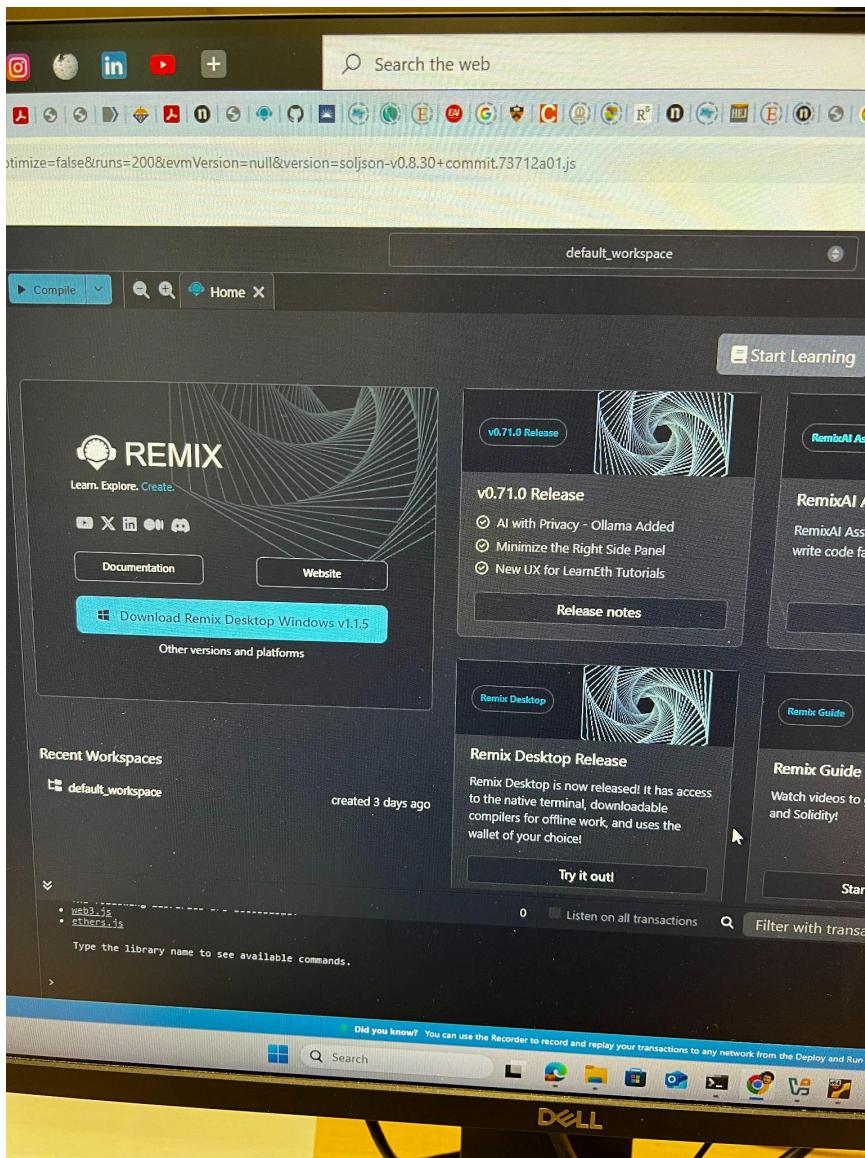


Blockchain and Smart Contracts

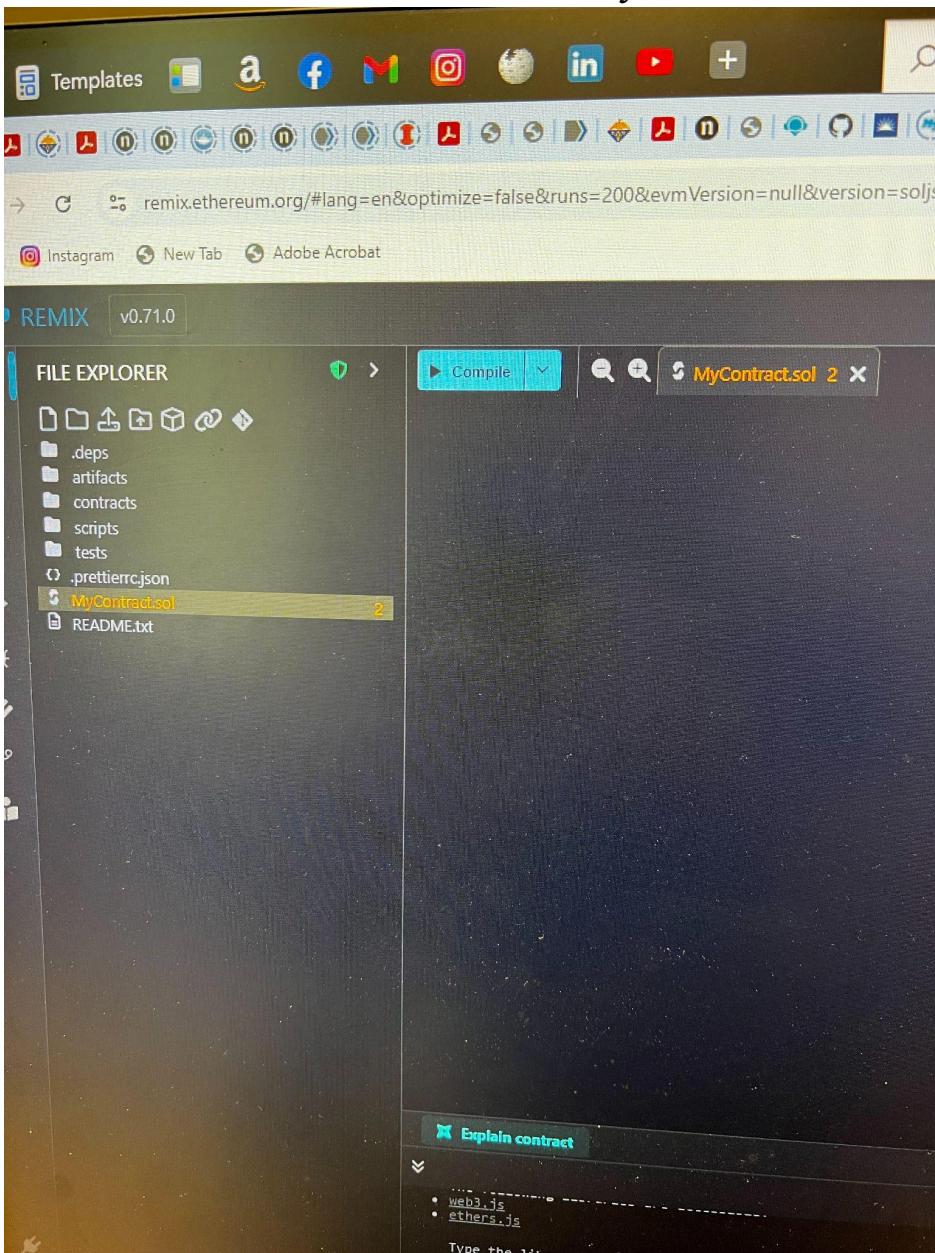
CSCI 5450

Assignment2: Connecting with a Public Testnet Blockchain
WASIM RAJA MONDAL

Open Remix IDE from your browser



Click on New File and name it as MyContract.sol



Write the following code in the file MyContract.sol I just created

The screenshot shows the Ethereum Remix IDE interface. The top bar has a search field and various browser icons. Below it, the URL is `remix.ethereum.org/#lang=en&optimize=false&runctions=200&evmVersion=null&version=soljson-v0.8.30+com`. The tabs include Instagram, New Tab, and Adobe Acrobat. The main window title is "REMX v0.71.0". The left sidebar is the "FILE EXPLORER" showing files like .deps, artifacts, contracts, scripts, tests, .prettierc.json, MyContract.sol (which is selected), and README.txt. The right pane contains the Solidity code for "MyContract.sol".

```
pragma solidity 0.4.20;

contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    // Store accounts that have voted
    mapping(address => bool) public voters;
    // Store Candidates
    // Fetch Candidate
    mapping(uint => Candidate) public candidates;
    // Store Candidates Count
    uint public candidatesCount;

    // voted event
    event votedEvent (
        uint indexed _candidateId
    );

    function Election () public {
        addCandidate("Candidate 1");
        addCandidate("Candidate 2");
    }

    function addCandidate (string _name) private {
        candidatesCount++;
    }
}
```

The bottom status bar shows "Scan Alert" and "Initialize as git repo", and a weather forecast for Rainy days ahead at 79°F.

Now, click on the Solidity compiler as shown in the screenshot below and select your compiler version. Since, the smart contract was written in 0.4.20, you should select the same version of compiler.

The screenshot shows the REMIX IDE interface. On the left, the Solidity Compiler section has a dropdown menu where '0.4.20+commit.3155dd80' is selected. The main area displays the following Solidity code:

```
1 pragma solidity 0.4.20;
2
3 contract Election {
4     // Model a Candidate
5     struct Candidate {
6         uint id;
7         string name;
8         uint voteCount;
9     }
10
11    // Store accounts that have voted
12    mapping(address => bool) public voted;
13    // Store Candidates
14    // Fetch Candidate
15    mapping(uint => Candidate) public candidates;
16    // Store Candidates Count
17    uint public candidatesCount;
18
19    // voted event
20    event votedEvent (
21        uint indexed _candidateId
22    );
23
24    function Election () public {
25        addCandidate("Candidate 1");
26        addCandidate("Candidate 2");
27    }
28
29    function addCandidate (string _name) public {
30        candidatesCount++;
    }
}

```

A tooltip 'Explain contract' is visible at the bottom of the code editor.

You will also see a Deploy button. In Remix IDE, you can deploy smart contracts directly to the blockchain without using the Truffle framework.

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

- Top Bar:** Shows the URL `remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=s`, and browser tabs for Instagram, New Tab, and Adobe Acrobat.
- Left Sidebar (DEPLOY & RUN TRANSACTIONS):**
 - ENVIRONMENT:** Set to "Remix VM (Prague)".
 - ACCOUNT:** Address `0x5B3...eddC4` with 100 ETH.
 - GAS LIMIT:** Set to "Custom" with value `3000000`.
 - VALUE:** Set to 0 Wei.
 - CONTRACT:** Selected contract is `Election - MyContract.sol`.
 - Buttons:** "Deploy" (orange), "Publish to IPFS", "At Address", and "Load contract from Address".
 - Notifications:** "Scam Alert" and "Initialize as git repo".
 - Weather:** "Rainy days ahead" at 79°F.
- Center Panel:** Displays the Solidity code for the `Election` contract.

```
function Election () public {
    addCandidate("Candidate 1");
    addCandidate("Candidate 2");
}

function addCandidate (string _name
candidatesCount++;
candidates[candidatesCount] = C
}

function vote (uint _candidateId) pu
// require that they haven't vot
require(!voters[msg.sender]);

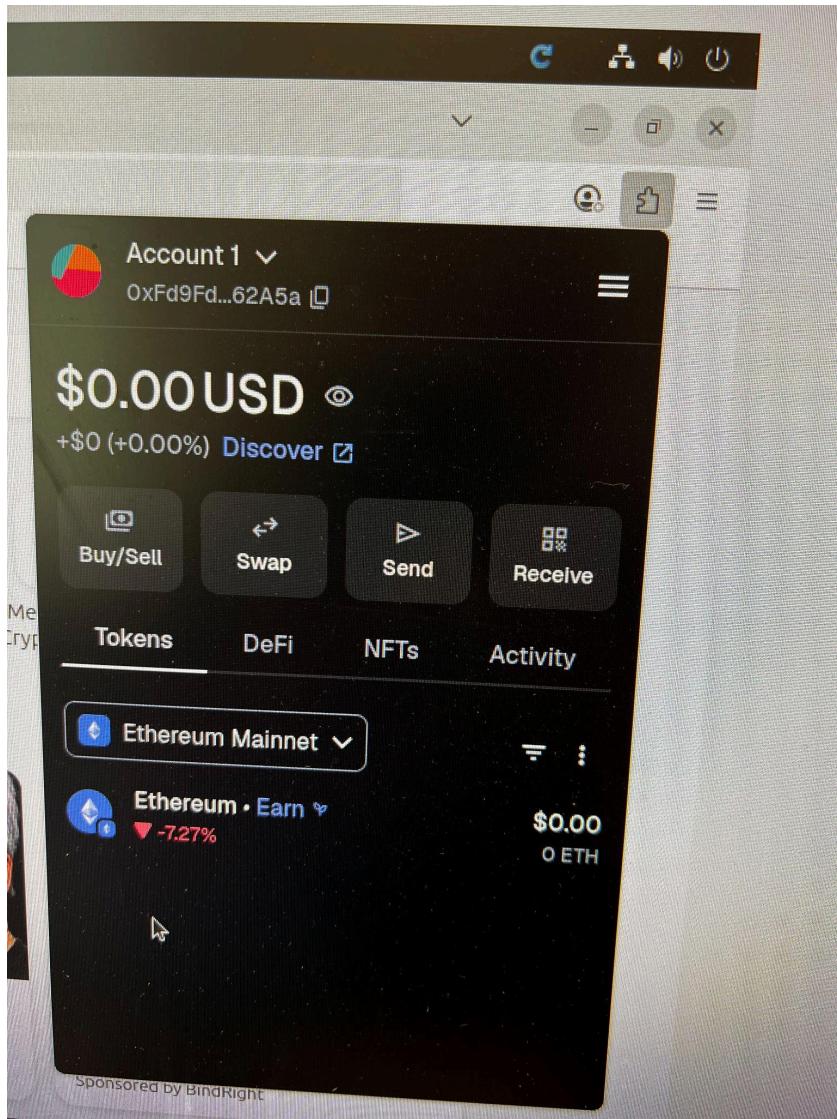
// require a valid candidate
require(_candidateId > 0 && _cand
}

// record that voter has voted
voters[msg.sender] = true;

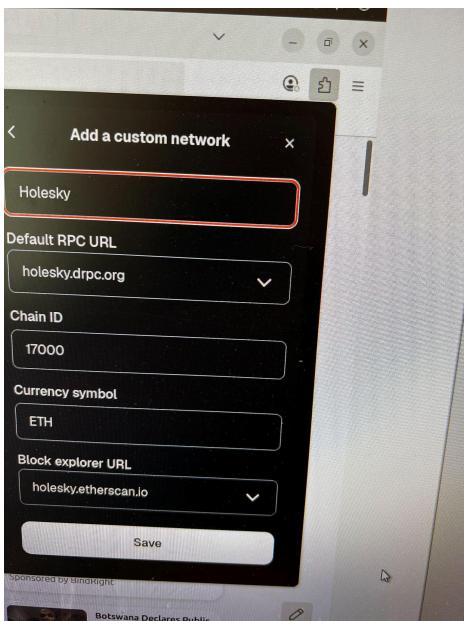
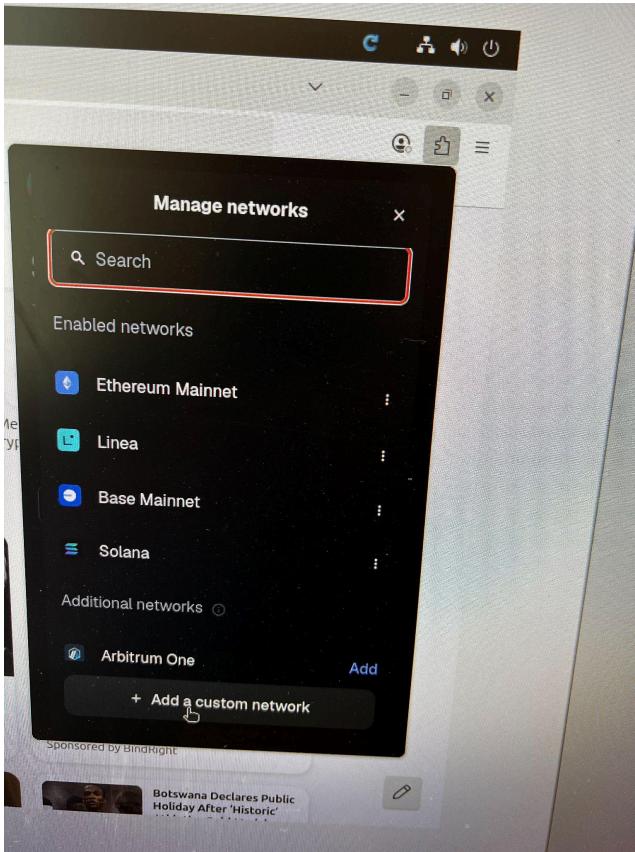
// update candidate vote Count
candidates[_candidateId].voteCount

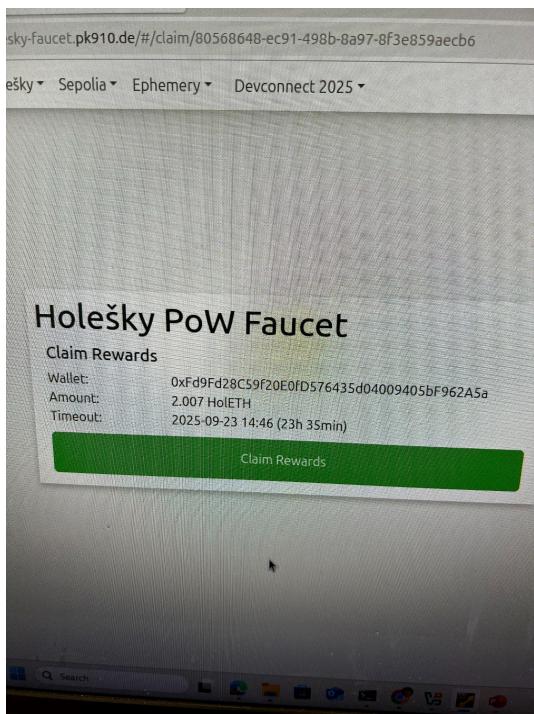
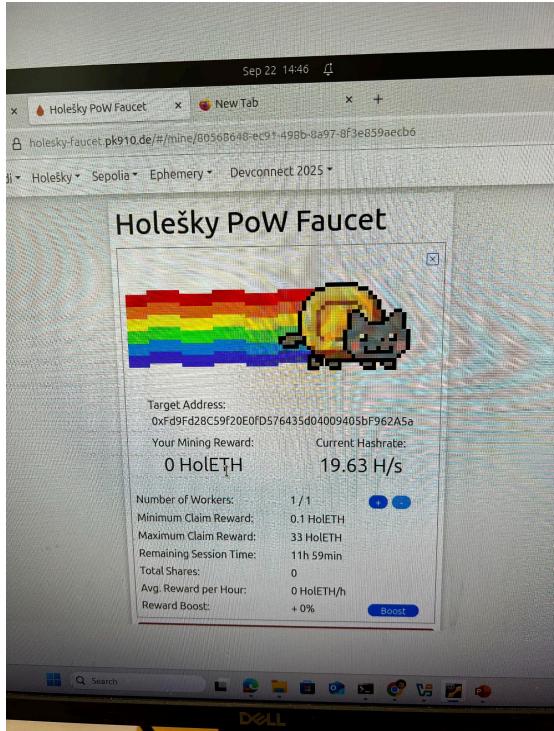
// trigger voted event
votedEvent(_candidateId);
}
```
- Bottom Panel:** "Explain contract" section with links to `web3.js` and `ethers.js`. It also includes a command-line interface placeholder: "Type the library name to see available commands."

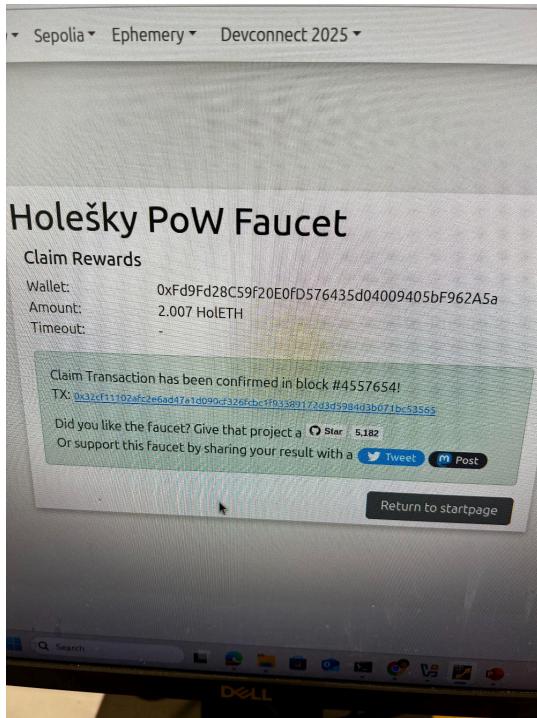
We should not deploy now because we do not have ETH in our account as shown below:



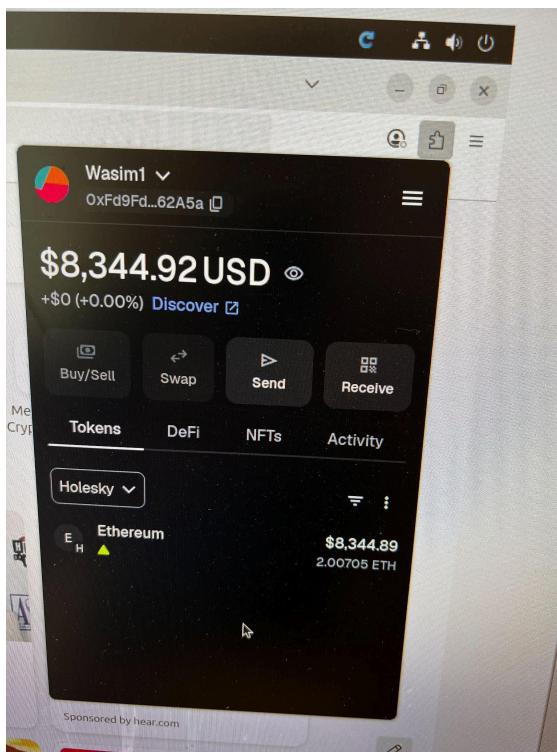
We need to add the Holesky network to our MetaMask account







The updated balance on the MetaMask account as shown below:



The contract has been deployed successfully, as confirmed by a green check mark in the console as shown in the screenshot below:

Sep 23 12:25

Holešky PoW Faucet election/contracts/ New Tab

.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.4.20

default workspace

Compile

```
21     );
22 }
23
24 function Election () public {   undefined gas
25     addCandidate("Candidate 1");
26     addCandidate("Candidate 2");
27 }
28
29 function addCandidate (string _name) private {
30     candidatesCount++;
31     candidates[candidatesCount] = Candidate(candidate
32 }
33
34 function vote (uint _candidateId) public {
35     // require that they haven't voted before
36     require(!voters[msg.sender]);
37
38     // require a valid candidate
39     require(_candidateId > 0 && _candidateId <= cand
40
41     // record that voter has voted
42     voters[msg.sender] = true;
43 }
```

Explain contract AI copilot

0 Listen on all transactions Filter with transaction hash or ad... [block:1 txIndex:-1 from: 0x5B3...eddC4 to: Election.(constructor) value: 0 wei data: 0x606...a0029 logs: 0 hash: 0x0f8...8ded7 Debug]

Select Context

Select context and anything!

MistralAI Create

Did you know? To learn new contract patterns and prototype, you can activate and try the cookbook plugin!

Search

This screenshot shows the Remix Ethereum IDE interface. The code editor displays a Solidity smart contract named 'Election'. The transaction history at the bottom shows a successful deployment of the contract to the Ethereum network. A green checkmark next to the transaction hash indicates the deployment was successful.

The details of the smart contract with its address, along with the available items, as shown below in the screenshot.

The screenshot shows a blockchain application interface with the following details:

- Deploy & Run** button (green)
- TRANSACTIONS** section:
 - Transactions recorded: 1
 - Deployed Contracts: 1
- Deployed Contracts** section:
 - ELECTION AT 0xD91...39138 (Contract Address)
 - Balance: 0 ETH
 - Available functions (methods):
 - vote (orange bar)
 - candidates (blue bar)
 - candidatesCount (blue bar)
 - voters (blue bar)
 - Low level interactions: CALLDATA
 - Transact button (orange)
- Right side of the interface displays a vertical list of numbers from 2 to 43.

As we can see above screenshot, in this deployed smart contract, we have four options. The one shown in orange is the vote function, which changes the blockchain state and therefore requires “gas” (gas is something we will cover later in class) to execute. The three shown in blue (candidates, candidatesCount, and voters) are not user-defined functions but public state variables. Solidity automatically generates getter functions for public variables, which is why Remix IDE allows us to call them. These getters are read-only and free to use, while the orange function writes to the blockchain and consumes gas

Insert a value of “1” to record a vote for Candidate 1 on the blockchain using this smart contract.

Sep 29

New Tab Remix - Ethereum IDE Holešky PoW Faucet

REMX v0.71.0 default_w

DEPLOY & RUN TRANSACTIONS

Transactions recorded 1 i

Deployed Contracts 1

ELECTION AT 0xD91...39138 (X)

Balance: 0 ETH

VOTE

_candidated: "1"

Calldata Parameters **transact**

candidates uint256

candidatesCount

voters address

Low level interactions

CALDATA Transact

Scam Alert Initialize as git repo

Compile Home

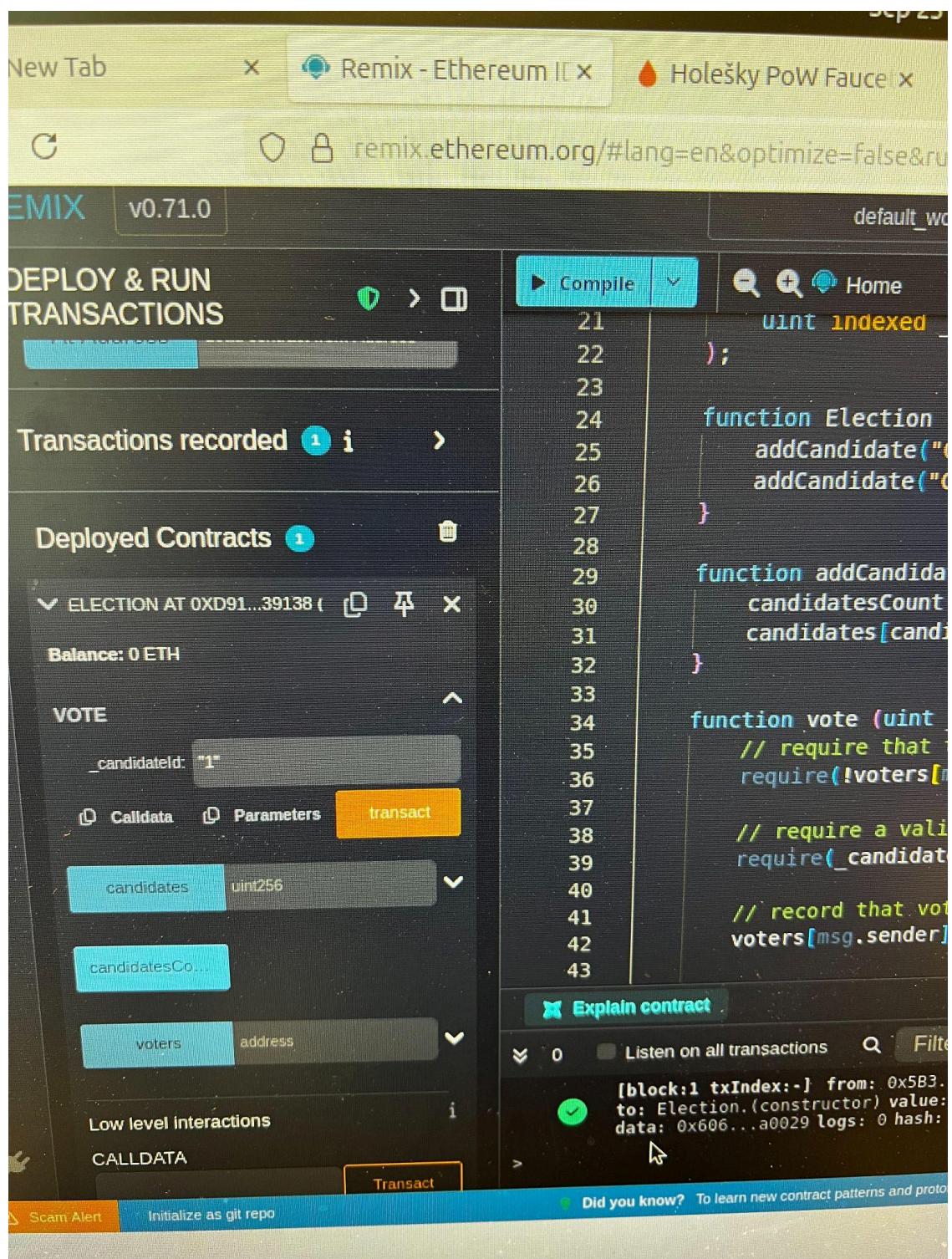
```
21     uint indexed;
22 );
23
24 function Election
25     addCandidate("C
26     addCandidate("C
27 }
28
29 function addCandida
30     candidatesCount
31     candidates[candi
32 }
33
34 function vote (uint
35     // require that
36     require(!voters[r
37
38     // require a valid
39     require(_candidat
40
41     // record that vot
42     voters[msg.sender]
43 }
```

Explain contract

0 Listen on all transactions Filter

[block:1 txIndex:-] from: 0x5B3... to: Election.(constructor) value: 0x606...a0029 logs: 0 hash:

Did you know? To learn new contract patterns and proto



Showing the confirmation of transact

```
23
24     function Election () public {
25         addCandidate("Candidate 1");
26         addCandidate("Candidate 2");
27     }
28
29     function addCandidate (string _name) private {
30         candidatesCount++;
31         candidates[candidatesCount] = Candidate(_name);
32     }
33
34     function vote (uint _candidateId) public {
35         // require that they haven't voted before
36         require(!voters[msg.sender]);
37
38         // require a valid candidate
39         require(_candidateId > 0 && _candidateId < candidatesCount);
40
41         // record that voter has voted
42         voters[msg.sender] = true;
43     }
```

Explain contract

AI copilot

0 Listen on all transactions Filter with transaction hash or address
to: Election.vote(uint256) 0xd91...39138
value: 0 wei data: 0x012...00001 logs: 1
hash: 0x4eb...e1614

Debug

Search

Now that you have cast a vote for Candidate 1, use the candidates option (listed under the vote function), which is a public variable with an auto-generated getter function. Enter the value 1 to see the information for Candidate 1, including the number of votes received.

The screenshot shows the Ethereum Remix IDE interface. The top bar displays tabs for "New Tab", "Remix - Ethereum", and "Holešky PoW Faucet". The URL in the address bar is "remix.ethereum.org/#lang=en&optimize=false&run=1". The left sidebar has icons for "Deploy & Run Transactions", "Contracts", "Blocks", "Logs", "Storage", "Calldata", and "Low level interactions". The main area is titled "REMX v0.71.0".

DEPLOY & RUN TRANSACTIONS

Balance: 0 ETH

VOTE

_candidateId: "1"

Calldata Parameters **transact**

CANDIDATES

"1"

Calldata Parameters **call**

0: uint256: id 1
1: string: name Candidate 1
2: uint256: voteCount 1

candidatesCount
voters address

Low level interactions
CALldata
Transact

Contract Source Code:

```
21     uint indexed;
22     );
23
24     function Election()
25     addCandidate("C");
26     addCandidate("C");
27     }
28
29     function addCandidate(string _name)
30     candidatesCount++;
31     candidates[candidatesCount] = _name;
32     }
33
34     function vote(uint _id)
35     // require that the voter exists
36     require(!voters[msg.sender]);
37
38     // require a valid candidate
39     require(_candidateId < candidatesCount);
40
41     // record that voter voted
42     voters[msg.sender]++;
43 }
```

Logs:

- 0 Listen on all transactions Filter
- tx: ELECTION.Candidates(1) data: 0x347...0001

Did you know? To learn new contract patterns and prototype

Do the same for Candidate 2 to show whether Candidate 2 has received any votes even if you have not cast a vote for Candidate 2 yet.

The screenshot shows the Remix Ethereum IDE interface. The top bar displays tabs for "New Tab", "Remix - Ethereum IDE", "Holešky PoW Faucet", and "remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=solida...". The title bar indicates the version "REMX v0.71.0" and the workspace name "default_workspace".

The left sidebar contains sections for "DEPLOY & RUN TRANSACTIONS" (Balance: 0 ETH) and "VOTE" (Candidate ID: "1"). The "VOTE" section includes "Calldata" and "Parameters" buttons, with "transact" highlighted. Below it is a "CANDIDATES" section with "Calldata" and "Parameters" buttons, and a "call" button.

The main area displays the Solidity code for the "Election" contract:

```
21  uint indexed _candidateId;
22
23
24  function Election () public {
25      addCandidate("Candidate 1");
26      addCandidate("Candidate 2");
27  }
28
29  function addCandidate (string _name) {
30      candidatesCount++;
31      candidates[candidatesCount] = _name;
32  }
33
34  function vote (uint _candidateId) {
35      // require that they haven't voted already
36      require(!voters[msg.sender]);
37
38      // require a valid candidate
39      require(_candidateId > 0 && _candidateId < candidatesCount);
3
41
42      // record that voter has voted
43      voters[msg.sender] = true;
    }
```

Below the code, there is an "Explain contract" button. At the bottom, a transaction log shows: "0 Listen on all transactions" and "Filter with transaction". The log details a transaction from "ELECTION" with "data: 0x347...00002". A note at the bottom right says: "Did you know? To learn new contract patterns and prototype, you can activate and deactivate them by clicking the gear icon in the top right corner of the editor."

If you click on the third option, candidatesCount, it will simply show how many candidates are present in the election.

The screenshot shows the REMIX Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel displays a 'VOTE' section with '_candidateId: "1"' and a 'CANDIDATES' section with '_id: "2"'. On the right, the code editor shows the Solidity contract:

```
21     uint indexed _can;
22     );
23
24     function Election () p
25         addCandidate("Candidate 1");
26         addCandidate("Candidate 2");
27     }
28
29     function addCandidate (string name) {
30         candidatesCount++;
31         candidates[candidateCount] = Candidate(name);
32     }
33
34     function vote (uint _candidateId) {
35         // require that they haven't voted yet
36         require(!voters[msg.sender]);
37
38         // require a valid candidateId
39         require(_candidateId > 0 && _candidateId < candidatesCount);
40
41         voters[msg.sender] = true;
42
43     }

```

The 'Explain contract' button is visible at the bottom of the code editor. The status bar at the bottom indicates a transaction count of 0 and a message about listening on all transactions.

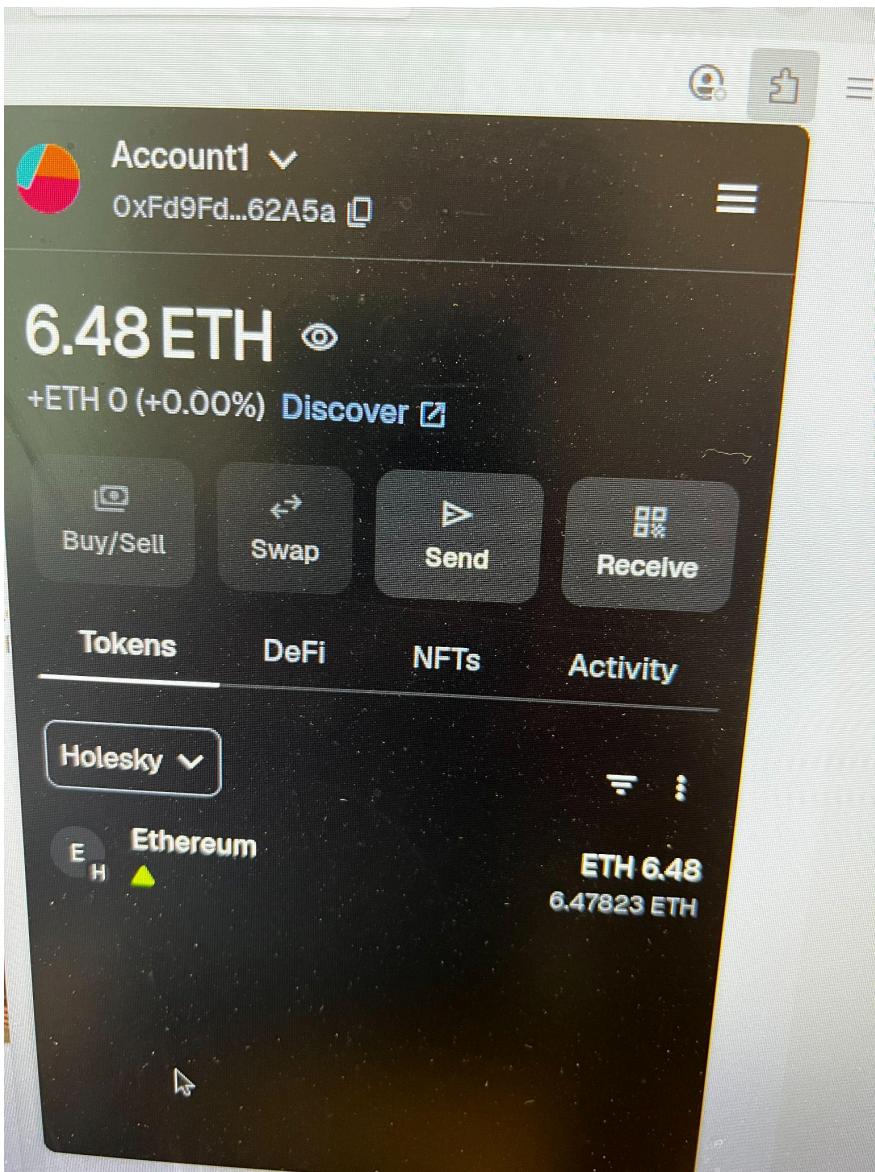
Finally, if you paste the account address of a voter (Account 1) from your MetaMask and click on the fourth option, voters, you will see information in the console about that voter (Account 1) and whether they have voted.

The screenshot shows the Ethereum Remix IDE interface. On the left, there's a sidebar for "DEPLOY & RUN TRANSACTIONS" with sections for "CANDIDATES" and "VOTERS". The "CANDIDATES" section has a dropdown set to "2" and a "call" button. Below it, parameters are listed: "0: uint256: id 2", "1: string: name Candidate 2", and "2: uint256: voteCount 0". The "VOTERS" section shows a dropdown with the account address "576435d04009405bF962A5a" and a "0: bool: false" field. On the right, the main area displays the Solidity code for the "Election" contract:

```
11 21     uint indexed _candidateCount;
12 22 );
13 23
14 24     function Election () public {
15 25         addCandidate("Candidate 1");
16 26         addCandidate("Candidate 2");
17 27     }
18 28
19 29     function addCandidate (string _name) public {
20 30         candidatesCount++;
21 31         candidates[_candidateCount] = Candidate(_name);
22 32     }
23 33
24 34     function vote (uint _candidateId) public {
25 35         // require that they haven't voted yet
26 36         require(!voters[msg.sender]);
27 37
28 38         // require a valid candidateId
29 39         require(_candidateId < candidatesCount);
29 40
30 41         // record that voter voted
31 42         voters[msg.sender] = true;
32 43     }
33
34
35
36
37
38
39
39
40
41
42
43
```

Below the code, there are buttons for "Compile", "Home", and "Explain contract". At the bottom, there are buttons for "Transact", "Low level interactions", and "CALLDATA". The status bar at the bottom indicates "0 Listen on all transactions" and "Filter with...".

Take a screenshot of your MetaMask wallet showing that your account has a balance of ETH.



Try voting for Candidate 1 again using the same account address. Can you revote? What message do you receive? Provide a screenshot as evidence.

I tried to revote as candidate 1. But it was not successful. This is shown in the screenshots below.

The screenshot shows the Remix Ethereum IDE interface. The top bar displays the date and time as "Sep 23 13:16". Below the bar, there are three tabs: "Remix - Ethereum", "Holešky PoW Faucet", and "election/contracts". The main workspace is titled "default_workspace" and contains a Solidity smart contract named "MyContract.sol". The code for the contract is as follows:

```
uint indexed _candidateId
);

function Election () public {
    addCandidate("Candidate 1");
    addCandidate("Candidate 2");
}

function addCandidate (string _name) private {
    candidatesCount++;
    candidates[candidatesCount] = Candidate(_name);
}

function vote (uint _candidateId) public {
    // require that they haven't voted before
    require(!voters[msg.sender]);

    // require a valid candidate
    require(_candidateId > 0 && _candidateId < candidatesCount);

    // record that voter has voted
    voters[msg.sender] = true;
}
```

The "RUN" tab on the left shows a green "Success" status with 0 recorded runs. The "Contracts" tab shows the deployed contract at address 0xd91...39138. The "ETH" tab shows a balance of 0. The "TRANSACTIONS" tab shows a transaction being prepared to call the "vote" function with candidate ID 1. The "CALL DATA" tab shows the function signature and parameters. The bottom right corner of the workspace shows a tooltip for a transaction: "0 Listen on all transactions to: Election.vote(uint256) 0xd91...39138 value: 0 wei data: 0x012...00001 logs: 0 hash: 0xd69...7291c".