Roll No. 45 Exam Seat No._____

VIVEKANANDEDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

HashuAdvani Memorial Complex, Collector's Colony, R. C. Marg,
Chembur, Mumbai – 400074. Contact No. 02261532532

Since 1962

CERTIFICATE

Certified that Mr./Miss Pushkar Prasad Sane

of SY MCA / A has

satisfactorily completed a course of the necessary experiments in

Blockchain Lab under my supervision

in the Institute of Technology in the academic year 2024 - 2025.

Principal Head of Department

Lab In-charge Subject Teacher

V.E.S. Institute of Technology, Collector Colony,
Chembur, Mumbai, Maharashtra 400047
Department of M.C.A

INDEX

Sr. No.	Contents	Date of Preparation	Date of Submission	Marks	Faculty Sign
1	Implement Caesar Cipher (Symmetric Encryption) and show the encryption as well as decryption process.	06-08-2024	13-08-2024		
2	Implementation of Asymmetric key algorithm (RSA).	06-08-2024	13-08-2024		
3	Implementation of SHA 256.	13-08-2024	21-08-2024		
4	Implementation of binary tree and merkle tree.	13-08-2024	21-08-2024		
5	Implementation of block (Genesis Block) and private chain using geth.	20-08-2024	04-09-2024		
6	Install Ganache(Personal block chain) and metamask (Show the installation Steps) . Compile and deploy a election smart contract in the personal blockchain using injected web3 environment (Metamask wallet). Use Remix online ide to compile and deploy the smart contact.	03-09-2024	01-10-2024		

7	1. Write a program in solidity to create a structure student with Roll no, Name, Class, Department, Course enrolled as variables. 2. Create a structure Consumer with Name, Address, Consumer ID, Units and Amount as members. Write a program in solidity to calculate the total electricity bill according to the given condition	08-10-2024	08-10-2024		
8	Execution of smart contract using truffle framework.	15-08-2024	16-09-2024		
9	Creation of Dapp in Ethereum Build DAPP's for 1. voting process.	22-10-2024	25-09-2024		
10	VacChain : Mini Project	23-10-2024	28-10-2024		

Final Grade	**Instructor Signature**

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 1**	
Title of Lab Assignment: Implement Caesar Cipher (Symmetric Encryption) and show the encryption as well as decryption process.		
DOP: 06-08-2024	**DOS: 13-08-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 1

Aim: Implement Caesar Cipher (Symmetric Encryption) and show the encryption as well as decryption process.

Theory:

What is Cryptography?

Cryptography is a technique of securing information and communications through the use of codes so that only those persons for whom the information is intended can understand and process it. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and the suffix “graphy” means “writing”. In Cryptography, the techniques that are used to protect information are obtained from mathematical concepts and a set of rule-based calculations known as algorithms to convert messages in ways that make it hard to decode them. These algorithms are used for cryptographic key generation, digital signing, and verification to protect data privacy, web browsing on the internet and to protect confidential transactions such as credit card and debit card transactions.

Features Of Cryptography:

1. Confidentiality: Information can only be accessed by the person for whom it is intended and no other person except him can access it.
2. Integrity: Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.
3. Non-repudiation: The creator/sender of information cannot deny his intention to send information at a later stage.
4. Authentication: The identities of the sender and receiver are confirmed. As well, the destination / origin of the information is confirmed.
5. Interoperability: Cryptography allows for secure communication between different systems and platforms.
6. Adaptability: Cryptography continuously evolves to stay ahead of security threats and technological advancements.

Types Of Cryptography:

1. Symmetric Key Cryptography
2. Asymmetric Key Cryptography

What is Symmetric Encryption?

It is an encryption system where the sender and receiver of a message use a single common key to encrypt and decrypt messages. Symmetric Key cryptography is faster and simpler but the problem is that the sender and receiver have to somehow exchange keys securely. The most popular symmetric key cryptography systems are Data Encryption Systems (DES) and Advanced Encryption Systems (AES).

Explanation about caesar cipher

The Caesar Cipher is one of the simplest and oldest methods of encrypting messages, named after Julius Caesar, who reportedly used it to protect his military communications. This technique involves shifting the letters of the alphabet by a fixed number of places. For example, with a shift of three, the letter ‘A’ becomes ‘D’, ‘B’ becomes ‘E’, and so on. Despite its simplicity, the Caesar Cipher formed the groundwork for modern cryptographic techniques. In this article, we’ll explore how the Caesar Cipher works, its significance, and its impact on the development of cryptography with its advantages and disadvantages

Advantages:

1. Easy to implement and use thus, making it suitable for beginners to learn about encryption.
2. Can be physically implemented, such as with a set of rotating disks or a set of cards, known as a scytale, which can be useful in certain situations.
3. Requires only a small set of pre-shared information.
4. Can be modified easily to create a more secure variant, such as by using multiple shift values or keywords.

Disadvantages:

1. It is not secure against modern decryption methods.
2. Vulnerable to known-plaintext attacks, where an attacker has access to both the encrypted and unencrypted versions of the same messages.
3. The small number of possible keys means that an attacker can easily try all possible keys until the correct one is found, making it vulnerable to a brute force attack.
4. It is not suitable for long text encryption as it would be easy to crack.
5. It is not suitable for secure communication as it is easily broken.
6. Does not provide confidentiality, integrity, and authenticity in a message.

Rules for the Caesar Cipher:

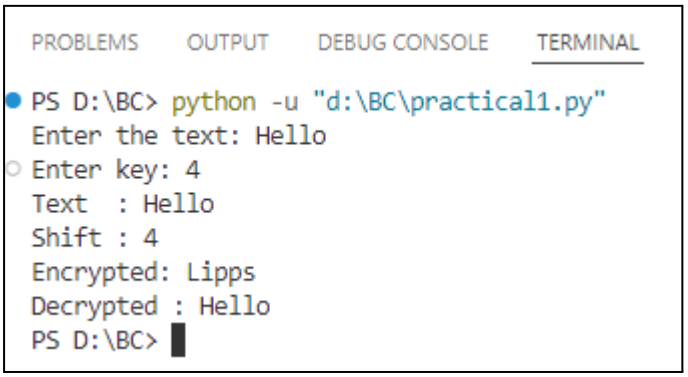
1. Choose a number between 1 and 25. This will be your “shift” value.
2. Write down the letters of the alphabet in order, from A to Z.
3. Shift each letter of the alphabet by the “shift” value. For example, if the shift value is 3, A would become D, B would become E, C would become F, and so on.
4. Encrypt your message by replacing each letter with the corresponding shifted letter. For example, if the shift value is 3, the word “hello” would become “khoor”.
5. To decrypt the message, simply reverse the process by shifting each letter back by the same amount. For example, if the shift value is 3, the encrypted message “khoor” would become “hello”.

Code:

```
def encrypt(text, key):
    result = ""

    # traverse text
    for i in range(len(text)):
        char = text[i]

        # Encrypt uppercase characters
        if (char.isupper()):
            result += chr((ord(char) + key - 65) % 26 + 65)

        # Encrypt lowercase characters
        elif char.islower():
            result += chr((ord(char) + key - 97) % 26 + 97)
        else:
            result += char

    return result
def decrypt(text, key):
    result = ""

    # traverse text
    for i in range(len(text)):
        char = text[i]
```

```
        # Encrypt uppercase characters
        if (char.isupper()):
            result += chr((ord(char) - key - 65) % 26 + 65)

        # Encrypt lowercase characters
        elif char.islower():
            result += chr((ord(char) - key - 97) % 26 + 97)
        else:
            result += char
    return result

#check the above function
text = input("Enter the text: ")
key = int(input("Enter key: "))
print ("Text  : " + text)
print ("Shift : " + str(key))

encrypted_text = encrypt(text, key)
print ("Cipher: " + encrypted_text)

decrypted_text = decrypt(encrypted_text, key)
print ("Original Text : " + decrypted_text)
```

Output:



Conclusion:

Implemented Caesar Cipher (Symmetric Encryption) and show the encryption as well as decryption process successfully

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 2**	
Title of Lab Assignment: Implementation of Asymmetric key algorithm (RSA).		
DOP: 06-08-2024	**DOS: 13-08-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 2

Aim: Implementation of Asymmetric key algorithm (RSA)

Theory:

RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes, the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography:

1. A client (for example browser) sends its public key to the server and requests some data.
2. The server encrypts the data using the client's public key and sends the encrypted data.
3. The client receives this data and decrypts it.

Advantages:

1. Security: RSA algorithm is considered to be very secure and is widely used for secure data transmission.
2. Public-key cryptography: RSA algorithm is a public-key cryptography algorithm, which means that it uses two different keys for encryption and decryption. The public key is used to encrypt the data, while the private key is used to decrypt the data.
3. Key exchange: RSA algorithm can be used for secure key exchange, which means that two parties can exchange a secret key without actually sending the key over the network.
4. Digital signatures: RSA algorithm can be used for digital signatures, which means that a sender can sign a message using their private key, and the receiver can verify the signature using the sender's public key.
5. Speed: The RSA technique is suited for usage in real-time applications since it is quite quick and effective.
6. Widely used: Online banking, e-commerce, and secure communications are just a few fields and applications where the RSA algorithm is extensively developed.

Disadvantages:

1. Slow processing speed: RSA algorithm is slower than other encryption algorithms, especially when dealing with large amounts of data.

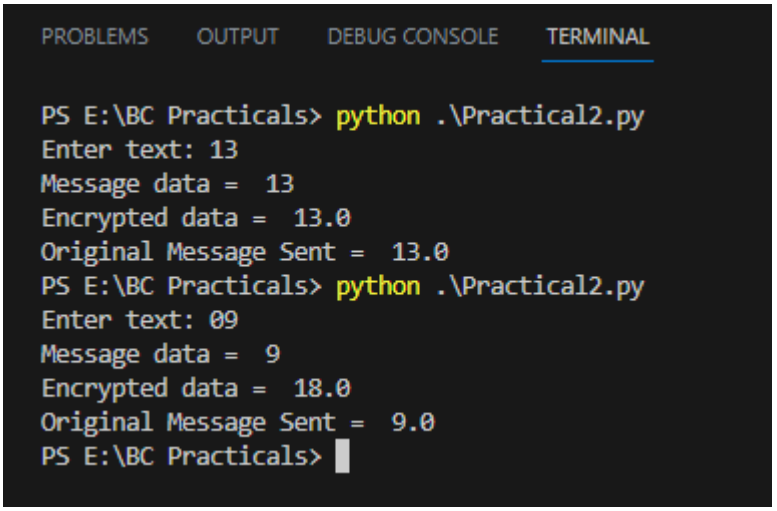
2. Large key size: RSA algorithm requires large key sizes to be secure, which means that it requires more computational resources and storage space.
3. Vulnerability to side-channel attacks: RSA algorithm is vulnerable to side-channel attacks, which means an attacker can use information leaked through side channels such as power consumption, electromagnetic radiation, and timing analysis to extract the private key.
4. Limited use in some applications: RSA algorithm is not suitable for some applications, such as those that require constant encryption and decryption of large amounts of data, due to its slow processing speed.
5. Complexity: The RSA algorithm is a sophisticated mathematical technique that some individuals may find challenging to comprehend and use.
6. Key Management: The secure administration of the private key is necessary for the RSA algorithm, although in some cases this can be difficult.
7. Vulnerability to Quantum Computing: Quantum computers have the ability to attack the RSA algorithm, potentially decrypting the data.

Code:

```
#Implementation of RSA Algorithm
import math
def gcd(a, h):
        temp = 0
        while(1):
                temp = a % h
                if (temp == 0):
                        return h
                a = h
                h = temp

p = 3
q = 7
n = p*q
e = 2
phi = (p-1)*(q-1)
while (e < phi):
        if(gcd(e, phi) == 1):
                break
```

```
        else:
                e = e+1
k = 2
d = (1 + (k*phi))/e

msg = int(input("Enter text: "))
print("Message data = ", msg)

c = pow(msg, e)
c = math.fmod(c, n)
print("Encrypted data = ", c)

m = pow(c, d)
m = math.fmod(m, n)
print("Original Message Sent = ", m)
```

Output:



Conclusion:

Successfully demonstrated the implementation of Asymmetric key algorithm (RSA).

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 3**	
Title of Lab Assignment: Implement SHA256 Algorithm.		
DOP: 13-08-2024	**DOS: 21-08-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 3

Aim: Implement SHA256 algorithm

Theory:

Hashing is the process of converting input data (or a "message") into a fixed-size string of bytes. This fixed-size output is typically a sequence of numbers and letters, known as a hash value, hash code, or simply hash. The function used to perform this conversion is called a hash function.

1. Deterministic: The same input will always produce the same hash value.
2. Fixed Output Length: Regardless of the input size, the output (hash value) is always of a fixed length.
3. Efficient Computation: The hash function should be able to produce the hash value quickly.
4. Preimage Resistance: It should be computationally infeasible to reverse the hash function to retrieve the original input from the hash value.
5. Collision Resistance: It should be difficult to find two different inputs that produce the same hash value.
6. Avalanche Effect: A small change in the input should produce a significantly different hash value.

Common Uses of Hashing

1. Cryptographic Applications:
   - ➢ Password Hashing: Passwords are hashed before storage in databases. During login, the input password is hashed and compared to the stored hash, protecting the actual password from exposure.
   - ➢ Digital Signatures: Hash functions are used to create a unique hash of a document. This hash is then encrypted to form a digital signature, which can be used to verify the authenticity and integrity of the document.

2. **Data Integrity Verification:**
   - ➢ Checksums: Hashes are used in checksums to detect errors or modifications in data during transmission or storage. For example, software distributions often include hash values so users can verify the integrity of the downloaded files.

3. Unique Identifiers:
    - ➢ Unique IDs: Hash functions are used to create unique identifiers for objects, such as database keys or session tokens.

Example of Hash Functions

1. MD5 (Message Digest Algorithm 5):
    - Produces a 128-bit hash value, often rendered as a 32-character hexadecimal number.
    - Example: Hash of "hello" is 5d41402abc4b2a76b9719d911017c592.
2. SHA-1 (Secure Hash Algorithm 1):
    - Produces a 160-bit hash value, often rendered as a 40-character hexadecimal number.
    - Example: Hash of "hello" is f7c3bc1d808e04732adf679965ccc34c.
3. SHA-256 (Secure Hash Algorithm 256-bit):
    - Part of the SHA-2 family, produces a 256-bit hash value.
    - Example: Hash of "hello" is 2cf24dba5fb0a30e26e83b2ac5b0e8a0.

Short Explanation about SHA256

SHA-256 (Secure Hash Algorithm 256-bit) is a cryptographic hash function that produces a 256-bit (32-byte) hash value, often represented as a 64-character hexadecimal string. It is part of the SHA-2 family of hash functions, which was designed to improve upon the security of the older SHA-1 algorithm.

Key Characteristics of SHA-256

1. **Fixed Output Length:**
    - Regardless of the input size, the output hash value is always 256 bits (32 bytes) long.
2. **Deterministic:**
    - The same input will always produce the same hash value.
3. **Pre-image Resistance:**
    - It is computationally infeasible to reverse the hash function and retrieve the original input from the hash value.
4. **Collision Resistance:**
    - It is difficult to find two different inputs that produce the same hash value.
5. **Avalanche Effect:**
    - A small change in the input drastically changes the hash value.

Example Usage

SHA-256 is commonly used in:

- **Digital Signatures:** To ensure data integrity and authenticity.
- **Password Hashing:** To securely store passwords.
- **Data Integrity Verification:** To check if data has been altered.

Explanation of Python libraries and methods used for implementing the program

In the context of implementing a program like the RSA encryption and decryption example provided earlier, several Python libraries and methods are used. Here's an explanation of the relevant libraries and their methods:

Python Libraries and Methods

1. **math Library**
   - **Purpose:** Provides mathematical functions.
   - **Method Used:**
     - gcd(a, b): Calculates the greatest common divisor of a and b. In the RSA algorithm, it's used to find a public exponent e that is coprime with the totient t.

   python
   from math import gcd

2. **hashlib Library**
   - **Purpose:** Provides hashing algorithms for cryptographic purposes.
   - **Method Used:**
     - hashlib.sha256(): Creates a new SHA-256 hash object. This is used to generate a SHA-256 hash of a message, ensuring data integrity and security.

   Example:
   import hashlib
   sha256 = hashlib.sha256()
   sha256.update(message.encode())
   hash_value = sha256.hexdigest()

3. **Python Built-in Functions**
    - **Purpose:** Built-in functions provide fundamental capabilities in Python.
    - **Methods Used:**
        - input(prompt): Reads a line from input, used to get user input.
        - int(): Converts a string to an integer, used to ensure numeric inputs.
        - print(): Outputs data to the console.

    Example:

    ```
    p = int(input("Enter the first prime number (p): "))
    q = int(input("Enter the second prime number (q): "))
    message = int(input("Enter the message (integer): "))
    ```

4. **Control Flow Statements**
    - **Purpose:** Direct the flow of execution based on conditions.
    - **Statements Used:**
        - if: Conditional statements to execute code blocks based on conditions.
        - while: Loops that continue execution as long as a condition is true.

    Example:

    ```
    if p <= 1 or q <= 1:
       print("Both p and q must be greater than 1.")
    ```

5. **Custom Functions**
    - **Purpose:** Encapsulate code into reusable components.
    - **Functions Defined:**
        - RSA(p, q, message): Implements the RSA algorithm for encryption and decryption.
        - main(): The main function to handle user input and call the RSA function.
        - is_prime(num): Checks if a number is prime, used for RSA key validation.

    Example:

    ```
    def RSA(p: int, q: int, message: int):
       # RSA algorithm implementation


    def main():
       # Input handling and RSA function call
    ```

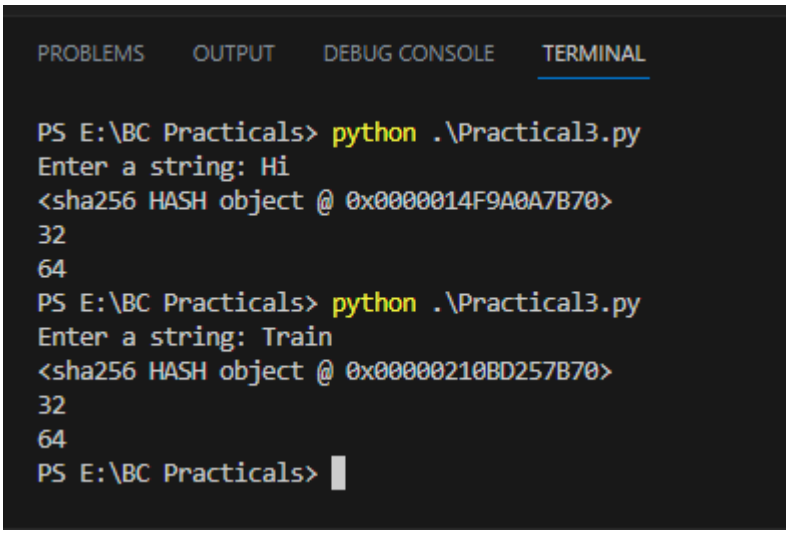
```
def is_prime(num):
    # Prime checking logic
```

1. **User Input Handling**: Uses input() to gather values for p, q, and message, and int() to convert these inputs into integers.
   python
   Copy code
   p = int(input("Enter the first prime number (p): "))
2. **Hashing (if used)**: Though not directly in the RSA example, if you were using hashlib, you'd use it to hash messages for security purposes.
   python
   Copy code
   sha256 = hashlib.sha256()
   sha256.update(message.encode())
   hash_value = sha256.hexdigest()
3. **Custom Logic**: Includes RSA for performing encryption and decryption, and is_prime to validate prime numbers used in the RSA algorithm.
   Example:
   # Implement RSA encryption and decryption
   def RSA(p: int, q: int, message: int):
   # Check if num is a prime number
   def is_prime(num):

Code:

```
import hashlib
str1 = input("Enter a string")
encodedstr = str1.encode()
result = hashlib.sha256(encodedstr)
print(result)
# result in hexadecimal format
result.hexdigest()
print(result.digest_size)
print(result.block_size)
```

Output:



Conclusion:

Successfully demonstrated the implementation of SHA256 algorithm.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 4**	
Title of Lab Assignment: Implementation of binary tree and merkle tree.		
DOP: 13-08-2024	**DOS: 21-08-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 4

Aim: Implementation of binary tree and merkle tree

Theory:

1. **Binary Tree**

   A binary tree is a hierarchical data structure in which each node has at most two children, referred to as the left child and the right child. This structure is one of the most fundamental concepts in computer science and is widely used due to its efficient data organization and retrieval capabilities. The topmost node in a binary tree is known as the root, and each node, except the root, has a parent node. The nodes without any children are called leaf nodes, and nodes that have at least one child are known as internal nodes.

   Binary trees are primarily used to implement binary search trees (BSTs), where the left child of a node contains values smaller than the node's value, and the right child contains values greater than the node's value. This ordering property allows for efficient searching, insertion, and deletion operations, with an average-case time complexity of O(log n) in balanced trees. However, in the worst case, such as in an unbalanced tree that resembles a linked list, the time complexity can degrade to O(n).

   Binary trees are also foundational for various other tree-based data structures, including AVL trees, red-black trees, and binary heaps. For example, AVL trees and red-black trees are self-balancing binary search trees that automatically maintain a balanced structure during insertions and deletions to ensure optimal search times. Binary heaps, on the other hand, are specialized binary trees used to implement priority queues, where the parent node is always greater than or equal to (in max-heaps) or less than or equal to (in min-heaps) its children.

   Beyond searching and sorting, binary trees are also used in expression parsing, where each node represents an operator, and the children represent operands. This application is crucial in compilers and interpreters for evaluating mathematical expressions. Additionally, binary trees play a key role in Huffman coding, a compression algorithm that generates an optimal prefix code based on the frequency of characters in a data set.

2. Merkle Tree

A Merkle tree, also known as a hash tree, is a cryptographic data structure that extends the concept of a binary tree to efficiently and securely verify the integrity of large data sets. In a Merkle tree, each leaf node represents a hash of a block of data, while each non-leaf (or internal) node contains a hash of the concatenation of its child nodes' hashes. This hierarchical structure culminates in a single hash at the top, known as the root hash or Merkle root, which serves as a unique fingerprint for the entire dataset

.

The primary advantage of a Merkle tree is its ability to verify the integrity of a large set of data with minimal data transmission and computational effort. To verify whether a particular piece of data is part of the dataset, one only needs to check the hashes along the path from the relevant leaf node to the Merkle root. This process, known as a Merkle proof, involves comparing the computed hashes to the stored hashes, ensuring that any tampering or corruption of the data will be immediately detected.

Merkle trees are extensively used in blockchain technology to ensure the integrity and consistency of transactions within blocks. In a blockchain, each block contains a Merkle root that represents the hash of all the transactions within that block. This allows for efficient verification of individual transactions without needing to download the entire blockchain, which is crucial for lightweight clients (SPV clients) in a distributed network.

In addition to blockchains, Merkle trees are also used in distributed systems and peer-to-peer networks, where they help ensure data integrity across distributed nodes. For example, in distributed file systems like IPFS or in peer-to-peer protocols like BitTorrent, Merkle trees enable nodes to verify that they have received unaltered and complete data from other peers.

Merkle trees also play a critical role in optimizing network bandwidth. By enabling partial data verification, they allow systems to transmit only the necessary parts of the tree for verification purposes, reducing the amount of data that needs to be exchanged between nodes.

Overall, the Merkle tree is a powerful tool in cryptographic and distributed systems, offering a scalable and secure method for verifying large data sets, ensuring data integrity, and enabling efficient data transmission in decentralized environments.

Code:

```
#Binary Tree
class Node:
    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data

    def insert(self, data):
      # Compare the new value with the parent node
        if self.data:
            if data < self.data:
                if self.left is None:
                    self.left = Node(data)
                else:
                    self.left.insert(data)
            elif data > self.data:
                if self.right is None:
                    self.right = Node(data)
                else:
                    self.right.insert(data)
        else:
            self.data = data

    # Print the tree
    def PrintTree(self):
        if self.left:
            self.left.PrintTree()
        print(self.data)
        if self.right:
            self.right.PrintTree()
```

```
    # Inorder traversal (Left -> Root -> Right)
    def inorderTraversal(self, root):
        res = []
        if root:
            res = self.inorderTraversal(root.left)
            res.append(root.data)
            res = res + self.inorderTraversal(root.right)
        return res

    # Preorder traversal (Root -> Left -> Right)
    def preorderTraversal(self, root):
        res = []
        if root:
            res.append(root.data)
            res = res + self.preorderTraversal(root.left)
            res = res + self.preorderTraversal(root.right)
        return res

    # Postorder traversal (Left -> Right -> Root)
    def postorderTraversal(self, root):
        res = []
        if root:
            res = self.postorderTraversal(root.left)
            res = res + self.postorderTraversal(root.right)
            res.append(root.data)
        return res

# Take root input from the user
root_value = int(input("Enter the root value: "))
root = Node(root_value)

# Insert values based on user input
while True:
    data = input("Enter a value to insert ")
    if data == '':
        break
```

```
    root.insert(int(data))
# Print traversals
print("\nBinary Tree:")
print("Inorder Traversal:", root.inorderTraversal(root))
print("Preorder Traversal:", root.preorderTraversal(root))
print("Postorder Traversal:", root.postorderTraversal(root))
```

#Merkle Tree

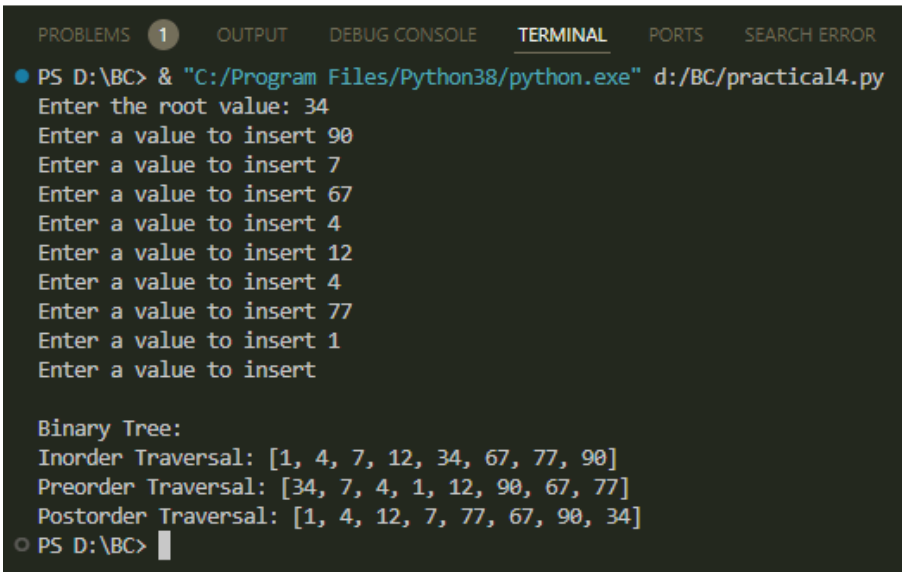
```
from typing import List
import hashlib

class Node:
def __init__(self, left, right, value: str, content) -> None:
        self.left: Node = left
        self.right: Node = right
        self.value = value
        self.content = content

    @staticmethod
    def hash(val: str) -> str:
        return hashlib.sha256(val.encode("utf-8")).hexdigest()

    def __str__(self):
        return str(self.value)

class MerkleTree:
    def __init__(self, values: List[str]) -> None:
        self.__buildTree(values)

    def __buildTree(self, values: List[str]) -> None:
        leaves: List[Node] = [Node(None, None, Node.hash(e), e) for e in values]
        if len(leaves) % 2 == 1:
            leaves.append(leaves[-1:][0])
        self.root: Node = self.__buildTreeRec(leaves)

    def __buildTreeRec(self, nodes: List[Node]) -> Node:
```

```
        half: int = len(nodes) // 2

        if len(nodes) == 2:
                return Node(nodes[0], nodes[1], Node.hash(nodes[0].value + nodes[1].value),
nodes[0].content + "+" + nodes[1].content)

        left: Node = self.__buildTreeRec(nodes[:half])
        right: Node = self.__buildTreeRec(nodes[half:])
        value: str = Node.hash(left.value + right.value)
        content: str = left.content + "+" + right.content
        return Node(left, right, value, content)

    def printTree(self) -> None:
        self.__printTreeRec(self.root)

    def __printTreeRec(self, node) -> None:
        if node is not None:
            if node.left is not None:
                print("Left: " + str(node.left))
                print("Right: " + str(node.right))
            else:
                print("Input")

            print("Value: " + str(node.value))
            print("Content: " + str(node.content))
            print("")
            self.__printTreeRec(node.left)
            self.__printTreeRec(node.right)

    def getRootHash(self) -> str:
        return self.root.value

def mixmerkletree() -> None:
    num_inputs = int(input("Enter the number of elements: "))
    elems = [input(f"Enter element {i + 1}: ") for i in range(num_inputs)]
```

```
    print("\nInputs: ")
    print(*elems, sep=" | ")
    print("")

    mtree = MerkleTree(elems)
    print("Root Hash: " + mtree.getRootHash() + "\n")
    mtree.printTree()

mixmerkletree()
```

Output:

Binary Tree



Merkle Tree

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS G:\Pushkar\MCA\Sem - 3\BC> & D:/Python/python.exe "g:/Pushkar/MCA/Sem - 3/BC/Practicals/MerkleTree.py"
Enter the number of elements: 4
Enter element 1: Hello
Enter element 2: This
Enter element 3: Is
Enter element 4: Pushkar

Inputs:
Hello | This  | Is  | Pushkar

Root Hash: aabe6f8c17fff884ac581959d008ab1262e765ff252069ace9a4cb63dcf9d919

Left: 4c9cb9aa539d775482b7bb96cec85223eca35e5e5901dcb78897ef74e58cf4d0
Right: a1c36a4c3b6a6e80b6bd7706fa29889ee34e50bb60243676a88a5fc702316c90
Value: aabe6f8c17fff884ac581959d008ab1262e765ff252069ace9a4cb63dcf9d919
Content: Hello+This +Is +Pushkar

Left: 185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969
Right: ddf93d8fff520ab6e142e8fff2872afc49f57822affdeaa8d9bffef14c020c59
Value: 4c9cb9aa539d775482b7bb96cec85223eca35e5e5901dcb78897ef74e58cf4d0
Content: Hello+This

Input
Value: 185f8db32271fe25f561a6fc938b2e264306ec304eda518007d1764826381969
Content: Hello

Input
Value: ddf93d8fff520ab6e142e8fff2872afc49f57822affdeaa8d9bffef14c020c59
Content: This

Left: 999a42025672ddc46ded0873c52d509fe549c207e4dfcdae2f807c2920c2d8d3
Right: 92ff973cc98b8f3e86f34c0c3a8407d5fb74ee304e297bd7579ce4ccb7584b7c
Value: a1c36a4c3b6a6e80b6bd7706fa29889ee34e50bb60243676a88a5fc702316c90
Content: Is +Pushkar

Input
Value: 999a42025672ddc46ded0873c52d509fe549c207e4dfcdae2f807c2920c2d8d3
Content: Is

Input
Value: 92ff973cc98b8f3e86f34c0c3a8407d5fb74ee304e297bd7579ce4ccb7584b7c
Content: Pushkar

PS G:\Pushkar\MCA\Sem - 3\BC>
```

Conclusion:

The implementation of the binary tree demonstrated the core functionalities of insertion and various tree traversal methods, including inorder, preorder, and postorder. The user input facilitated dynamic construction of the binary tree, allowing for a flexible and interactive demonstration of the tree's operations. The code effectively illustrates how binary trees can be used for efficient data organization and retrieval.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 5**	
Title of Lab Assignment: Implementation of block(Genesis Block) and private chain using geth.		
DOP: 20-08-2024	**DOS: 04-09-2024**	
CO Mapped: **CO3, CO4**	**PO Mapped:** **PO1, PO2, PO3, PO7, PO9, PSO1**	**Signature:**

Practical No. 5

Aim: Implementation of block(Genesis Block) and private chain using geth.

Description:

What is a node?

- A node is generally a point of intersection or connection in a telecommunications network. A node may also mean any system or physical device that is connected to a network and can execute certain functions like creating, receiving or sending information via a communication channel. The explanation of a node varies depending on the protocol layer being referred to.
- For example, a basic resident network may consist of a file server, two laptops and a fax machine. In this case, the network has four nodes, each equipped with a MAC address to uniquely identify them.
- The most popular usage of the term “node” is seen in the blockchain space. In this guide, we will explain what nodes are in more detail, including the different types of blockchain nodes being used today.
- In Ethereum, a user can run three different kinds of nodes: light, full and archive. Their differences lie in how fast they can synchronize with the entire network.
- There are many ways to run your own Ethereum node, but some popular hardware that can work on the network are DAppNode and Avado. Ethereum nodes have almost the same requirements as Bitcoin nodes, only that the former requires less computing power.
- Note that before you run an Ethereum node, it is advisable to check your bandwidth limitations first.
- Ethereum nodes are essential in keeping its blockchain network secure and reliable, as well as transparent. In fact, anyone can view the nodes and their performances on the network via Etherscan’s node tracker.
- In order to receive block rewards, you would have to run an Ethereum staking node.

Geth:

Geth (Go Ethereum) is a command line interface for running Ethereum node implemented in Go Language. Using Geth you can join Ethereum network, transfer ether between accounts or even mine ethers.

Genesis Block:

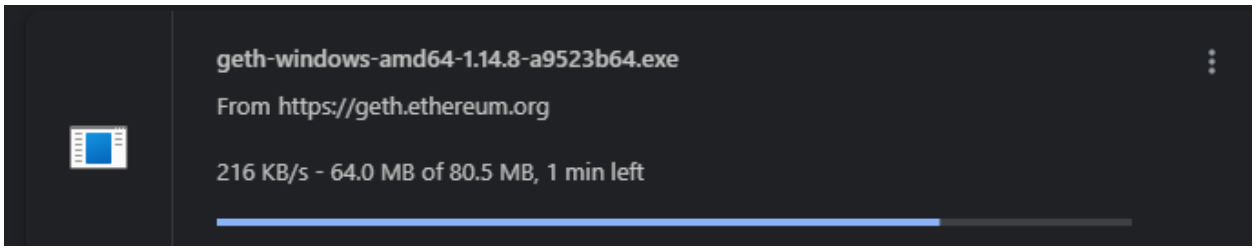
- A genesis block refers to the first block in a blockchain and is usually hardcoded into its application's software. A blockchain has multiple "blocks" (containing validated transactions and recorded activity data) linked together by a metaphorical chain.
- Each "block" of a crypto asset contains referential data for the previous one and derives its value/legitimacy from its predecessor.The genesis block, thus, refers to the first block (Block 0 or Block 1) of a new blockchain, to which all other subsequent blocks are attached.
- A genesis block is unique as it is the only block in a blockchain that does not reference a predecessor block, and in almost all cases, the first mining rewards it unlocks are unspendable.
- Genesis blocks have special significance, as they form the very foundation of a blockchain and often contain interesting stories or hidden meanings. For instance, Bitcoin's genesis block contains the now-famous message "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks" — a reference to the deteriorating financial conditions of that time and the rationale for creating cryptocurrencies like Bitcoin and Ethereum. Bitcoin's genesis block in 2009 contained 50 BTC.
- The Bitcoin genesis block is very intriguing not just for its included message, but also due to the fact that the next block was timestamped nearly six days later (the average time is 10 minutes).

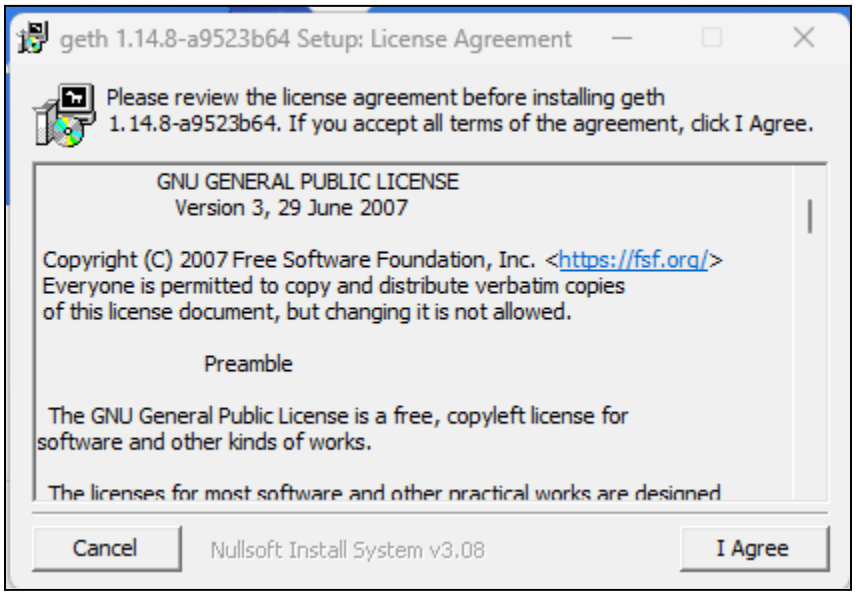
Commands used in this practical:

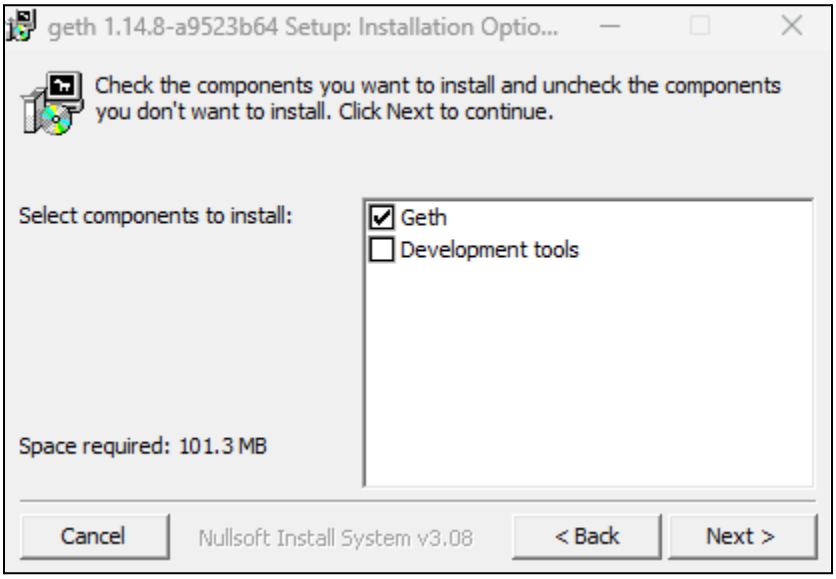
- **geth --datadir chaindata init genesis.json:** Initializes geth into chaindata.
- **geth --datadir=./chaindata/:** Used to run geth.
- **geth attach ipc:\\.\pipe\geth.ipc:** IPC to interact with geth.
- **personal.newAccount():** Create a new account.
- **eth.accounts:** Get information about all the accounts present.
- **eth.coinbase:** Get information about the coinbase account.
- **eth.getBalance(eth.accounts[0]):** Get the current balance of an account.
- **miner.start() / miner.stop():** Start/Stop the mining process.
- **eth.blockNumber:** Get the blockNumber.
- **personal.unlockAccount(eth.accounts[0]):** Unlock the coinbase account for transaction.

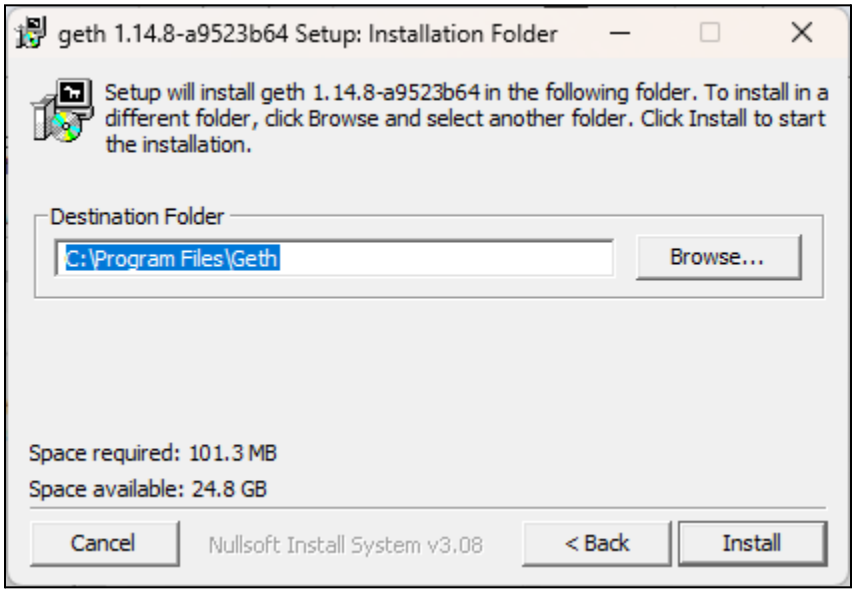
- **eth.sendTransaction({from: eth.coinbase, to: eth.accounts[1], value: web3.toWei(10,“ether”)}):** Command for transaction. Enter the account number to which we want to transfer the ethers to.
- **eth.getTransaction(txHash):** get the information about the transaction. Enter the hash instead of “txHash”.
- **web3.fromWei(eth.getBalance(eth.accounts[1]), “ether”):** get balance of the second account in ethers.
- **eth.getBlock(“latest”):** Get information about the latest block.
- **eth.getBlock(388):** Get information about a specific block.

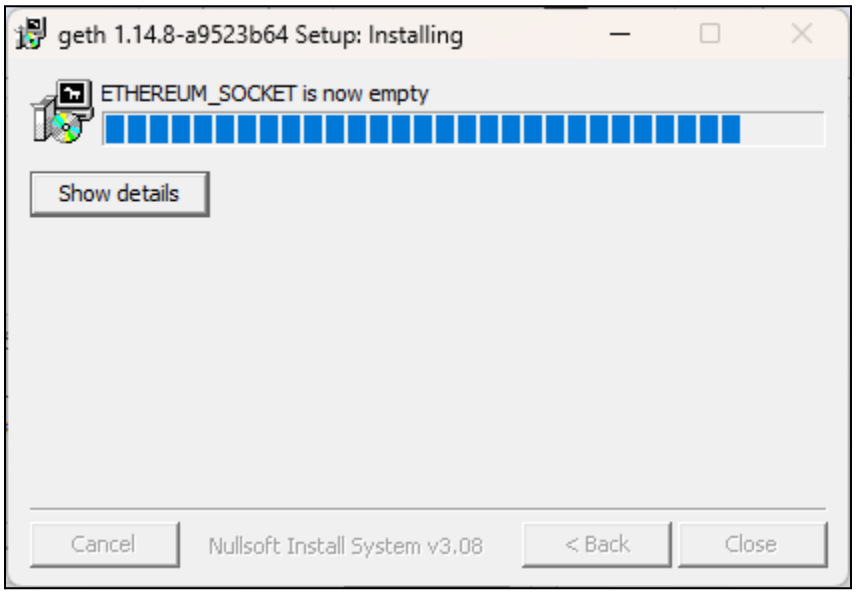
Output:

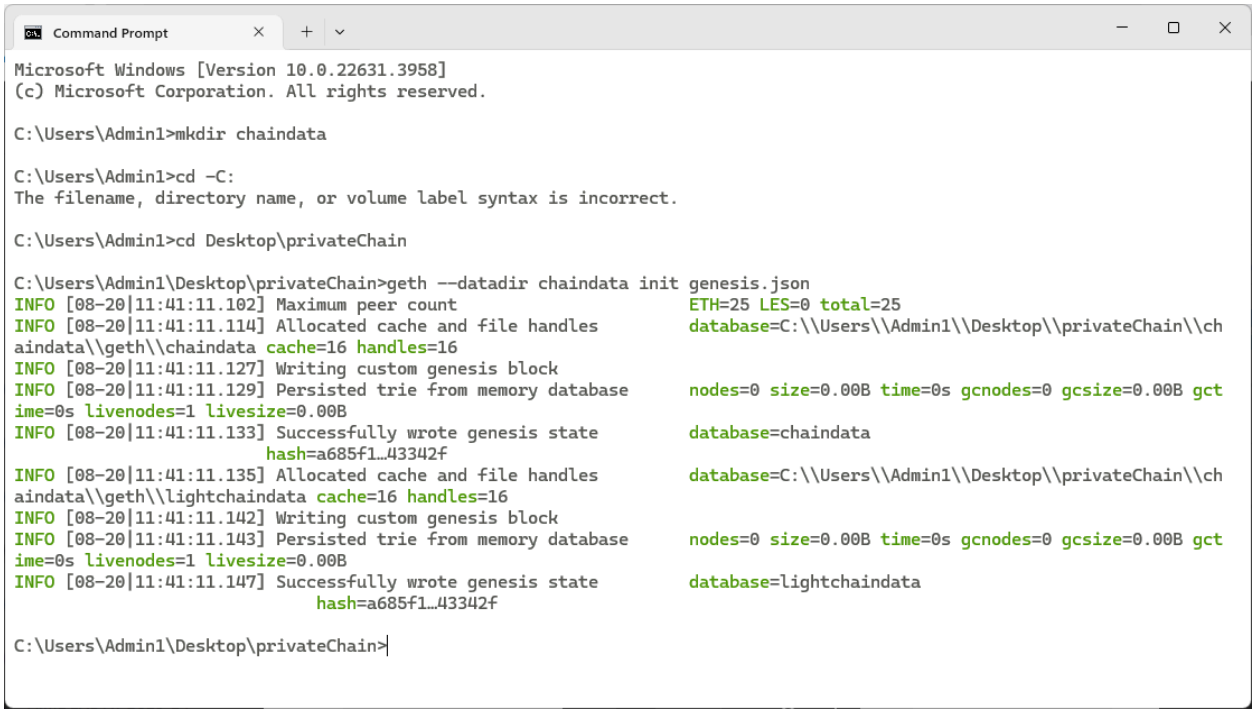


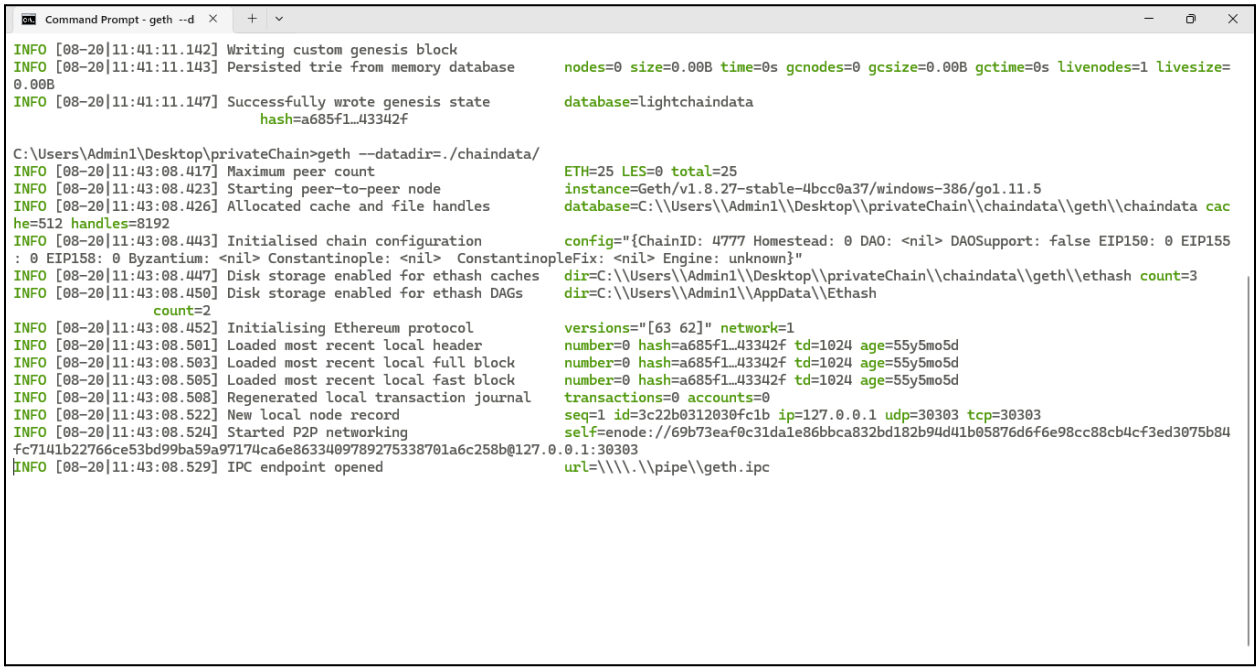



geth 1.14.8-a9523b64 Setup: Installation Optio...
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Geth
Development tools
Space required: 101.3 MB
Cancel
Nullsoft Install System v3.08
< Back
Next >


geth 1.14.8-a9523b64 Setup: Installation Folder
Setup will install geth 1.14.8-a9523b64 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.
Destination Folder
C:\Program Files\Geth
Browse...
Space required: 101.3 MB
Space available: 24.8 GB
Cancel
Nullsoft Install System v3.08
< Back
Install

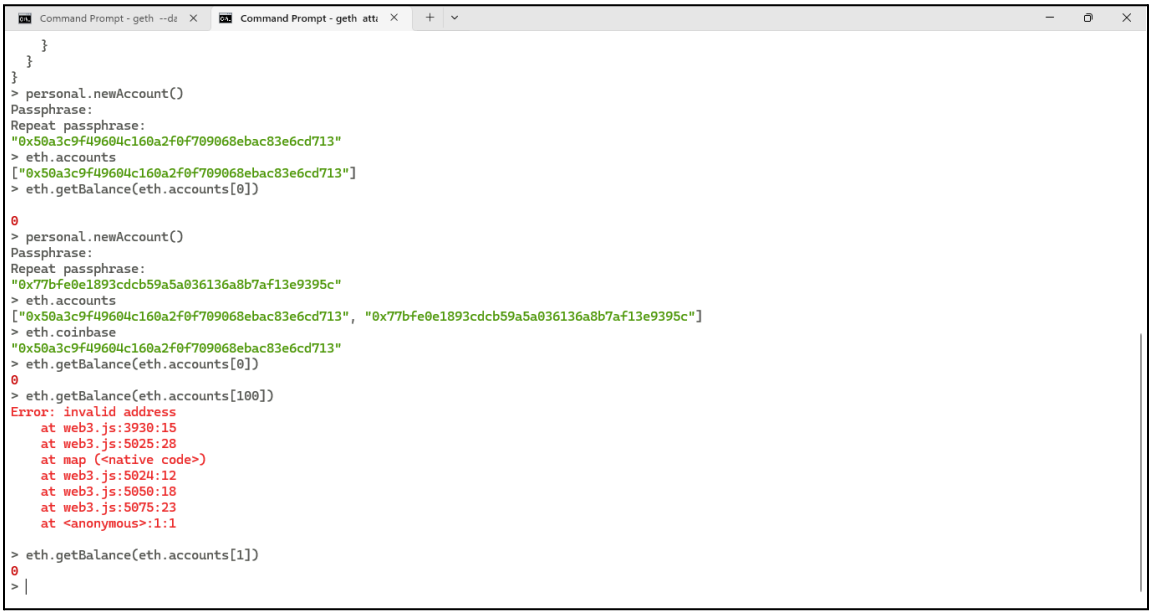

geth 1.14.8-a9523b64 Setup: Installing
ETHEREUM_SOCKET is now empty
Show details
Cancel
Nullsoft Install System v3.08
< Back
Close


Command Prompt
Microsoft Windows [Version 10.0.22631.3958]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Admin1>mkdir chaindata
C:\Users\Admin1>cd -C:
The filename, directory name, or volume label syntax is incorrect.
C:\Users\Admin1>cd Desktop\privateChain
C:\Users\Admin1\Desktop\privateChain>geth --datadir chaindata init genesis.json
INFO [08-20|11:41:11.102] Maximum peer count ETH=25 LES=0 total=25
INFO [08-20|11:41:11.114] Allocated cache and file handles database=C:\\Users\\Admin1\\Desktop\\privateChain\\chaindata\\geth\\chaindata cache=16 handles=16
INFO [08-20|11:41:11.127] Writing custom genesis block
INFO [08-20|11:41:11.129] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [08-20|11:41:11.133] Successfully wrote genesis state database=chaindata hash=a685f1…43342f
INFO [08-20|11:41:11.135] Allocated cache and file handles database=C:\\Users\\Admin1\\Desktop\\privateChain\\chaindata\\geth\\lightchaindata cache=16 handles=16
INFO [08-20|11:41:11.142] Writing custom genesis block
INFO [08-20|11:41:11.143] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [08-20|11:41:11.147] Successfully wrote genesis state database=lightchaindata hash=a685f1…43342f
C:\Users\Admin1\Desktop\privateChain>



- **geth attach ipc:\\.\pipe\geth.ipc**

- **personal.newAccount()**
- **eth.getBalance(eth.accounts[0])**



- **personal.newAccount():**

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
```

- **eth.accounts**
- **eth.coinbase**

```
> eth.accounts
["0x571b970ee70a066f08e46b9e8dd5a95e2ace0e9e", "0x0dfde928a98cb359c2a299f37b5d4c38bdd7c568"]
> eth.coinbase
"0x571b970ee70a066f08e46b9e8dd5a95e2ace0e9e"
```

- **eth.getBalance(eth.accounts[0])**

```
> eth.getBalance(eth.accounts[0])
0
> eth.getBalance(eth.accounts[1])
0
```

- **miner.start() / miner.stop()**

```
> miner.start()
null
> miner.stop()
true
> eth.blockNumber
>
```

- **web3.fromWei(eth.getBalance(eth.accounts[1]), “ether”)**

```
> web3.fromWei(eth.getBalance(eth.coinbase),"ether")
0
>
```

- **personal.unlockAccount(eth.accounts[0])**

```
> personal.unlockAccount(eth.accounts[0])
Unlock account 0x571b970ee70a066f08e46b9e8dd5a95e2ace0e9e
Passphrase:
true
> eth.sendTransaction({from:eth.coinbase,to:eth.accounts[1],value:web3.toWei(10,"ether")})
"0x1483666a1748003966ab3a631e8a5f277efb3e66c3d41b03d8992ee7885c6d47"
```

```
> miner.start()
null
> miner.stop()
true
> eth.blockNumber
25
```

```
> eth.getBalance(eth.accounts[0])
115000000000000000000
```

- **eth.sendTransaction({from: eth.coinbase, to: eth.accounts[1], value: web3.toWei(10,“ether”)})**

```
> eth.getTransaction("0x1483666a1748003966ab3a631e8a5f277efb3e66c3d41b03d8992ee7885c6d47")
{
  blockHash: "0x8db6317b84503973d67e991a5dbc413d3ef62bbce1f99abdb7ed92cb5476ea91",
  blockNumber: 7,
  from: "0x571b970ee70a066f08e46b9e8dd5a95e2ace0e9e",
  gas: 90000,
  gasPrice: 18000000000,
  hash: "0x1483666a1748003966ab3a631e8a5f277efb3e66c3d41b03d8992ee7885c6d47",
  input: "0x",
  nonce: 0,
  r: "0x72fe974dbbc11849db6223a71ad738649e825d51e7722673f11592410329cb52",
  s: "0x58389eec7b9db94c08c429476bea2c91b88aad3bff18882be1495eff73293193",
  to: "0x0dfde928a98cb359c2a299f37b5d4c38bdd7c568",
  transactionIndex: 0,
  v: "0x2576",
  value: 100000000000000000
}
> eth.getBlock("latest").gasLimit
7807027
>
```

```
> web3.fromWei(eth.getBalance(eth.accounts[1]),"ether")
10
>
```

- **GET LATEST BLOCK:**

  Command:

  Eth.getBlock(“latest”)

```
> eth.getBlock("latest")
{
  difficulty: 132352,
  extraData: "0xd783010702846765746885676f312e398777696e646f7773",
  gasLimit: 7648507,
  gasUsed: 0,
  hash: "0x4bb1cc2262436f81589c233613270ec7eded6ed2330175b1ac78a6f4ae84aeaa",
  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  miner: "0x571b970ee70a066f08e46b9e8dd5a95e2ace0e9e",
  mixHash: "0x2d5972cc96ab0542739c08052bda557a0c628c78a49d0193fb61a9402085e2a1",
  nonce: "0x0e6d3d5d4d5abb96",
  number: 46,
  parentHash: "0x0fdc259b35634dd6580ae07efbcfec0958954e84dbcc7a517f2e506b898968ea",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 535,
  stateRoot: "0x4443da6991e61f2fd8c3ee33d24af0802f4b1997d7b23a88e0cd9160b257c361",
  timestamp: 1724869883,
  totalDifficulty: 6050304,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

- **GET SPECFIC BLOCK INFO:**

  Command:

  Eth.getBlock(2)

```
> eth.getBlock(2)
{
  difficulty: 131136,
  extraData: "0xd783010702846765746885676f312e398777696e646f7773",
  gasLimit: 7984386,
  gasUsed: 0,
  hash: "0x9ee5b4c2916ad9b91faf8bd1ad57cd4327298992560e7d8b185fe5078b165375",
  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  miner: "0x571b970ee70a066f08e46b9e8dd5a95e2ace0e9e",
  mixHash: "0x1a8e99f5a3f58b57084b8586e674bb7e3b335951e93f451bd80e5842e800660a",
  nonce: "0x735a624d110b973b",
  number: 2,
  parentHash: "0xfdac5d93f973f6f76470a99e3f4ed7aa9ccbe4052c5b44bad77703265aaa686b",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 535,
  stateRoot: "0x40a24129b975de81492e8597562064a547857902d44e014e6652268d17c07b12",
  timestamp: 1724867964,
  totalDifficulty: 263232,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

Conclusion:

Setting up a private Ethereum network involves installing Geth, creating a genesis block, and initializing the blockchain. This setup allows for mining, sending transactions, and deploying smart contracts in a controlled environment, offering a hands-on way to explore Ethereum's features without using the main network.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 6**	
Title of Lab Assignment: Election Contract		
DOP: 03-09-2024	**DOS: 01-10-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 6

Aim: Develop an Election contract using solidity programming. Create a struct called Candidate, the struct members are ID, name and the vote-count. The smart contract should have the functions like addcandidate, show-candidates, vote, candidatescount and the voters function to verify the status of the casted vote using the Ethereum account address. Further, compile the contract and deploy to the personal Blockchain network using Ganache.

Theory:

Ganache is a personal Ethereum blockchain used for testing and development. It allows developers to deploy and test smart contracts in a local environment without needing to interact with the real Ethereum network. This tool simulates the blockchain by providing test accounts with Ether, faster block times, and a simplified transaction and mining process. Ganache is part of the Truffle Suite, making it easier for developers to create decentralized applications (DApps), test smart contracts, and explore blockchain features before deploying them on a live network. Its key benefit is providing a controlled, local environment where blockchain development can take place without financial risks or delays due to real-world transaction processing.

MetaMask is a cryptocurrency wallet that allows users to interact with the Ethereum blockchain through a browser extension or mobile app. It manages Ethereum-based assets and provides a user-friendly interface for sending and receiving Ether (ETH) and other Ethereum-based tokens. MetaMask also functions as a bridge between decentralized applications (DApps) and the Ethereum network, enabling users to sign transactions, approve smart contract interactions, and manage multiple Ethereum accounts. It is commonly used in blockchain development because it connects easily to both live Ethereum networks and local blockchains, such as those set up by Ganache. MetaMask plays a key role in Web3 development by enabling users to interact securely with decentralized platforms through their private wallets.

Smart contracts are self-executing programs on a blockchain that automatically enforce agreements or perform actions when predefined conditions are met. Written primarily in the Solidity programming language for Ethereum, smart contracts are used to handle transactions, manage digital assets, and perform automated functions without the need for intermediaries. These contracts are immutable once deployed, meaning that their code cannot be changed, ensuring the rules and logic will always operate as initially defined. For

example, smart contracts can be used for a wide range of applications, such as voting systems, decentralized finance (DeFi), supply chain management, and more. They reduce the need for trust between parties by relying on the blockchain's transparent and secure infrastructure.

The **Web3 environment** refers to the decentralized internet where users interact directly with blockchain-based applications (DApps) and smart contracts. Unlike traditional web applications (Web2), which rely on centralized servers, Web3 enables peer-to-peer interactions on decentralized networks. MetaMask is a key component of this Web3 environment because it acts as a wallet and gateway to interact with these decentralized platforms. By injecting Ethereum accounts directly into the browser, MetaMask allows users to sign transactions, approve smart contracts, and manage their digital assets in a decentralized manner. It is widely used in the Ethereum ecosystem for connecting users to blockchain applications in a secure, non-custodial manner, meaning the user controls their private keys and assets.

Remix IDE is a browser-based integrated development environment (IDE) for writing, compiling, deploying, and testing Ethereum smart contracts. It is one of the most popular tools for developing smart contracts using Solidity, Ethereum's programming language. Remix offers a user-friendly interface with a set of powerful features such as syntax highlighting, debugging, and contract analysis. One of its key advantages is that it does not require installation, making it accessible directly from the browser. Remix also integrates with tools like MetaMask and personal blockchains (like Ganache) for deploying and interacting with contracts in both local and real Ethereum environments. It is an essential tool for Ethereum developers, particularly for learning, prototyping, and testing smart contract functionality.

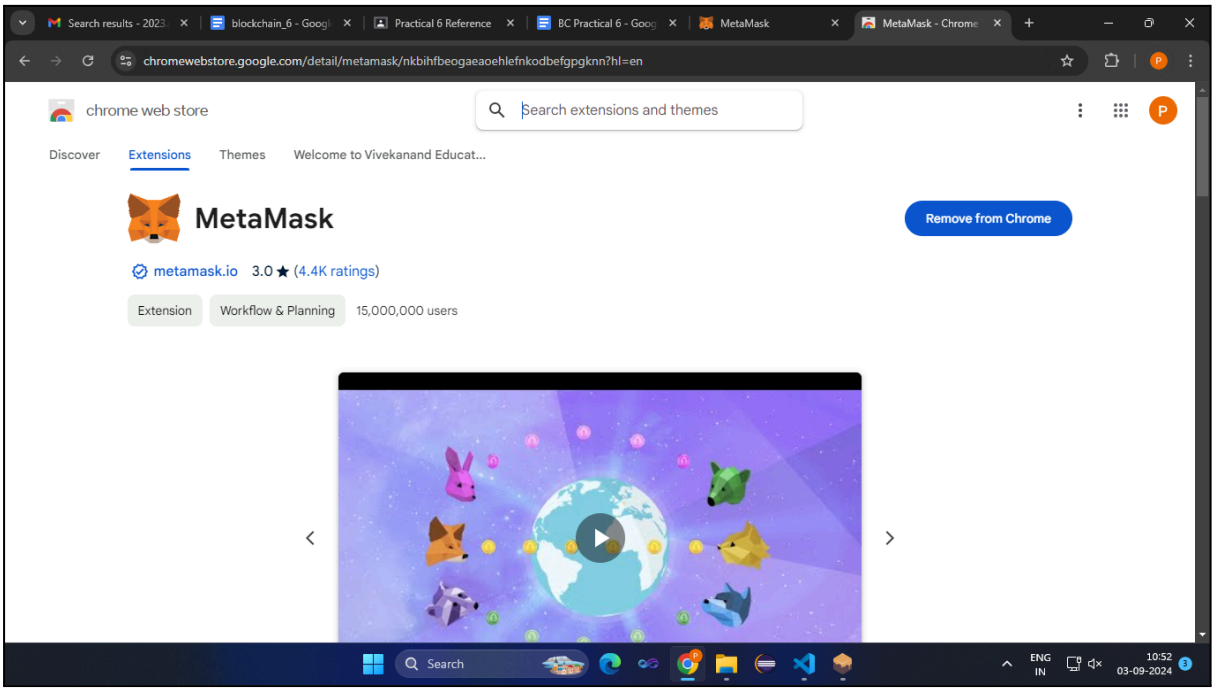
Code:

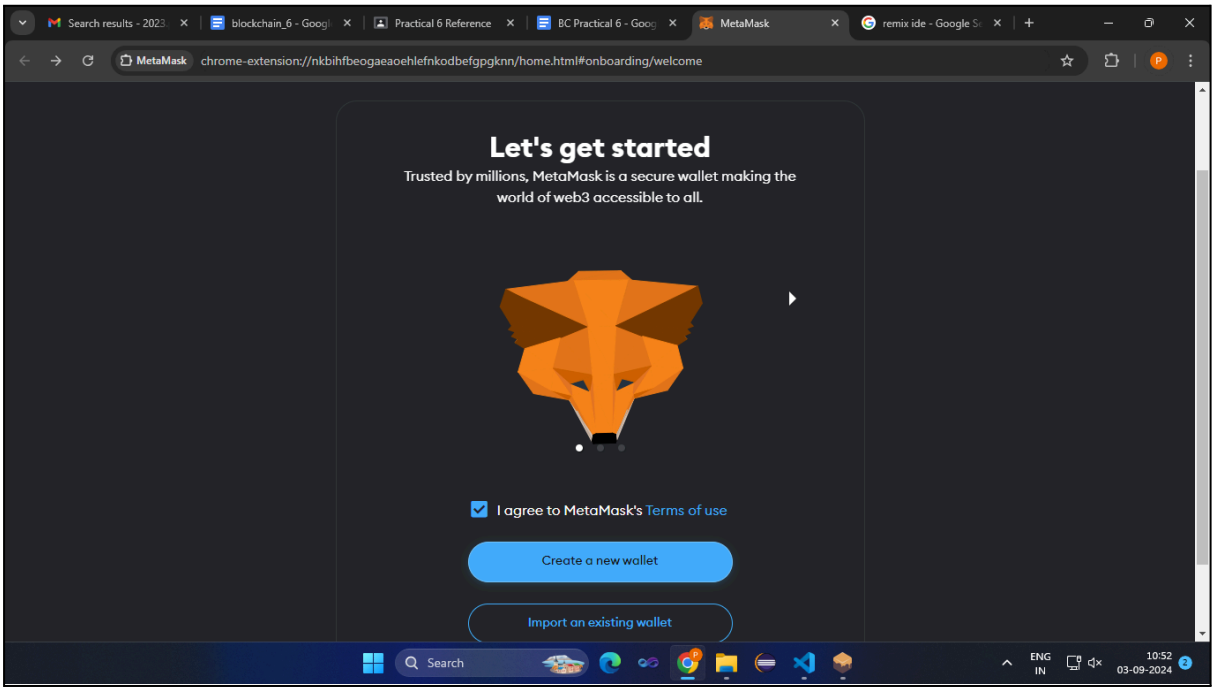
```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Election {
    // Model a Candidate
    struct Candidate {
        //three propertiees for candidates
        uint id;
        string name;
```

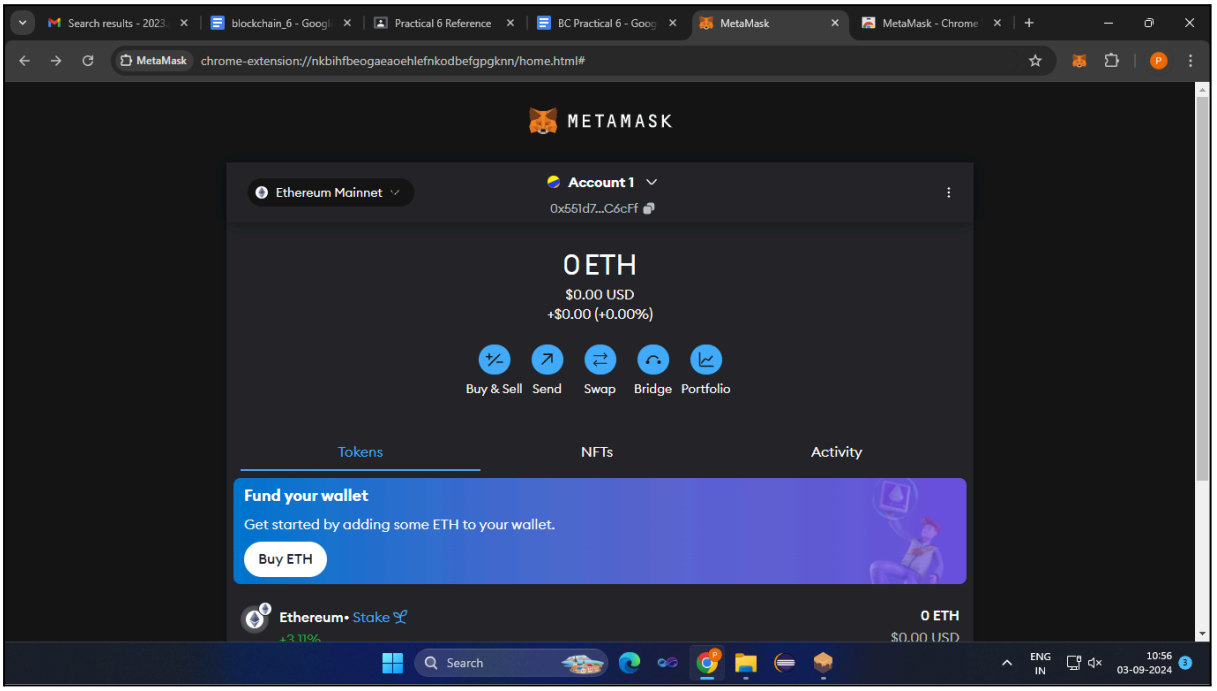
```
        uint voteCount;
    }

    // Store accounts that have voted
    //voters list
     mapping(address => bool) public voters; //here address is account address, each user will have unique and single address
    // Store Candidates, Fetch Candidate, candidate list
    mapping(uint => Candidate) public candidates;
    // Store Candidates Count
    uint public candidatesCount;

    // voted event, will fetch the candidate id whom you have voted
    event votedEvent (
        uint indexed _candidateId
    );
    constructor()  {
    addCandidate("N MODI, BJP");
    addCandidate("A kejriwal, AAP");
    addCandidate("Rahul G, Congress");
    addCandidate("Nikhil, JDS");
    }

    function addCandidate (string memory _name) private {
    candidatesCount ++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    }
    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;
    // update candidate vote Count
    candidates[_candidateId].voteCount ++;
```

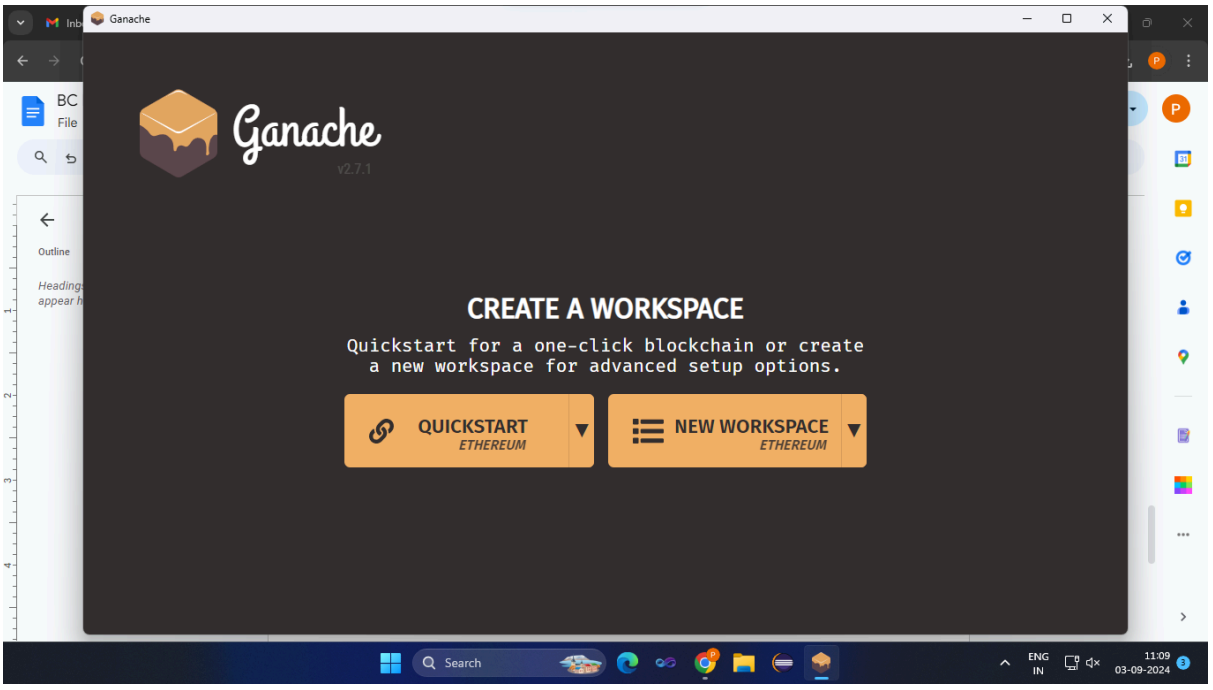
```
    // trigger voted event emit
    emit votedEvent(_candidateId);
    }
}
```

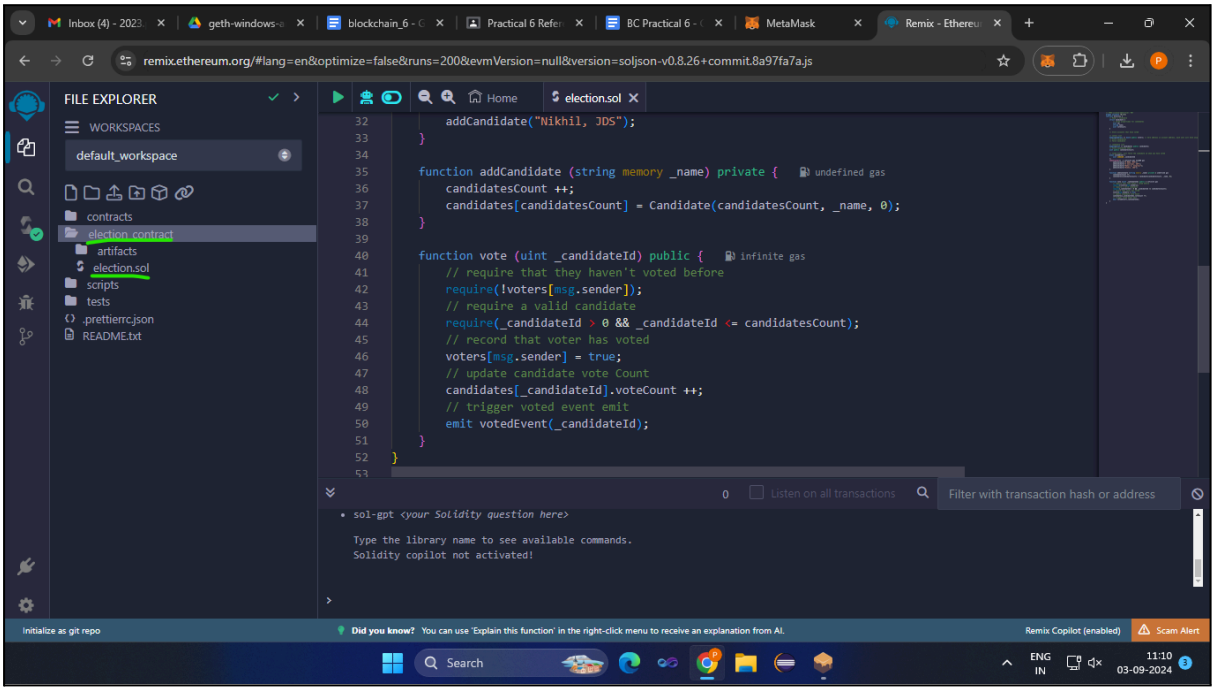
Output:

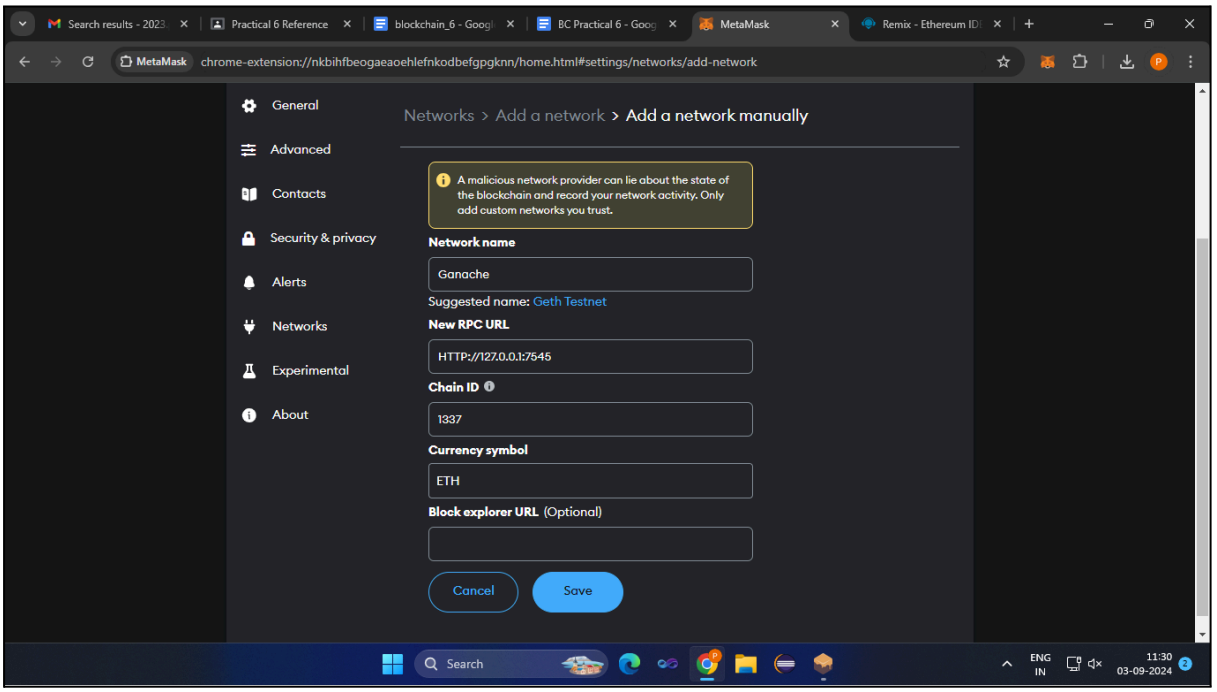


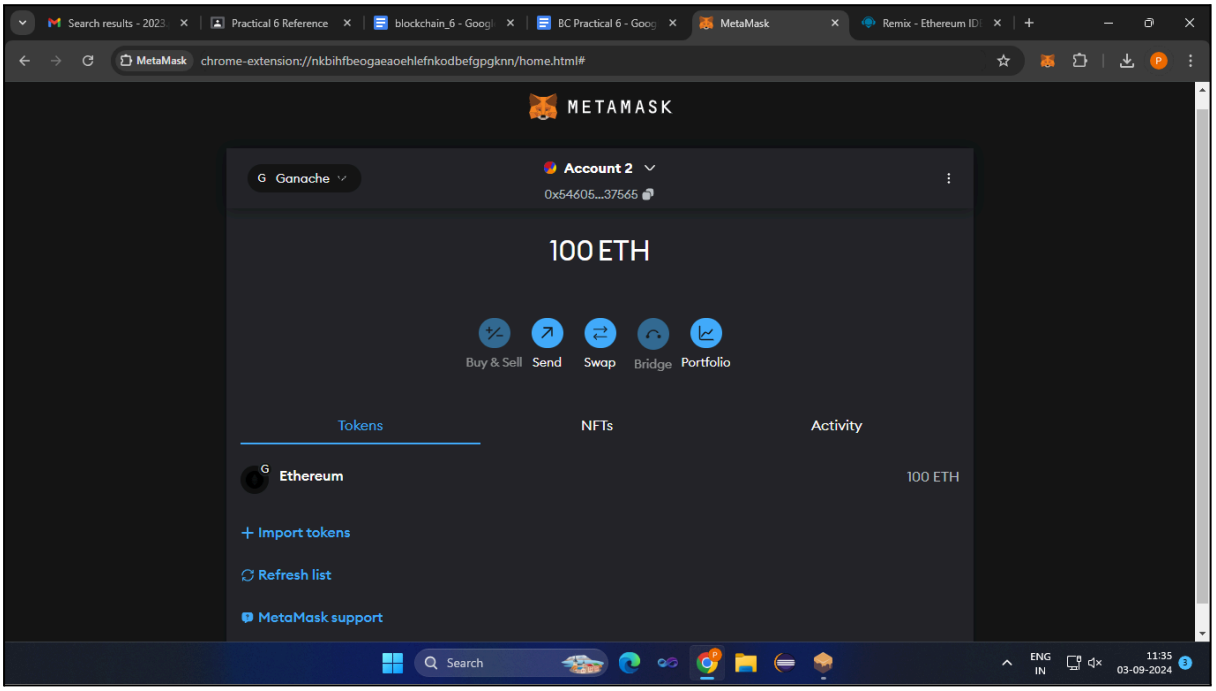


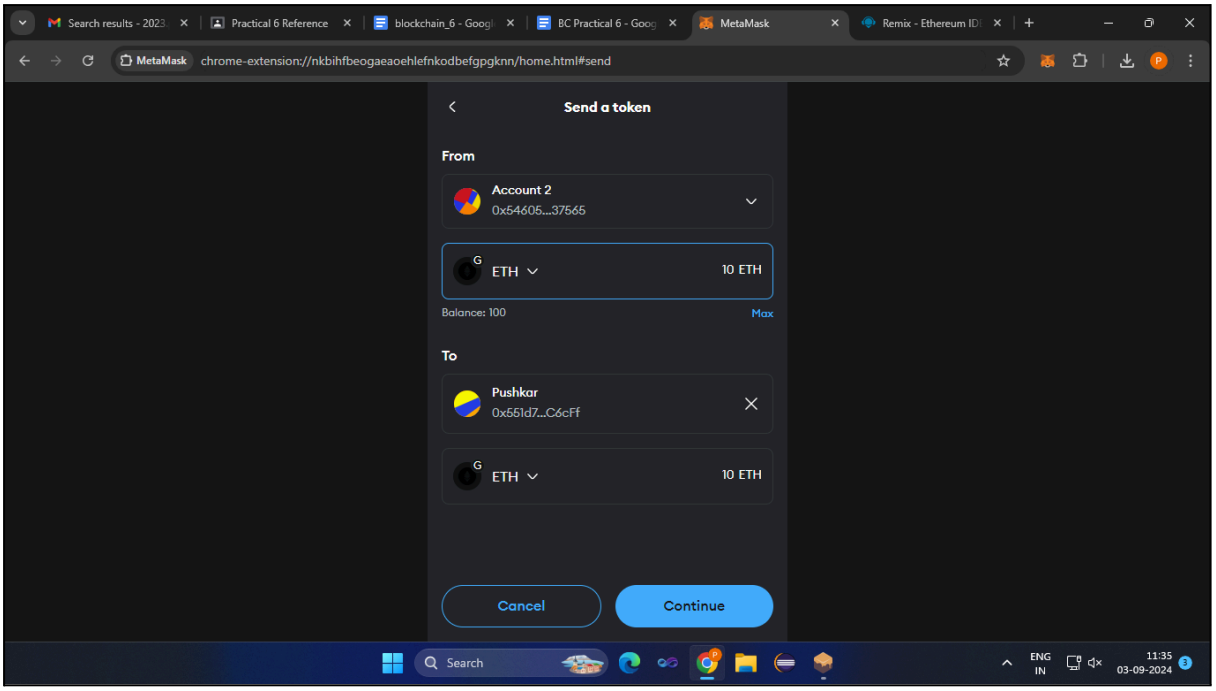

METAMASK
Ethereum Mainnet
Account 1
0x551d7...C6cFf
0 ETH
$0.00 USD
+$0.00 (+0.00%)
Buy & Sell
Send
Swap
Bridge
Portfolio
Tokens
NFTs
Activity
Fund your wallet
Get started by adding some ETH to your wallet.
Buy ETH
Ethereum • Stake
0 ETH
$0.00 USD

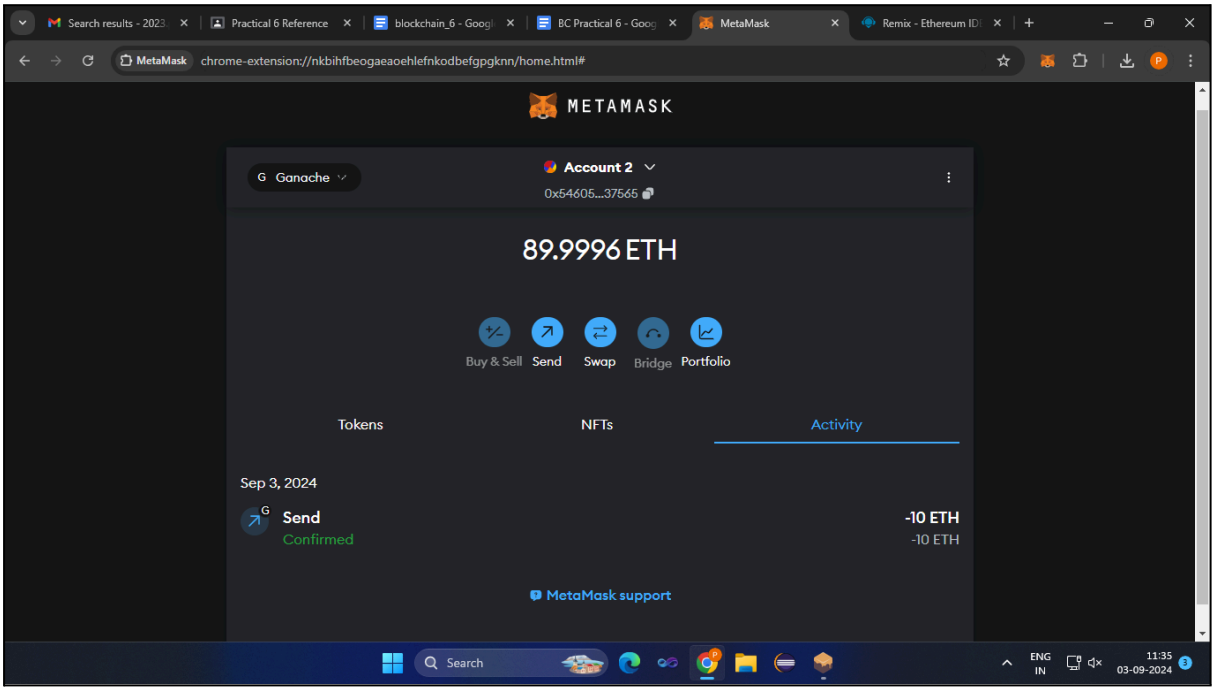

Ganache
v2.7.1
CREATE A WORKSPACE
Quickstart for a one-click blockchain or create
a new workspace for advanced setup options.
QUICKSTART
ETHEREUM
NEW WORKSPACE
ETHEREUM

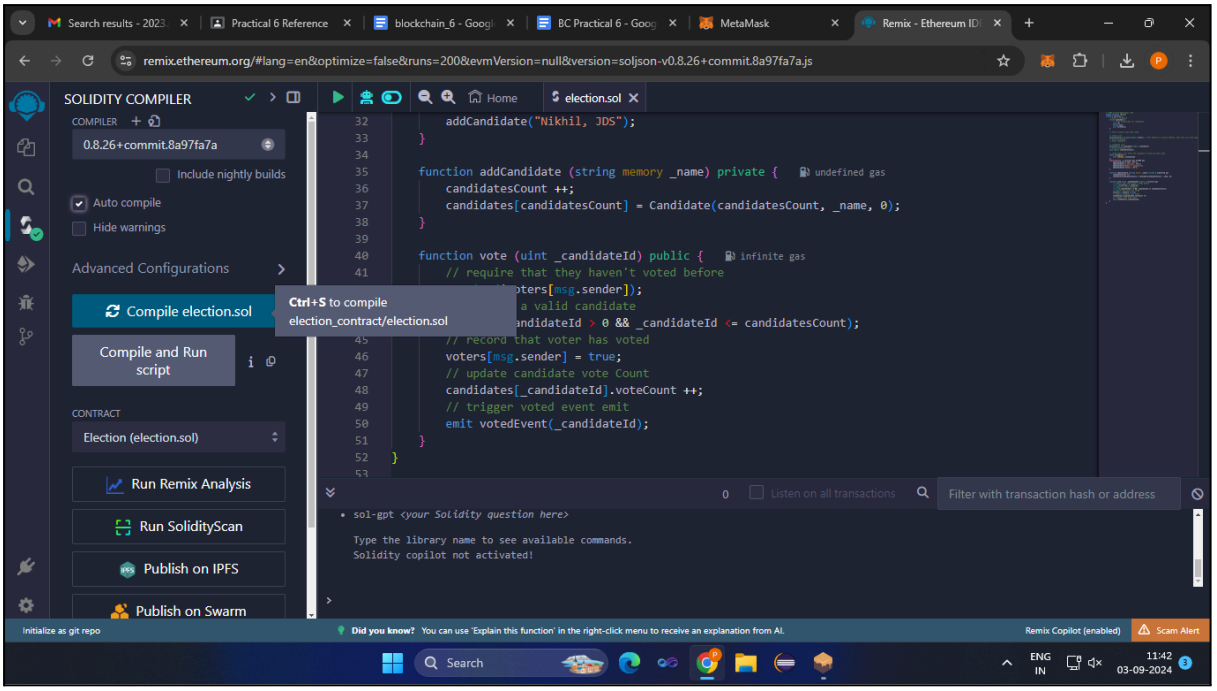

FILE EXPLORER
WORKSPACES
default_workspace
contracts
election_contract
artifacts
election.sol
scripts
tests
.prettierrc.json
README.txt
election.sol
addCandidate("Nikhil, JDS");
}
function addCandidate (string memory _name) private {
candidatesCount ++;
candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}
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;
// update candidate vote Count
candidates[_candidateId].voteCount ++;
// trigger voted event emit
emit votedEvent(_candidateId);
}
}
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Solidity copilot not activated!
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert

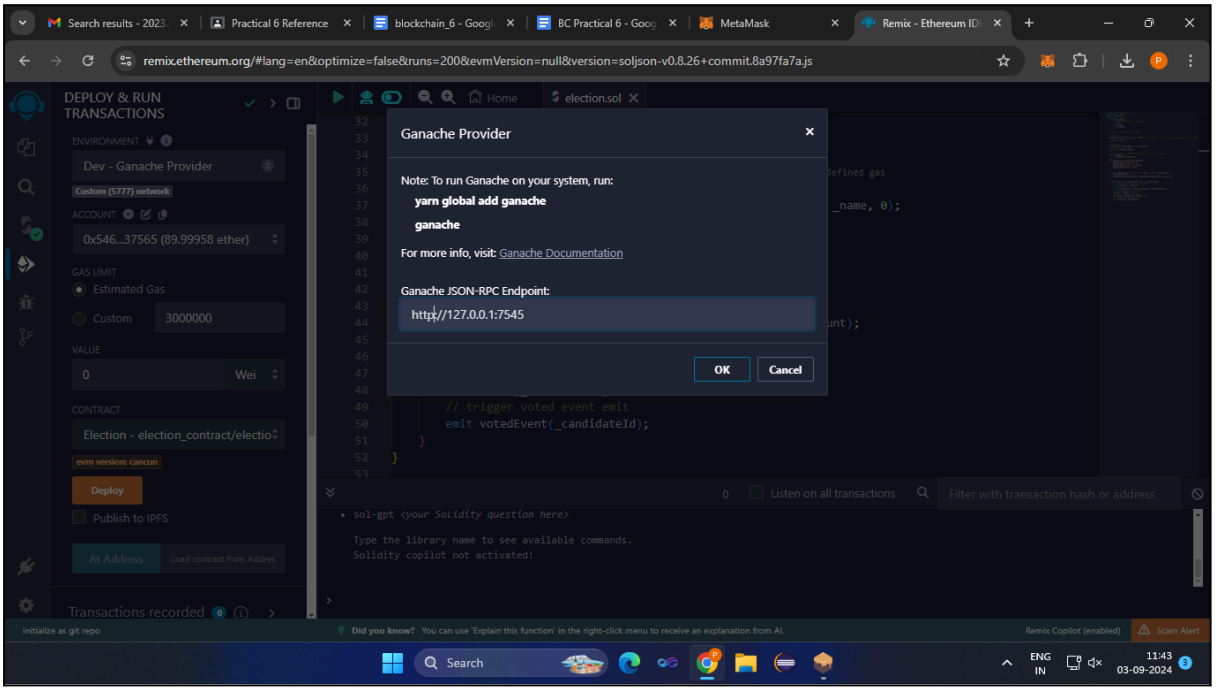

chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#settings/networks/add-network
General
Advanced
Contacts
Security & privacy
Alerts
Networks
Experimental
About
Networks > Add a network > Add a network manually
A malicious network provider can lie about the state of the blockchain and record your network activity. Only add custom networks you trust.
Network name
Ganache
Suggested name: Geth Testnet
New RPC URL
HTTP://127.0.0.1:7545
Chain ID
1337
Currency symbol
ETH
Block explorer URL (Optional)
Cancel
Save

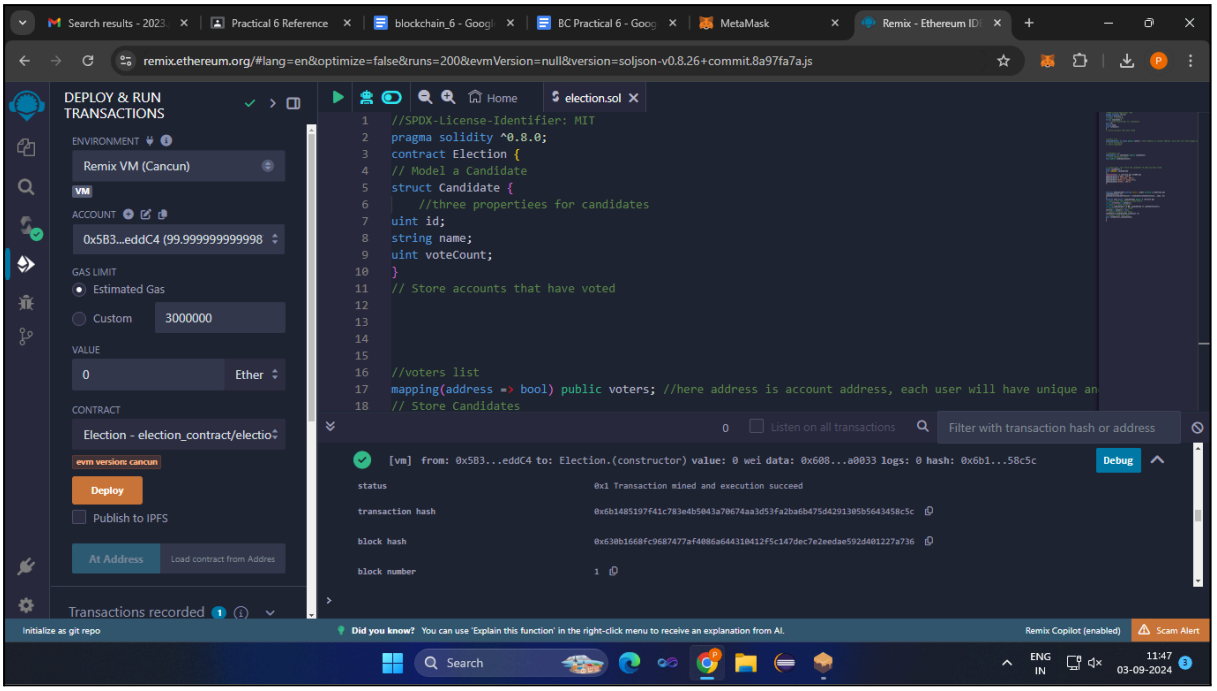

METAMASK
Ganache
Account 2
0x54605...37565
100 ETH
Buy & Sell
Send
Swap
Bridge
Portfolio
Tokens
NFTs
Activity
Ethereum
100 ETH
Import tokens
Refresh list
MetaMask support

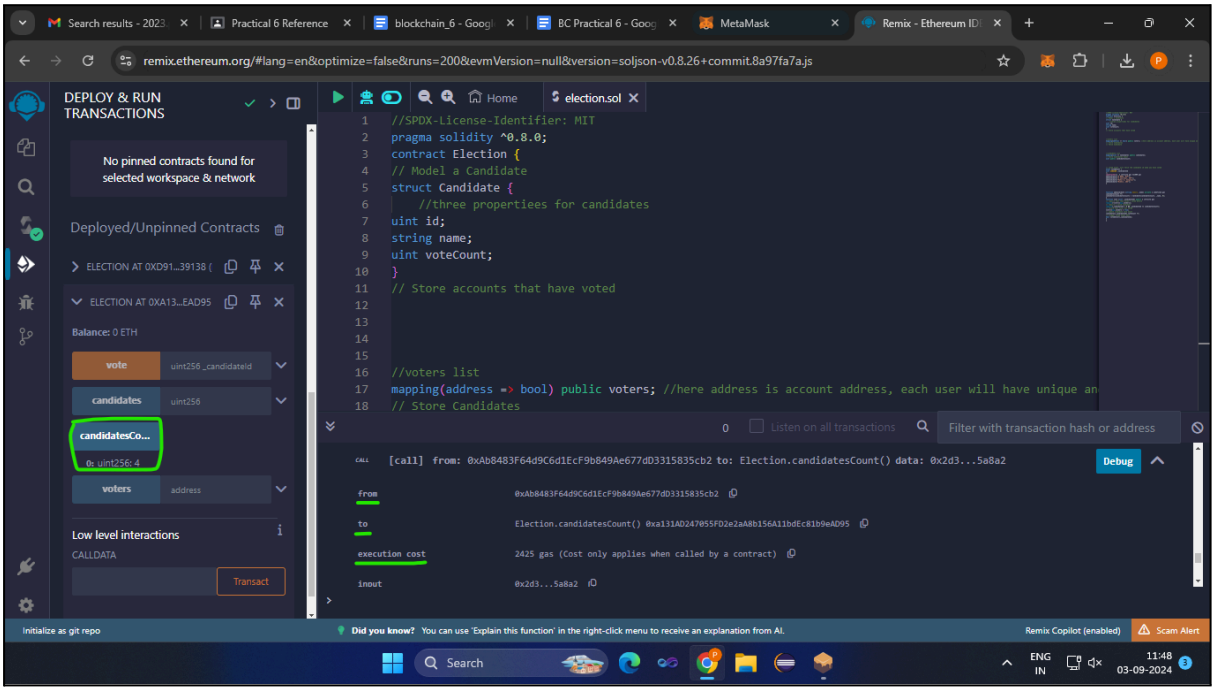

Send a token
From
Account 2
0x54605...37565
ETH
10 ETH
Balance: 100
Max
To
Pushkar
0x551d7...C6cFf
ETH
10 ETH
Cancel
Continue

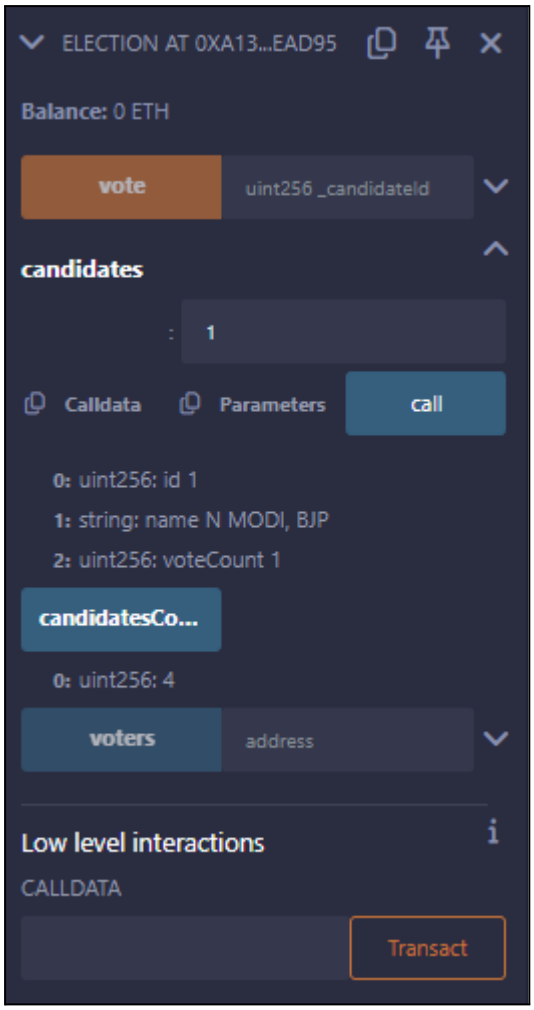

METAMASK
Ganache
Account 2
0x54605...37565
89.9996 ETH
Buy & Sell
Send
Swap
Bridge
Portfolio
Tokens
NFTs
Activity
Sep 3, 2024
Send
Confirmed
-10 ETH
-10 ETH
MetaMask support

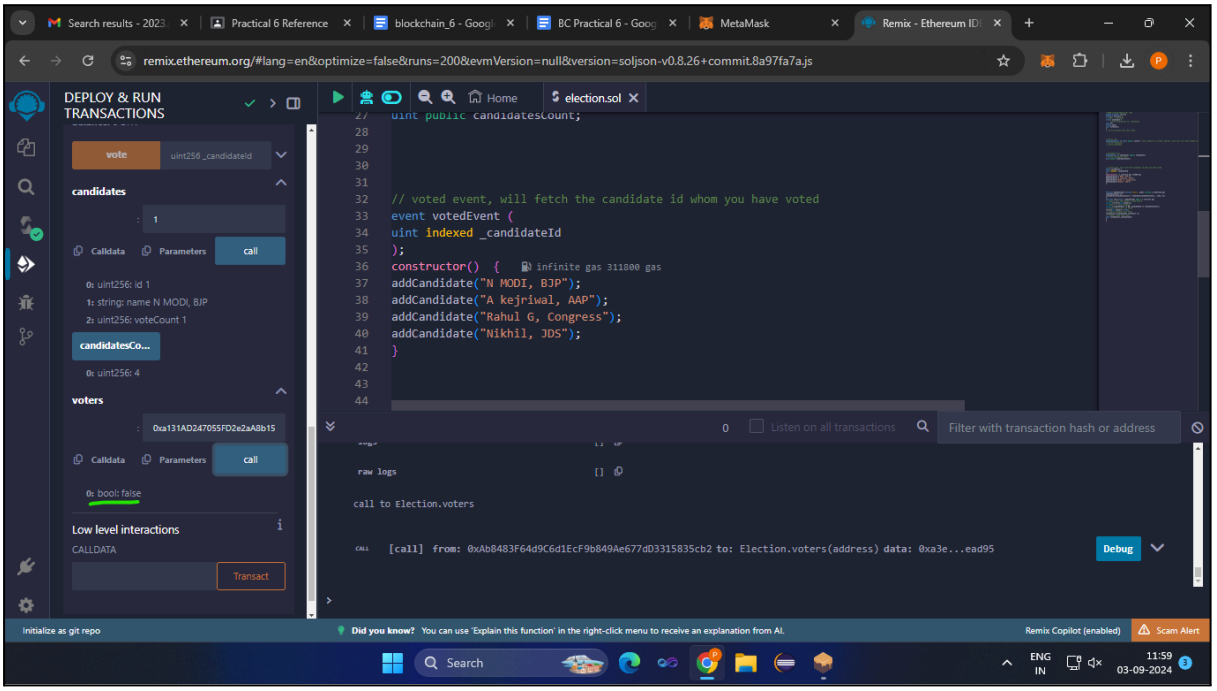

SOLIDITY COMPILER
0.8.26+commit.8a97fa7a
Auto compile
Hide warnings
Advanced Configurations
Compile election.sol
Ctrl+S to compile
election_contract/election.sol
Compile and Run script
CONTRACT
Election (election.sol)
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
election.sol
addCandidate("Nikhil, JDS");
function addCandidate (string memory _name) private {
candidatesCount ++;
candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
function vote (uint _candidateId) public {
// require that they haven't voted before
// record that voter has voted
voters[msg.sender] = true;
// update candidate vote Count
candidates[_candidateId].voteCount ++;
// trigger voted event emit
emit votedEvent(_candidateId);
sol-gpt <your Solidity question here>
Type the library name to see available commands.
Solidity copilot not activated!

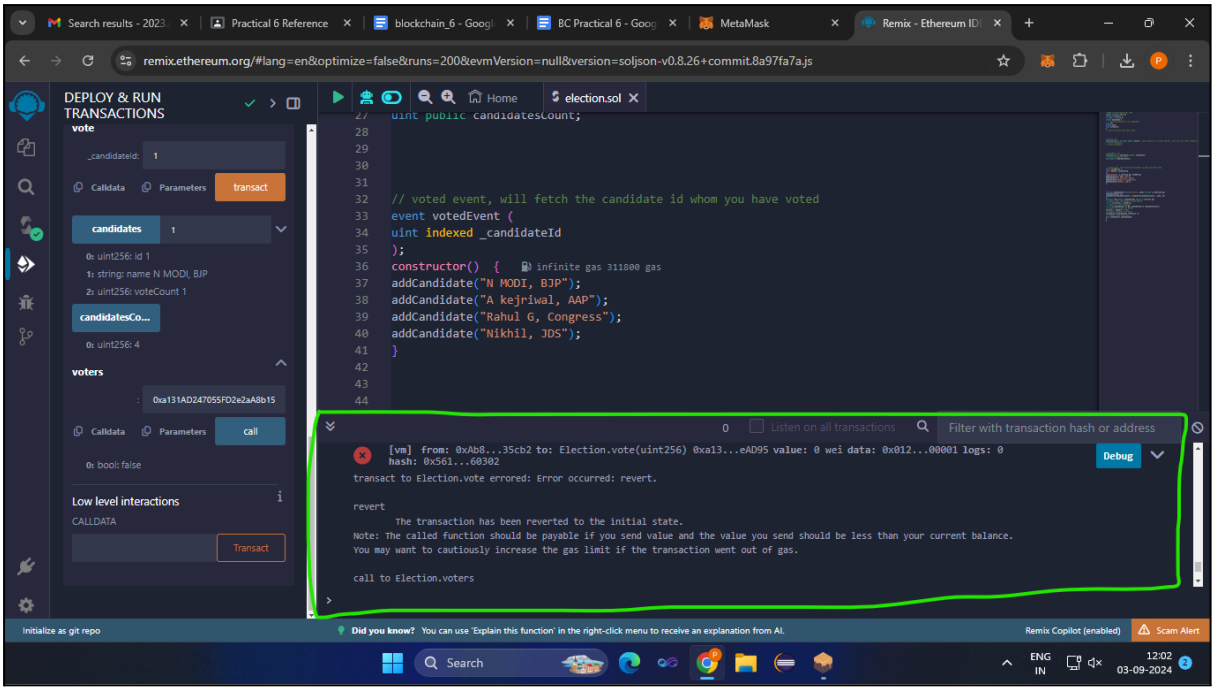

Ganache Provider
Note: To run Ganache on your system, run:
yarn global add ganache
ganache
For more info, visit: Ganache Documentation
Ganache JSON-RPC Endpoint:
http://127.0.0.1:7545
OK
Cancel
DEPLOY & RUN TRANSACTIONS
Dev - Ganache Provider
election.sol

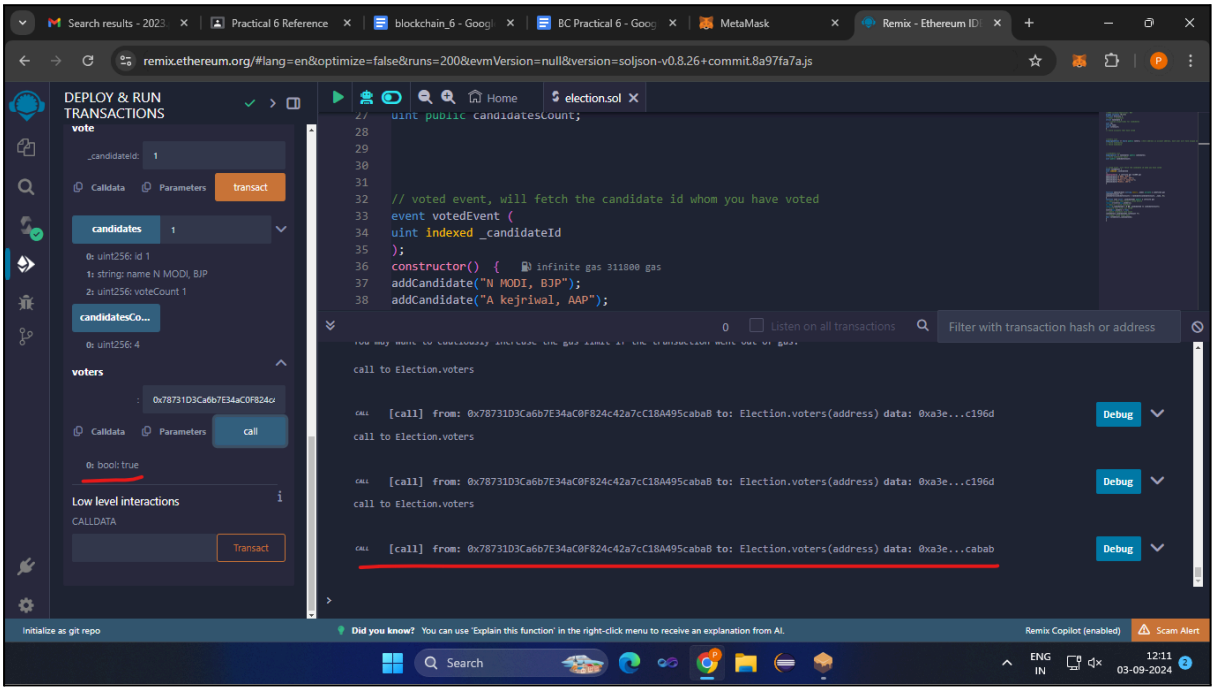

DEPLOY & RUN TRANSACTIONS
Remix VM (Cancun)
election.sol
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Election {
// Model a Candidate
struct Candidate {
//three propertiees for candidates
uint id;
string name;
uint voteCount;
}
// Store accounts that have voted
//voters list
mapping(address => bool) public voters; //here address is account address, each user will have unique an
// Store Candidates
[vm] from: 0x5B3...eddC4 to: Election.(constructor) value: 0 wei data: 0x608...a0033 logs: 0 hash: 0x6b1...58c5c
Debug
status 0x1 Transaction mined and execution succeed
transaction hash 0x6b1485197f41c783e4b5043a70674aa3d53fa2ba6b475d4291305b5643458c5c
block hash 0x630b1668fc9687477af4086a644310412f5c147dec7e2eedae592d401227a736
block number 1

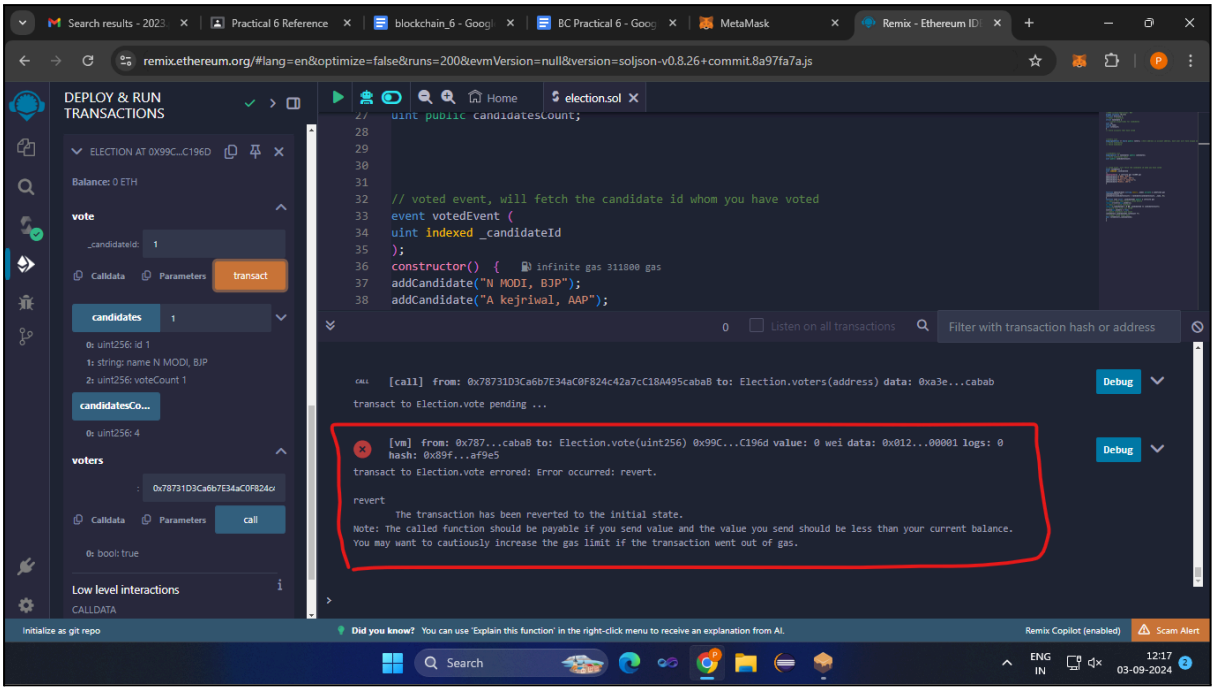

DEPLOY & RUN TRANSACTIONS
No pinned contracts found for selected workspace & network
Deployed/Unpinned Contracts
ELECTION AT 0XA13...EAD95
Balance: 0 ETH
vote
uint256 _candidateId
candidates
uint256
candidatesCo...
0: uint256: 4
voters
address
Low level interactions
CALLDATA
Transact
election.sol
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Election {
// Model a Candidate
struct Candidate {
//three properties for candidates
uint id;
string name;
uint voteCount;
}
// Store accounts that have voted
//voters list
mapping(address => bool) public voters; //here address is account address, each user will have unique an
// Store Candidates
Filter with transaction hash or address
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: Election.candidatesCount() data: 0x2d3...5a8a2
Debug
from
to
execution cost
2425 gas (Cost only applies when called by a contract)
input
0x2d3...5a8a2
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert

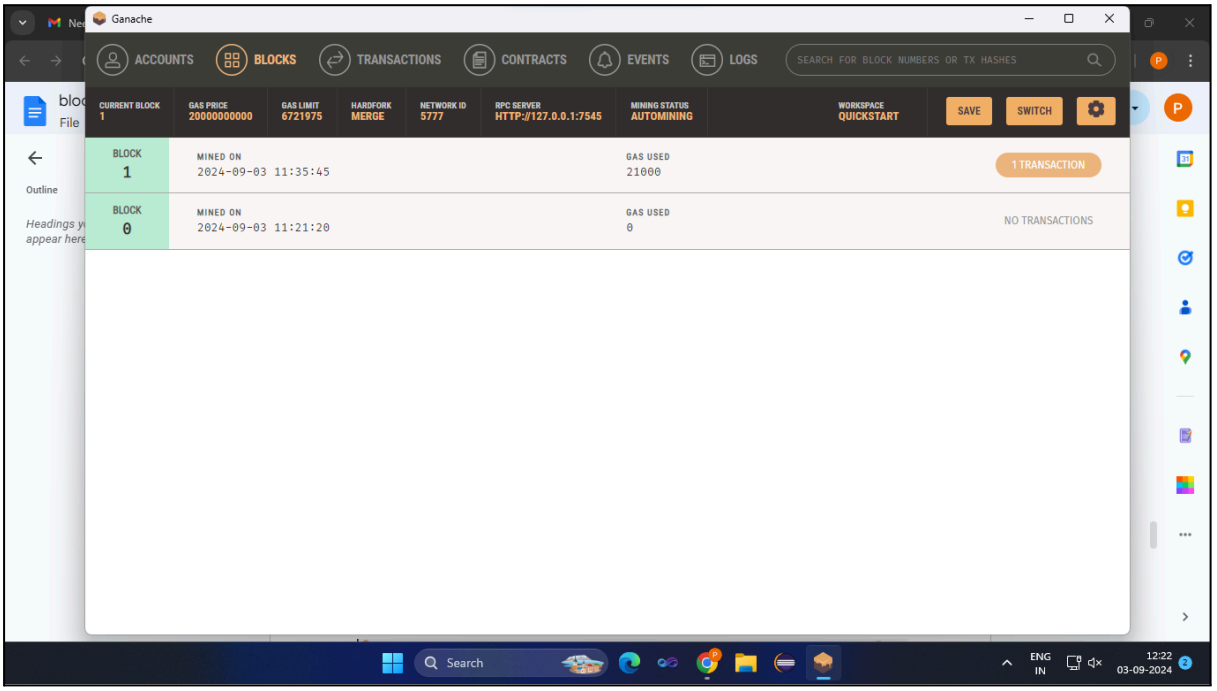

ELECTION AT 0XA13...EAD95
Balance: 0 ETH
vote
uint256 _candidateId
candidates
1
Calldata
Parameters
call
0: uint256: id 1
1: string: name N MODI, BJP
2: uint256: voteCount 1
candidatesCo...
0: uint256: 4
voters
address
Low level interactions
CALLDATA
Transact

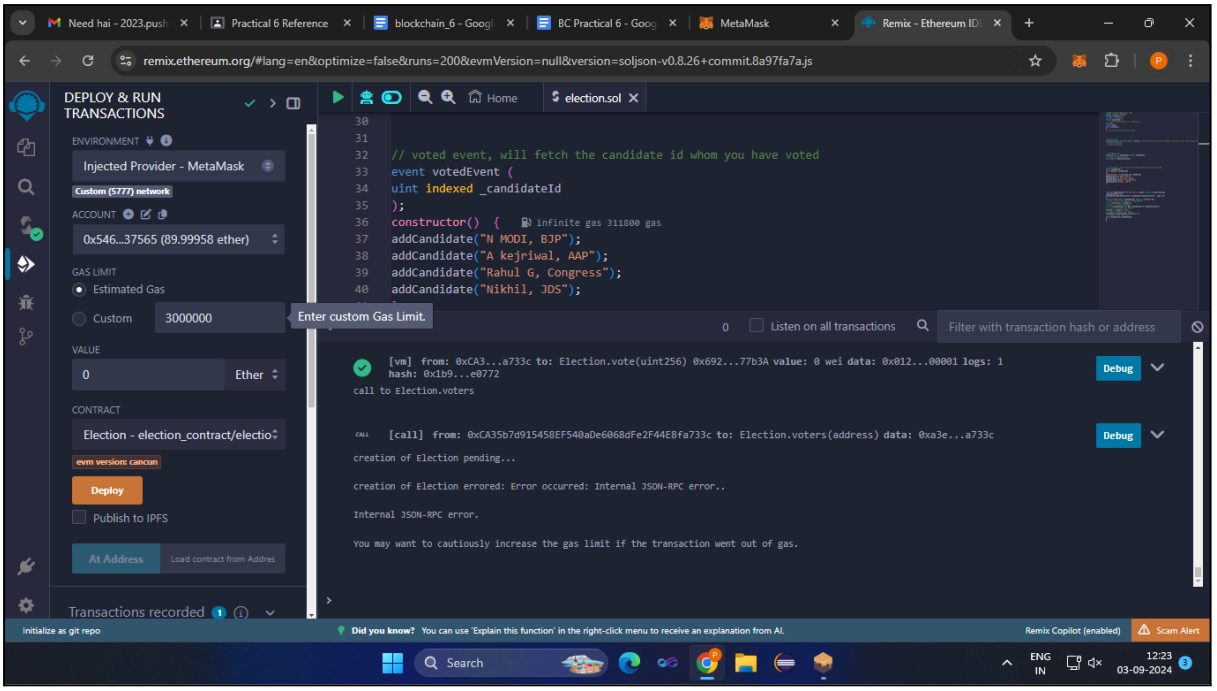

DEPLOY & RUN TRANSACTIONS
vote
uint256 _candidateId
candidates
1
Calldata
Parameters
call
0: uint256: id 1
1: string: name N MODI, BJP
2: uint256: voteCount 1
candidatesCo...
0: uint256: 4
voters
0xa131AD247055FD2e2aA8b15
Calldata
Parameters
call
0: bool: false
Low level interactions
CALLDATA
Transact
election.sol
// voted event, will fetch the candidate id whom you have voted
event votedEvent (
uint indexed _candidateId
);
constructor() { infinite gas 311800 gas
addCandidate("N MODI, BJP");
addCandidate("A kejriwal, AAP");
addCandidate("Rahul G, Congress");
addCandidate("Nikhil, JDS");
}
Listen on all transactions
Filter with transaction hash or address
raw logs
call to Election.voters
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: Election.voters(address) data: 0xa3e...ead95
Debug
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert
Search
ENG IN
11:59
03-09-2024

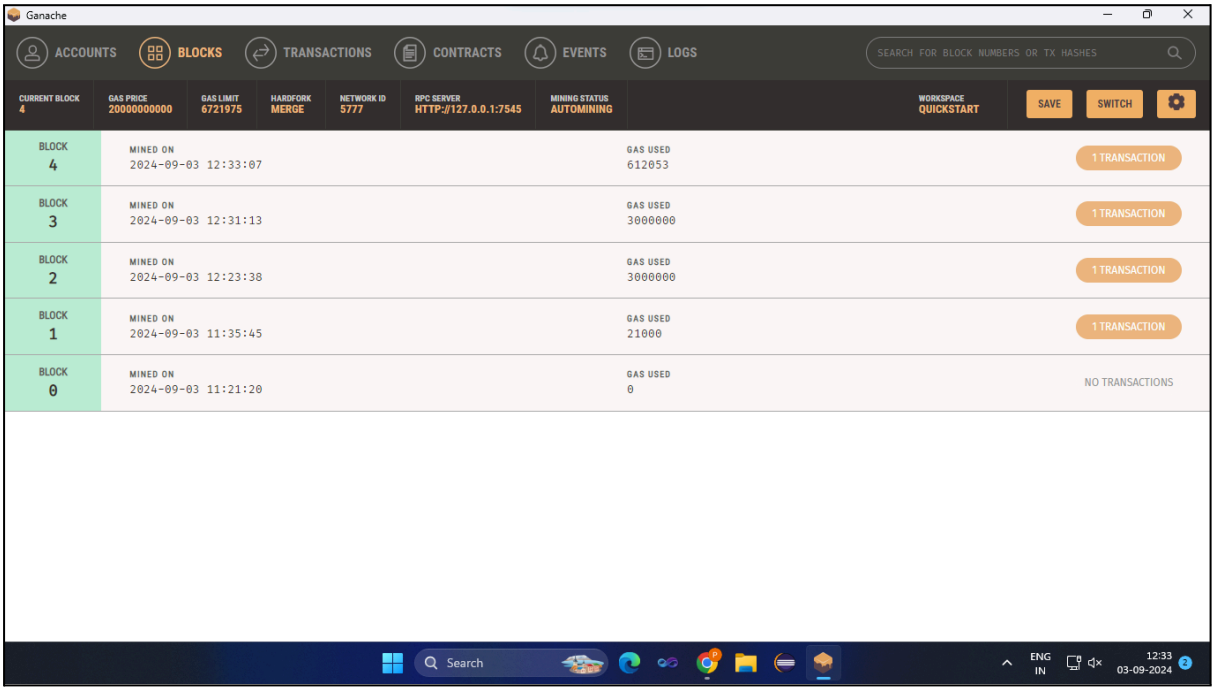

DEPLOY & RUN TRANSACTIONS
vote
_candidateId: 1
Calldata
Parameters
transact
candidates
1
0: uint256: id 1
1: string: name N MODI, BJP
2: uint256: voteCount 1
candidatesCo...
0: uint256: 4
voters
0xa131AD247055FD2e2aA8b15
Calldata
Parameters
call
0: bool: false
Low level interactions
CALLDATA
Transact
election.sol
// voted event, will fetch the candidate id whom you have voted
event votedEvent (
uint indexed _candidateId
);
constructor() { infinite gas 311800 gas
addCandidate("N MODI, BJP");
addCandidate("A kejriwal, AAP");
addCandidate("Rahul G, Congress");
addCandidate("Nikhil, JDS");
}
Listen on all transactions
Filter with transaction hash or address
[vm] from: 0xAb8...35cb2 to: Election.vote(uint256) 0xa13...eAD95 value: 0 wei data: 0x012...00001 logs: 0
hash: 0x561...60302
Debug
transact to Election.vote errored: Error occurred: revert.
revert
The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
You may want to cautiously increase the gas limit if the transaction went out of gas.
call to Election.voters
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert
Search
ENG IN
12:02
03-09-2024

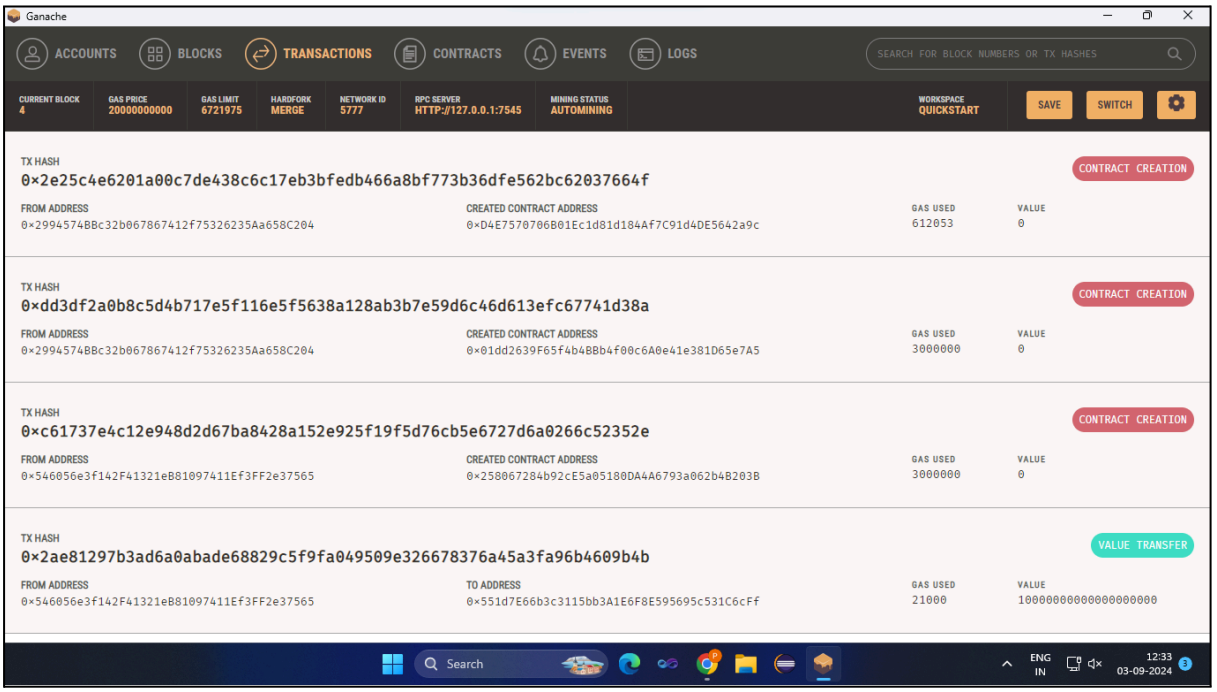

DEPLOY & RUN TRANSACTIONS
vote
_candidateId: 1
Calldata Parameters transact
candidates 1
0: uint256: id 1
1: string: name N MODI, BJP
2: uint256: voteCount 1
candidatesCo...
0: uint256: 4
voters
0x78731D3Ca6b7E34aC0F824c
Calldata Parameters call
0: bool: true
Low level interactions
CALLDATA
Transact
election.sol
// voted event, will fetch the candidate id whom you have voted
event votedEvent (
uint indexed _candidateId
);
constructor() { infinite gas 311800 gas
addCandidate("N MODI, BJP");
addCandidate("A kejriwal, AAP");
Filter with transaction hash or address
call to Election.voters
[call] from: 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB to: Election.voters(address) data: 0xa3e...c196d
Debug
call to Election.voters
[call] from: 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB to: Election.voters(address) data: 0xa3e...c196d
Debug
call to Election.voters
[call] from: 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB to: Election.voters(address) data: 0xa3e...cabab
Debug
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert
Search
ENG IN
12:11 03-09-2024

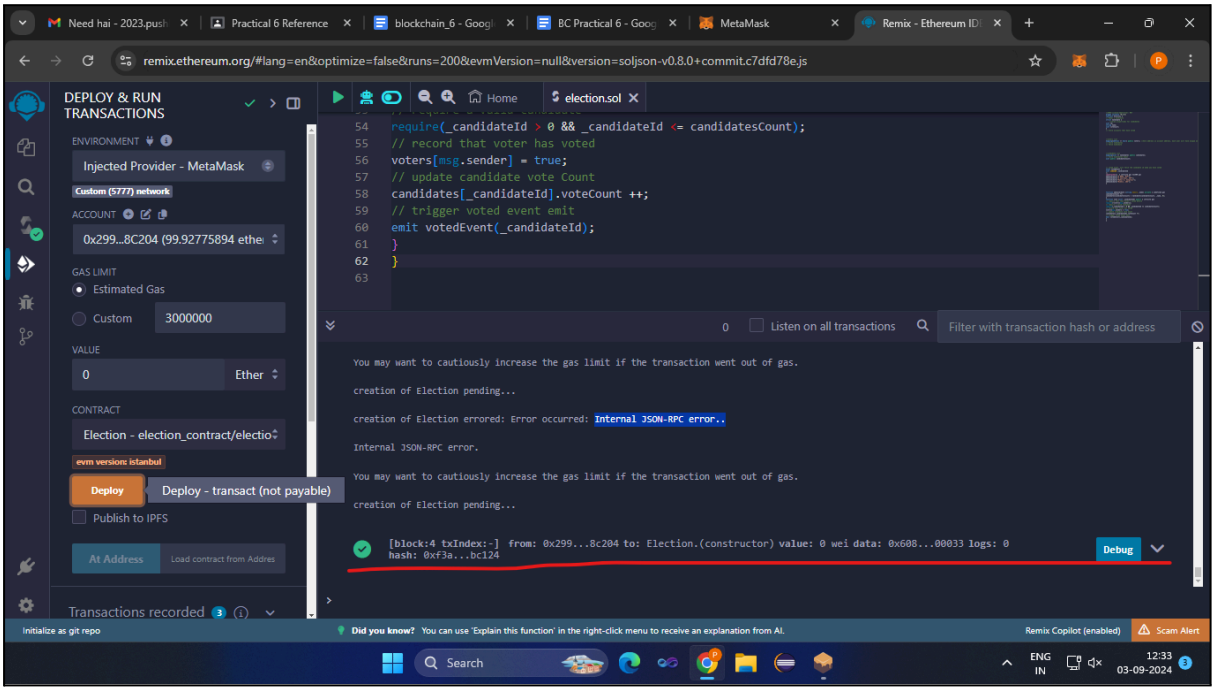

DEPLOY & RUN TRANSACTIONS
ELECTION AT 0X99C...C196D
Balance: 0 ETH
vote
_candidateId: 1
Calldata Parameters transact
candidates 1
0: uint256: id 1
1: string: name N MODI, BJP
2: uint256: voteCount 1
candidatesCo...
0: uint256: 4
voters
0x78731D3Ca6b7E34aC0F824c
Calldata Parameters call
0: bool: true
Low level interactions
CALLDATA
election.sol
// voted event, will fetch the candidate id whom you have voted
event votedEvent (
uint indexed _candidateId
);
constructor() { infinite gas 311800 gas
addCandidate("N MODI, BJP");
addCandidate("A kejriwal, AAP");
Filter with transaction hash or address
[call] from: 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB to: Election.voters(address) data: 0xa3e...cabab
Debug
transact to Election.vote pending ...
[vm] from: 0x787...cabaB to: Election.vote(uint256) 0x99C...C196d value: 0 wei data: 0x012...00001 logs: 0 hash: 0x89f...af9e5
Debug
transact to Election.vote errored: Error occurred: revert.
revert
The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
You may want to cautiously increase the gas limit if the transaction went out of gas.
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert
Search
ENG IN
12:17 03-09-2024


Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 1
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MERGE
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE QUICKSTART
SAVE
SWITCH
BLOCK 1
MINED ON 2024-09-03 11:35:45
GAS USED 21000
1 TRANSACTION
BLOCK 0
MINED ON 2024-09-03 11:21:20
GAS USED 0
NO TRANSACTIONS
Search
ENG IN
12:22 03-09-2024


remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.26+commit.8a97fa7a.js
DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Injected Provider - MetaMask
Custom (5777) network
ACCOUNT
0x546...37565 (89.99958 ether)
GAS LIMIT
Estimated Gas
Custom
3000000
Enter custom Gas Limit.
VALUE
0
Ether
CONTRACT
Election - election_contract/electio
evm version: cancun
Deploy
Publish to IPFS
At Address
Load contract from Addres
Transactions recorded
election.sol
// voted event, will fetch the candidate id whom you have voted
event votedEvent (
uint indexed _candidateId
);
constructor() { infinite gas 311800 gas
addCandidate("N MODI, BJP");
addCandidate("A kejriwal, AAP");
addCandidate("Rahul G, Congress");
addCandidate("Nikhil, JDS");
Listen on all transactions
Filter with transaction hash or address
[vm] from: 0xCA3...a733c to: Election.vote(uint256) 0x692...77b3A value: 0 wei data: 0x012...00001 logs: 1
hash: 0x1b9...e0772
call to Election.voters
Debug
[call] from: 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c to: Election.voters(address) data: 0xa3e...a733c
creation of Election pending...
creation of Election errored: Error occurred: Internal JSON-RPC error..
Internal JSON-RPC error.
You may want to cautiously increase the gas limit if the transaction went out of gas.
Initialize as git repo
Did you know? You can use 'Explain this function' in the right-click menu to receive an explanation from AI.
Remix Copilot (enabled)
Scam Alert
Search
ENG IN
12:23 03-09-2024


Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 4
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MERGE
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE QUICKSTART
SAVE
SWITCH
BLOCK 4 MINED ON 2024-09-03 12:33:07 GAS USED 612053 1 TRANSACTION
BLOCK 3 MINED ON 2024-09-03 12:31:13 GAS USED 3000000 1 TRANSACTION
BLOCK 2 MINED ON 2024-09-03 12:23:38 GAS USED 3000000 1 TRANSACTION
BLOCK 1 MINED ON 2024-09-03 11:35:45 GAS USED 21000 1 TRANSACTION
BLOCK 0 MINED ON 2024-09-03 11:21:20 GAS USED 0 NO TRANSACTIONS
Search
ENG IN
12:33 03-09-2024


Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 4
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MERGE
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545
MINING STATUS AUTOMINING
WORKSPACE QUICKSTART
SAVE
SWITCH
TX HASH 0x2e25c4e6201a00c7de438c6c17eb3bfedb466a8bf773b36dfe562bc62037664f CONTRACT CREATION
FROM ADDRESS 0x2994574BBc32b067867412f75326235Aa658C204
CREATED CONTRACT ADDRESS 0xD4E7570706B01Ec1d81d184Af7C91d4DE5642a9c
GAS USED 612053
VALUE 0
TX HASH 0xdd3df2a0b8c5d4b717e5f116e5f5638a128ab3b7e59d6c46d613efc67741d38a CONTRACT CREATION
FROM ADDRESS 0x2994574BBc32b067867412f75326235Aa658C204
CREATED CONTRACT ADDRESS 0x01dd2639F65f4b4BBb4f00c6A0e41e381D65e7A5
GAS USED 3000000
VALUE 0
TX HASH 0xc61737e4c12e948d2d67ba8428a152e925f19f5d76cb5e6727d6a0266c52352e CONTRACT CREATION
FROM ADDRESS 0x546056e3f142F41321eB81097411Ef3FF2e37565
CREATED CONTRACT ADDRESS 0x258067284b92cE5a05180DA4A6793a062b4B203B
GAS USED 3000000
VALUE 0
TX HASH 0x2ae81297b3ad6a0abade68829c5f9fa049509e326678376a45a3fa96b4609b4b VALUE TRANSFER
FROM ADDRESS 0x546056e3f142F41321eB81097411Ef3FF2e37565
TO ADDRESS 0x551d7E66b3c3115bb3A1E6F8E595695c531C6cFf
GAS USED 21000
VALUE 1000000000000000000
Search
ENG IN
12:33 03-09-2024



Conclusion:

In this way , the installation of Ganache(Personal block chain) and MetaMask, Compilation and deployment of an election smart contract in the personal blockchain using injected web3 environment(MetaMask wallet) has been done successfully.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 7**	
Title of Lab Assignment: Simple Experiments using Solidity Program Constructs		
DOP: 08-10-2024	**DOS: 08-10-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 7

Aim: Simple Experiments using Solidity Program Constructs (if-then, while etc...)

1. Write a program in solidity to create a structure student with Roll no, Name, Class, Department, Course enrolled as variables.
    a. Add information of 5 students.
    b. Search for a student using Roll no.
    c. Display all information

2. Create a structure Consumer with Name, Address, Consumer ID, Units and Amount as members. Write a program in solidity to calculate the total electricity bill according to the given condition.
   For first 50 units Rs. 0.50/unit
   For next 100 units Rs. 0.75/unit
   For next 100 units Rs. 1.20/unit
   For unit above 250 Rs. 1.50/unit
   An additional surcharge of 20% is added to the bill. Display the information of 5 such consumers along with their units consumed and amount.

Theory:

Smart Contracts: Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They run on blockchain platforms, such as Ethereum, allowing for trustless transactions and automation of processes. Smart contracts ensure that conditions are met before executing actions, which eliminates the need for intermediaries.

Data Structures: In Solidity, data structures are essential for organizing and managing complex data efficiently. The two primary types of data structures used in Solidity are:

1. Structs:
    a. Structs are custom data types that group multiple variables under a single name. They are used to define complex entities with various attributes.
    b. For example, the Consumer struct in the ElectricityBillRegistry contract encapsulates all relevant details of an electricity consumer, such as consumer ID, name, address, units consumed, and amount due.

2. Arrays:
    a. Arrays are used to store multiple values of the same type. In the contract, an array of Consumer structs is used to maintain a list of all consumers, facilitating easy access and manipulation of consumer data.

Functionality in Contracts: Smart contracts utilize functions to manipulate data structures:

a. Adding Data: Functions like addConsumer allow users to add new instances of data (consumers) to the contract.
b. Retrieving Data: Functions like getConsumers return the entire list of consumers, providing structured access to data.
c. Calculating Values: Internal functions can perform calculations (e.g., calculateBill) based on the data stored in structs.

Importance of Structs in Smart Contracts:

a. Clarity: Structs enhance code readability by grouping related data, making the contract easier to understand.
b. Efficiency: They optimize data management by reducing the complexity of handling multiple variables.
c. Encapsulation: Structs encapsulate data and behavior, allowing for better organization of contract logic.

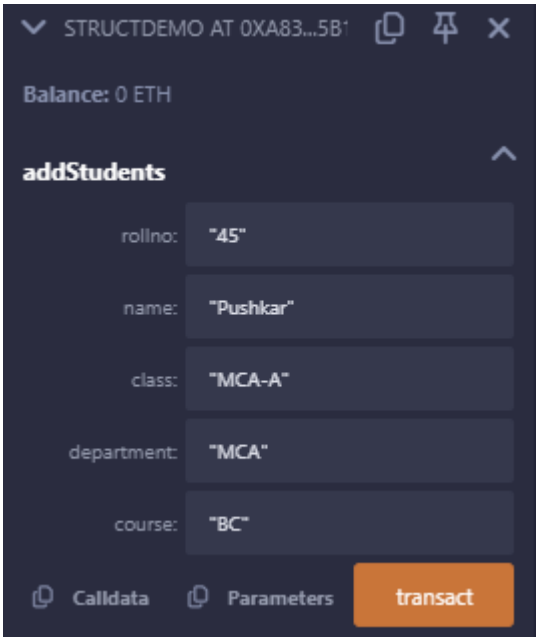
1. **Student.sol**

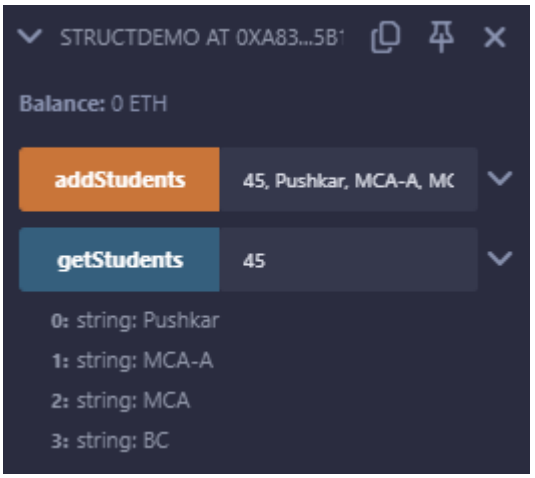
Code:

```
// SPDX-License-Identifier: GPL-3.0
// Students Details
pragma solidity ^0.8.26;

// Creating a Smart Contract
contract StructDemo{
    // Structure of Students
    struct Students{
        // State variables
        int rollno;
        string name;
        string class;
        string department;
```

```
        string course;
        }

        Students []studs;
        // Function to add
        // Students details
        function addStudents(int rollno, string memory name, string memory class,
string memory department, string memory course) public {
                Students memory e = Students(rollno, name, class, department,
course);
                studs.push(e);
        }

        // Function to get
        // details of Students
        function getStudents( int rollno ) public view returns (string memory, string
memory, string memory, string memory){
                uint i;
                for(i=0; i < studs.length; i++) {
                        Students memory e = studs[i];

                        // Looks for a matching
                        // Students id
                        if(e.rollno == rollno) {
                                return(e.name, e.class, e.department, e.course);
                        }
                }

                // If provided Students
                // id is not present
                // it returns Not
                // Found
                return("Not Found", "Not Found", "Not Found", "Not Found");
        }
}
```

Output:





2. **Consumer.sol**

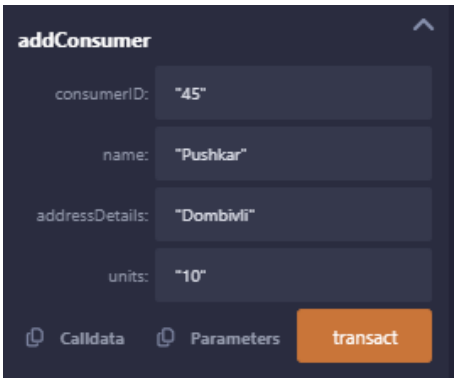
Code:

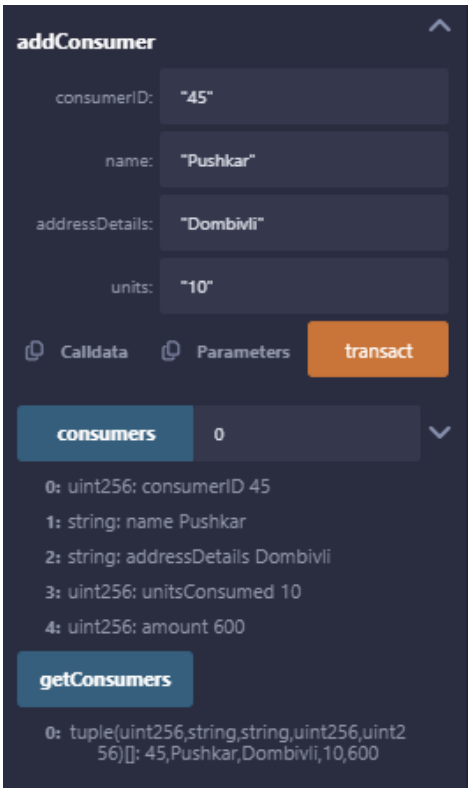
```
// SPDX-License-Identifier: GPL-3.0
// Solidity program
// to store Consumer Details
pragma solidity ^0.8.26;

// Creating a Smart Contract
contract ElectricityBillRegistry {
    // Structure of consumer
    struct Consumer {
```

```
        // State variables
        uint256 consumerID;
        string name;
        string addressDetails;
        uint256 unitsConsumed;
        uint256 amount;
    }

    Consumer[] public consumers;

    // Function to add consumer details
    function addConsumer(uint256 consumerID, string memory name, string memory
addressDetails, uint256 units) public {
        uint256 amount = calculateBill(units);
        Consumer memory c = Consumer(consumerID, name, addressDetails, units,
amount);
        consumers.push(c);
    }

    // Function to calculate the electricity bill
    function calculateBill(uint256 units) private pure returns (uint256) {
        uint256 bill = 0;

        if (units <= 50) {
            bill = units * 50; // Rs. 0.50/unit
        } else if (units <= 150) {
            bill = (50 * 50) + ((units - 50) * 75); // Rs. 0.75/unit for units 51-150
        } else if (units <= 250) {
            bill = (50 * 50) + (100 * 75) + ((units - 150) * 120); // Rs. 1.20/unit for units
151-250
        } else {
            bill = (50 * 50) + (100 * 75) + (100 * 120) + ((units - 250) * 150); // Rs.
1.50/unit for units above 250
        }
```

```
        // Apply additional surcharge of 20%
        bill = bill + (bill * 20 / 100);
        return bill;
    }

    // Function to get details of all consumers
    function getConsumers() public view returns (Consumer[] memory) {
        return consumers;
    }
}
```

Output:



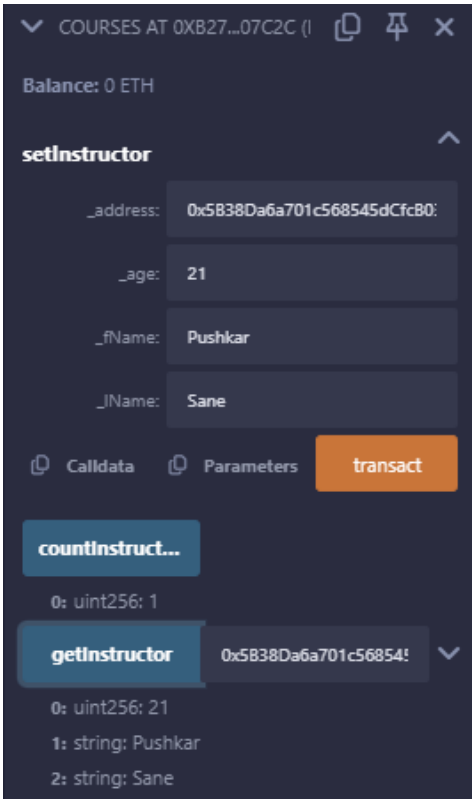


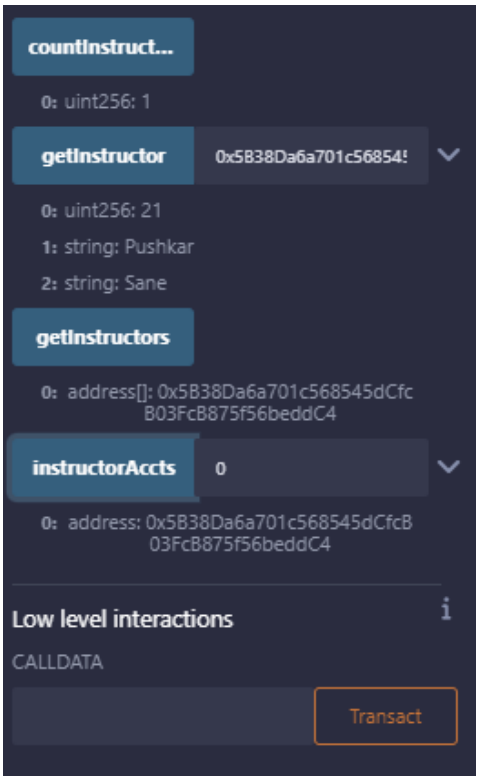
3. **Instructor.sol**

Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.26;
contract Courses {
    struct Instructor {
        uint age;
        string fName;
        string lName;
    }

    mapping (address => Instructor) instructors;
    address[] public instructorAccts;
    function setInstructor(address _address, uint _age, string memory _fName, string
memory _lName) public {
    //var instructor = instructors[_address];
    instructors[_address].age = _age;
    instructors[_address].fName = _fName;
    instructors[_address].lName = _lName;
        instructorAccts.push(_address);
    }

    function getInstructors() view public returns(address[] memory) {
        return instructorAccts;
    }

    function getInstructor(address _address) view public returns (uint, string memory,
string memory) {
        return (instructors[_address].age, instructors[_address].fName,
instructors[_address].lName);
    }

    function countInstructors() view public returns (uint) {
        return instructorAccts.length;
    }
}
```

Output:





Conclusion:

Successfully demonstrated simple experiments using Solidity Program Constructs.

Name of Student : Pushkar Sane		
Roll Number : 45		**LAB Assignment Number:** 8
Title of LAB Assignment : Execution of smart contract using truffle framework.		
DOP : 15-08-2024		**DOS :** 16-09-2024
CO Mapped : CO5	**PO Mapped:** PO1,PO2, PO3, PO4, PO7, PO9, PSO1, PSO2	**Signature:**

PRACTICAL 8

Aim: Execution of smart contract using truffle framework.

Theory:

Truffle framework:

- Truffle is a world-class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier.
- Truffle is widely considered the most popular tool for blockchain application development with over 1.5 million lifetime downloads. Truffle supports developers across the full lifecycle of their projects, whether they are looking to build on Ethereum, Hyperledger, Quorum, or one of an ever-growing list of other supported platforms.
- Paired with Ganache, a personal blockchain, and Drizzle, a front-end dApp development kit, the full Truffle suite of tools promises to be an end-to-end dApp development platform.

  1. Built-in smart contract compilation, linking, deployment and binary management.

  Automated contract testing for rapid development.

  2. Scriptable, extensible deployment & migrations framework.
  3. Network management for deploying to any number of public & private networks.
  4. Package management with EthPM & NPM, using the ERC190 standard.
  5. Interactive console for direct contract communication.
  6. Configurable build pipeline with support for tight integration.
  7. External script runner that executes scripts within a Truffle environment

Smart Contract:

A smart contract is a stand-alone script usually written in Solidity and compiled into binary or JSON and deployed to a specific address on the blockchain. In the same way that we can call a specific URL endpoint of a RESTful API to execute some logic through an HttpRequest, we can similarly execute the deployed smart contract at a specific address by submitting the correct data along with the necessary Ethereum to call the deployed and compiled Solidity function.

Installation Steps:

- **npm install -g truffle**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\24_MCA\BC> node -v
v20.6.1
PS D:\24_MCA\BC> npm install -g truffle
npm WARN deprecated testrpc@0.0.1: testrpc has been renamed to ganache-cli, please use this package from now on.
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promises now, pleas
e switch to that.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated apollo-datasource@3.3.2: The `apollo-datasource` package is part of Apollo Server v2 and v3, which are now deprecat
ed (end-of-life October 22nd 2023 and October 22nd 2024, respectively). See https://www.apollographql.com/docs/apollo-server/previous-v
ersions/ for more details.
npm WARN deprecated apollo-server-errors@3.3.1: The `apollo-server-errors` package is part of Apollo Server v2 and v3, which are now de
precated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the `@apollo
/server` package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
npm WARN deprecated apollo-server-plugin-base@3.7.2: The `apollo-server-plugin-base` package is part of Apollo Server v2 and v3, which
are now deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in th
e `@apollo/server` package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
npm WARN deprecated apollo-server-types@3.8.0: The `apollo-server-types` package is part of Apollo Server v2 and v3, which are now depr
ecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the `@apollo/s
erver` package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
npm WARN deprecated apollo-server-express@3.12.1: The `apollo-server-express` package is part of Apollo Server v2 and v3, which are now
 deprecated (end-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the `@apo
llo/server` package. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
npm WARN deprecated apollo-server@3.12.1: The `apollo-server` package is part of Apollo Server v2 and v3, which are now deprecated (end
-of-life October 22nd 2023 and October 22nd 2024, respectively). This package's functionality is now found in the `@apollo/server` pack
age. See https://www.apollographql.com/docs/apollo-server/previous-versions/ for more details.
```

- **truffle --version**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

D:\24_MCA\BC>truffle --version
Truffle v5.11.5 (core: 5.11.5)
Ganache v7.9.1
Solidity v0.5.16 (solc-js)
Node v20.6.1
Web3.js v1.10.0
```

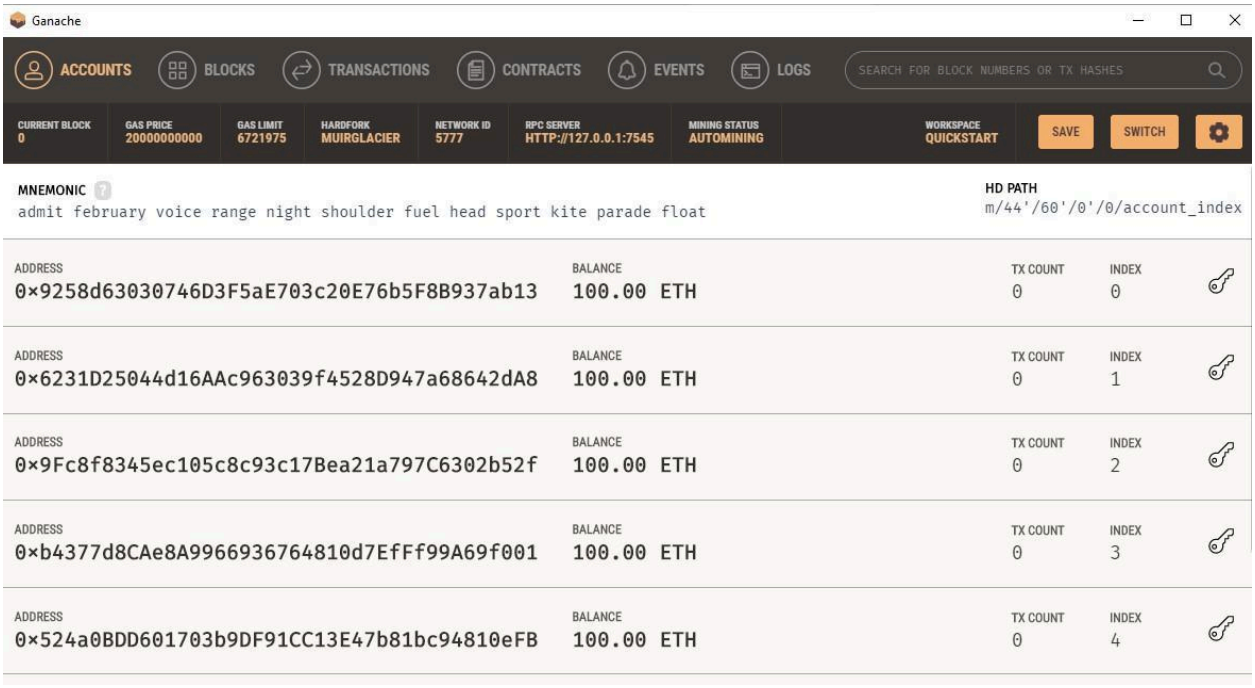
- **truffle init**

```
D:\24_MCA\BC>truffle init

Starting init...
================

> Copying project files to D:\24_MCA\BC

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs
```

MyContract.sol

pragma solidity >=0.5.0 <0.9.0; contract MyContract { string value; constructor() public { value = "myValue"; } function get() public view returns(string memory) { return value; } function set(string memory _value) public { value = _value;
}
}

- **Start Ganache:**



- **truffle compile**

```
D:\24_MCA\BC>truffle compile

Compiling your contracts...
===========================
✓ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
> Compiling .\contracts\MyContract.sol
> Artifacts written to D:\24_MCA\BC\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang

D:\24_MCA\BC>
```

Migration > 2_deploy_contract.js var MyContract = artifacts.require("./MyContract.sol"); module.exports = function(deployer)

{

deployer.deploy(MyContract);

};

- **truffle migrate**

```
D:\24_MCA\BC>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol
> Artifacts written to D:\24_MCA\BC\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_contracts.js
=====================

   Deploying 'MyContract'
   ----------------------
   > transaction hash:    0x2decc255f79e9eae062916e228d1da2204c69727306d9056f422a03ca45c1b4a
   > Blocks: 0            Seconds: 0
   > contract address:    0xfB0cCca437E9cc1Ad19a9E41A00e30518E1632ca
   > block number:        1
   > block timestamp:     1697618467
   > account:             0x2c5E7238B8CE19c55a6258EaD154a73a91A38A33
   > balance:             99.99492518
   > gas used:            253741 (0x3df2d)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00507482 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00507482 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.00507482 ETH
```

- **truffle console**

```
D:\24_MCA\BC>truffle console
truffle(development)> MyContract.deployed().then((instance) => { app = instance } )
undefined
truffle(development)> app.get()
'myValue'
truffle(development)> app.set('Kalpana')
{
  tx: '0xc677eef3212942918e83aec8a8d40c122f3af02db9a06ccc7ab76c529e7c9c56',
  receipt: {
    transactionHash: '0xc677eef3212942918e83aec8a8d40c122f3af02db9a06ccc7ab76c529e7c9c56',
    transactionIndex: 0,
    blockHash: '0x9b58e10e616a4212a9c65fd6ce7ad6628ef427359b0a6846720048a12f99514c',
    blockNumber: 2,
    from: '0x2c5e7238b8ce19c55a6258ead154a73a91a38a33',
    to: '0xfb0ccca437e9cc1ad19a9e41a00e30518e1632ca',
    gasUsed: 29102,
    cumulativeGasUsed: 29102,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []          ▯
}
truffle(development)> app.get()
'Kalpana'
truffle(development)>
```

1) Create a Bank Account contract and implement the following services:

Deposit

Withdraw (keep a condition that only the owner of the contract can withdraw)

Receive Ether

Transfer Ether

Check Balance

Code:

Mycontract.sol:

```
pragma  solidity  ^0.5.0;
contract MyContract
{ uint balance=31300; function getBalance() public
   view returns (uint)
   {  return  balance;  }  function  deposit(uint
   newDeposit) public
   { balance=balance+newDeposit;
   } }
```

```
pragma solidity ^0.5.0; contract
FC1
{ function receiveDeposit() payable public
    { } function getbalance() public view returns
    (uint)
    { return address(this).balance;
   } }

pragma solidity ^0.5.0; contract
FC2
{     address     owner;
    constructor() public
    {     owner=msg.sender;     }     function
    receiveDeposit() payable public
    { } function getbalance() public view returns
    (uint) { return address(this).balance;
    } function withdraw(uint funds) public
    { msg.sender.transfer(funds);
   } }

pragma solidity ^0.5.0; contract
FC3
{     address     owner;
    constructor() public
    {  owner=msg.sender;
    } modifier ifOwner()
    { if(owner != msg.sender)
        {
           revert(); }
        else
        {
           _;
```

```
    } } function receiveDeposit() payable
  public
  { } function getbalance() public view returns
  (uint)
  { return address(this).balance;
  } function withdraw(uint funds) public ifOwner
  { msg.sender.transfer(funds);
  }
}
```

2_deploy_contracts.sol

```
var MyContract = artifacts.require("./MyContract.sol");
var FC1=artifacts.require("./FC1.sol")
var FC2=artifacts.require("./FC2.sol")
var FC3=artifacts.require("./FC3.sol")
var Test = artifacts.require("./Test.sol"); module.exports =
function(deployer)
{
deployer.deploy(MyContract);
deployer.deploy(FC1);
deployer.deploy(FC2);
deployer.deploy(FC3);
deployer.deploy(Test);
};
```

Output:

Compile:

Migrate:

```
D:\24_MCA\BC>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\Test.sol
> Artifacts written to D:\24_MCA\BC\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang

D:\24_MCA\BC>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\Test.sol
> Artifacts written to D:\24_MCA\BC\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_contracts.js
=====================
```

Check Balance: truffle

console

Mycontract.deployed().then((instance)=>{app = instance}) app.getBalance()

```
D:\24_MCA\BC>truffle console
truffle(development)> MyContract.deployed().then((instance) => { app = instance } )
undefined
```

```
truffle(development)> app.getBalance()
BN {
  negative: 0,
  words: [ 31300, <1 empty item> ],
  length: 1,
  red: null
}
```

Deposit 5000:

```
truffle(development)> app.deposit(5000)
{
  tx: '0xbb6baf68e97b4b8e035d47c4530931c5a5f357e914a2283b330c1466b58eb428',
  receipt: {
    transactionHash: '0xbb6baf68e97b4b8e035d47c4530931c5a5f357e914a2283b330c1466b58eb428',
    transactionIndex: 0,
    blockHash: '0xf8fd833d9480c148041e7468a22c7d34d592fb8c8974f81b1bbb2fc611d92ae8',
    blockNumber: 4,
    from: '0x2c5e7238b8ce19c55a6258ead154a73a91a38a33',
    to: '0x79f206b158d43ad0908b6a6b9daf7cec0ab4c1ed',
    gasUsed: 27269,
    cumulativeGasUsed: 27269,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
```

```
truffle(development)> app.getBalance()
BN {
  negative: 0,
  words: [ 36300, <1 empty item> ],
  length: 1,
  red: null
}
```

```
D:\24_MCA\BC>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\Test.sol
> Artifacts written to D:\24_MCA\BC\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang

D:\24_MCA\BC>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\Test.sol
> Artifacts written to D:\24_MCA\BC\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_contracts.js
=====================
```

```
2_deploy_contracts.js
=====================

   Replacing 'MyContract'
   ----------------------
   > transaction hash:    0xe241ac0fac861f5019a22ba06649259bd5316e273113e02ba5991e33879982e6
   > Blocks: 0            Seconds: 0
   > contract address:    0x010864f8a1a6689787221cA47d4A956e9ac8C636
   > block number:        10
   > block timestamp:     1697620740
   > account:             0x2c5E7238B8CE19c55a6258EaD154a73a91A38A33
   > balance:             99.9749101
   > gas used:            123713 (0x1e341)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00247426 ETH


   Replacing 'FC1'
   ---------------
   > transaction hash:    0x54d99333f9b9f6c08a2238ad26053d5144acb62240b48364d999850581051693
   > Blocks: 0            Seconds: 0
   > contract address:    0x827c413E4aDd31D1f9Ca4121E0FE27532De43328
   > block number:        11
   > block timestamp:     1697620741
   > account:             0x2c5E7238B8CE19c55a6258EaD154a73a91A38A33
   > balance:             99.97299664
   > gas used:            95673 (0x175b9)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00191346 ETH
```

```
Replacing 'FC2'
---------------
> transaction hash:    0x1bb8055433cfdb01a704011afef38b0b187278334bf45f899c41e87e73f24848
> Blocks: 0            Seconds: 0
> contract address:    0xFD2bDe6CB2336198C8989093c0f6d30B1ce3fC53
> block number:        12
> block timestamp:     1697620742
> account:             0x2c5E7238B8CE19c55a6258EaD154a73a91A38A33
> balance:             99.9699925
> gas used:            150207 (0x24abf)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.00300414 ETH


Replacing 'FC3'
---------------
> transaction hash:    0x7896e7a66777cc941a71910f22662e18894e8856e418b8147ddf5251a130ab5e
> Blocks: 0            Seconds: 0
> contract address:    0x64B8DBCeF3248bbCfDDA28aA9Cd494eF8f07a019
> block number:        13
> block timestamp:     1697620743
> account:             0x2c5E7238B8CE19c55a6258EaD154a73a91A38A33
> balance:             99.96659584
> gas used:            169833 (0x29769)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.00339666 ETH
```

```
  Replacing 'Test'
  ----------------
  > transaction hash:    0x28e56a428dd93d7df3ebfb7d5b198d16ad05f3d0ae81e8a43f6466dfcba38955
  > Blocks: 0            Seconds: 0
  > contract address:    0x69968fC9ED33455b1EC75a49dD48164356A78571
  > block number:        14
  > block timestamp:     1697620743
  > account:             0x2c5E7238B8CE19c55a6258EaD154a73a91A38A33
  > balance:             99.96344522
  > gas used:            157531 (0x2675b)
  > gas price:           20 gwei
  > value sent:          0 ETH
  > total cost:          0.00315062 ETH

  > Saving artifacts
  -------------------------------------
  > Total cost:          0.01393914 ETH

Summary
=======
> Total deployments:   5
> Final cost:          0.01393914 ETH
```

```
D:\24_MCA\BC>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\Test.sol
> Artifacts written to D:\24_MCA\BC\build\contracts
> Compiled successfully using:
   - solc: 0.5.1+commit.c8a2cb62.Emscripten.clang

D:\24_MCA\BC>truffle console
truffle(development)> MyContract.deployed().then((instance) => { app = instance } )
undefined
truffle(development)> FC1.deployed().then((instance) => { app1 = instance } )
undefined
truffle(development)> app1.receiveDeposit()
{
  tx: '0x40b1f2b14574c0ba09c27cad4911d16c51f4bb85398946e67e51ec7f6f6155d3',
  receipt: {
    transactionHash: '0x40b1f2b14574c0ba09c27cad4911d16c51f4bb85398946e67e51ec7f6f6155d3',
    transactionIndex: 0,
    blockHash: '0xe22b74eb09b6c77c196c43c4b09e53deafbdf84f1c2f031462b70e2f4bdc255d',
    blockNumber: 15,
    from: '0x2c5e7238b8ce19c55a6258ead154a73a91a38a33',
    to: '0x827c413e4add31d1f9ca4121e0fe27532de43328',
    to: '0x827c413e4add31d1f9ca4121e0fe27532de43328',
    gasUsed: 21189,
```

```
    gasUsed: 21189,
    cumulativeGasUsed: 21189,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00',
    rawLogs: []
  },
  logs: []
}
truffle(development)>
```

Receive Ether:

```
truffle(development)> app.getBalance()
BN {
  negative: 0,
  words: [ 32100, <1 empty item> ],
  length: 1,
  red: null
}
truffle(development)>
```

Withdraw 900:

```
truffle(development)> app.withdraw(900)
{
  tx: '0x494a542ab12aed32c48ed55f2cdbbb5d1761be50ec96984cf7d982a52d81bedd',
  receipt: {
    transactionHash: '0x494a542ab12aed32c48ed55f2cdbbb5d1761be50ec96984cf7d982a52d81bedd',
    transactionIndex: 0,
    blockHash: '0x895d45919c60437cd1374d8525d3698fd85875490e98d8952130f7f3355b10ce',
    blockNumber: 13,
    from: '0x582e1d44f087cd64f140ded3b1e95e24abdd04d7',
    to: '0xb4e870ad5408e3c9dfe8564b4bf7927fccb78b70',
    gasUsed: 27264,
    cumulativeGasUsed: 27264,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
truffle(development)>

truffle(development)> app.getBalance()
BN {
  negative: 0,
  words: [ 31200, <1 empty item> ],
  length: 1,
  red: null
}
truffle(development)>
```

2) Create a Smart contract to simulate function overloading . Execute the contract using the truffle framework.

Code:

Test.sol:

pragma solidity ^0.5.0; contract Test { function getSum(uint
a, uint b) public pure returns(uint){

return a + b;

} function getSum(uint a, uint b, uint c) public pure returns(uint){

```
    return a + b + c;
  }
  function callSumWithTwoArguments(uint a, uint b) public pure returns(uint){
    return getSum(a,b);
  } function callSumWithThreeArguments(uint a, uint b, uint c) public pure returns(uint){
    return getSum(a,b,c);
  }
}
```

2_deploy_contract.js:

```
var Test = artifacts.require("./Test.sol"); module.exports =
function(deployer)
{ deployer.deploy(Test);
};
```

Output:

```
truffle(development)> truffle compile

Compiling your contracts...

===========================

> Compiling .\contracts\Mycontract.sol

> Compiling .\contracts\Mycontract.sol
> Compiling .\contracts\overloading.sol
> Artifacts written to D:\mca12\Blockchain\prac8\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

truffle(development)>
```

Migrate

```
D:\mca12\Blockchain\prac8>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Mycontract.sol
> Compiling .\contracts\Mycontract.sol
> Compiling .\contracts\Test.sol
> Artifacts written to D:\mca12\Blockchain\prac8\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)

   > block timestamp:     1698267010
   > account:             0x582e1d44f087cd64F140deD3B1e95E24abdd04D7
   > balance:             99.97201244
   > gas used:            145135 (0x236ef)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0029027 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0029027 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.0029027 ETH


D:\mca12\Blockchain\prac8>
```

Create instance of Test:

```
D:\mca12\Blockchain\prac8>truffle console
truffle(development)> Test.deployed().then((instance)=>{app = instance})
undefined
truffle(development)>
```

Get sum of 2 numbers:

```
D:\mca12\Blockchain\prac8>truffle console
truffle(development)> Test.deployed().then((instance)=>{app = instance})
undefined
truffle(development)> app.callSumWithTwoArguments(4,5)
BN { negative: 0, words: [ 9, <1 empty item> ], length: 1, red: null }
truffle(development)>
```

Get sum of three numbers:

```
truffle(development)> app.callSumWithThreeArguments(2,6,8)
BN { negative: 0, words: [ 16, <1 empty item> ], length: 1, red: null }
truffle(development)>
```

Conclusion: I have successfully understood how to construct a smart contract using Solidity and Truffle framework.

Name of Student : Pushkar Sane		
Roll Number : 45		**LAB Assignment Number: 9**
Title of LAB Assignment : Creation of Dapp in Ethereum.		
DOP : 22-10-2024		**DOS : 25-09-2024**
CO Mapped : **CO5**	**PO Mapped:** **PO1,PO2, PO3, PO4,** **PO7, PO9, PSO1, PSO2**	**Signature:**

PRACTICAL 9

Aim: Creation of Dapp in Ethereum

Theory:

Decentralized Applications:

Decentralized Applications (or DApps) are applications that do not rely on a centralized backend running in AWS or Azure that power traditional web and mobile applications (outside of hosting the frontend code itself). Instead, the application interacts directly with a blockchain which can be thought of distributed cluster of nodes analogous to applications interacting directly with a “masterless” cluster of Cassandra nodes with full replication on every peer in an untrusted peer-to-peer network.

These blockchain nodes do not require a leader which would defeat the purpose of being truly decentralized. Unlike leader election in various consensus protocols like Raft and Paxos, blockchain transactions are sent to and processed by “random” nodes via Proof of Work or Proof of Stake. These nodes are untrusted nodes running in an arbitrary sized network on various computer devices around the world. Such technology can enable true decentralized ledgers and systems of records.

DApps are the frontend apps which interact with these blockchain over an API. For Ethereum, this API is a JSON-RPC layer called the Ethereum Web3 API which Moesif supports natively.

The fundamental requirements for DAPPS:

1. Open Source: dApps should be open source and its codebase should be freely available for all. Any changes in the structure or working of the app should only be taken by agreement of the majority.

2. Decentralized: dApps should be decentralized with all the information and operations stored on a public and decentralized Blockchain which would ensure security and transparency.

3. Incentive: dApps should offer some sort of incentives to their users in the form of cryptographic tokens. These are a sort of liquid assets and they provide incentives for users to support the Blockchain dApp ecosystem.

4. Protocol: dApps should have a particular protocol to demonstrate proof of value. This means showing the value of a particular process in a way that can be easily verified by others.

Ethereum is a cryptocurrency platform in the market just like Bitcoin. It is an open-source & decentralized blockchain featuring working on smart contracts. It has its own cryptocurrency known as ether. The smart contracts in Ethereum are written in solidity.
TestRPC The Ethereum TestRPC is like a manual emulator for blockchain. It provides blockchain interaction without running on an actual Ethereum node. It also provides all the features required for testing of your blockchain. It is based on Node.js.

Web3.js Web3.js is an Ethereum-based JavaScript API that allows us to interact with a local or remote Ethereum node using different servers such as HTTP. It interacts with the Ethereum blockchain and can retrieve data from the blockchain as required by the developer.

Advantages:

- Many of the advantages of dApps center around the program's ability to safeguard user privacy. With decentralized apps, users do not need to submit their personal information to use the function the app provides. DApps use smart contracts to complete the transaction between two anonymous parties without the need to rely on a central authority.
- Proponents interested in free speech point out that dApps can be developed as alternative social media platforms. A decentralized social media platform would be resistant to censorship because no single participant on the blockchain can delete messages or block messages from being posted.
- Ethereum is a flexible platform for creating new dApps, providing the infrastructure needed for developers to focus their efforts on finding innovative uses for digital applications. This could enable rapid deployment of dApps in a variety of industries including banking and finance, gaming, social media, and online shopping.

1. VOTING PROCESS

1.create a project directory for our dApp in the command line like this:

$ mkdir election

$ cd election

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\MCA_24> mkdir election


    Directory: D:\MCA_24


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        01-11-2023     13:34                election


PS D:\MCA_24> cd election
PS D:\MCA_24\election>
```

2. From within your project directory, install the pet shop box from the command line like this:

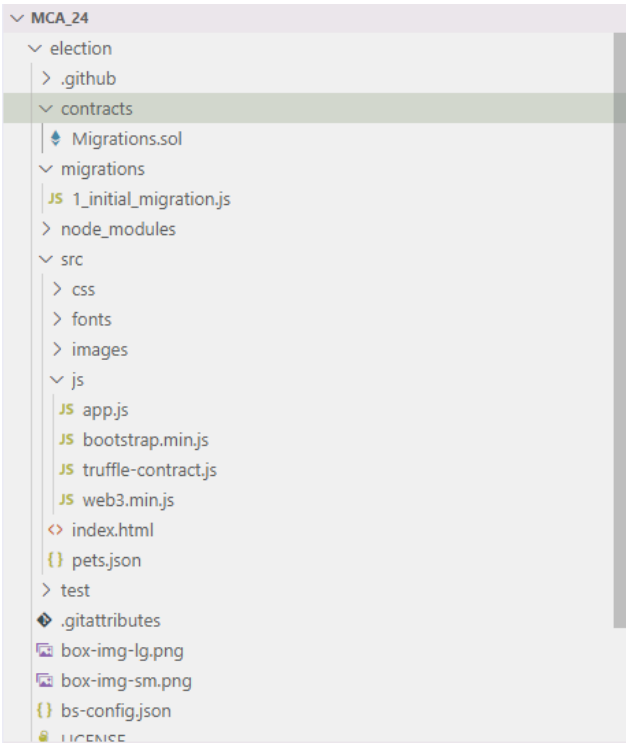
$ truffle unbox pet-shop

```
PS D:\MCA_24\election> truffle unbox pet-shop

Starting unbox...
=================

✓ Preparing to download box
✓ Downloading
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile
npm WARN deprecated set-value@2.0.0: Critical bug fixed in v3.0.1, please upgrade to the latest version.
npm WARN deprecated mixin-deep@1.3.1: Critical bug fixed in v2.0.1, please upgrade to the latest version.
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated set-value@0.4.3: Critical bug fixed in v3.0.1, please upgrade to the latest version.
npm WARN deprecated chokidar@2.0.4: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated source-map-url@0.4.0: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated source-map-resolve@0.5.2: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated axios@0.17.1: Critical security vulnerability fixed in v0.21.1. For more information, see https://githu
✓ Cleaning up temporary files
✓ Setting up box

Unbox successful, sweet!
```

```
Commands:

  Compile:        truffle compile
  Migrate:        truffle migrate
  Test contracts: truffle test
  Run dev server: npm run dev

PS D:\MCA_24\election>
```

3.Open Ganache

4.Start developing the solidity code, test code and client-side code for the application:



Code:

Contracts-> Election.sol

```
pragma solidity >=0.4.2 <=0.8.0;
contract Election {
// Model a Candidate
struct Candidate {
uint id;
string name;
uint voteCount;
}
// Read/write candidates
mapping(uint => Candidate) public candidates;
// Store accounts that have voted
mapping(address => bool) public voters;
```

```
// Store Candidates Count
uint public candidatesCount;
constructor() public {
addCandidate("Atharva");
addCandidate("Rutik");
}
event votedEvent(uint indexed _candidateId);
function addCandidate (string memory _name) private {
candidatesCount ++;
candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}
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;
// update candidate vote Count
candidates[_candidateId].voteCount ++;
// trigger voted event
emit votedEvent(_candidateId);
}
}
```

migration-> 2_deploy_contracts.js

```
var Election = artifacts.require("./Election.sol");

module.exports = function(deployer) {
  deployer.deploy(Election);
};
```

test-> election.js

```
var Election = artifacts.require("./Election.sol");
```

```
contract("Election", function(accounts) {
var electionInstance;
it("initializes with two candidates", function() {
return Election.deployed().then(function(instance) {
return instance.candidatesCount();
}).then(function(count) {
assert.equal(count, 2);
});
});
it("it initializes the candidates with the correct values", function() {
return Election.deployed().then(function(instance) {
electionInstance = instance;
return electionInstance.candidates(1);
}).then(function(candidate) {
assert.equal(candidate[0], 1, "contains the correct id");
assert.equal(candidate[1], "Candidate 1", "contains the correct name");
assert.equal(candidate[2], 0, "contains the correct votes count");
return electionInstance.candidates(2);
}).then(function(candidate) {
assert.equal(candidate[0], 2, "contains the correct id");
assert.equal(candidate[1], "Candidate 2", "contains the correct name");
assert.equal(candidate[2], 0, "contains the correct votes count");
});
});
it("allows a voter to cast a vote", function() {
    return Election.deployed().then(function(instance) {
    electionInstance = instance;
    candidateId = 1;
    return electionInstance.vote(candidateId, { from: accounts[0] });
    }).then(function(receipt) {
    assert.equal(receipt.logs.length, 1, "an event was triggered");
    assert.equal(receipt.logs[0].event, "votedEvent", "the event type is correct");
```

```
    assert.equal(receipt.logs[0].args._candidateId.toNumber(), candidateId, "the candidate id is correct");
    return electionInstance.voters(accounts[0]);
    }).then(function(voted) {
    assert(voted, "the voter was marked as voted");
    return electionInstance.candidates(candidateId);
    }).then(function(candidate) {
    var voteCount = candidate[2];
    assert.equal(voteCount, 1, "increments the candidate's vote count");
    })
});
it("throws an exception for invalid candidates", function() {
return Election.deployed().then(function(instance) {
electionInstance = instance;
return electionInstance.vote(99, { from: accounts[1] })
}).then(assert.fail).catch(function(error) {
assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
return electionInstance.candidates(1);
}).then(function(candidate1) {
var voteCount = candidate1[2];
assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
return electionInstance.candidates(2);
}).then(function(candidate2) {
var voteCount = candidate2[2];
assert.equal(voteCount, 0, "candidate 2 did not receive any votes");
});
});
it("throws an exception for double voting", function() {
return Election.deployed().then(function(instance) {
electionInstance = instance;
candidateId = 2;
electionInstance.vote(candidateId, { from: accounts[1] });
return electionInstance.candidates(candidateId);
```

```
}).then(function(candidate) {
var voteCount = candidate[2];
assert.equal(voteCount, 1, "accepts first vote");
// Try to vote again
return electionInstance.vote(candidateId, { from: accounts[1] });
}).then(assert.fail).catch(function(error) {
assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
return electionInstance.candidates(1);
}).then(function(candidate1) {
   var voteCount = candidate1[2];
assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
return electionInstance.candidates(2);
}).then(function(candidate2) {
var voteCount = candidate2[2];
assert.equal(voteCount, 1, "candidate 2 did not receive any votes");
});
});
});
```

src-> index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
     <!-- The above 3 meta tags *must* come first in the head; any other head content must come
*after* these tags -->
    <title>Pete's Pet Shop</title>
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
```

```
  <!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
  <![endif]-->
</head>
<body>
  <div class="container" style="width: 650px;">
  <div class="row">
    <div class="col-lg-12">
      <h1 class="text-center">Election Results</h1>
      <hr/>
      <br/>
      <div id="loader">
      <p class="text-center">Loading...</p>
      </div>
      <div id="content" style="display: none;">
      <table class="table">
      <thead>
      <tr>
      <th scope="col">#</th>
      <th scope="col">Name</th>
      <th scope="col">Votes</th>
      </tr>
      </thead>
      <tbody id="candidatesResults">
      </tbody>
    </table>
    <hr/>
    <form onSubmit="App.castVote(); return false;">
    <div class="form-group">
    <label for="candidatesSelect">Select Candidate</label>
    <select class="form-control" id="candidatesSelect">
    </select>
```

```
    </div>
    <button type="submit" class="btn btn-primary">Vote</button>
    <hr />
    </form>
    <p id="accountAddress" class="text-center"></p>
    </div>
    </div>
    </div>
    </div>
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<!-- Include all compiled plugins (below), or include individual files as needed
-->
<script src="js/bootstrap.min.js"></script>
<script src="js/web3.min.js"></script>
<script src="js/truffle-contract.js"></script>
<script src="js/app.js"></script>
</body>
</html>
```

src->js-> app.js

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',
  hasVoted: false,
  init: function() {
    return App.initWeb3();
},
initWeb3: function() {
// TODO: refactor conditional
if (typeof web3 !== 'undefined') {
```

```
// If a web3 instance is already provided by Meta Mask.
App.web3Provider = web3.currentProvider;
web3 = new Web3(web3.currentProvider);
} else {
// Specify default instance if no web3 instance provided
App.web3Provider = new
Web3.providers.HttpProvider('http://localhost:7545');
web3 = new Web3(App.web3Provider);
}
return App.initContract();
},
initContract: function() {
  $.getJSON("Election.json", function(election) {
  // Instantiate a new truffle contract from the artifact
  App.contracts.Election = TruffleContract(election);
  // Connect provider to interact with contract
  App.contracts.Election.setProvider(App.web3Provider);
  App.listenForEvents();
return App.render();
});
},
// Listen for events emitted from the contract
listenForEvents: function() {
App.contracts.Election.deployed().then(function(instance) {
// Restart Chrome if you are unable to receive this event
// This is a known issue with Metamask
// https://github.com/MetaMask/metamask-extension/issues/2393
instance.votedEvent({}, {
  fromBlock: 0,
  toBlock: 'latest'
  }).watch(function(error, event) {
  console.log("event triggered", event)
  // Reload when a new vote is recorded
```

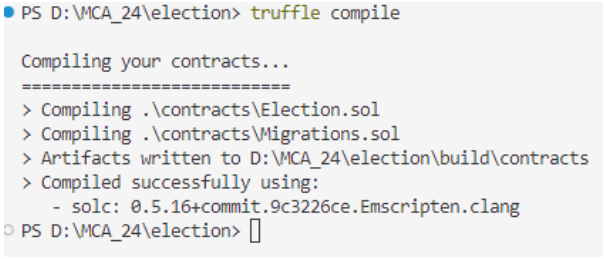
```
  App.render();
  });
  });
  },
  render: function() {
    var electionInstance;
    var loader = $("#loader");
    var content = $("#content");
    loader.show();
    content.hide();
// Load account data
web3.eth.getCoinbase(function(err, account) {
if (err === null) {
App.account = account;
$("#accountAddress").html("Your Account: " + account);
}
});
// Load contract data
App.contracts.Election.deployed().then(function(instance) {
  electionInstance = instance;
  return electionInstance.candidatesCount();
  }).then(function(candidatesCount) {
  var candidatesResults = $("#candidatesResults");
  candidatesResults.empty();
  var candidatesSelect = $('#candidatesSelect');
candidatesSelect.empty();
for (var i = 1; i <= candidatesCount; i++) {
electionInstance.candidates(i).then(function(candidate) {
var id = candidate[0];
var name = candidate[1];
var voteCount = candidate[2];
// Render candidate Result
```

```
var candidateTemplate = "<tr><th>" + id + "</th><td>" + name +"</td><td>" + voteCount + "</td></tr>"
candidatesResults.append(candidateTemplate);
// Render candidate ballot option
var candidateOption = "<option value='" + id + "' >" + name + "</ option>"
candidatesSelect.append(candidateOption);
});
}
return electionInstance.voters(App.account);
}).then(function(hasVoted) {
// Do not allow a user to vote
if(hasVoted) {
$('form').hide();
}
loader.hide();
content.show();
}).catch(function(error) {
console.warn(error);
});
},
castVote: function() {
  var candidateId = $('#candidatesSelect').val();
  App.contracts.Election.deployed().then(function(instance) {
  return instance.vote(candidateId, { from: App.account });
}).then(function(result) {
  // Wait for votes to update
  $("#content").hide();
  $("#loader").show();
  }).catch(function(err) {
  console.error(err);
  });
  }
  };
```

```
$(function() {
  $(window).load(function() {
  App.init();
  });
  });
```

Output:

- **truffle compile**
- **truffle migrate**
- **truffle console**
- **Election.deployed().then(function(instance) { app = instance })**
- **Truffle test**
- **npm run dev**



```
PS D:\MCA_24\election> truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to D:\MCA_24\election\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x4fae3f1e0f66e23a7958d0b829d31e1bc6e928b7a6e6bc39aa63cb4cad8242cf
   > Blocks: 0            Seconds: 0
   > contract address:    0x8AF51Cb3967660b4Ca6eBe4a259bFF124183535C
   > block number:        1
   > block timestamp:     1698827487
   > account:             0x986fa481d29aEe7D2590C537E03d39b290D39e29
   > balance:             99.99616114
   > gas used:            191943 (0x2edc7)
   > gas price:           20 gwei
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00383886 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00383886 ETH


2_deploy_contracts.js
=====================

   Deploying 'Election'
   --------------------
   > transaction hash:    0x8c15d22fe2f41aeabbc0312d1094fbef6c4d89cdc32f903b5988d21db1daa6da
   > Blocks: 0            Seconds: 0
   > contract address:    0x214558159ec60184eadeE517eA6ceeD62c51c639
   > block number:        3
   > block timestamp:     1698827488
   > account:             0x986fa481d29aEe7D2590C537E03d39b290D39e29
   > balance:             99.987601
   > gas used:            385669 (0x5e285)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00771338 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00771338 ETH

Summary
=======
> Total deployments:   2
> Final cost:          0.01155224 ETH


PS D:\MCA_24\election>
```

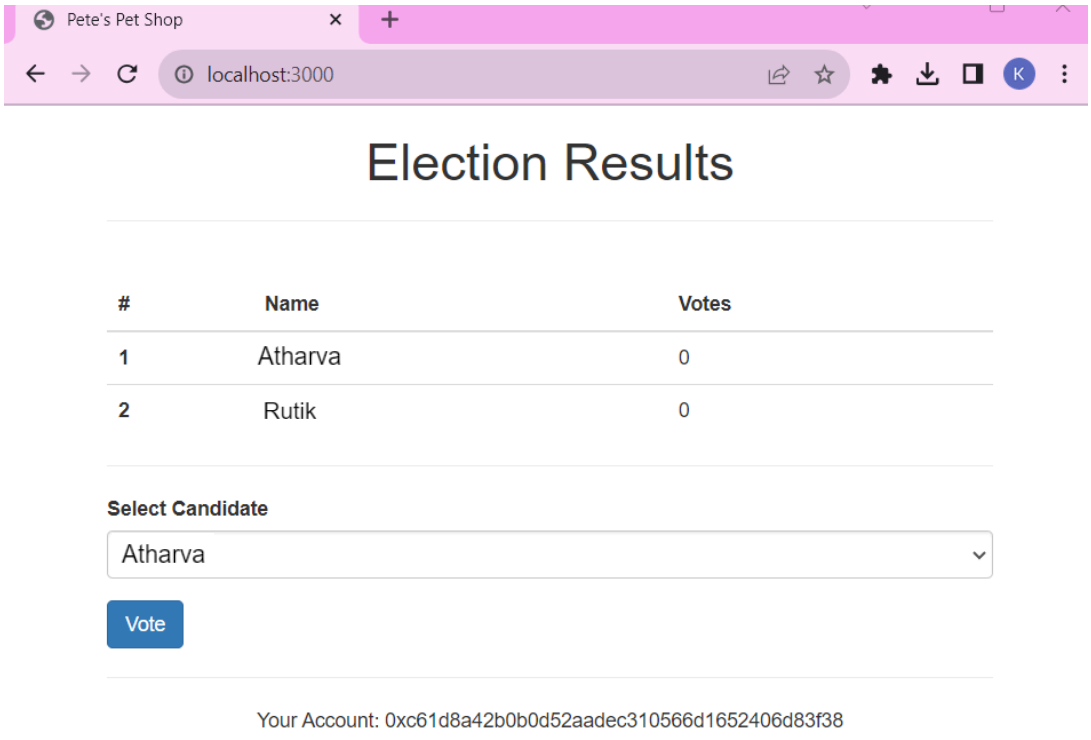
Election.deployed().then(function(instance) { app = instance })

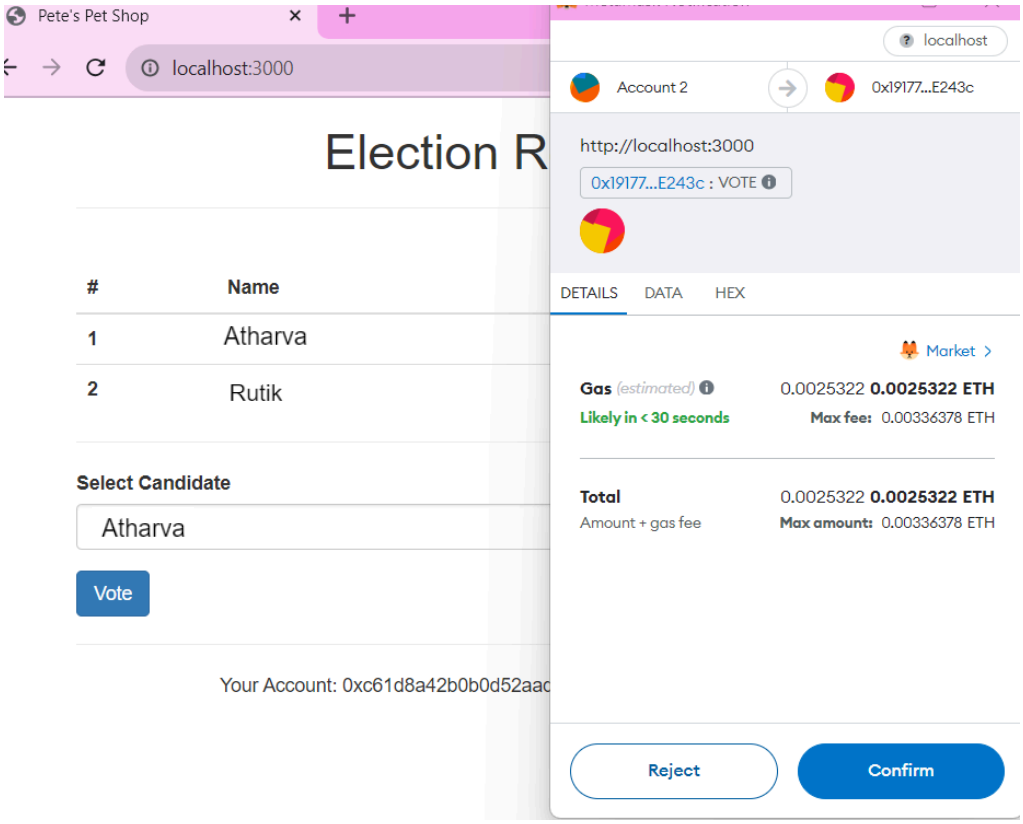
```
PS D:\MCA_24\election> truffle console
truffle(development)> Election.deployed().then(function(instance) { app = instance })
undefined
truffle(development)> .exit
PS D:\MCA_24\election>
```

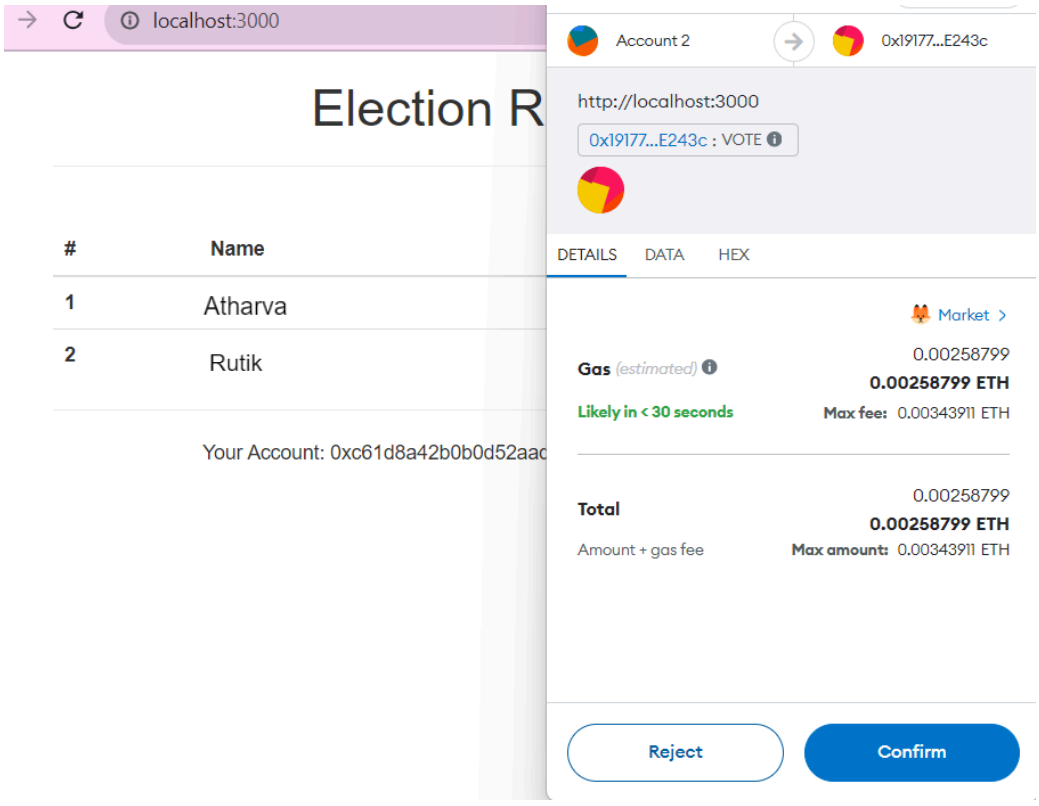
```
PS D:\MCA_24\election> truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\Exam\AppData\Local\Temp\test--22944-DdlfxjU7XFft
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


  Contract: Election
    ✓ initializes with two candidates (91ms)
    ✓ it initializes the candidates with the correct values (209ms)
    ✓ allows a voter to cast a vote (707ms)
    ✓ throws an exception for invalid candiates (374ms)
    1) throws an exception for double voting
    > No events were emitted


  4 passing (2s)
  1 failing

  1) Contract: Election
       throws an exception for double voting:
     AssertionError: error message must contain revert
      at D:\MCA_24\election\test\election.js:72:1
      at processTicksAndRejections (node:internal/process/task_queues:95:5)
```

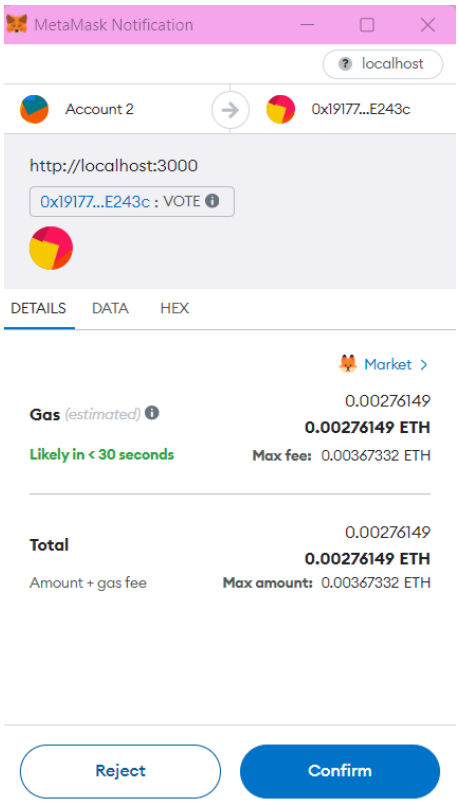
```
PS D:\MCA_24\election> npm run dev

> pet-shop@1.0.0 dev
> lite-server

** browser-sync config **
{
  injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server: {
    baseDir: [ './src', './build/contracts' ],
    middleware: [ [Function (anonymous)], [Function (anonymous)] ]
  }
}
[Browsersync] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://192.168.34.92:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://localhost:3001
 --------------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
```

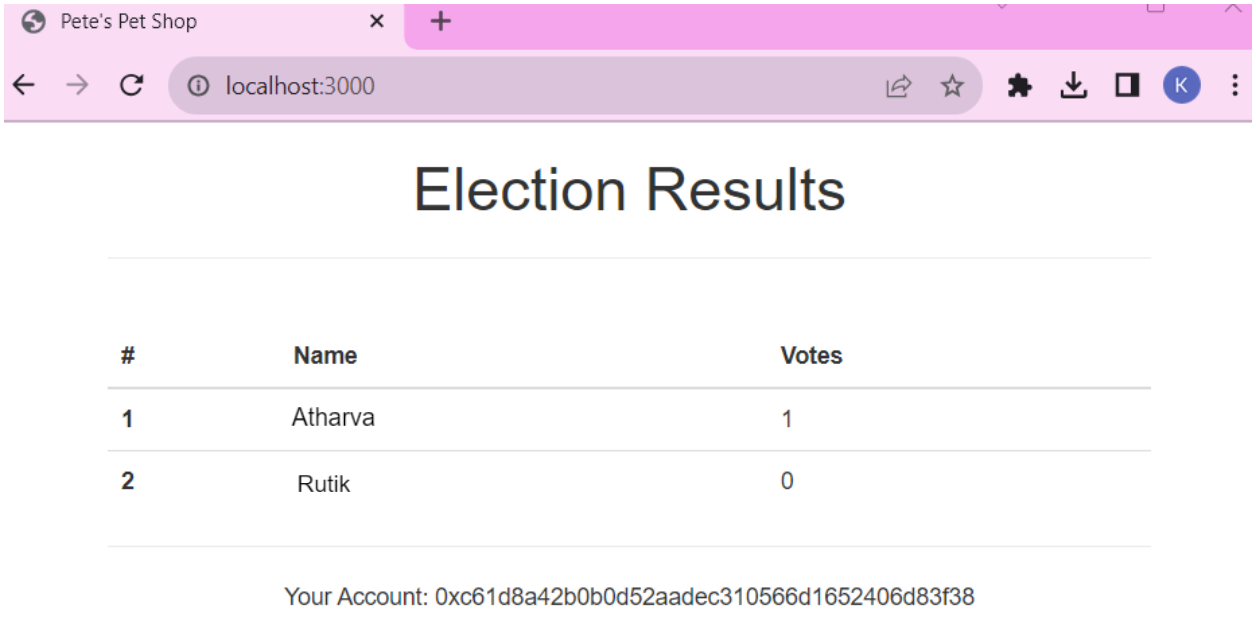
```
23.11.01 14:07:00 200 GET /index.html
23.11.01 14:07:00 200 GET /js/bootstrap.min.js
23.11.01 14:07:00 200 GET /css/bootstrap.min.css
23.11.01 14:07:00 200 GET /js/app.js
23.11.01 14:07:00 200 GET /js/truffle-contract.js
23.11.01 14:07:00 200 GET /js/web3.min.js
23.11.01 14:07:01 200 GET /Election.json
23.11.01 14:07:01 404 GET /favicon.ico
```



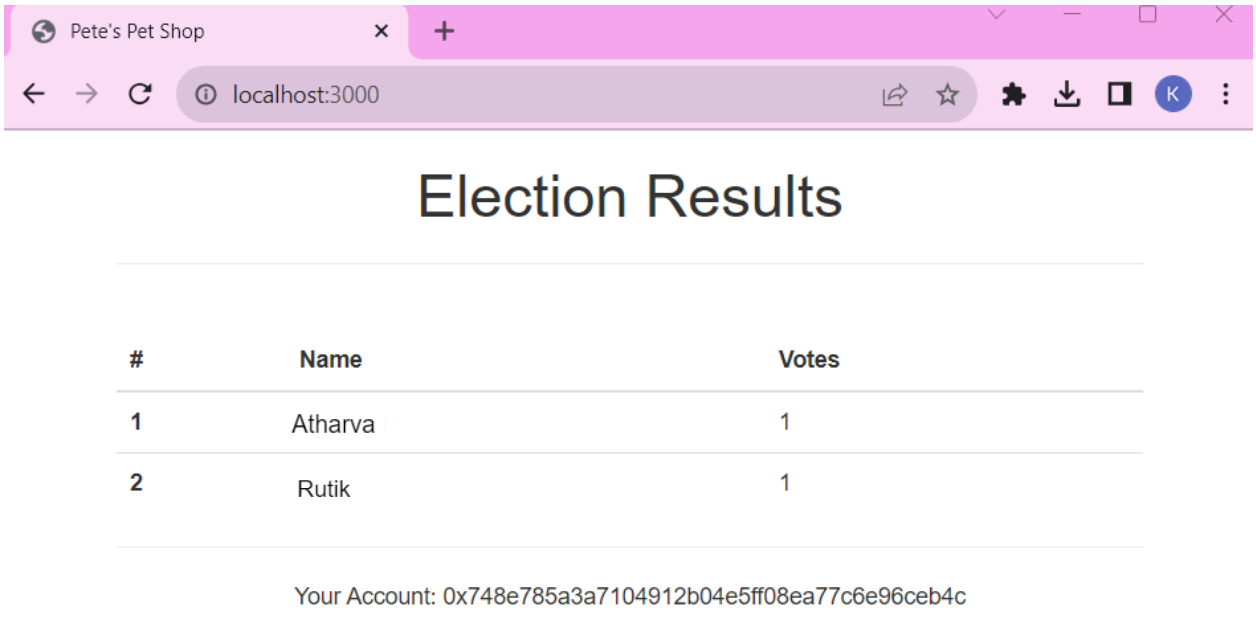

Pete's Pet Shop
localhost:3000
Election R
Name
1 Atharva
2 Rutik
Select Candidate
Atharva
Vote
Your Account: 0xc61d8a42b0b0d52aac
localhost
Account 2
0x19177...E243c
http://localhost:3000
0x19177...E243c : VOTE
DETAILS DATA HEX
Market
Gas (estimated)
0.0025322 0.0025322 ETH
Likely in < 30 seconds
Max fee: 0.00336378 ETH
Total
0.0025322 0.0025322 ETH
Amount + gas fee
Max amount: 0.00336378 ETH
Reject
Confirm


localhost:3000
Election R
Name
1 Atharva
2 Rutik
Your Account: 0xc61d8a42b0b0d52aac
Account 2
0x19177...E243c
http://localhost:3000
0x19177...E243c : VOTE
DETAILS DATA HEX
Market
Gas (estimated)
0.00258799
0.00258799 ETH
Likely in < 30 seconds
Max fee: 0.00343911 ETH
Total
0.00258799
0.00258799 ETH
Amount + gas fee
Max amount: 0.00343911 ETH
Reject
Confirm





We can also create another account and vote again:



Conclusion: I have successfully built DAPPS for the voting process .

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 10**	
Title of Lab Assignment: Create Dapps in Ethereum.		
DOP: 20-10-2024	**DOS: 28-10-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 10

Aim: Create Dapps in Ethereum.

Theory:

1. **Introduction**
    a. **Purpose**

    VacChain aims to provide a blockchain-based solution for managing and verifying vaccine certifications. The project offers peer-to-peer verification, secure data storage, and decentralized handling of vaccine-related records.

    b. **Scope**
        i. Blockchain technology ensures tamper-proof, transparent vaccine records.
        ii. Supports user registration, login, and certificate uploads.
        iii. Offers dashboards for users and administrators.
        iv. Peer nodes communicate for record synchronization.
        v. Microservices handle various operations independently for scalability.
    c. **Definitions, Acronyms, and Abbreviations**
        i. Blockchain: Distributed ledger for secure and decentralized data management.
        ii. EJS: Embedded JavaScript templating for generating HTML pages.
        iii. Node: A peer in the blockchain network responsible for validating and syncing data.
        iv. Microservice: A small, independent service managing specific tasks.
2. **Overall Description**
    a. **Product Perspective**

    VacChain integrates blockchain with traditional web services for managing vaccine certificates. The system consists of:

    - Client-side: User interfaces for interacting with the platform.
    - Server-side: Blockchain nodes and microservices that ensure backend data integrity and synchronization

    b. **Product Features**
        i. User Management: Registration, login, and profile management.
        ii. Certificate Management: Upload, store, and validate vaccine certificates.
        iii. Dashboard: Displays vaccine-related statistics and user details.

iv. Peer Communication: Nodes exchange information to maintain a consistent blockchain ledger.

c. User Classes and Characteristics

i. End Users: Individuals uploading and verifying vaccine certificates.
ii. Administrators: Manage platform operations, users, and blockchain nodes.
iii. Peers: Other nodes in the network handling data synchronization.

3. Functional Requirements

a. User Registration and Login

i. FR1: Users should be able to register using email and password.
ii. FR2: Users should be able to log in and access their dashboard.

b. Certificate Management

i. FR3: Users can upload vaccine certificates in PDF format.
ii. FR4: The system should validate uploaded certificates against a predefined schema.

c. Dashboard

i. FR5: Users and admins should have separate dashboards displaying relevant statistics.
ii. FR6: Admins can view and manage uploaded certificates.

d. Blockchain Node Communication

i. FR7: Peers should exchange data to ensure blockchain consistency.
ii. FR8: Microservices handle individual components like authentication and peer updates.

4. Non-Functional Requirements

a. Performance Requirements

i. The system should handle up to 100 concurrent users without performance degradation.
ii. Data synchronization between peers should complete within 2 seconds.

b. Security Requirements

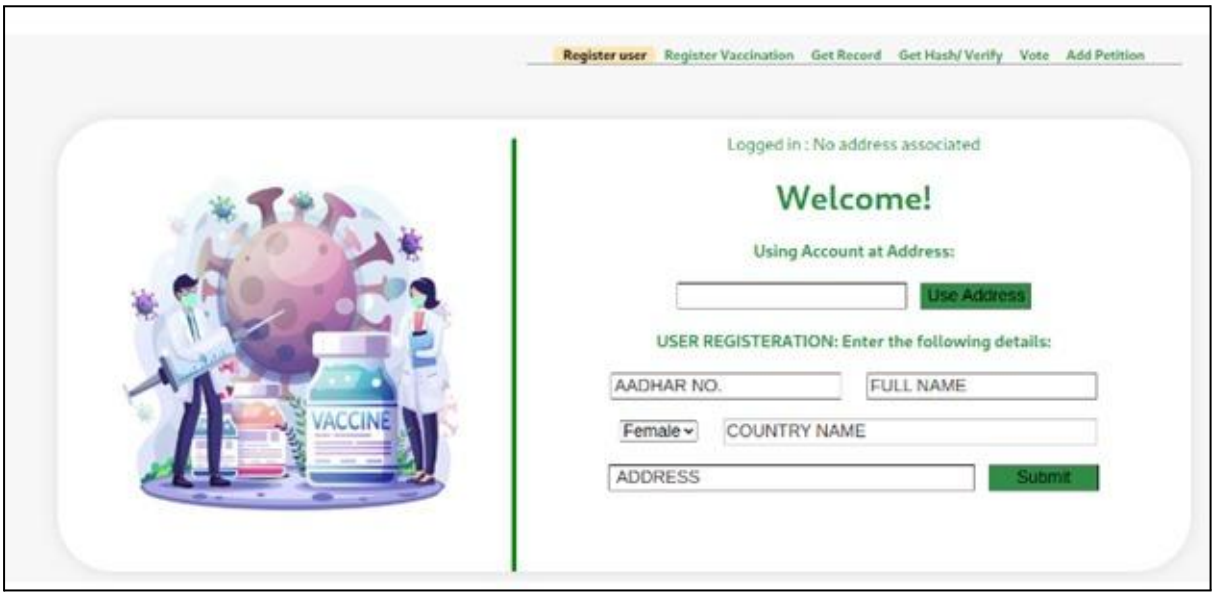
i. All sensitive data, such as passwords, should be encrypted.
ii. Blockchain nodes should only communicate over secure channels.

c. Usability Requirements

i. The UI should be responsive and work on desktops and mobile devices.
ii. Certificate uploads should provide feedback within 5 seconds.

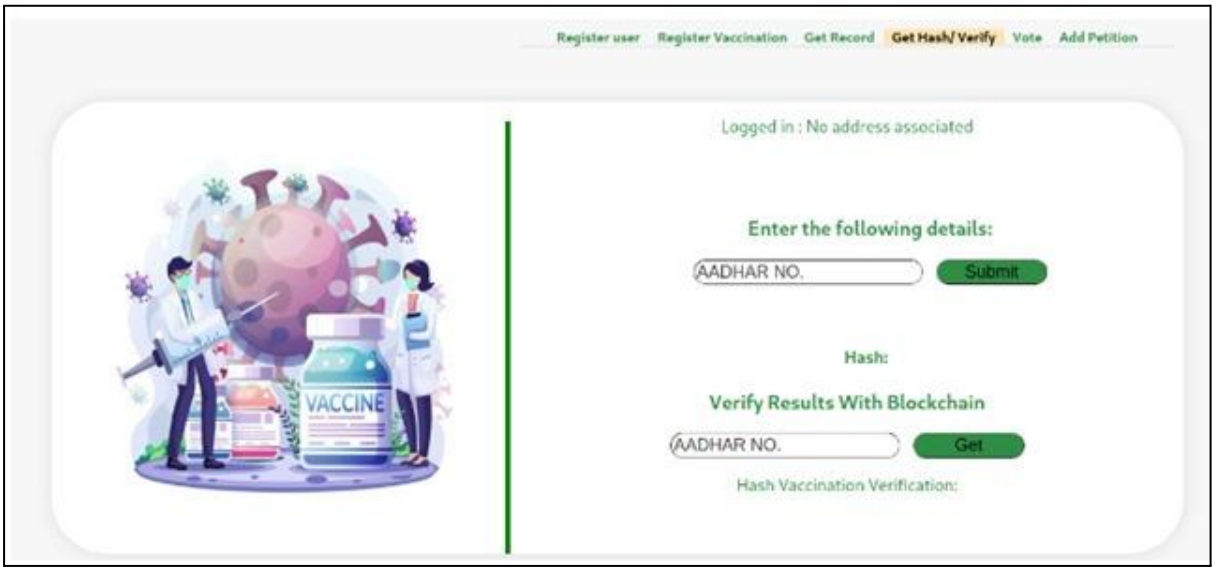
5. **System Architecture**
    - Frontend: EJS-based templates for login, registration, and dashboards.
    - Backend: Node.js-based microservices handling various functionalities.
    - Blockchain: Custom blockchain logic implemented for certificate storage
6. **Constraints**
    - The blockchain ledger cannot be altered once data is stored.
    - Each node must remain online to maintain network integrity
7. **Assumptions and Dependencies**
    - All users must have internet access to interact with the system.
    - The blockchain nodes need to be online for smooth operation.
8. **Appendix**
    - Technologies Used: Node.js, EJS, Blockchain, Microservices.
    - Future Enhancements: Support for mobile app integration, advanced analytics on vaccine data.

Output:




Register user
Register Vaccination
Get Record
Get Hash/ Verify
Vote
Add Petition
Logged in : No address associated
Register Vaccination
Enter Vaccination Details:
AADHAR NO.
VACCINATION NAME
BATCH ID
LOCATION
mm / dd / yyyy
Submit
VACCINE


Register user
Register Vaccination
Get Record
Get Hash/ Verify
Vote
Add Petition
Logged in : No address associated
Only Vaccination Authority/User
AADHAR NO.
Get
Click to view details!
VACCINE



Conclusion:

VacChain is a secure, blockchain-based system for managing vaccine certificates, providing tamper-proof storage and peer-to-peer verification. Users can register, upload certificates, and access dashboards, while administrators manage platform operations. The platform uses microservices for scalability and real-time data synchronization across blockchain nodes. Future enhancements include mobile support and analytics for expanded functionality.