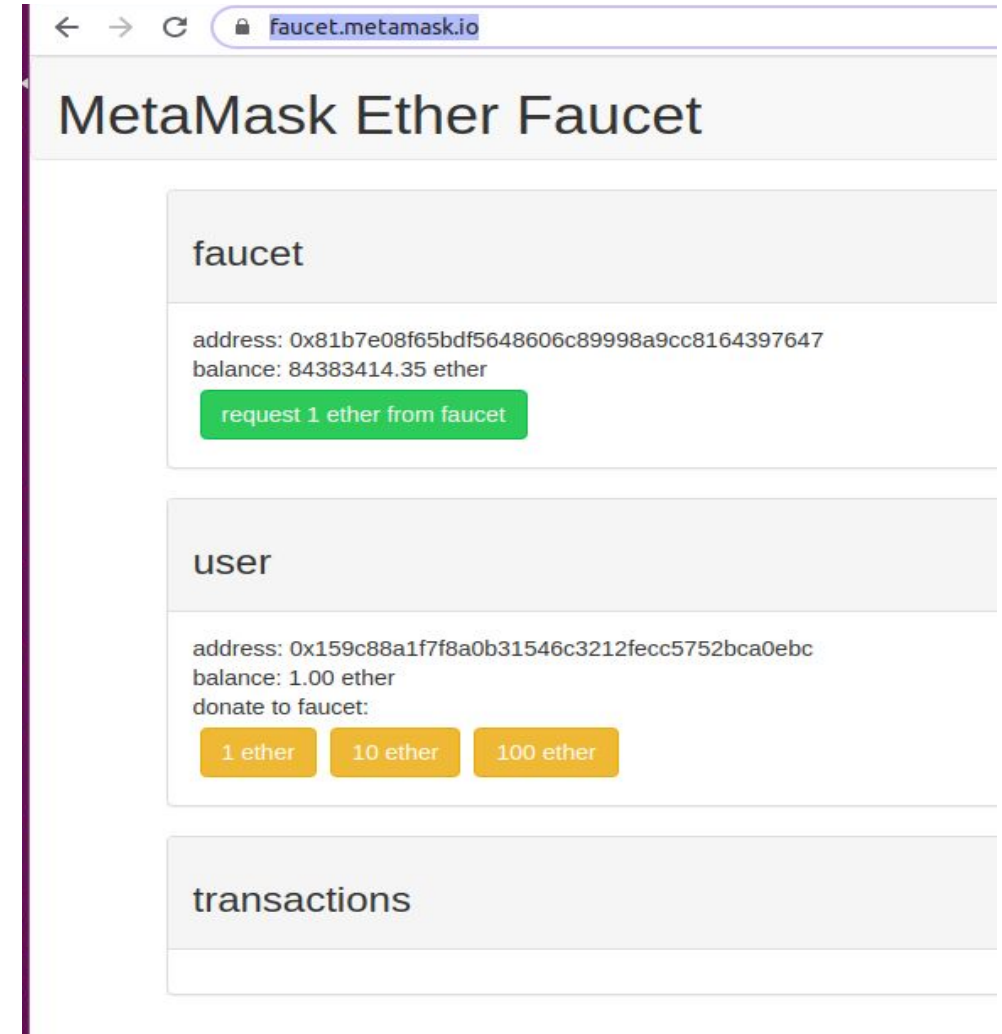
After selecting ropsten network select a account and copy the address



Add test ethers to Ropsten account

Go to :

<https://faucet.metamask.io/>



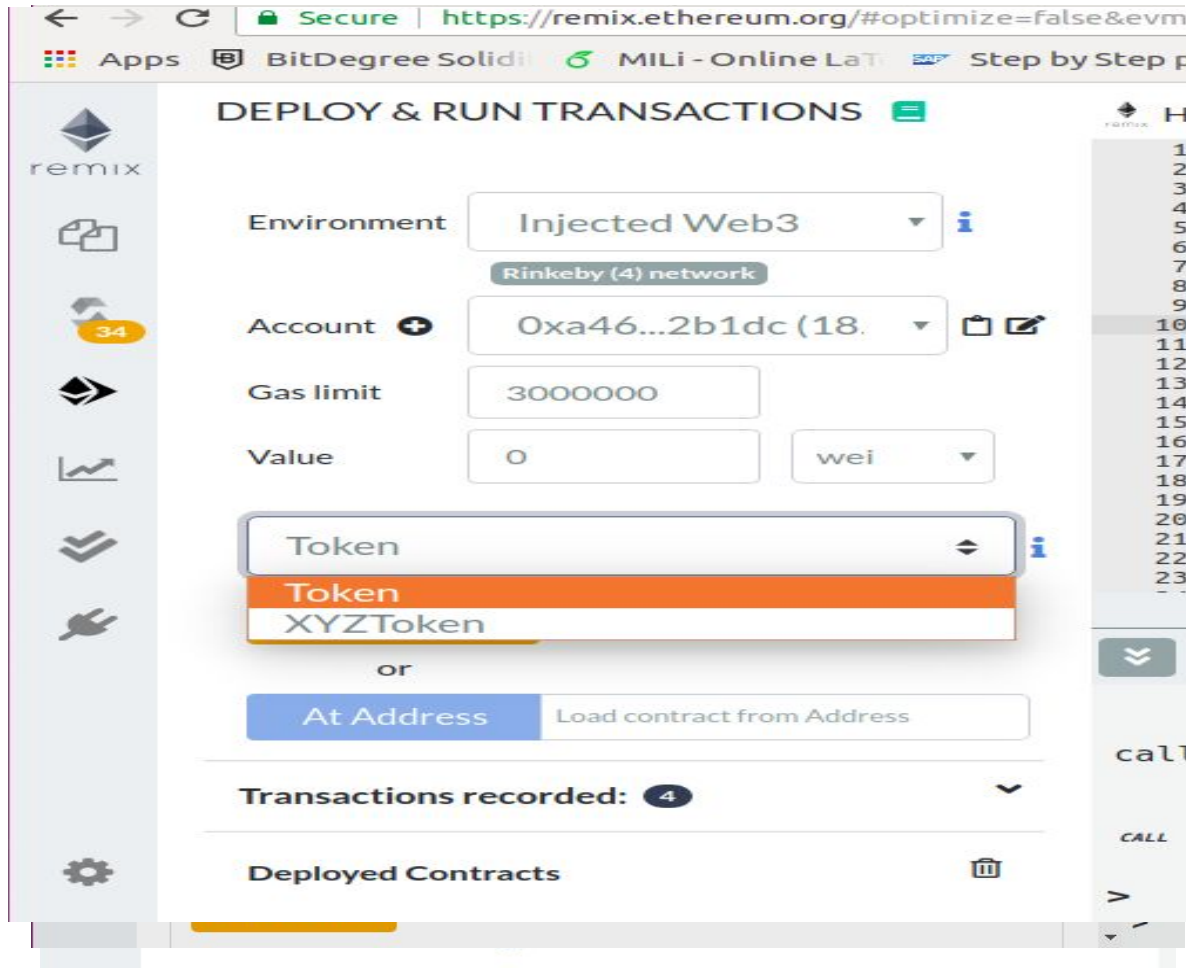
The screenshot shows the MetaMask Ether Faucet interface in a web browser. The address bar displays 'faucet.metamask.io'. The page title is 'MetaMask Ether Faucet'. It features three main sections: 'faucet', 'user', and 'transactions'. The 'faucet' section shows the faucet's address and balance, with a green button to request 1 ether. The 'user' section shows the user's address and balance, with three yellow buttons to donate 1, 10, or 100 ether to the faucet. The 'transactions' section is currently empty.

faucet
address: 0x81b7e08f65bdf5648606c89998a9cc8164397647
balance: 84383414.35 ether
<button>request 1 ether from faucet</button>

user
address: 0x159c88a1f7f8a0b31546c3212fecc5752bca0ebc
balance: 1.00 ether
donate to faucet:
<button>1 ether</button> <button>10 ether</button> <button>100 ether</button>

transactions

Deploy Smart Contract using Remix

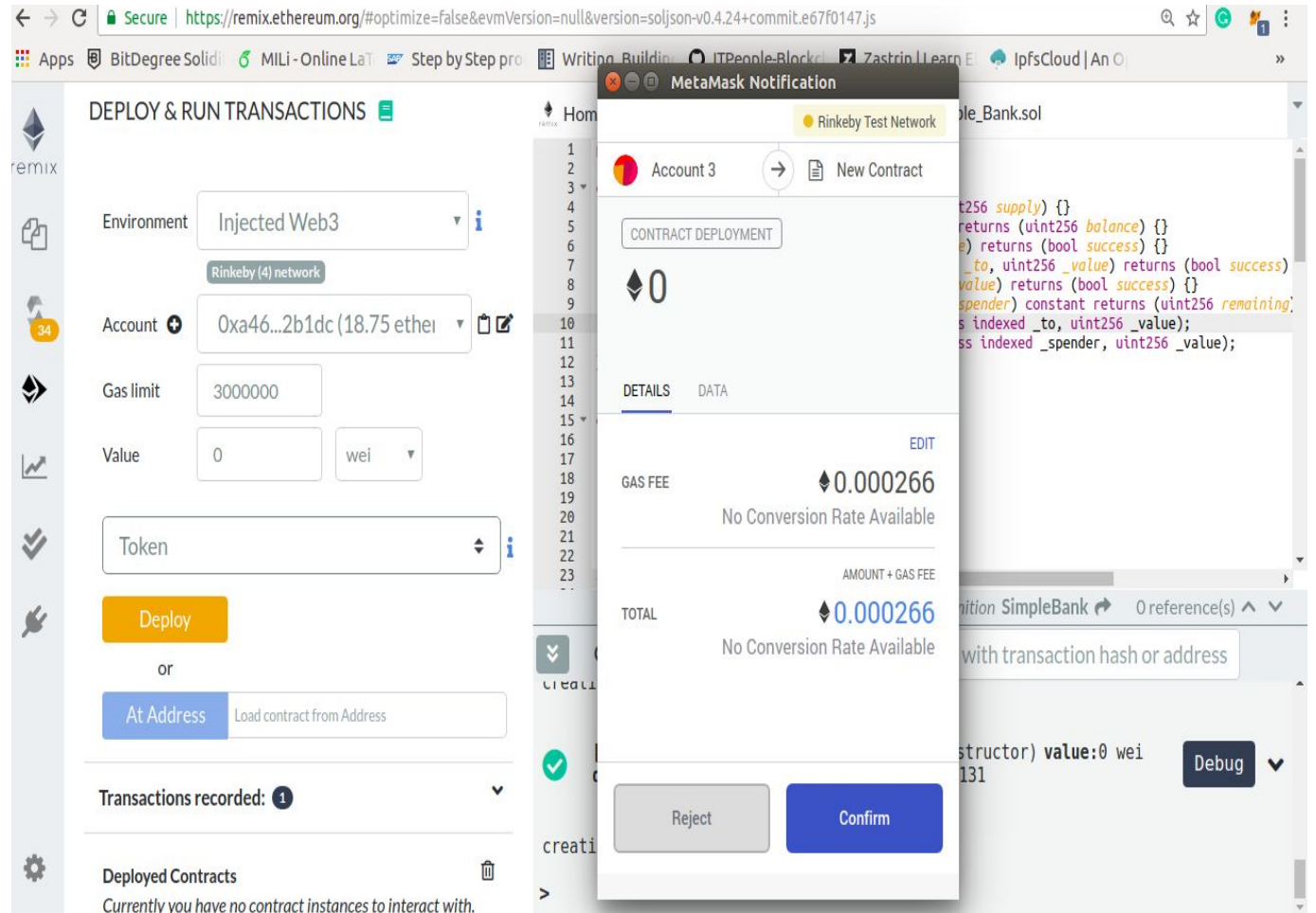


Follow below Steps to deploy and run the transaction:

1. Click on the “**Deploy and Run Transactions**” Tab
2. Select “**Injected Web3**” environment - this step will connect Remix to Metamask
3. Note there are 2 contracts to deploy
4. First deploy Token contract
5. And then deploy XYZToken contract

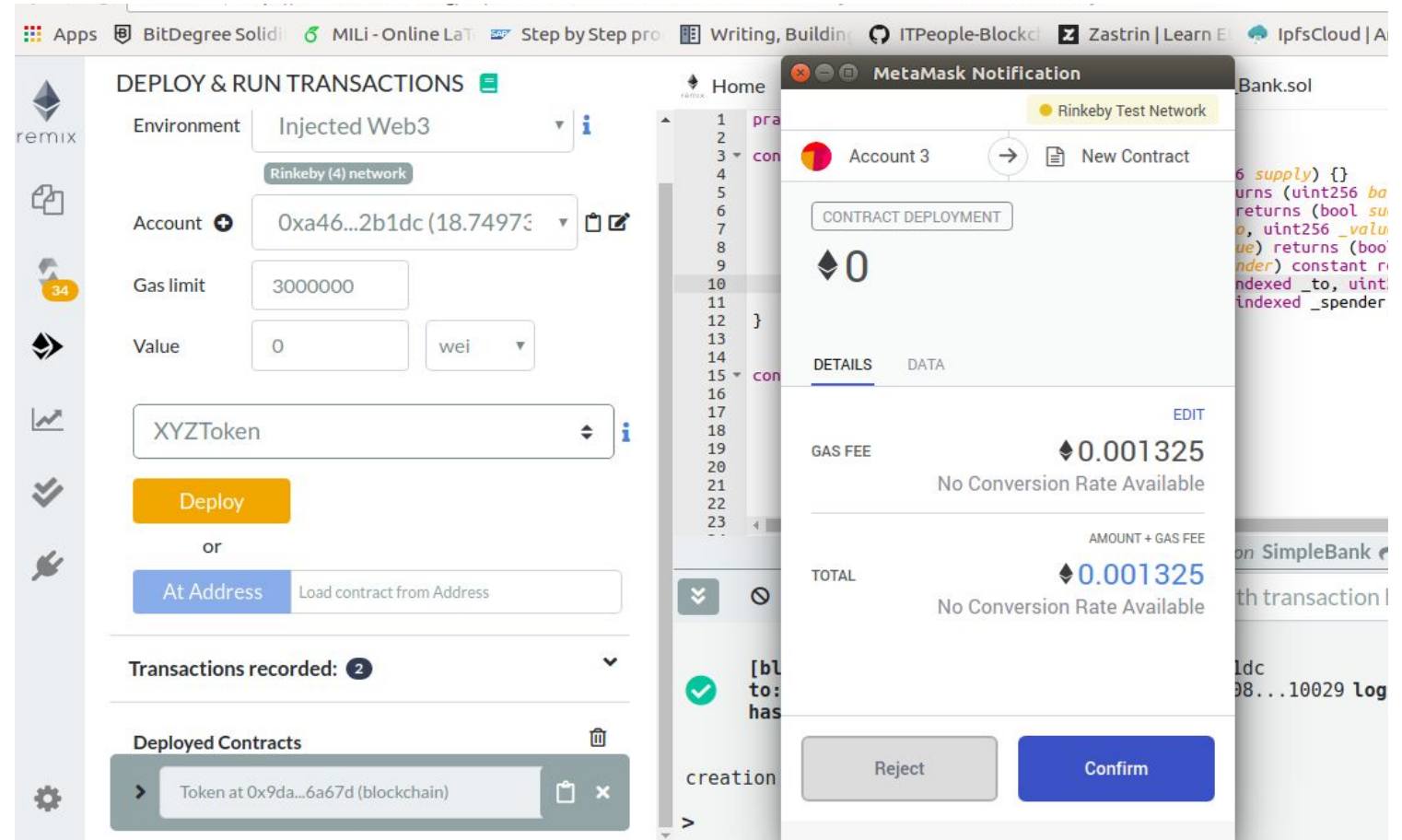
Deploy Token Contract

1. When you click on deploy Metamask will send you notification
2. You need to confirm the transaction
3. This will deduct the transaction cost from your account
4. Contract now deployed on rinkeyby network



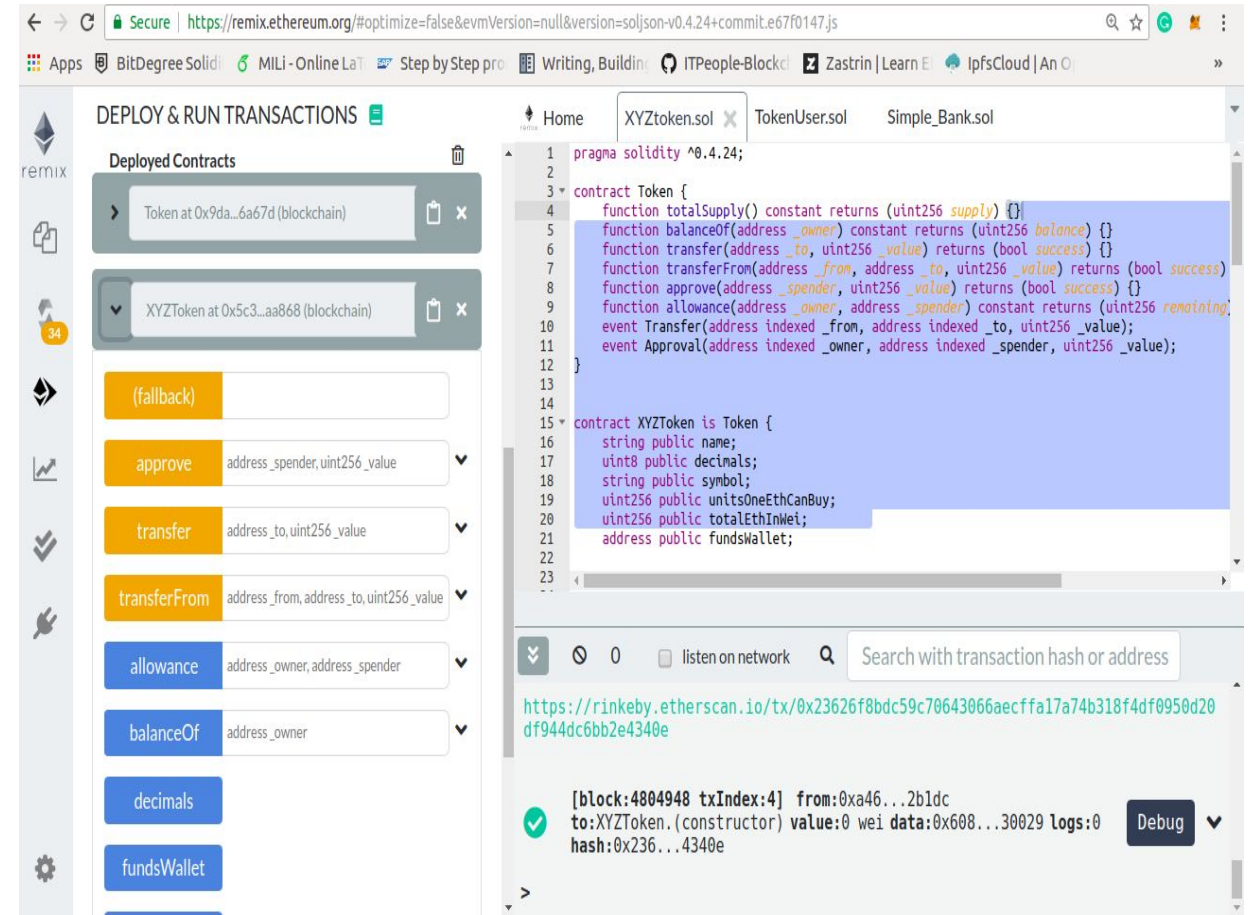
Deploy XYZToken Contract

Same as Token contract
deploy XYZToken
contract as well



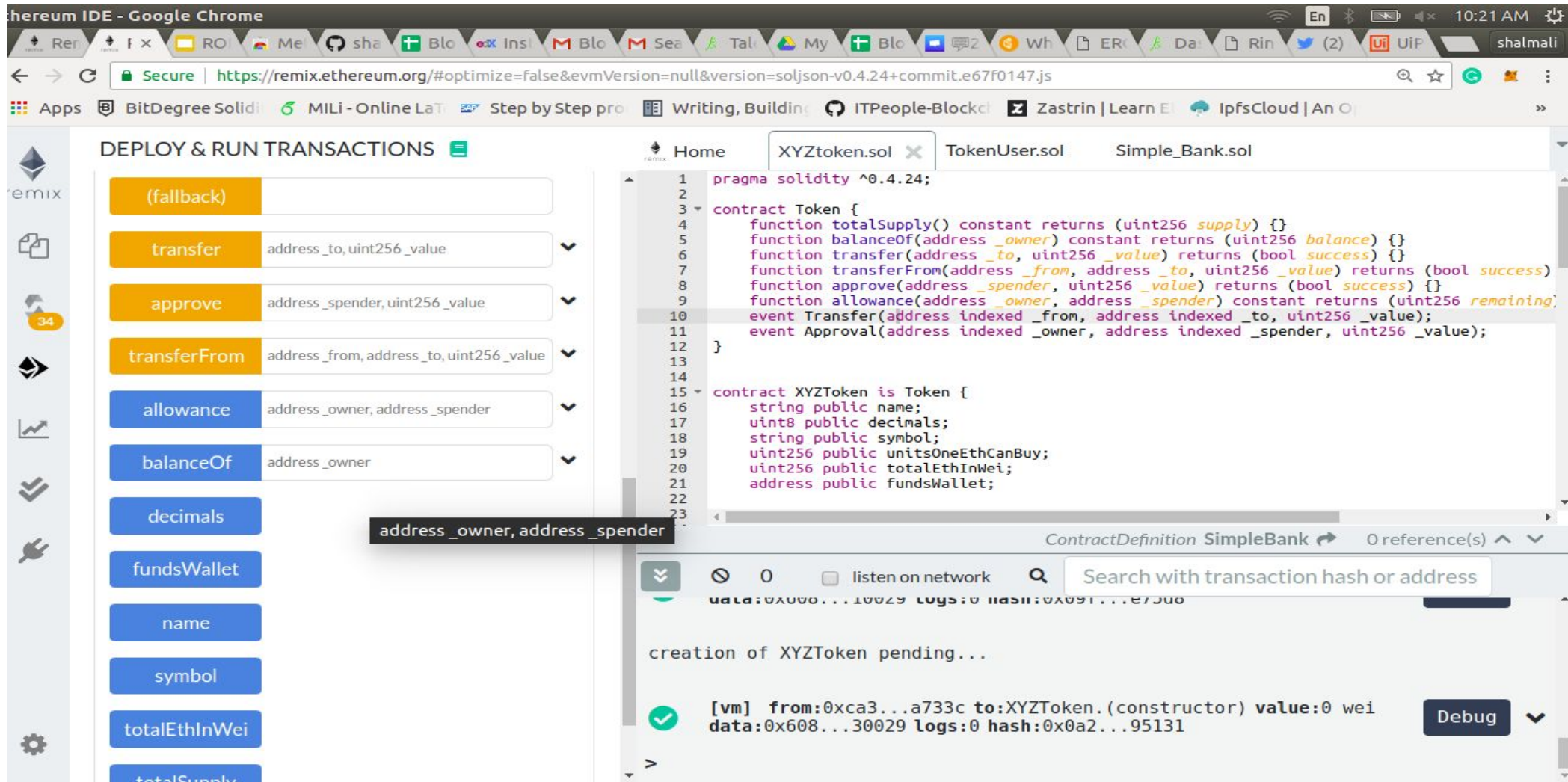
Test the functionality of the XYZToken Contract

Here Token contract act as a interface and actual functionalities are implemented in XYZToken.



The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the 'XYZToken' contract deployed at address 0x5c3...aa868 on the blockchain. Below this, a list of functions is available for interaction: (fallback), approve, transfer, transferFrom, allowance, balanceOf, decimals, and fundsWallet. The main editor on the right shows the Solidity code for the 'Token' and 'XYZToken' contracts. The 'Token' contract defines functions like totalSupply, balanceOf, transfer, transferFrom, approve, and allowance, along with Transfer and Approval events. The 'XYZToken' contract inherits from 'Token' and adds public variables for name, decimals, symbol, unitsOneEthCanBuy, totalEthInWei, and fundsWallet. At the bottom, the transaction log shows a successful deployment of the XYZToken constructor at block 4804948, transaction index 4, with a value of 0 wei and a gas limit of 30029.

Test the functionality of the XYZToken Contract



The screenshot shows the Remix IDE interface in Google Chrome. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel with a list of functions: (fallback), transfer, approve, transferFrom, allowance, balanceOf, decimals, fundsWallet, name, symbol, totalEthInWei, and totalSupply. The main editor displays the Solidity code for the XYZToken contract, which inherits from the Token contract. The code defines the XYZToken constructor with parameters: name, decimals, symbol, unitsOneEthCanBuy, totalEthInWei, and fundsWallet. The right sidebar shows the 'ContractDefinition SimpleBank' panel with a search bar and a list of transactions. The transaction list shows a successful transaction for the creation of XYZToken, with a value of 0 wei and a hash of 0x0a2...95131.

```

pragma solidity ^0.4.24;

contract Token {
    function totalSupply() constant returns (uint256 supply) {}
    function balanceOf(address _owner) constant returns (uint256 balance) {}
    function transfer(address _to, uint256 _value) returns (bool success) {}
    function transferFrom(address _from, address _to, uint256 _value) returns (bool success) {}
    function approve(address _spender, uint256 _value) returns (bool success) {}
    function allowance(address _owner, address _spender) constant returns (uint256 remaining);
    event Transfer(address indexed _from, address indexed _to, uint256 _value);
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);
}

contract XYZToken is Token {
    string public name;
    uint8 public decimals;
    string public symbol;
    uint256 public unitsOneEthCanBuy;
    uint256 public totalEthInWei;
    address public fundsWallet;
}

```

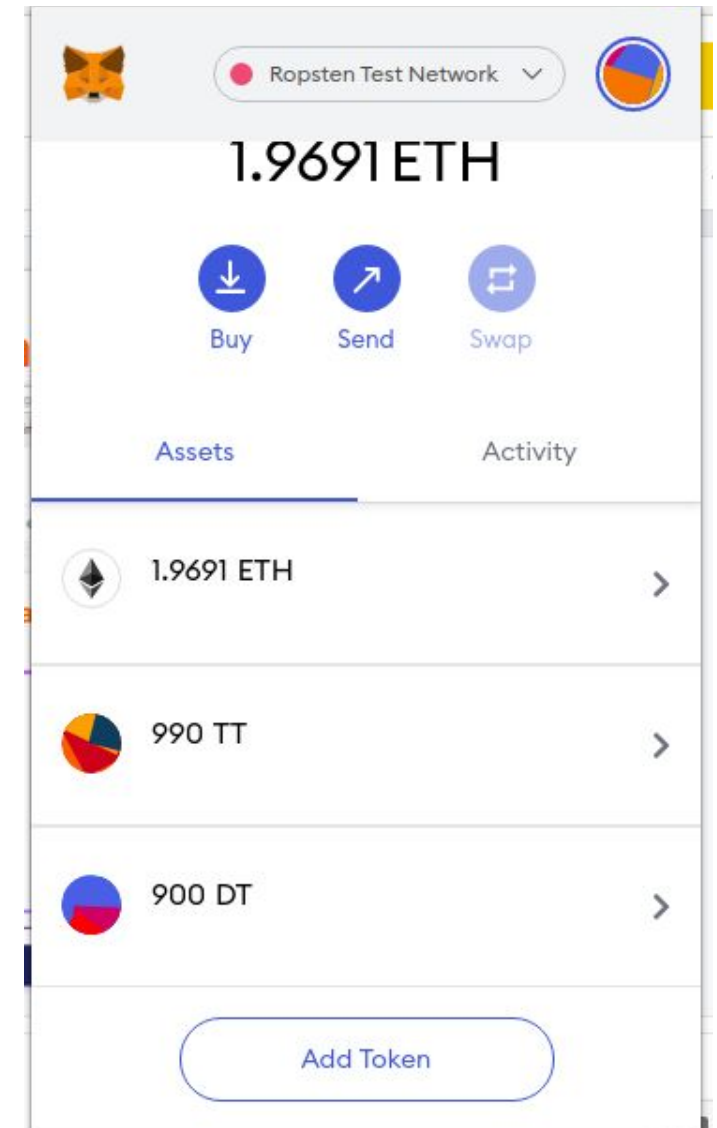
Transaction List:

- creation of XYZToken pending...
- [vm] from:0xca3...a733c to:XYZToken.(constructor) value:0 wei
data:0x608...30029 logs:0 hash:0x0a2...95131

How to add your token to Metamask (a browser wallet)

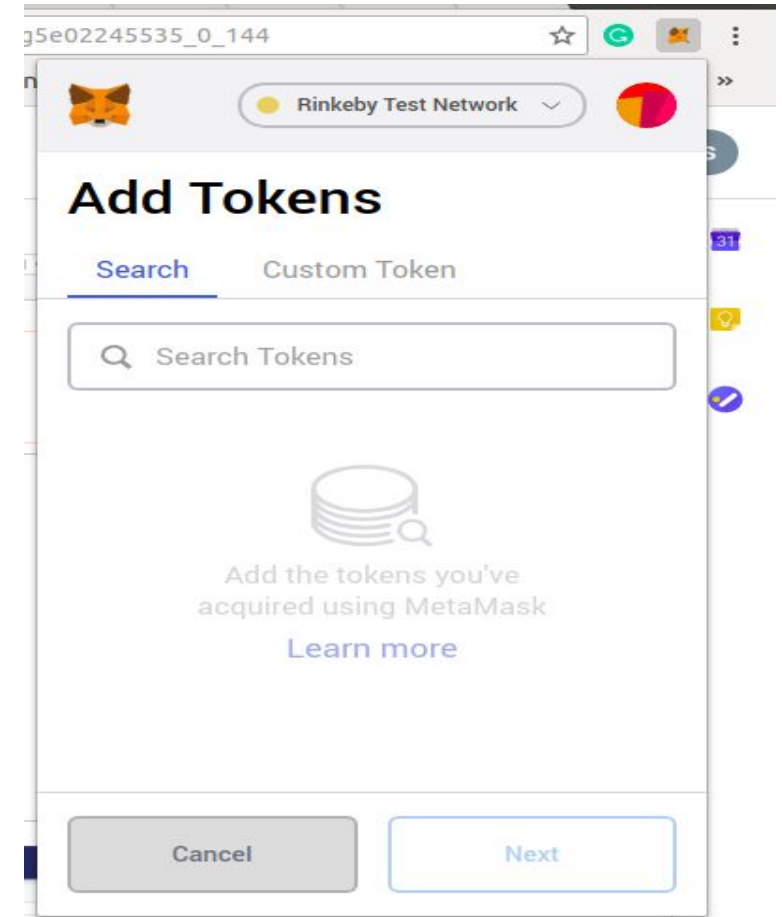
Follow below steps to add your token to metamask

1. Click on Add Token



How to add your token to Metamask (a browser wallet)

Click on Custom Token



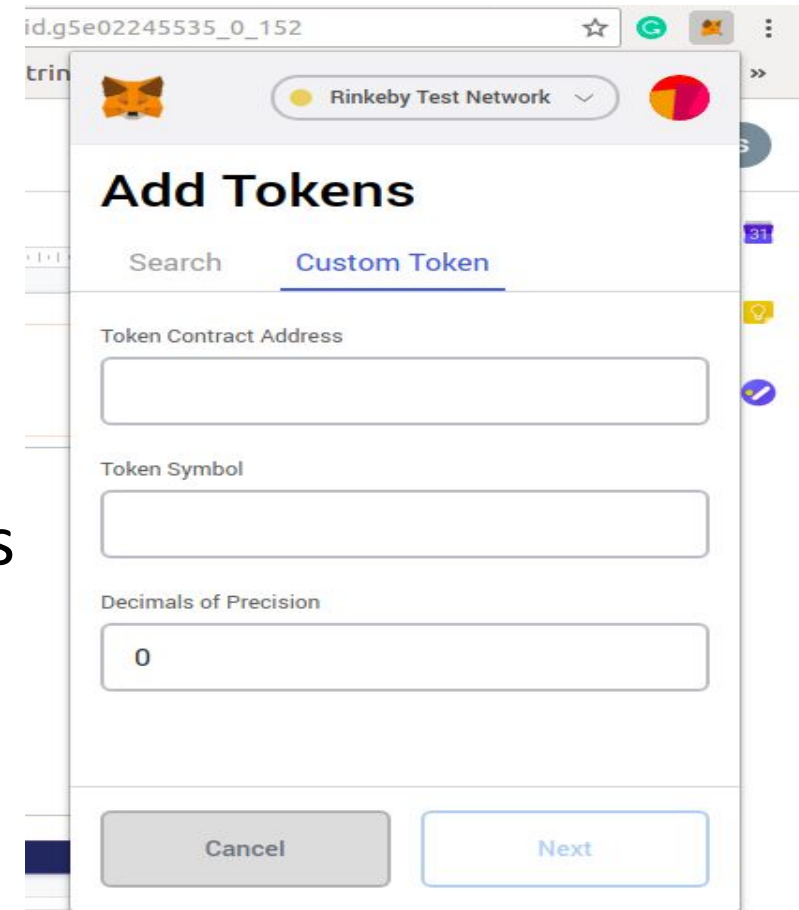
How to add your token to Metamask (a browser wallet)

Now to add token :

1. Copy XYZToken Contract address from remix



2. Paste the address in the Token Contract Address text box in the metamask



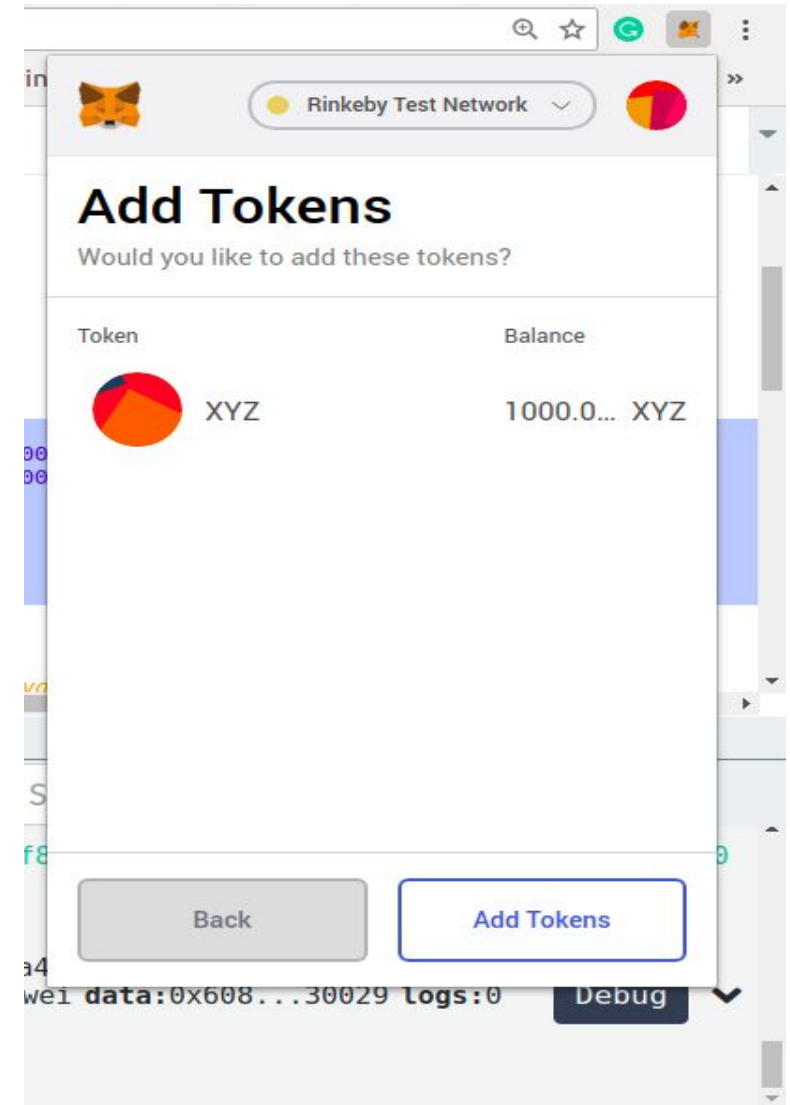
How to add your token to Metamask (a browser wallet)

As soon as you paste contract address in the metamask rest details will be automatically fetched.

The screenshot displays the Remix IDE interface in Google Chrome. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows a list of 'Deployed Contracts' including 'Token at 0x9da...6a67d (blockchain)' and 'XYZToken at 0x5c3...aa868 (blockchain)'. Below this, various transaction actions like '(fallback)', 'approve', 'transfer', 'transferFrom', 'allowance', 'balanceOf', 'decimals', and 'fundsWallet' are listed. The central editor shows the Solidity code for 'XYZToken.sol', which defines a 'Token' contract with attributes like 'name', 'decimals', 'symbol', and 'fundsWallet'. The 'constructor' sets initial balances and supply. The right sidebar shows the 'Add Tokens' dialog in the Metamask browser extension, set to the 'Rinkeby Test Network'. The dialog has fields for 'Token Contract Address' (0x5c35136fcab7732e0dee242cbf2cc02c), 'Token Symbol' (XYZ), and 'Decimals of Precision' (18). A 'Next' button is visible at the bottom right of the dialog. The bottom of the browser window shows a transaction confirmation from Rinkeby Etherscan.

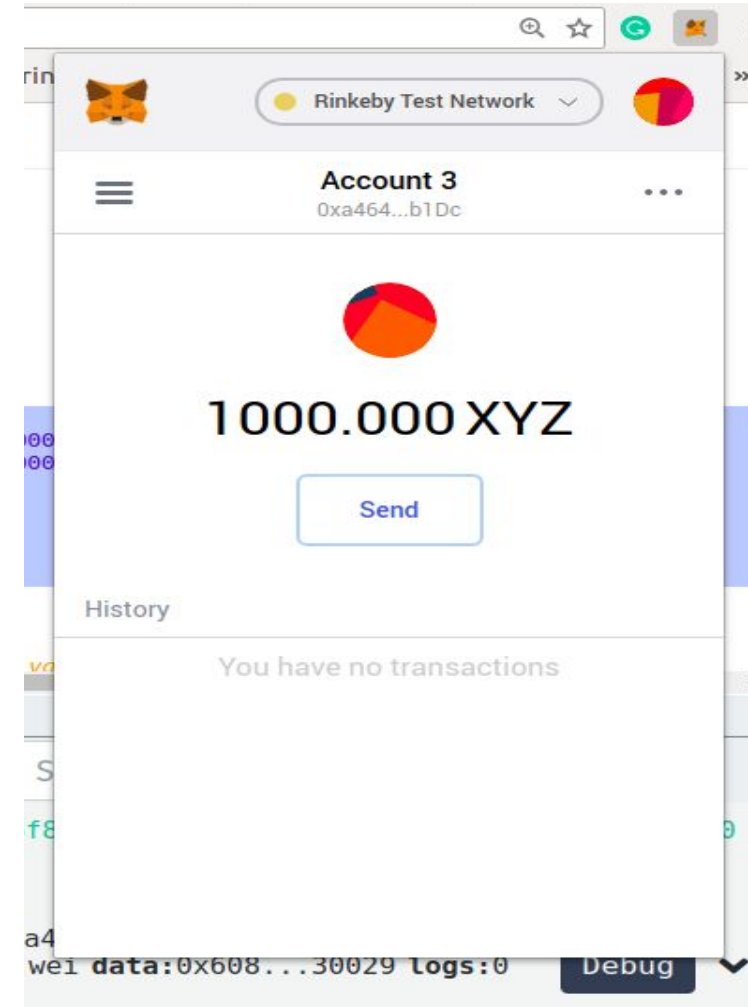
How to add your token to Metamask (a browser wallet)

1. Click on next
2. Click on Add Tokens



How to add your token to Metamask (a browser wallet)

1. XYZ Token added to account 3
2. Metamask can track the token in this account now.



Task for Participants

1. Create your own token with your name on Rikeyby Network

Hint - 2

- Deploy the contract same as XYZToken and test the functionalities

Final Steps

1. Create your own token
2. Add the token to Metamask
3. Copy complete code in the word document
4. Add the screenshot of metamask with your token in the document and convert the document into pdf format
5. Goto LMS -> Building Smart Contracts-> Week 9 ->Weekly Assignments -> Assignment Upload

Happy Learning!