Roll No. **42** Exam Seat No._____

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

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

Since 1962

CERTIFICATE

Certified that Mr. **CHRISTY PHILIP [ROLL NO: 42]** of **SYMCA-2B** has satisfactorily completed a course of the necessary experiments in **MCAE331 – Blockchain Lab** under the supervision of **Mrs. Vaishali Gatty** in the Institute of Technology in the academic year **2022- 2023.**

Principal Head of Department

Lab In-charge Subject Teacher

V.E.S. Institute of Technology,
Collector Colony, Chembur, Mumbai

Department of M.C.A

MCAE331 - BLOCKCHAIN LAB INDEX

Sr. No	Contents	Date Of Preparation	Date Of Submission	Marks	Sign
1	Implement ceaser cipher (symmetric encryption) and show the encryption as well as decryption process.	26/08/2022	30/08/2022	10	
2	Implement RSA and show the encryption as well as decryption process.	30/08/2022	13/09/2022	10	
3	To Implement SHA 256 Algorithm	06/09/2022	13/09/2022	10	
4	Implementation of Merkle Tree	13/09/2022	20/09/2022	10	
5	Install Geth Ethereum Node (Show installation steps also). Create a Genesis block and a private chain. Use geth commands to create user accounts, mine, transact etc.	27/09/2022	27/09/2022	10	
6	To implement the installation of ganache, nietamask and remix ide and deploy smart contract using injected web 3 environment.	04/10/2022	11/10/2022	10	
7	To study solidity and implement some examples of it.	18/10/2022	03/11/2022	10	
8	Smart Contract using Truffle Framework.	03/11/2022	22/11/2022	10	
9	Create Dapps in Ethereum	03/11/2022	22/11/2022	10	
10	Mini Project	03/11/2022	22/11/2022	10	

PRACTICAL 1

AIM: - Implement Ceaser Cipher (Symmetric Encryption) and show the encryption as well as decryption process.

THEORY: -

What is Cryptograpy?

Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”.

In Cryptography the techniques which are use 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 it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.

What is Symmetric Encryption

Symmetric encryption uses a single key to encrypt and decrypt. If you encrypt a zip file, then decrypt with the same key, you are using symmetric encryption. Symmetric encryption is also called “secret key” encryption because the key must be kept secret from third parties.

Ceaser Cipher

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter with a fixed number of positions down the alphabet.

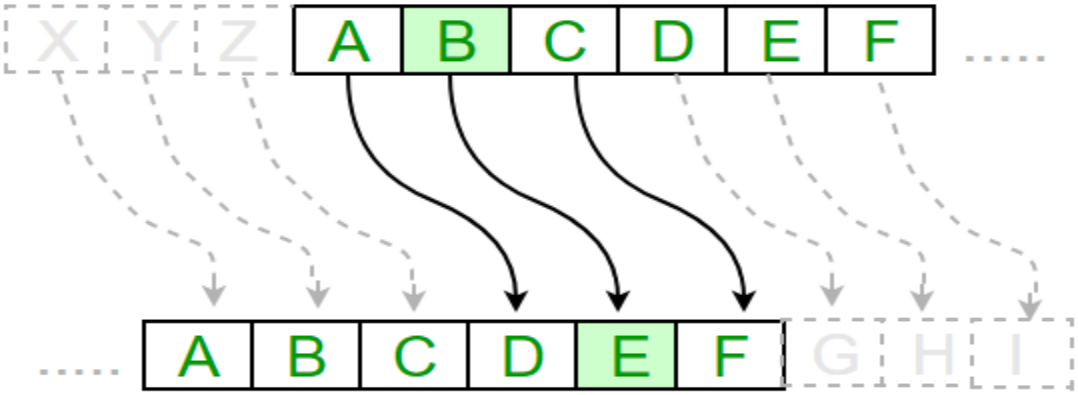
For example with a shift of 1, A would be replaced by B, B would become C, and so on.

The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.

Thus to cipher a given text we need an integer value, known as a shift which indicates the number of positions each letter of the text has been moved down.

The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25.

Encryption of a letter by a shift n can be described mathematically as.



Example: -

Text : ABCDEFGHIJKLMNOPQRSTUVWXYZ
Shift: 23
Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW

Text : ATTACKATONCE
Shift: 4
Cipher: EXXEGOEXSRGI

Algorithm for Caesar Cipher:

Input:

A String of lower case letters, called Text.

An Integer between 0-25 denoting the required shift.

Procedure:

Traverse the given text one character at a time .

For each character, transform the given character as per the rule, depending on whether we're encrypting or decrypting the text. Return the new string generated.

CODE: -

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


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

        if (char.isupper()):
            result += chr((ord(char) + shift-65) % 26 + 65)

        else:
            result += chr((ord(char) + shift - 97) % 26 + 97)

    return result




def decrypt(text,s):
    result = ""
```

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

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

        # Encrypt lowercase characters
        else:
            result += chr((ord(char) + s - 97) % 26 + 97)

    return result

shift=7
s = 26-shift
text="ATTACK"
text2=encrypt(text,shift)
print ("Text  : " + text)
print ("Shift : " + str(shift))
print ("Cipher: " + encrypt(text,shift))
print ("DeCipher: " + decrypt(text2,s))
```

OUTPUT: -

```
Text  : ATTACK
Shift : 7
Cipher: HAAHJR
DeCipher: ATTACK
```

CONCLUSION: -

From this practical I have learned to implement Ceaser Cipher.

PRACTICAL 2

AIM: - Implementation of RSA Algorithm(Asymmetric Encryption)

THEORY: -

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

An example of asymmetric cryptography :

A client (for example browser) sends its public key to the server and requests for some data.

The server encrypts the data using client's public key and sends the encrypted data.

Client receives this data and decrypts it.

Since this is asymmetric, nobody else except browser can decrypt the data even if a third party has public key of browser.

The idea! The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Algorithm:

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

N=p*q

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: Public key
The specified pair of numbers n and e forms the RSA public key and it is made public.

Step 4: Private Key
Private Key d is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −
ed = 1 mod (p-1) (q-1)
The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula
Consider a sender who sends the plain text message to someone whose public key is (n,e). To encrypt the plain text message in the given scenario, use the following syntax −
C = Pe mod n

Decryption Formula
The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver C has the private key d, the result modulus will be calculated as −
Plaintext = Cd mod n

CODE: -

```
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 = 12.0
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: -

```
Message data =  12.0
Encrypted data =  3.0
Original Message Sent =  12.0
```

CONCLUSION: -

From this practical I have learned to implement RSA Algorithm using Python.

PRACTICAL 3

AIM: - Implement SHA 256

THEORY: -

1) HASHING: -

Hash functions are extremely useful and appear in almost all information security applications.

A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.

Values returned by a hash function are called message digest or simply hash values.

2) SHA 256: -

SHA 256 is very complex, so we're not going to go into too much detail here. However, we can understand it from a broad perspective. FIPS-180 describes SHA algorithms as involving two essential stages:

1) Preprocessing — This is where the message is padded, broken down into smaller blocks, and initialization values are set.

2) Hash computation — This process involves a series of operations that result in a series of hash values. The 256-bit resulting hash digest we get at the very end is generated by computing these various hash values.

SHA 256 follows the steps given below:

First, data is converted into binary. Binary code uses 0s and 1s to store information. For example, the letter 'a' is written as '01000001' in this basic computer language.

The binary data is divided into blocks of 512 bits. If the block is smaller than 512, it'll be expanded to that size by adding bits of "padding." If it's larger, it'll be broken into blocks of 512 bits. (If the last block isn't exactly 512 bits, padding is added to the last block to make it 512 bits.)

The message is further divided into smaller blocks that are 32 bits each.

Sixty-four iterations (rounds) of compression functions are performed, wherein the hash values generated above are rotated in a specific pattern and additional data gets added.

New hash values are created from the output of the previous operations.
In the last round, one final 256-bit hash value is produced — this hash digest is the end product of SHA 256.
Check out this article on hash functions in cryptography and how they work for more information about how hashing algorithms work.
Final Thoughts on SHA 256 Algorithm
Even though we didn't get into the nitty-gritty of how the SHA 256 algorithm works in this article, you should now understand that SHA 256 is a very useful function. SHA 256 converts data into fixed-length, virtually irreversible hash values, and is mainly used to verify the authenticity data.

As we mentioned earlier, no one has been able to crack SHA 256 to date, and it's used in some of the most secure networks in the world. One day, SHA 256 might break, but for the moment, we can rely on it to keep our data safe.

3) LIBRARIES AND FUNCTIONS: -

1) Hashlib: -

The Python hashlib module is an interface for hashing messages easily. This contains numerous methods which will handle hashing any raw message in an encrypted format.
The core purpose of this module is to use a hash function on a string, and encrypt it so that it is very difficult to decrypt it.
Typically, an encrypted string is long enough such that it is almost impossible to get back the original string.

2) Encode: -

The encode() method encodes the string, using the specified encoding. If no encoding is specified, UTF-8 will be used.
Syntax: -
string.encode(encoding=encoding, errors=errors)

3) Hexdigest: -

hexdigest() : Returns the encoded data in hexadecimal format.

4) Clear(): -

Releases all resources used by the HashAlgorithm class.

(Inherited from HashAlgorithm)

5) ComputeHash(Byte[]): -

Computes the hash value for the specified byte array.

(Inherited from HashAlgorithm)

6) ComputeHash(Byte[], Int32, Int32): -

Computes the hash value for the specified region of the specified byte array.

(Inherited from HashAlgorithm)

7) ComputeHash(Stream): -

Computes the hash value for the specified Stream object.

(Inherited from HashAlgorithm)

8) ComputeHashAsync(Stream, CancellationToken) : -

Asynchronously computes the hash value for the specified Stream object.

(Inherited from HashAlgorithm)

9) Create(): -

Creates an instance of the default implementation of SHA256.

10) Create(String): -

Creates an instance of a specified implementation of SHA256.

11) Dispose(): -

Releases all resources used by the current instance of the HashAlgorithm class.

(Inherited from HashAlgorithm)

12) Dispose(Boolean): -

Releases the unmanaged resources used by the HashAlgorithm and optionally releases the managed resources.

(Inherited from HashAlgorithm)

CODE: -

```
import hashlib

str = "GeeksforGeeks"

# Encoding and sending to SHA256
result = hashlib.sha256(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA256 is : ")
print(result.hexdigest())

print ("\r")
```

OUTPUT: -

```
The hexadecimal equivalent of SHA256 is :
ab4b1a43c502cd686cc97a1af06a6077d5a92f843f0809aba83e0702ec5f0c95
```

CONCLUSION: -

From this practical I have learned to implement SHA 256 using Python.

PRACTICAL 4

AIM: - Implementation of Binary Tree and Merkle Tree using SHA-256

THEORY: -

BINARY TREE: -

A binary tree is a tree data structure in which each parent node can have at most two children. Each node of a binary tree consists of three items:

1) data item

2) address of left child

3) address of right child

Types of Binary Tree

1. Full Binary Tree

A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.

2. Perfect Binary Tree

A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.

3. Complete Binary Tree

A complete binary tree is just like a full binary tree, but with two major differences

1) Every level must be completely filled

2) All the leaf elements must lean towards the left.

3) The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.

4. Degenerate or Pathological Tree

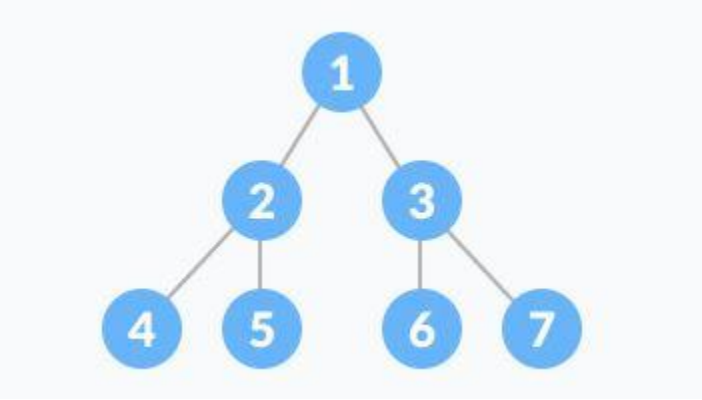
A degenerate or pathological tree is the tree having a single child either left or right.

5. Skewed Binary Tree

A skewed binary tree is a pathological/degenerate tree in which the tree is either dominated by the left nodes or the right nodes. Thus, there are two types of skewed binary tree: left-skewed binary tree and right-skewed binary tree.

6. Balanced Binary Tree

It is a type of binary tree in which the difference between the height of the left and the right subtree for each node is either 0 or 1.



MERKLE TREE: -

Merkle tree also known as hash tree is a data structure used for data verification and synchronization.

It is a tree data structure where each non-leaf node is a hash of it's child nodes. All the leaf nodes are at the same depth and are as far left as possible.

It maintains data integrity and uses hash functions for this purpose.

Hash Functions:

So before understanding how Merkle trees work, we need to understand how hash functions work.

A hash function maps an input to a fixed output and this output is called hash. The output is unique for every input and this enables fingerprinting of data. So, huge amounts of data can be easily identified through their hash.

This is a binary merkel tree, the top hash is a hash of the entire tree.

This structure of the tree allows efficient mapping of huge data and small changes made to the data can be easily identified.

If we want to know where data change has occurred then we can check if data is consistent with root hash and we will not have to traverse the whole structure but only a small part of the structure.

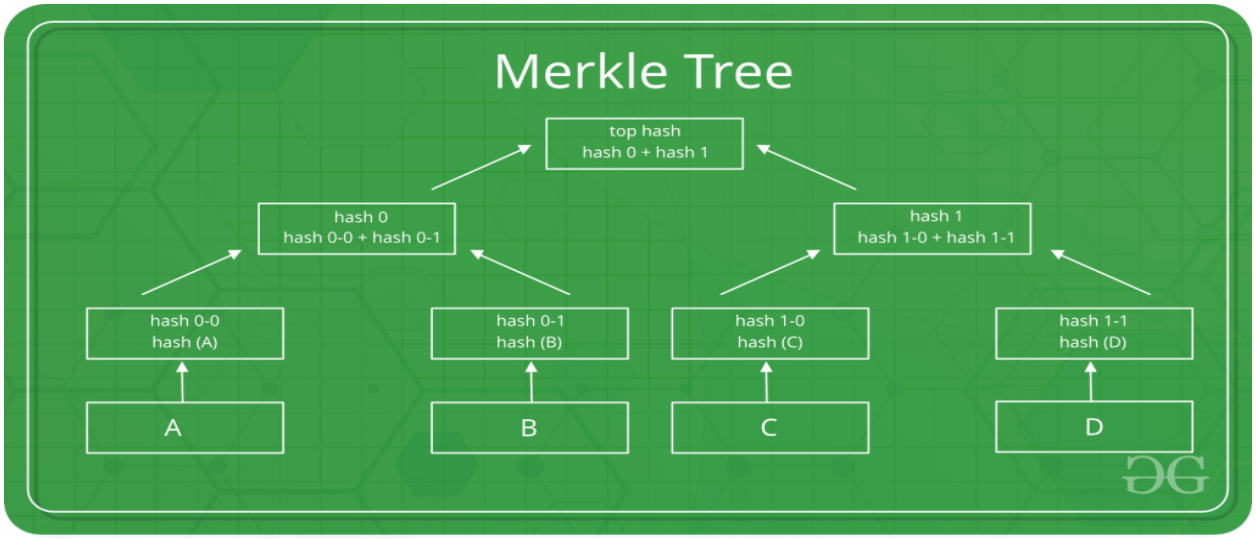
The root hash is used as the fingerprint for the entire data.

For a Binary Merkel tree

Operation	Complexity
Space	O(n)
Searching	O(logn)
Traversal	O(n)
Insertion	O(logn)
Deletion	O(logn)
Synchronization	O(logn)

APPLICATIONS:

- Merkle trees are useful in distributed systems where same data should exist in multiple places.
- Merkle trees can be used to check inconsistencies.
- Apache Cassandra uses Merkle trees to detect inconsistencies between replicas of entire databases.
- It is used in bitcoin and blockchain.



A) BINARY TREE: -

CODE: -

```
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
root = Node(27)
root.insert(14)
root.insert(35)
root.insert(10)
root.insert(19)
root.insert(31)
root.insert(42)
print(root.inorderTraversal(root))
```

OUTPUT: -

```
[10, 14, 19, 27, 31, 35, 42]
```

B) MERKLE TREE: -

CODE: -

```
from typing import List
import typing
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])# duplicate last elem if odd number of elements
            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 = self.__buildTreeRec(nodes[:half]).content+"+"+self.__buildTreeRec(nodes[half:]).content
        return Node(left, right, value,content)
    def printTree(self)-> None:
        self.__printTreeRec(self.root)
    def __printTreeRec(self, node)-> None:
        if node != None:
            if node.left != 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:
                elems = ["Mix", "Merkle", "Tree","From","Onur Atakan ULUSOY","https://github.com/onuratakan/mixmerkletree","GO"]
    print("Inputs: ")
    print(*elems, sep = " | ")
```

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

OUTPUT: -

```
Inputs:
Mix | Merkle | Tree | From | Onur Atakan ULUSOY | https://github.com/onuratakan/mixmerkletree | GO

Root Hash: 2d895361b378a8a7f6dfe918953e270fd6b7d9bd9e18ded42ce6b4375026d277

Left: 941ea3b5f29a54923b15e852d307e8f1ea719a72c1c90958182bddf4eadb200b
Right: c04ac4dc80287139319ed96eac4d0f4f2e27c7e349f7c848b0cd72508bc95868
Value: 2d895361b378a8a7f6dfe918953e270fd6b7d9bd9e18ded42ce6b4375026d277
Content: Mix+Merkle+Tree+From+Onur Atakan ULUSOY+https://github.com/onuratakan/mixmerkletree+GO+GO

Left: 0f0f51729ff9f7b5d2ebbed20ab898834d83f18334811a9016bc72a2a690035e
Right: 66edd3db394018a4b7e452e99c351734abe106e9033a17a83c009414a9ddcb59
Value: 941ea3b5f29a54923b15e852d307e8f1ea719a72c1c90958182bddf4eadb200b
Content: Mix+Merkle+Tree+From

Left: dce6eb4447ef655f33b7baeb0a16ab3d27257c65bf45f760a44b9bff8d5b6663
Right: 728beae3756e9ea17b15ecbc095a5c0d7bba2c3182b83606baf199560ba868ba
Value: 0f0f51729ff9f7b5d2ebbed20ab898834d83f18334811a9016bc72a2a690035e
Content: Mix+Merkle

Input
Value: dce6eb4447ef655f33b7baeb0a16ab3d27257c65bf45f760a44b9bff8d5b6663
Content: Mix

Input
Value: 728beae3756e9ea17b15ecbc095a5c0d7bba2c3182b83606baf199560ba868ba
Content: Merkle
```

Left: b85f5fabf97d3b18eab12b1746fcff04590986982ae5f205a171880c5c21202b
Right: 218197693424e0154cefc0af31aed96c084b987e08136e91d5528ddbb5461e24
Value: 66edd3db394018a4b7e452e99c351734abe106e9033a17a83c009414a9ddcb59
Content: Tree+From

Input
Value: b85f5fabf97d3b18eab12b1746fcff04590986982ae5f205a171880c5c21202b
Content: Tree

Input
Value: 218197693424e0154cefc0af31aed96c084b987e08136e91d5528ddbb5461e24
Content: From

Left: 32ba559410e9d1c432589550028f5e965cb84349d56a3b68145ba161ed3e128c
Right: e5cc95450ffd1c71004150074c32195dbf80666a10544fc8c965a305a8a26ee1
Value: c04ac4dc80287139319ed96eac4d0f4f2e27c7e349f7c848b0cd72508bc95868
Content: Onur Atakan ULUSOY+https://github.com/onuratakan/mixmerkletree+GO+GO

Left: b45211393b0857d40548cf48bb64c86fdb940dc94cd47445d852f5d522318b73
Right: 612bd0f794de2ea56722c4e659168bcbba5f5c172fa3ef3d9cb339cf26fdf193
Value: 32ba559410e9d1c432589550028f5e965cb84349d56a3b68145ba161ed3e128c
Content: Onur Atakan ULUSOY+https://github.com/onuratakan/mixmerkletree

Input
Value: b45211393b0857d40548cf48bb64c86fdb940dc94cd47445d852f5d522318b73
Content: Onur Atakan ULUSOY

Input
Value: 612bd0f794de2ea56722c4e659168bcbba5f5c172fa3ef3d9cb339cf26fdf193
Content: https://github.com/onuratakan/mixmerkletree

```
Left: a0f949497a74dde0c63a35cd9ab147012ffa007b019557ea5cecb9a243d0c5de
Right: a0f949497a74dde0c63a35cd9ab147012ffa007b019557ea5cecb9a243d0c5de
Value: e5cc95450ffd1c71004150074c32195dbf80666a10544fc8c965a305a8a26ee1
Content: GO+GO

Input
Value: a0f949497a74dde0c63a35cd9ab147012ffa007b019557ea5cecb9a243d0c5de
Content: GO

Input
Value: a0f949497a74dde0c63a35cd9ab147012ffa007b019557ea5cecb9a243d0c5de
Content: GO

None
```

CONCLUSION: -

From this practical I have learned to implement Binary Tree and Merkle Tree.

PRACTICAL 5

AIM: - Creating a ethereum node using Geth

THEORY: -

ETHERUM

At its core, Ethereum is a decentralized global software platform powered by blockchain technology. It is most commonly known for its native cryptocurrency, ether (ETH).

Ethereum can be used by anyone to create any secured digital technology. It has a token designed to pay for work done supporting the blockchain, but participants can also use it to pay for tangible goods and services if accepted.

Ethereum is designed to be scalable, programmable, secure, and decentralized. It is the blockchain of choice for developers and enterprises creating technology based upon it to change how many industries operate and how we go about our daily lives.

It natively supports smart contracts, an essential tool behind decentralized applications.

Many decentralized finance (DeFi) and other applications use smart contracts in conjunction with blockchain technology.

Learn more about Ethereum, its token ETH, and how they are an integral part of non-fungible tokens, decentralized finance, decentralized autonomous organizations, and the metaverse.

GETH

Geth Ethereum is the command-line interface for implementing an Ethereum node in Google's Go programming language. Geth serves as a node in the blockchain that helps the user to mine Ether and creates software that runs on Ethereum Virtual Machine. Geth helps to mine real ether, create contracts, make transactions, transfer funds between addresses, etc. Geth is supported on almost all operating systems like Linux, macOS, and Windows.

CODE: -

genesis.json

```
{

"config": {

"chainId": 4777,

"homesteadBlock": 0,

"eip150Block": 0,

"eip155Block": 0,

"eip158Block": 0

},

"alloc" : {},

"difficulty" : "0x400",

"extraData" : "",

"gasLimit" : "0x7A1200",

"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",

"timestamp" : "0x00"

}
{

"coinbase" : "0x0000000000000000000000000000000000000001",

"difficulty" : "0x20000",

"extraData" : "",
```

"gasLimit" : "0x2fefd8",

"nonce" : "0x0000000000000042",

"mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

"parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",

"timestamp" : "0x00",

"alloc": {},

"config": {

"chainId": 15,

"homesteadBlock": 0,

"eip155Block": 0,

"eip158Block": 0,

"eip150Block": 0

}

}

OUTPUT: -

Note: -

Create a new folder on your desktop called “Private Chain”.

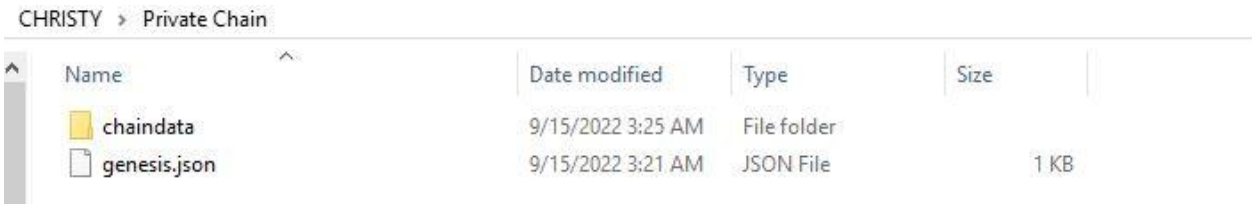
Open command prompt in this folder and create a data directory folder for our chaindata by typing “Mkdir chaindata”.

Next, we need to create and save our genesis.json block in our Private Chain folder, as the genesis block will be used to initialize our private network and store data in the

data directory folder “chaindata”.

Open up notepad, copy & paste the code below into a new file called “genesis.json” and save this file in our Private Chain folder.

Path should be



Server

```
E:\CHRISTY\Private Chain>geth --datadir chaindata init genesis.json
```

```
E:\CHRISTY\Private Chain>geth --datadir=./chaindata/
```

Client

New terminal should be opened path is not necessary

```
C:\Users\user>geth attach ipc:\\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.25-stable-69568c55/windows-amd64/go1.18.5
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
 datadir: E:\CHRISTY\Private Chain\chaindata
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> eth.accounts
[]
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xde3d94c337d5db91cf8b37b49729e33ab18eda63"
> eth.accounts
["0xde3d94c337d5db91cf8b37b49729e33ab18eda63"]
> eth.coinbase
"0xde3d94c337d5db91cf8b37b49729e33ab18eda63"
> eth.getBalance(eth.accounts[0])
0
```

```
> miner.start()
null
>
```

On server side

```
45.674ms
INFO [09-15|04:04:09.202] ⛏ ⛓ mined potential block                  number=2725 hash=27ff8f..f5099e
INFO [09-15|04:04:09.392] Successfully sealed new block           number=2726 sealhash=dda257..f0f69e hash=a18e00..eae088 elapsed=235.629ms
INFO [09-15|04:04:09.392] ⛏ ⛓ block reached canonical chain          number=2719 hash=309e0a..8b565a
INFO [09-15|04:04:09.403] Commit new sealing work                 number=2727 sealhash=1b3c18..b541f5 uncles=0 txs=0 gas=0     fees=0       elapsed=
11.495ms
INFO [09-15|04:04:09.423] Commit new sealing work                 number=2727 sealhash=1b3c18..b541f5 uncles=0 txs=0 gas=0     fees=0       elapsed=
31.371ms
INFO [09-15|04:04:09.429] ⛏ ⛓ mined potential block                  number=2726 hash=a18e00..eae088
INFO [09-15|04:04:09.549] Successfully sealed new block           number=2727 sealhash=1b3c18..b541f5 hash=0ce0c4..dd562c elapsed=156.420ms
INFO [09-15|04:04:09.549] ⛏ ⛓ block reached canonical chain          number=2720 hash=6f7e3f..9fca29
INFO [09-15|04:04:09.567] Commit new sealing work                 number=2728 sealhash=5beaca..0d64c3 uncles=0 txs=0 gas=0     fees=0       elapsed=
18.829ms
```

```
> miner.stop()
null
```

On server side

```
INFO [09-15|04:04:32.189] Looking for peers                        peercount=0 tried=7  static=0
INFO [09-15|04:04:42.224] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:04:52.249] Looking for peers                        peercount=0 tried=5  static=0
INFO [09-15|04:05:02.307] Looking for peers                        peercount=0 tried=7  static=0
INFO [09-15|04:05:12.324] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:05:22.333] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:05:32.356] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:05:42.366] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:05:52.379] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:06:02.408] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:06:12.426] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:06:22.480] Looking for peers                        peercount=0 tried=6  static=0
INFO [09-15|04:06:32.513] Looking for peers                        peercount=0 tried=5  static=0
INFO [09-15|04:06:42.529] Looking for peers                        peercount=0 tried=7  static=0
```

Client

```
> eth.getBalance(eth.accounts[0])
1.2815e+22
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x1cfc0d21b7424cec89efd83991ef0d21281ff0bd"
> eth.blockNumber
2563

> personal.unlockAccount(eth.accounts[0])
Unlock account 0xde3d94c337d5db91cf8b37b49729e33ab18eda63
Passphrase:
true
```

```
> eth.sendTransaction({from:eth.coinbase,to:eth.accounts[1],value:web3.toWei(10,"ether")})
"0x3def803a823dab1df9724e88e1fb49f42130daf8c1ddd9a9fdd922d2216726e9"
> miner.start()
null
```

On server side

```
INFO [09-15|04:04:20.331] Successfully sealed new block           number=2736 sealhash=9606ad..27cda7 hash=5fc228..e2cfaf elapsed=958.739ms
INFO [09-15|04:04:20.331] □ □□ block reached canonical chain          number=2729 hash=c27048..422376
INFO [09-15|04:04:20.350] Commit new sealing work                 number=2737 sealhash=496cab..d3aa59 uncles=0 txs=0 gas=0     fees=0       elapsed=
18.327ms
INFO [09-15|04:04:20.353] □ □□ mined potential block                  number=2736 hash=5fc228..e2cfaf
INFO [09-15|04:04:20.391] Commit new sealing work                 number=2737 sealhash=496cab..d3aa59 uncles=0 txs=0 gas=0     fees=0       elapsed=
59.720ms
INFO [09-15|04:04:22.147] Looking for peers                       peercount=0 tried=5  static=0
INFO [09-15|04:04:23.169] Successfully sealed new block           number=2737 sealhash=496cab..d3aa59 hash=721112..316893 elapsed=2.837s
INFO [09-15|04:04:23.169] □ □□ block reached canonical chain          number=2730 hash=02620c..0323cd
INFO [09-15|04:04:23.191] Commit new sealing work                 number=2738 sealhash=092b42..c4d97e uncles=0 txs=0 gas=0     fees=0       elapsed=
```

Client

```
> miner.stop()
null
```

On server side

```
INFO [09-15|04:27:45.275] Looking for peers                       peercount=0 tried=6  static=0
INFO [09-15|04:27:55.311] Looking for peers                       peercount=0 tried=5  static=0
INFO [09-15|04:28:05.317] Looking for peers                       peercount=0 tried=7  static=0
INFO [09-15|04:28:15.375] Looking for peers                       peercount=0 tried=6  static=0
INFO [09-15|04:28:25.403] Looking for peers                       peercount=0 tried=6  static=0
INFO [09-15|04:28:35.454] Looking for peers                       peercount=0 tried=5  static=0
INFO [09-15|04:28:45.490] Looking for peers                       peercount=0 tried=5  static=0
INFO [09-15|04:28:55.525] Looking for peers                       peercount=0 tried=6  static=0
INFO [09-15|04:29:05.556] Looking for peers                       peercount=0 tried=7  static=0
INFO [09-15|04:29:15.579] Looking for peers                       peercount=0 tried=6  static=0
```

Client

```
> web3.fromWei(eth.getBalance(eth.accounts[1]),"ether")
10
> eth.getBlock("latest")
{
  difficulty: 495509,
  extraData: "0xda83010a19846765746888676f312e31382e358777696e646f7773",
  gasLimit: 30000000,
  gasUsed: 0,
  hash: "0xa52741c44fcdf026d925101f91d550a5924d00669229f1ca67ff02c5a1ec328b",
  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000",
  miner: "0xde3d94c337d5db91cf8b37b49729e33ab18eda63",
  mixHash: "0x299e062a2964a55be94aec622d793c4b50ed0230e73fd94af55c9f5e486bda75",
  nonce: "0x7d7dff2e05441e48",
  number: 2738,
  parentHash: "0x721112ad28b1f807b6436d68652d334fca87c1835e2794a74e7f7521b6316893",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 541,
  stateRoot: "0x1b7c113abf6fe27ae13ab89b52e5ca91fc690979a8e12289e5bddd88a6b5774d",
  timestamp: 1663195621,
  totalDifficulty: 749799710,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

```
To exit, press ctrl-d or type exit
> eth.accounts
["0xde3d94c337d5db91cf8b37b49729e33ab18eda63", "0x1cfc0d21b7424cec89efd83991ef0d21281ff0bd"]
> eth.getTransaction("0x3def803a823dab1df9724e88e1fb49f42130daf8c1ddd9a9fdd922d2216726e9")
{
  blockHash: "0x24eacf5a1460981912b9344458e2c4bf9b9a2580dd57259bb6a28bdcb6bc6198",
  blockNumber: 2564,
  chainId: "0x12a9",
  from: "0xde3d94c337d5db91cf8b37b49729e33ab18eda63",
  gas: 21000,
  gasPrice: 1000000000,
  hash: "0x3def803a823dab1df9724e88e1fb49f42130daf8c1ddd9a9fdd922d2216726e9",
  input: "0x",
  nonce: 0,
  r: "0x6e128aeaaa46f392ecc91578534547bcc005db1ba55ab93267a51c818b650ce9",
  s: "0x2e23de94fcfd540bf35c560764cff12470bd6b9a2fc643f5be2020b4410677da",
  to: "0x1cfc0d21b7424cec89efd83991ef0d21281ff0bd",
  transactionIndex: 0,
  type: "0x0",
  v: "0x2575",
  value: 10000000000000000000
}
>
```

CONCLUSION: -

From this practical I have learned to implementing a ethereum node using Geth

PRACTICAL 6

AIM: - TO IMPLEMENT THE INSTALLATION OF GANACHE, METAMASK AND REMIX IDE AND DEPLOY SMART CONTRACT USING INJECTED WEB 3 ENVIRONMENT

THEORY: -

1) Ganache:

Ganache is used for setting up a personal Ethereum Blockchain for testing your Solidity contracts. It provides more features when compared to Remix. Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps in a safe and deterministic environment. On the other hand, Ganache is a high-end development tool used to run your own local blockchain for both Ethereum and Corda dApp development. Ganache is helpful in all parts of the development process. The local chain allows you to develop, deploy and test your projects and smart contracts in a deterministic and safe environment. There are two different “versions” of Ganache, one desktop application, and one command-line tool. The desktop application is called Ganache UI, and it supports development for both Ethereum and Corda; meanwhile, the command-line tool is called ganache-CLI, which solely supports Ethereum development. Furthermore, all the different versions of Ganache are available for Mac, Windows, and Linux.

2) Metamask:

MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications. MetaMask is developed by ConsenSys Software Inc., a blockchain software company focusing on Ethereum-based tools and infrastructure. MetaMask allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser.

3) Remix IDE:

Remix IDE (Integrated Development Environment) is a web application that can be used to write, debug, and deploy Ethereum Smart Contracts. Smart contract is a technology that consist in formalize contract rules through an algorithm, this code is written in solidity language, and is executed in a data decentralized platform named blockchain, which is simulated in the IDE remix ethereum, allowing automatic events, real time information, transparency, rastreability, immutability and data security.

STEPS: -

1) Download Ganache from trufflesuite.com.

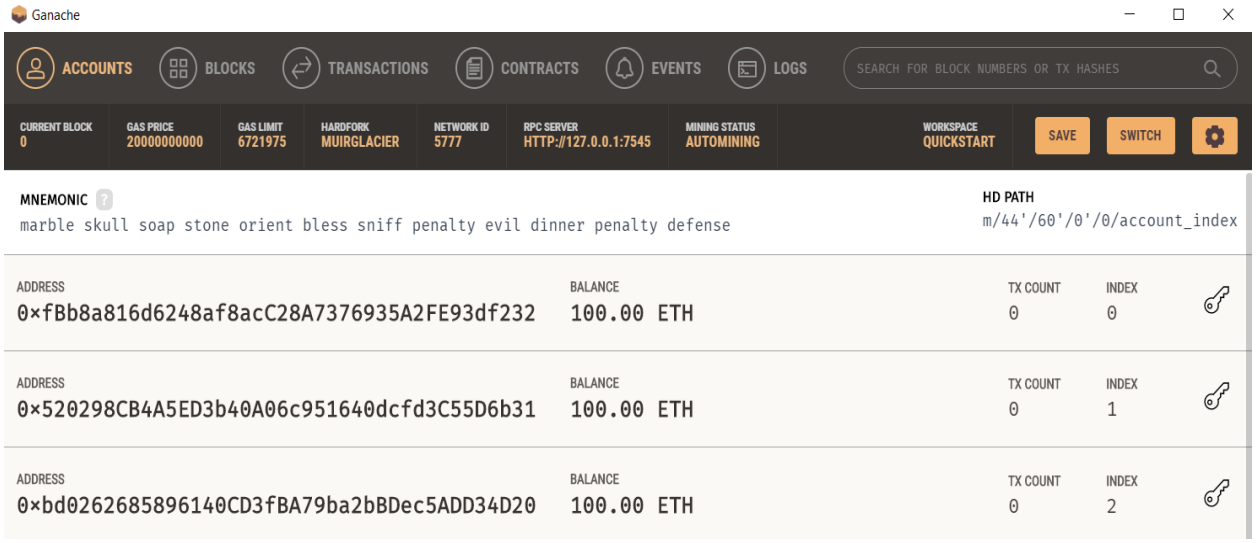
2) Add Metamask Extension in your Browser.

3) If you are new then go for creating a wallet in the Metamask which is now added in your Browser as an extension or already having a wallet then go for import

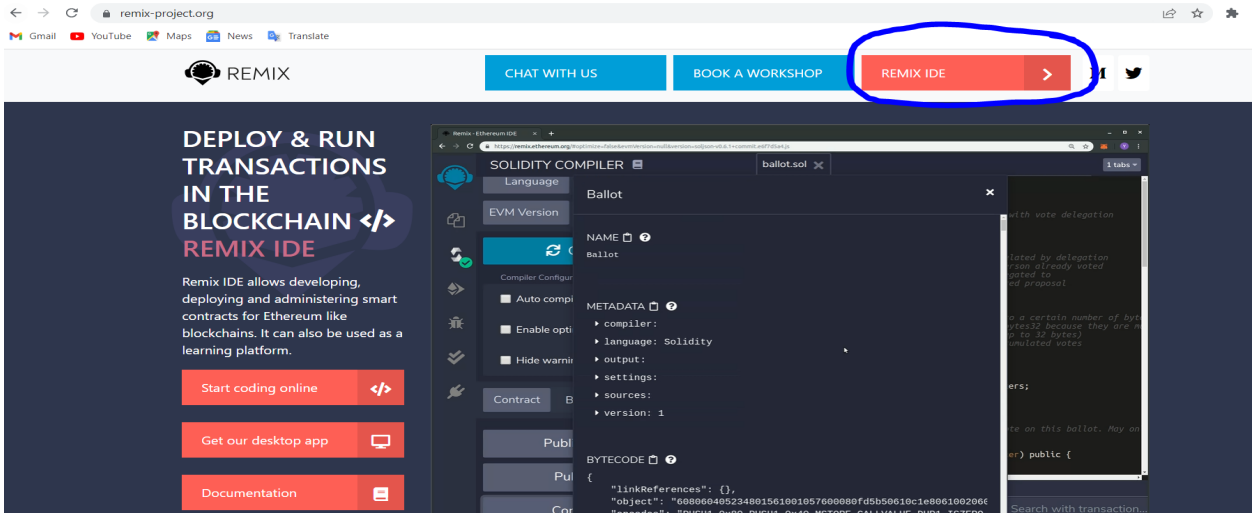
4) Creating the wallet will include following steps

- Enter password(remember the password)
- After the password is entered click Next
- A Secret Recovery Phrase(Click on Reveal Words and take a note of the phrase)
- Now choose the words in the phrase as you click Next
- After Confirming Phrase you will be shone Congratulations you have all done message.

5) Launch the Ganache by clicking Quickstart below UI is shown

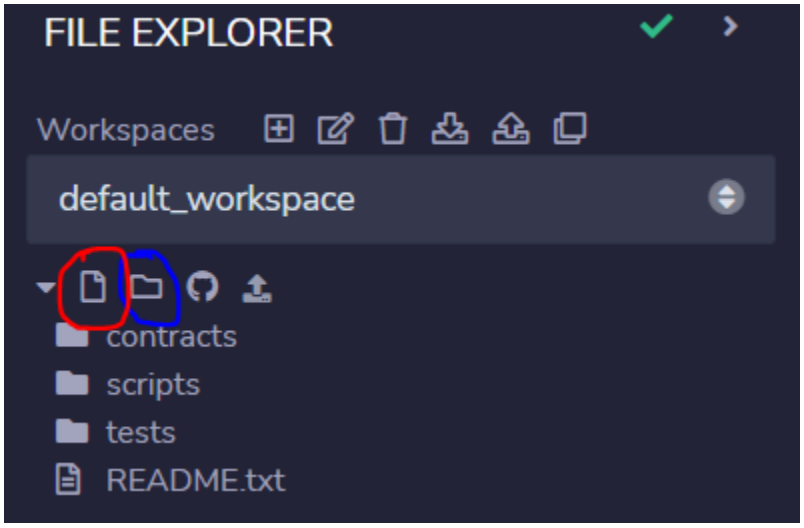


6) Choose the below figure circled option REMIX IDE.



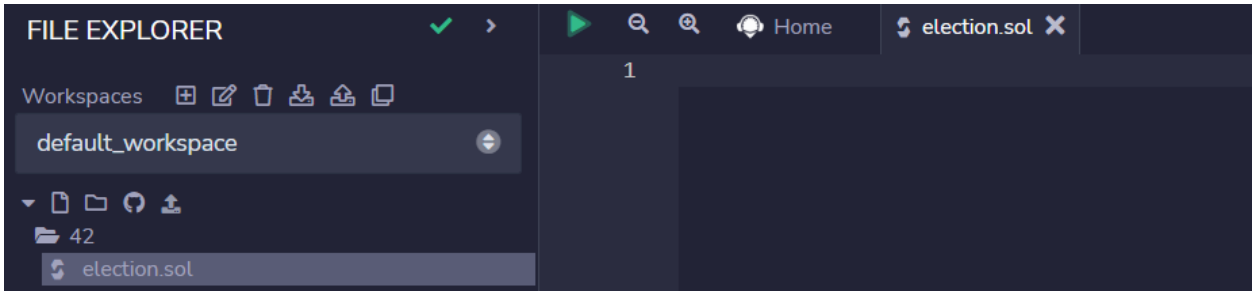
7) Create a folder in the IDE

8) Create a folder with a name and inside it create a file(For creating folder click on Blue circled icon and for File click on Red circled icon)

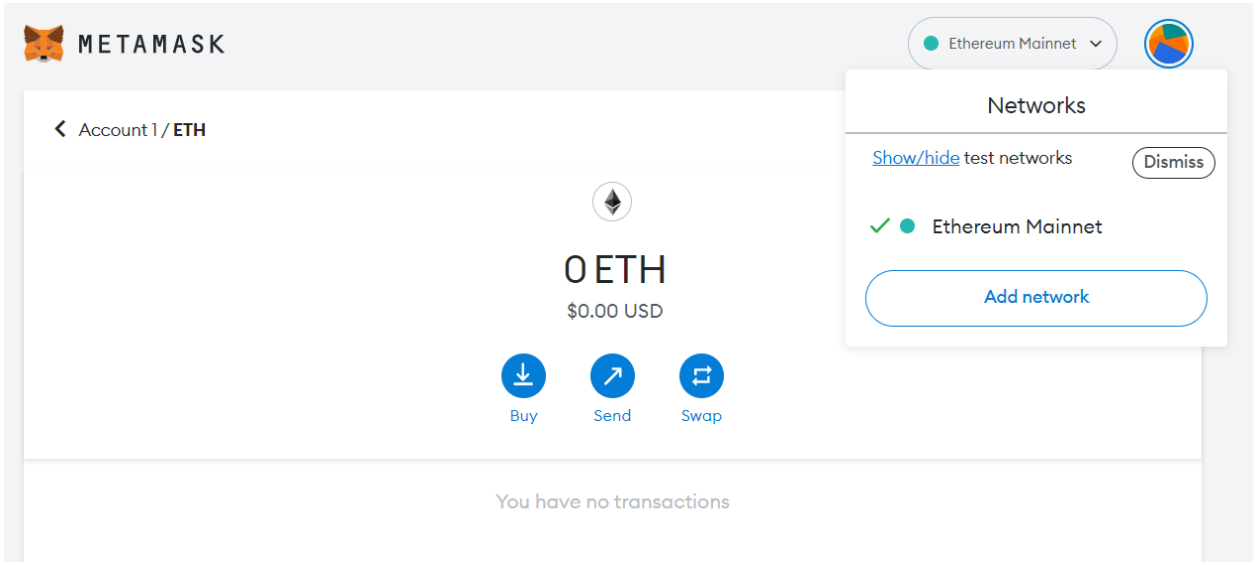


9) Save the file with .sol extension

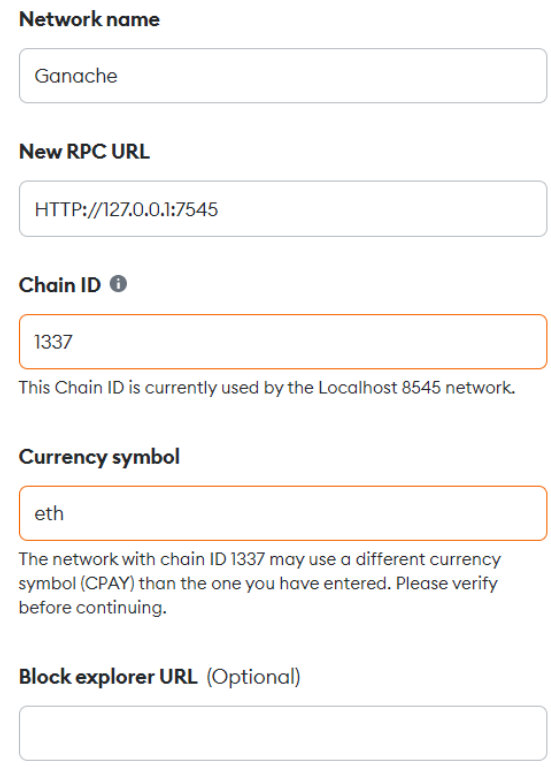
10) After the file is created the editor will get open.



11) Now come to below UI do add network.

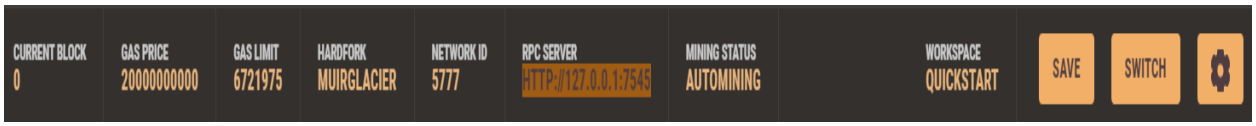


12) Enter the details as follow after doing add network.



Name: - Ganache

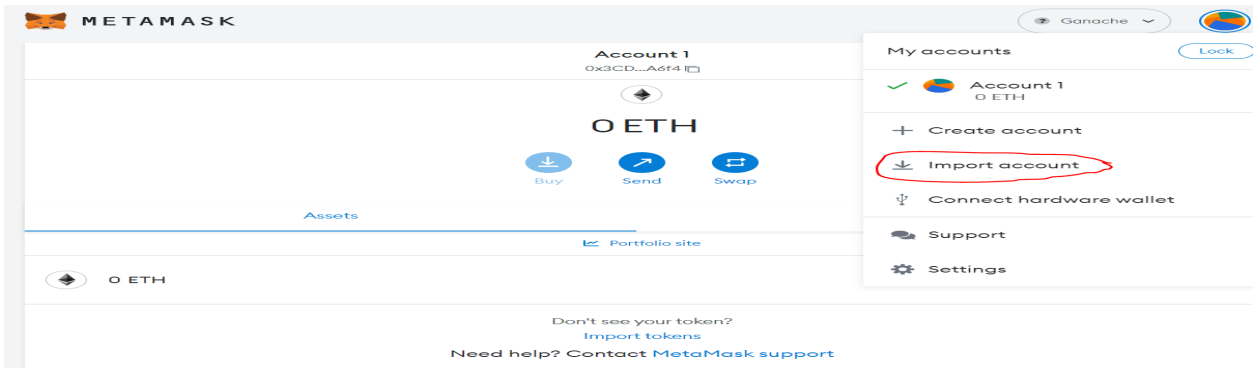
RPC URL will be getting from Ganache refer below image in which RPC Server is highlighted



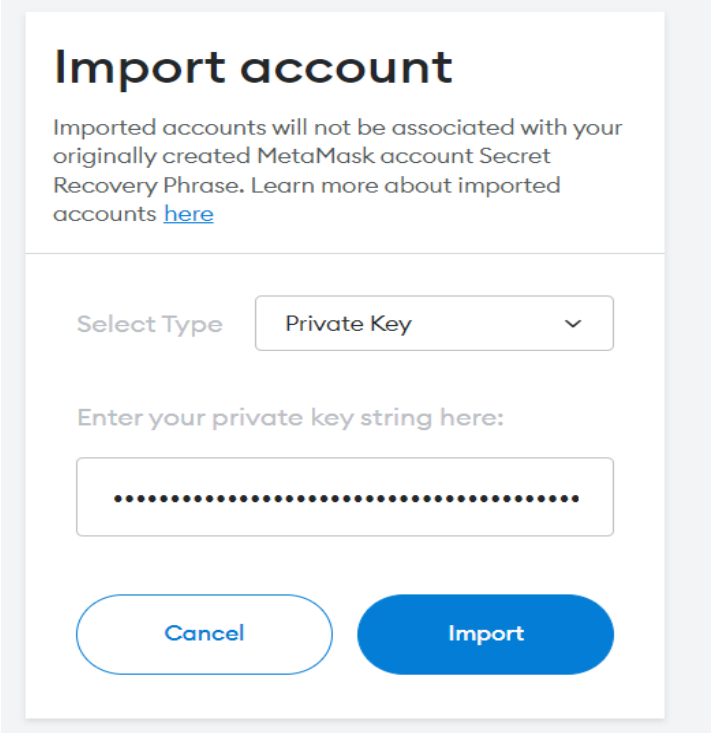
Chain ID: - 1337(Keep it same)

Currency Symbol: - eth

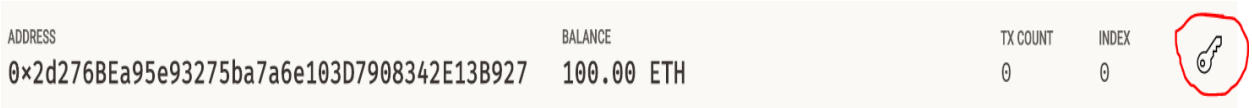
13) Now we have to Import the account by choosing following option



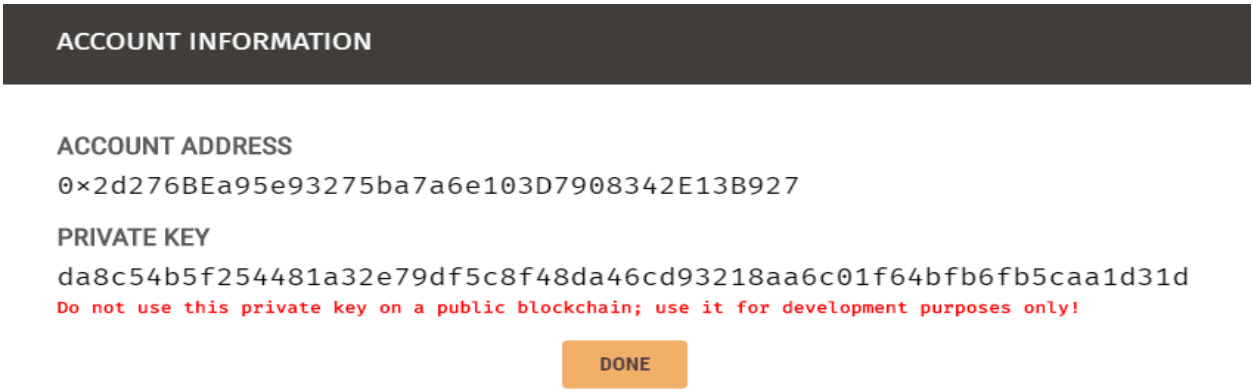
14) This prompt appears



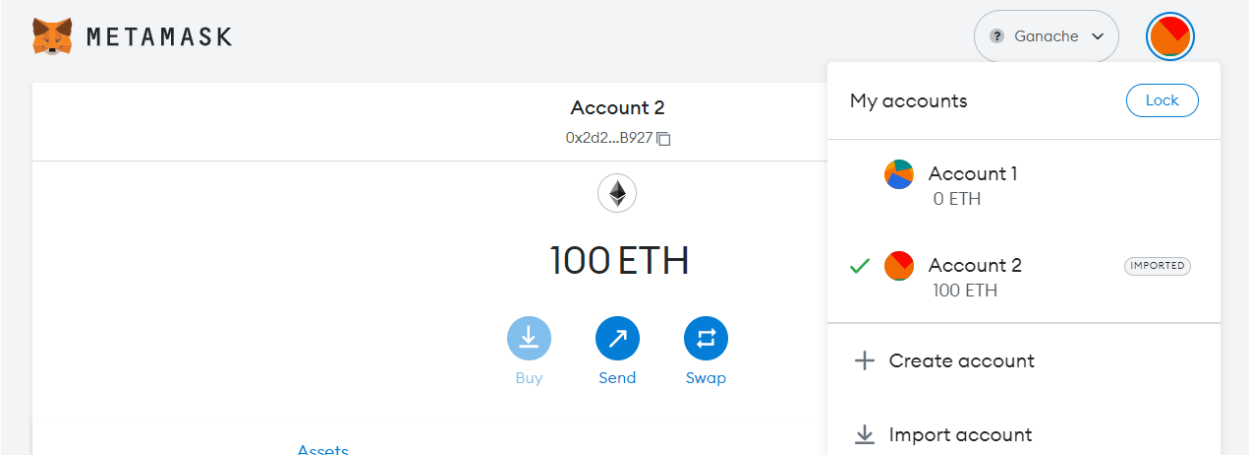
15) Goto Ganache click on the circled icon of key



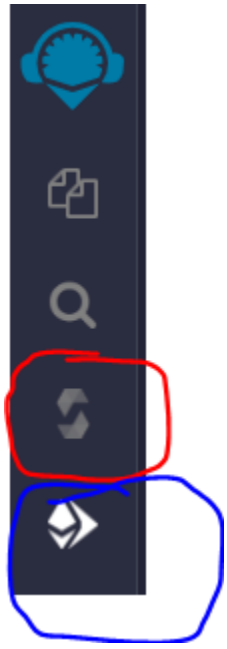
16) Copy paste the private key as shown in below image and paste it in Enter your private key string here which is shown in Step 14.



17) By default the account with the Ethers is chosen.



18) Now go to REMIX IDE

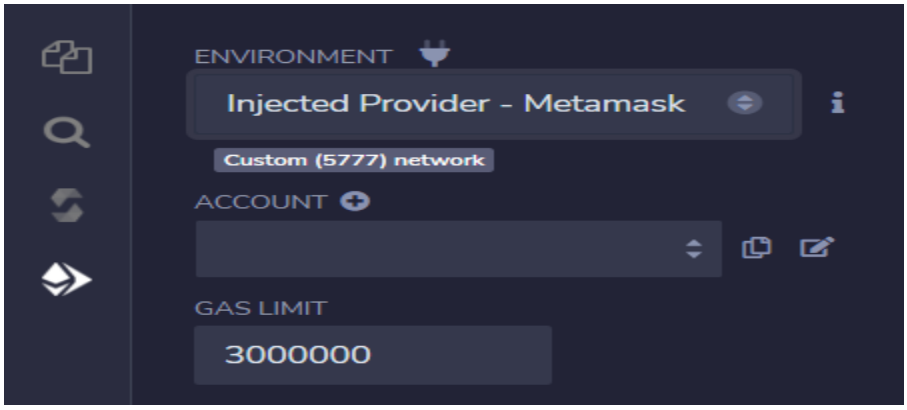


Red Circled-Compiling Code

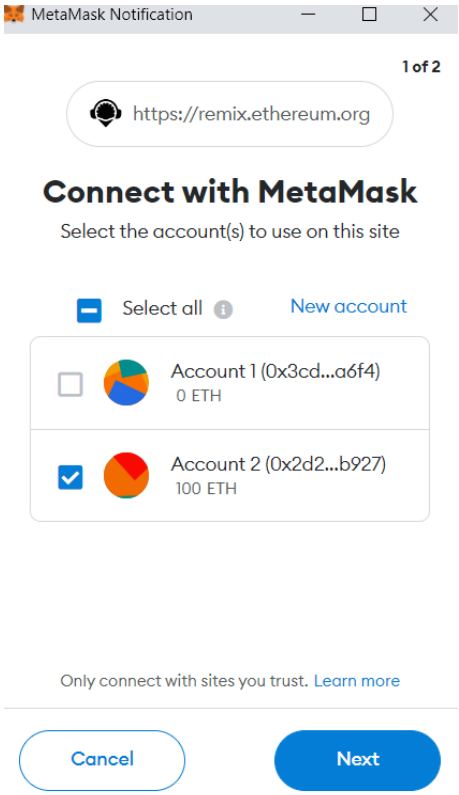
Blue Circled-Deployment

19) Select the account follow the image to select the account

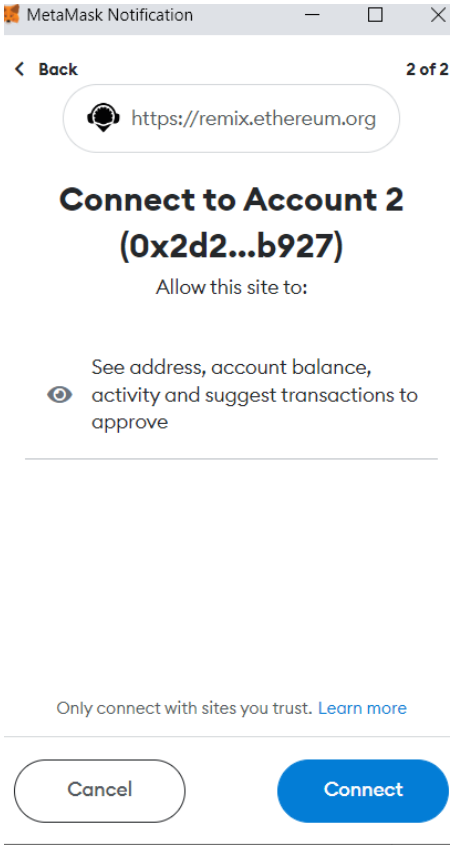
Environment should have Injected Provider-Metamask selected



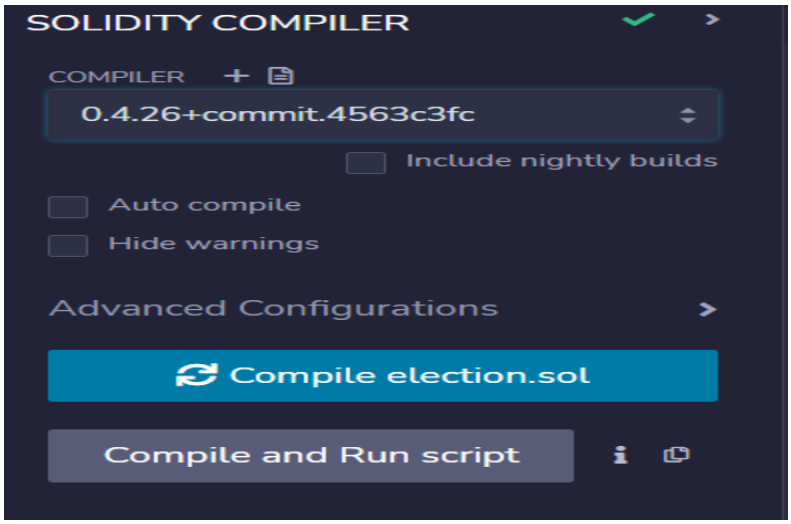
The prompt appears on the right hand side of your desktop.



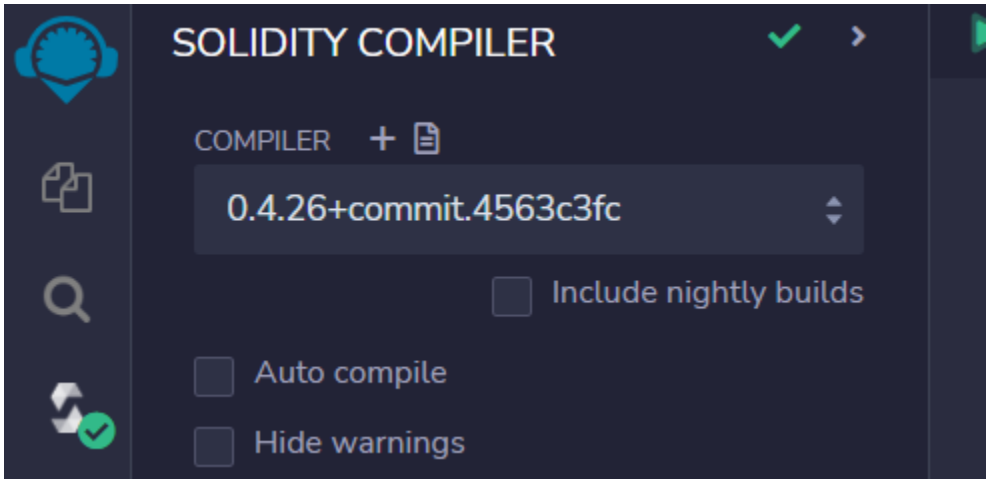
20) Now by clicking Next you will be shown prompt as in below image click Connect.



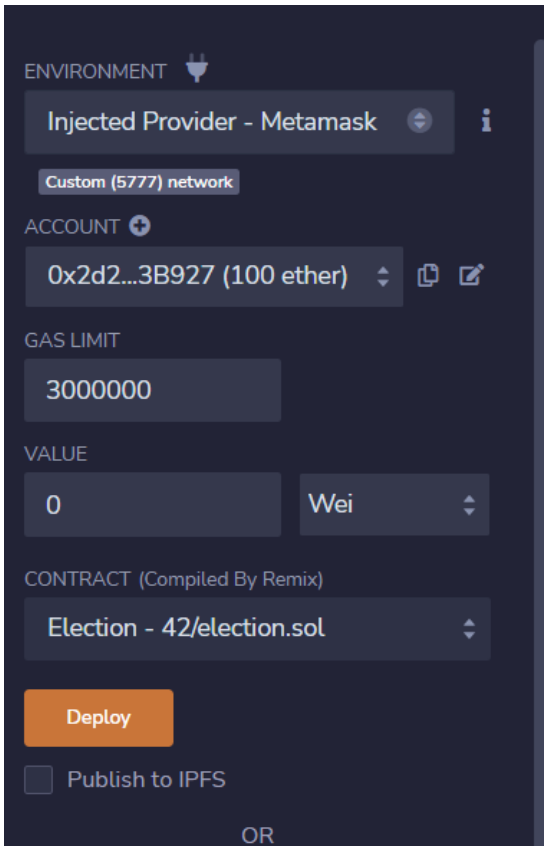
21) Now in REMIX IDE WE need to compile the code(which is in the CODE Section) so for the code used in this practical use the compiler version of the below image.



22) By the Green tick on compiler shows error in code is not been detected



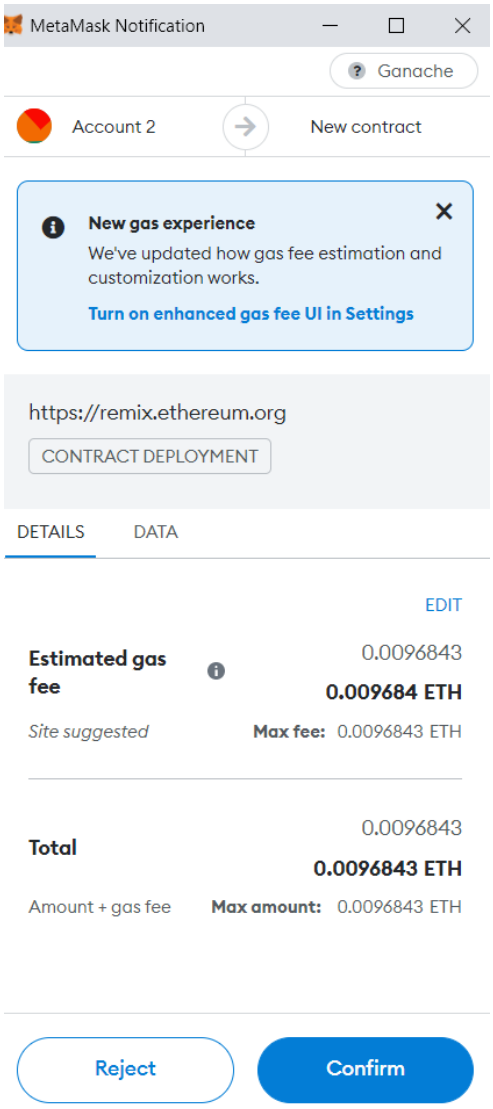
23) Now goto the deploy section and click Deploy.



1) Environment: - Injected Provider-Metamask

2) Account- The Account which we created

24) Deploying will deduct some amount of Ethers.



25) After clicking Confirm in REMI IDE Goto deploy section Now the testing section is shown in OUTPUT.

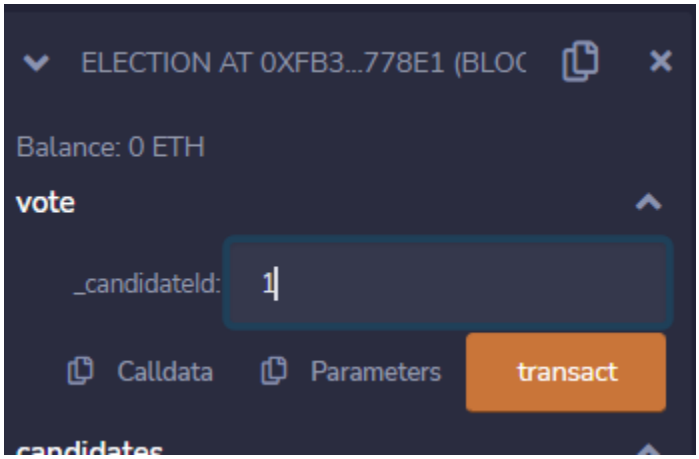
CODE: -

```
pragma solidity ^0.4.2;
contract Election {
// Model a Candidate
struct Candidate {
uint id;
string name;
```

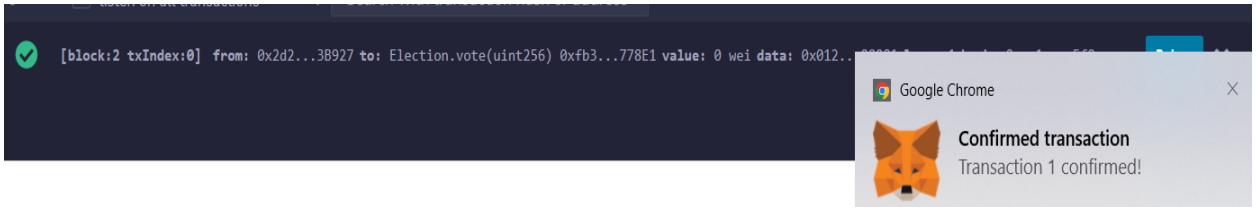
```
uint voteCount;
}
// Store accounts that have voted
mapping(address => bool) public voters;
// Store Candidates
// Fetch Candidate
mapping(uint => Candidate) public candidates;
// Store Candidates Count
uint public candidatesCount;

// voted event
event votedEvent (
uint indexed _candidateId
);
constructor() public {
addCandidate("N MODI, BJP");
addCandidate("A kejriwal, AAP");
addCandidate("Rahul G, Congress");
addCandidate("Nikhil, JDS");

}

function addCandidate (string _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: -

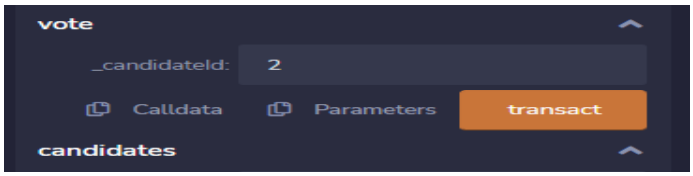
1) In vote section choose candidate id after clicking confirm on metamask right hand side prompt

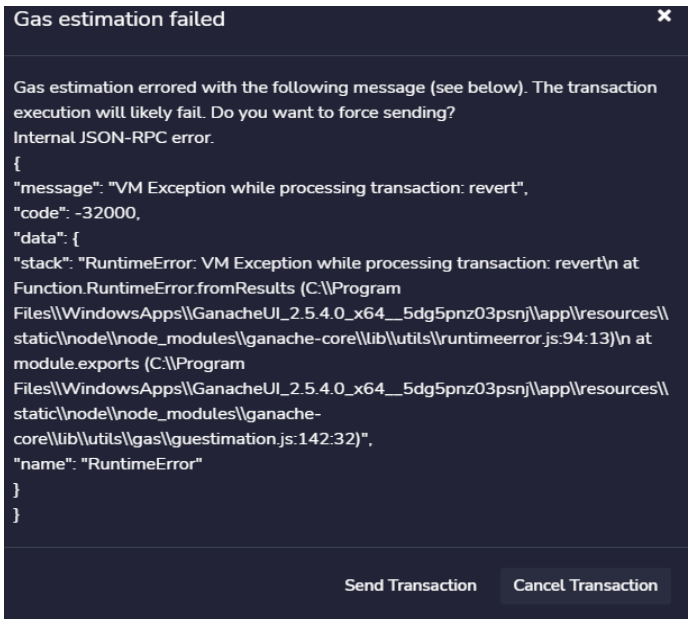


Transaction success message is shown

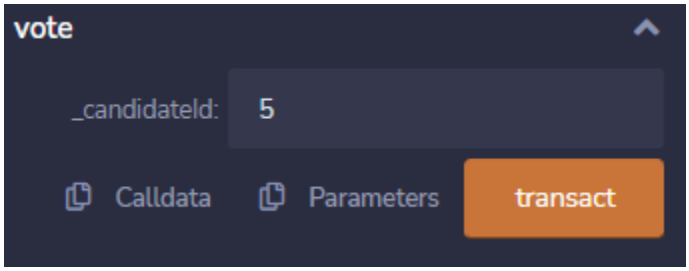


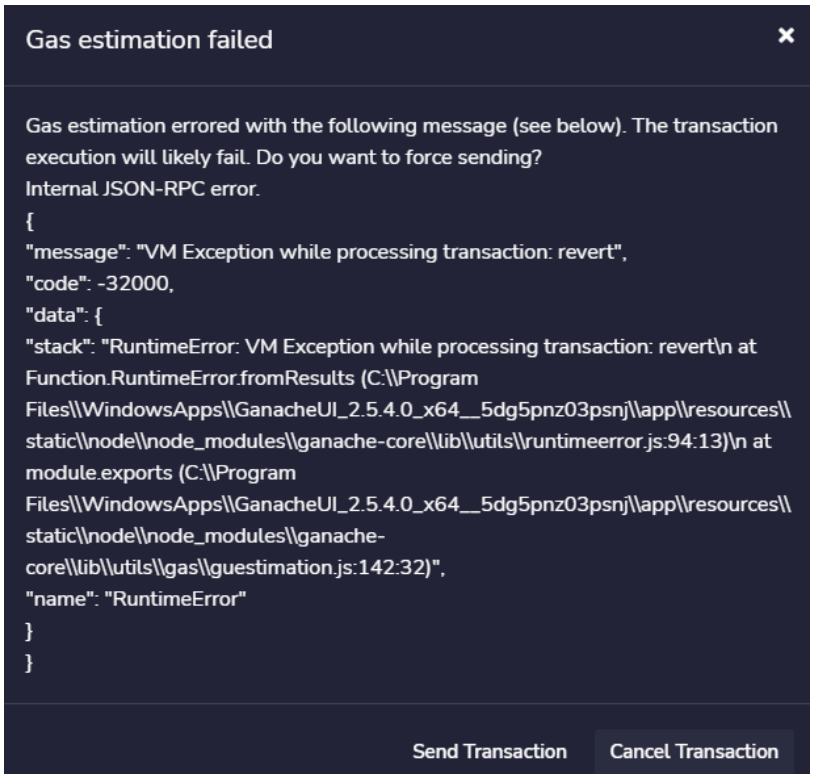
2) If we again try to vote error is shown with a different ID



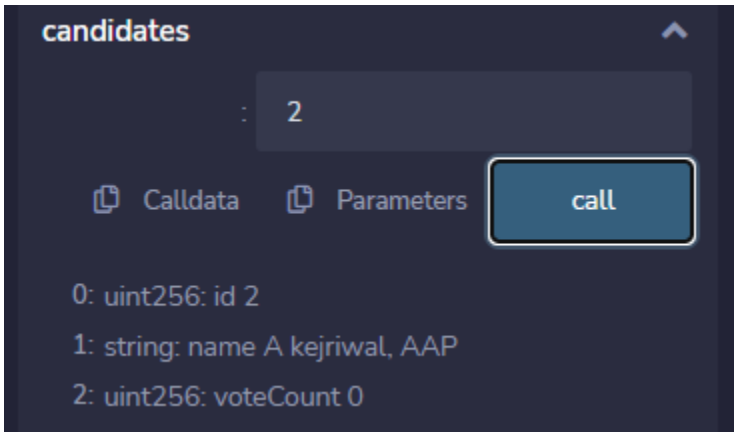


3) If we try to vote beyond the Id then error is shown.

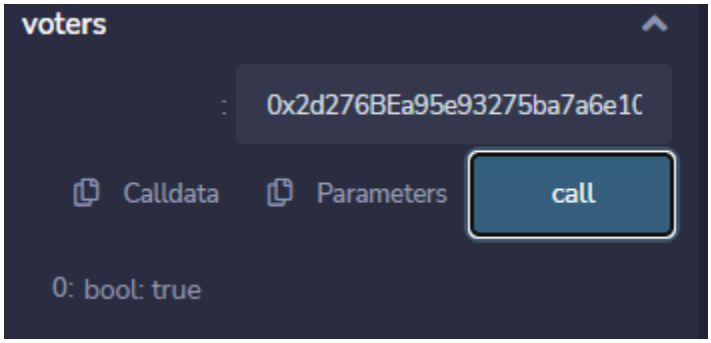




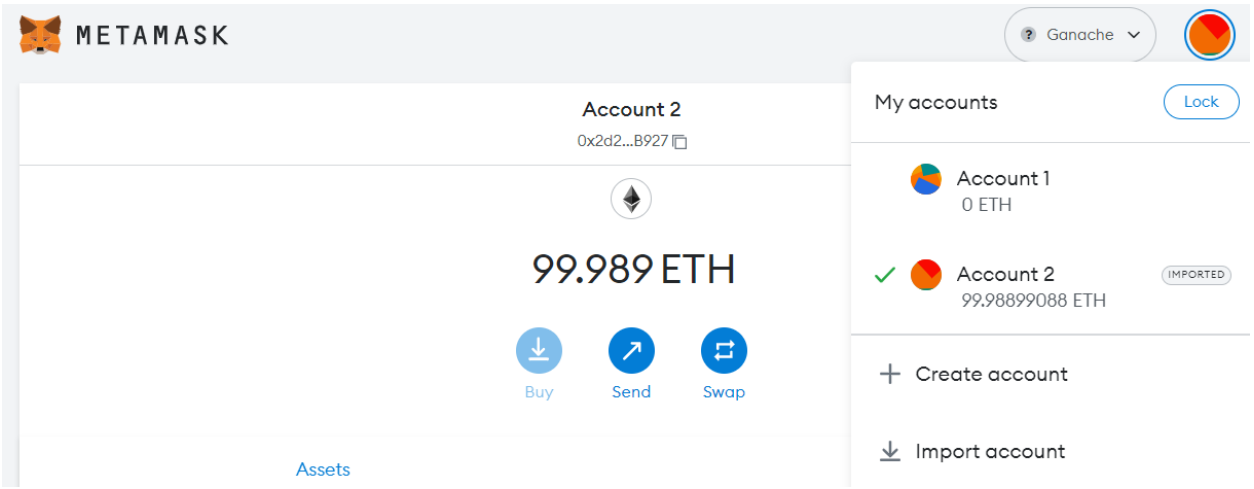
4) In candidate section we can check the candidate details



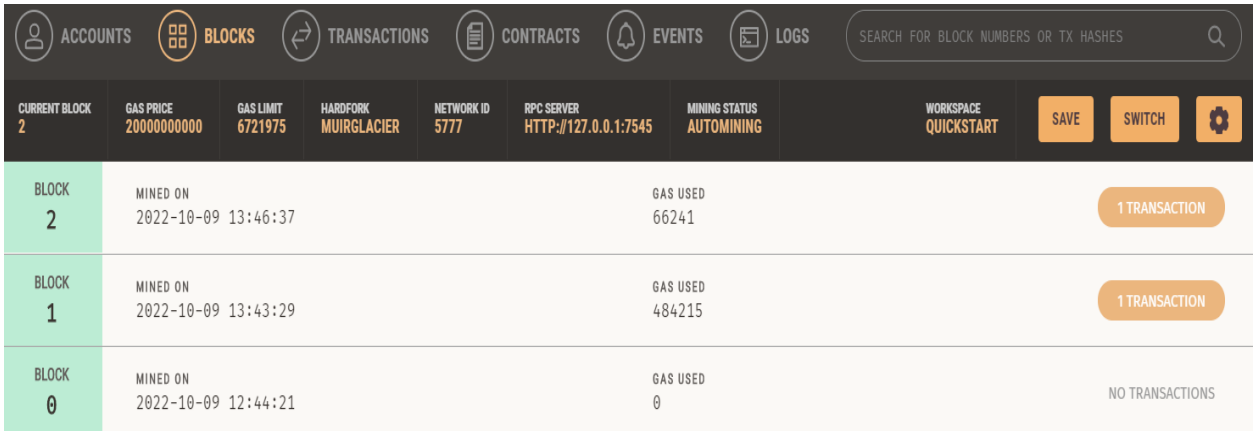
5) In voters section we can check whether the candidate has voted or not true indicates he has voted false indicates he has not



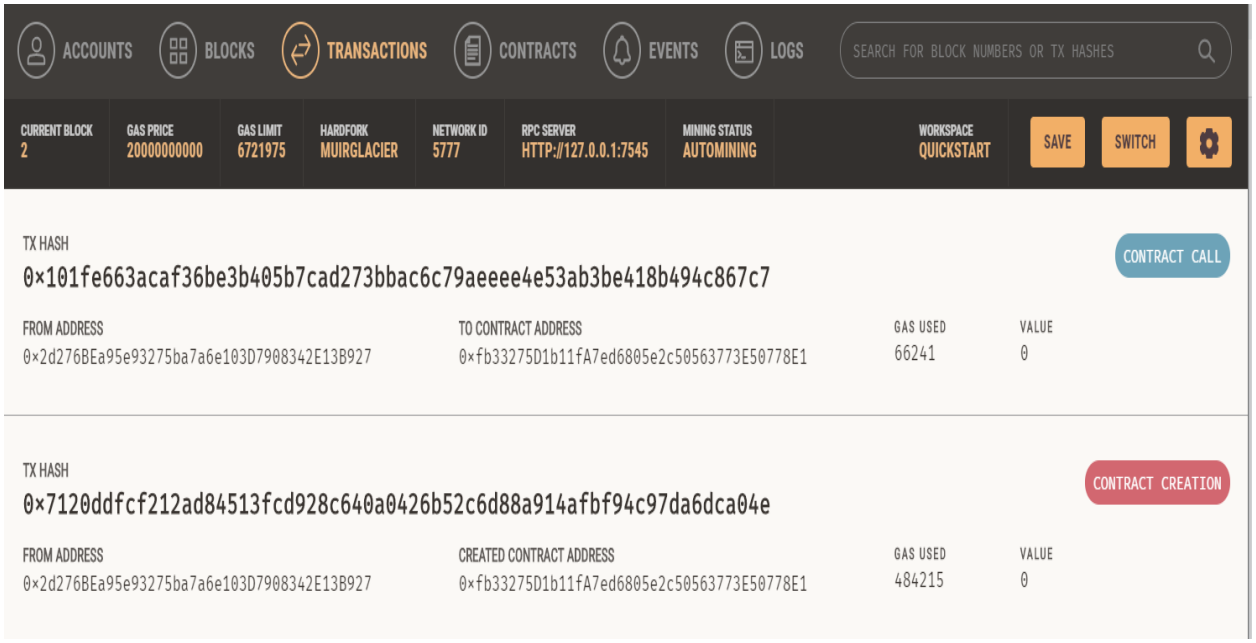
6) Balance



7) Goto Ganache check in Blocks



8) Goto Ganache check in Transactions.



CONCLUSION: -

From this practical I have learned to implement the installation of Ganache, Metamask and REMIX IDE and deploy smart contract using injected web 3 environment

PRACTICAL 7

AIM: - To study solidity and implement some examples of it

THEORY: -

SOLIDITY: -

Solidity is a contract-oriented, high-level programming language for implementing smart contracts. Solidity is highly influenced by C++, Python and JavaScript and has been designed to target the Ethereum Virtual Machine (EVM).Solidity is statically typed, supports inheritance, libraries and complex user-defined types programming language. You can use Solidity to create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

What is a Smart Contract?

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible. The concept of smart contracts was first proposed by Nick Szabo in 1994. Szabo is a legal scholar and cryptographer known for laying the groundwork for digital currency.

A) WAP in solidity to find area and circumference of a circle, rectangle & square

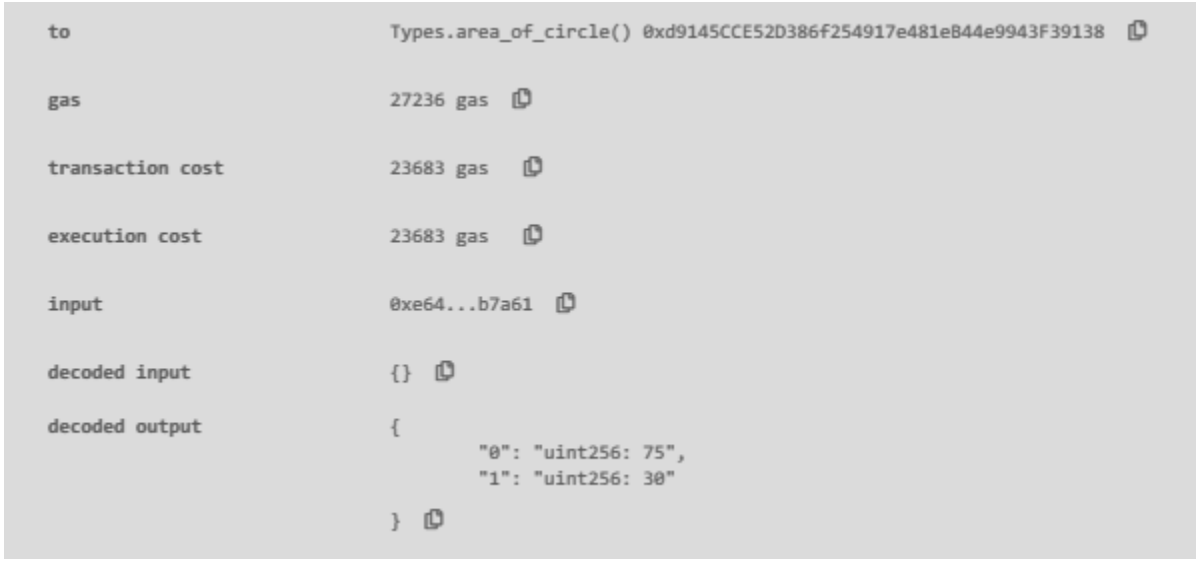
CODE: -

```
pragma solidity ^0.5.0;
 contract Types {
    uint l = 50;
    uint w = 25;

    uint s = 9;

    uint r = 5;

function area_of_rectangle() public returns(uint, uint)
 {
 uint area = l * w;
 uint perimeter = 2 * (l + w);
 return(area,perimeter);
```

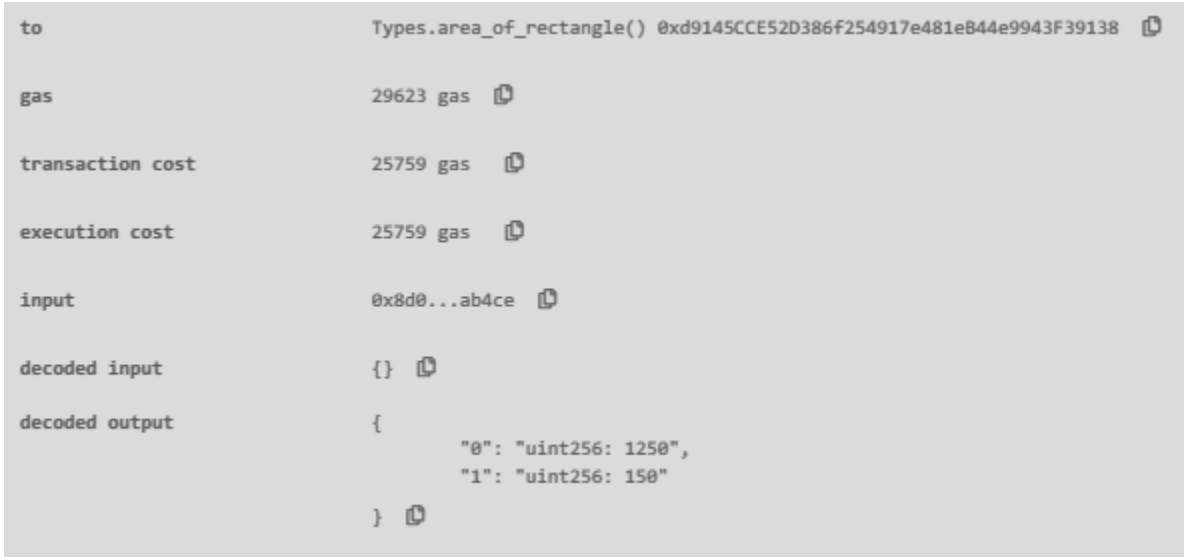
```
 }

function area_of_sqaure() public returns(uint, uint)
 {
 uint area = s * s;
 uint perimeter = 4 * s;
 return(area,perimeter);
 }

function area_of_circle() public returns(uint,uint)
 {
 uint area = 3 * r * r;
 uint circumference = 2 * 3  * r;
 return(area,circumference);
 }
}
```

OUTPUT: -

Area of circle and it's circumference



Area of rectangle and it's perimeter



Area of square and it's perimeter

B) WAP in solidity to input the basic salary of an employee and calculate gross salary according to given conditions:

a) Base Salary <= 10000 HRA=20% DA = 80%

b) Base Salary is between 10001 to 20000:HRA =25% DA=90%

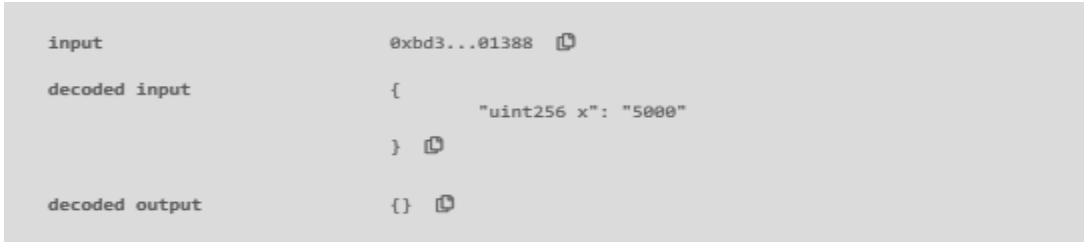
c) Base Salary>=20001 HRA=30% DA=95%

CODE: -

```
pragma solidity ^0.5.0;
//pragma experimental ABIEncoderV2;
contract Salary {
```

```
uint256 hra;
uint256 da;
uint256 bs;
function salcal(uint x) public{
   bs=x;
if(bs<=10000){
   hra=bs * 20/100;
   da=bs * 80/100;
}
else if(bs>=10001 && bs<=20000){
   hra=bs * 25/100;
   da=bs * 90/100;
}
else{
   hra=bs * 30/100;
   da=bs* 95/100;
}
}
function fin() public returns(uint256){
uint256 gross=bs+da+hra;
return gross;
}
}
```

OUTPUT: -



After pressing fin

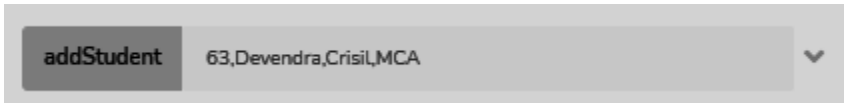
C) Write a program in solidity to create a structure student with Roll no, Name,Class, Department,Course enrolled as variables.
i) Add information about 5 students.
ii)Search for a student using Roll no
iii)Display all information

CODE: -

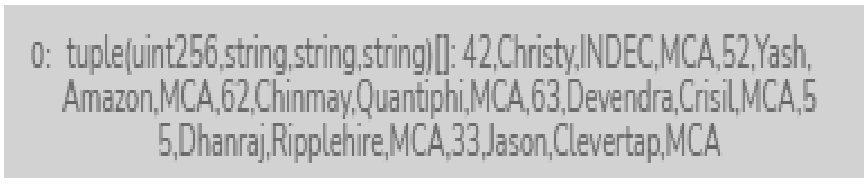
```
pragma solidity ^0.5.0;
pragma experimental ABIEncoderV2;
contract q3 {
struct student {
uint256 rollNo;
string name;
string placedCompany;
string degreeCourse;
}
student[] s;
function addStudent(
uint256 rollNo,
string memory name,
string memory placedCompany,
string memory degreeCourse
) public {
s.push(student(rollNo, name, placedCompany, degreeCourse));
}
function getStudentByRollNo(uint256 rollNumber)
public
view
returns (uint256, string memory)
{
uint256 i = 0;
for (i = 0; i < s.length; i++) {
if (s[i].rollNo == rollNumber) {
return (s[i].rollNo, s[i].name);
```

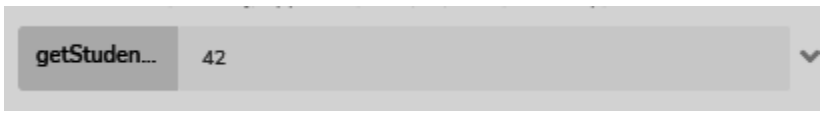
```
}
}
return (s[0].rollNo, s[0].name);
}
function getAllStudents() public view returns (student[] memory) {
return s;
}
}
```

OUTPUT: -



Get all students





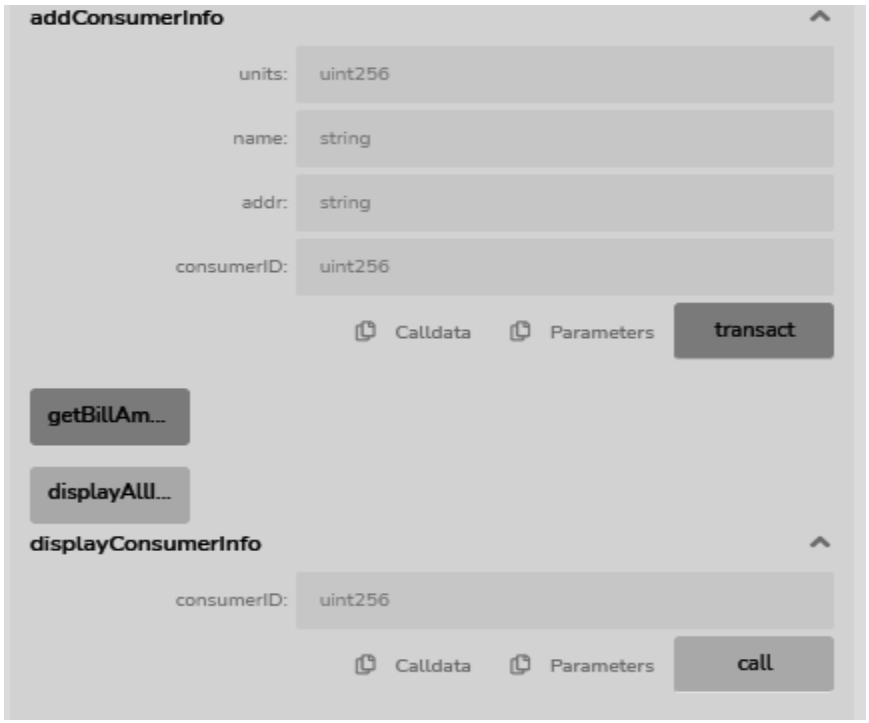
D) Create a structure Consumer with Name, Address, Consumer ID, Units and Amount as members.Write a program in solidity to calculate the total electricity bill according to the given condition:
For first 50 units Rs. 0.50/unit
For next 100 units Rs. 0.75/unit
For next 100 units Rs. 1.20/unit
For unit 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.

CODE: -

```
pragma solidity ^0.5.0;
pragma experimental ABIEncoderV2;
contract q4 {
struct Consumer {
uint256 units;
string name;
string addr;
uint256 consumerID;
uint256 amount;
}
Consumer[] consumer;
function addConsumerInfo(
uint256 units,
string memory name,
string memory addr,
uint256 consumerID
) public {
consumer.push(Consumer(units, name, addr, consumerID, 0));
}
function getBillAmount() public {
uint256 i = 0;
uint256 amount = 0;
uint256 surcharge = 0;
uint256 units = 0;
for (i = 0; i < consumer.length; i++) {
units = consumer[i].units;
if (units <= 50) {
amount = ((units * 1) / 2);
} else if (units <= 150) {
amount = 25 + (((units - 50) * 3) / 4);
} else if (units <= 250) {
amount = 100 + (((units - 150) * 6) / 5);
} else {
amount = 220 + (((units - 250) * 3) / 2);
}
surcharge = (amount * 20) / 100;
amount = amount + surcharge;
consumer[i].amount = amount;
}
```

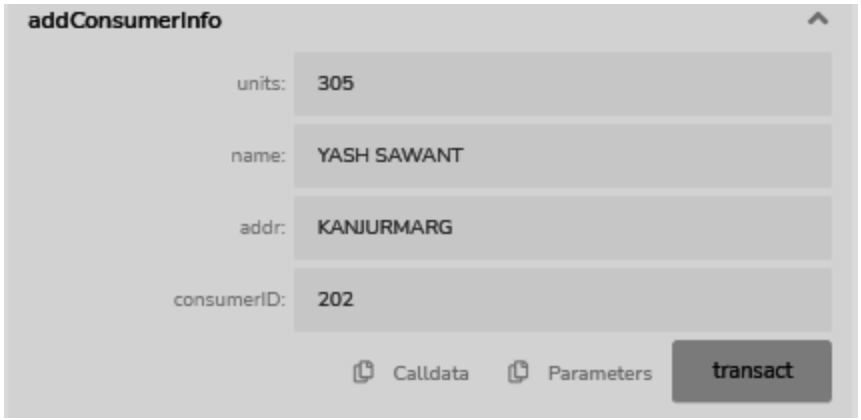
```
}
function displayConsumerInfo(uint256 consumerID)
public
view
returns (
uint256,
string memory,
string memory,
uint256,
uint256
)
{
uint256 i = 0;
for (i = 0; i < consumer.length; i++) {
if (consumer[i].consumerID == consumerID) {
return (
consumer[i].units,
consumer[i].name,
consumer[i].addr,
consumer[i].consumerID,
consumer[i].amount
);
}
}
return (
consumer[0].units,
consumer[0].name,
consumer[0].addr,
consumer[0].consumerID,
consumer[0].amount
);
}
function displayAllInfo() public view returns (Consumer[] memory) {
return consumer;
}
}
```

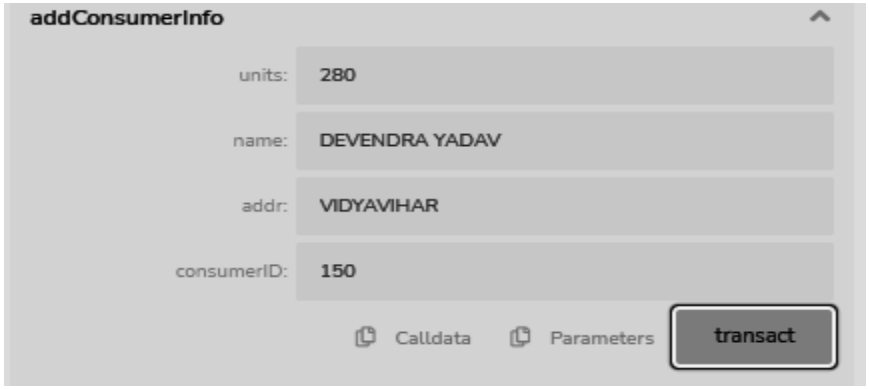
OUTPUT: -


addConsumerInfo
units: uint256
name: string
addr: string
consumerID: uint256
Calldata
Parameters
transact
getBillAm...
displayAll...
displayConsumerInfo
consumerID: uint256
Calldata
Parameters
call


addConsumerInfo
units: 250
name: CHRISTY PHILIP
addr: DOMBIVLI
consumerID: 402
Calldata
Parameters
transact


addConsumerInfo
units: 204
name: CHINMAY VYAPARI
addr: DOMBIVLI
consumerID: 388
Calldata
Parameters
transact

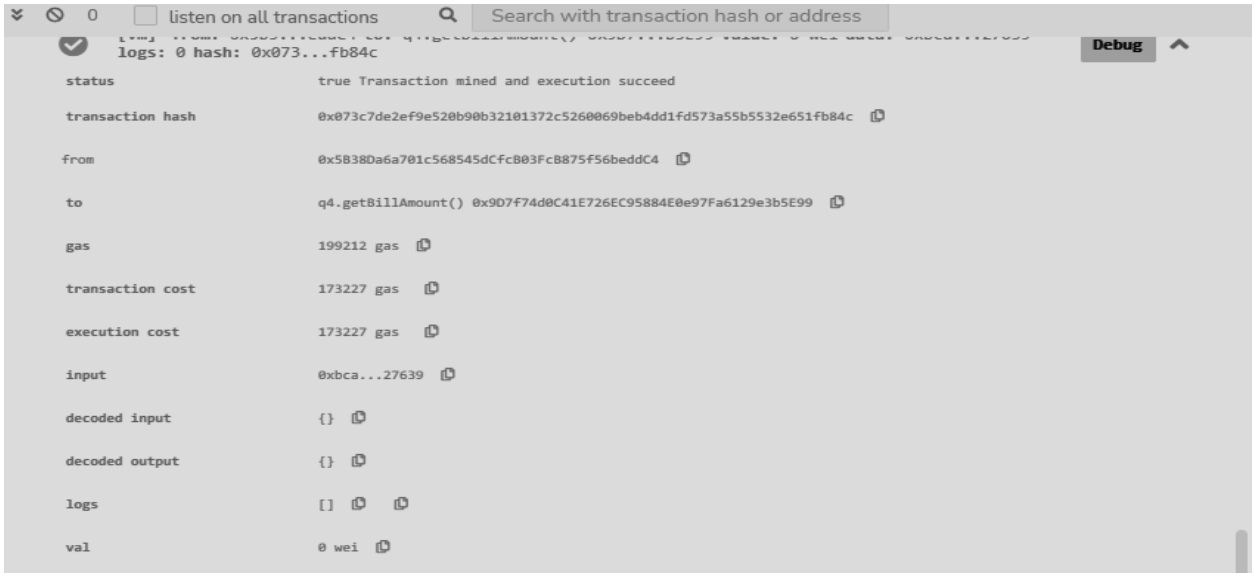

addConsumerInfo
units: 305
name: YASH SAWANT
addr: KANJURMARG
consumerID: 202
Calldata
Parameters
transact


addConsumerInfo
units: 280
name: DEVENDRA YADAV
addr: VIDYAVIHAR
consumerID: 150
Calldata
Parameters
transact

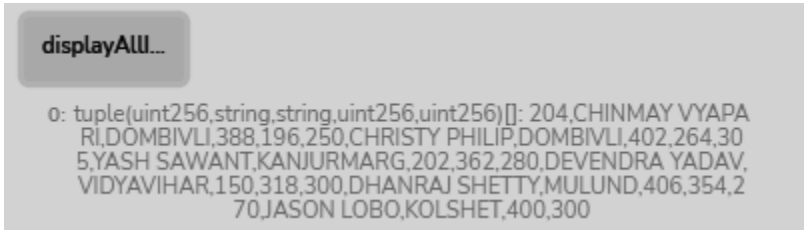

addConsumerInfo
units: 300
name: DHANRAJ SHETTY
addr: MULUND
consumerID: 406
Calldata
Parameters
transact


addConsumerInfo
units: 270
name: JASON LOBO
addr: KOLSHET
consumerID: 400
Calldata
Parameters
transact

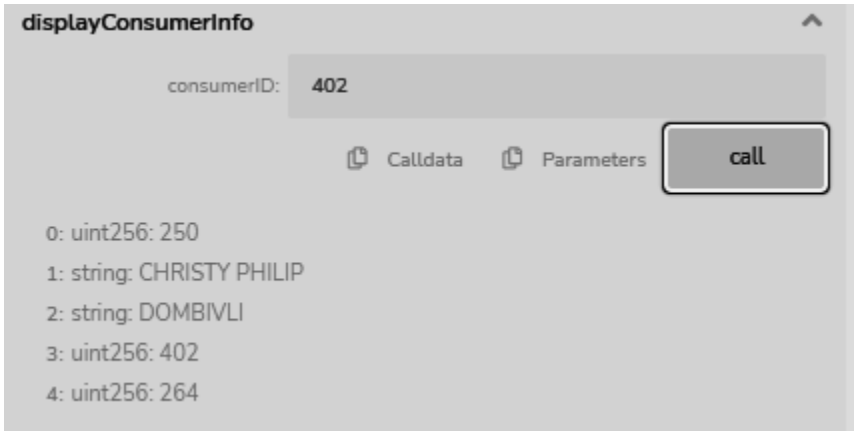
GET BILL AMOUNT



DISPLAY ALL



DISPLAY CONSUMER INFO



CONCLUSION: -

From this practical I have learned solidity and implemented some examples of it.

PRACTICAL 8

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

You can install Truffle with NPM in your command line like this:

Command: - npm install -g truffle

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.

1) Install Node JS and Truffle Suite to develop and migrate the smart contracts into the Private Blockchain network. Make use of Truffle tools like compile, migrate

and test for compilation, migration and testing the smart contracts through Blockchain.

Developing smart contracts using Truffle

- Now that you have a private blockchain running, you need to configure your
- environment for developing smart contracts.
- The first dependency you'll need is Node Package Manager, or NPM, which comes
- with Node.js.
- You can see if you have node already installed by
- **Command: -** node –v
- Otherwise, Download from: URL: https://nodejs.org/en/

Truffle Framework:

Truffle Framework, which provides a suite of tools for developing Ethereum smart contacts with the Solidity programming language.

- Smart Contract Management
- Automated Testing
- Deployment & Migrations
- Network Management
- Script Runner
- Client Side Development

STEPS: -

1) Command: -node -v

```
C:\Users\Philip>node -v
v16.16.0
```

2) Command: -npm install -g truffle

```
C:\Users\Philip>npm install -g truffle
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
npm WARN deprecated multibase@0.6.1: This module has been superseded by the multiformats module
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promises now, please swi
hat.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated multicodec@0.5.7: This module has been superseded by the multiformats module
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated cids@0.7.5: This module has been superseded by the multiformats module
npm WARN deprecated multicodec@1.0.4: This module has been superseded by the multiformats module
 problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.3.2: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, whic
wn to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@2.0.1: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, whic
wn to be problematic.  See https://v8.dev/blog/math-random for details.

added 751 packages, and audited 787 packages in 3m

87 packages are looking for funding
  run `npm fund` for details

9 moderate severity vulnerabilities

To address issues that do not require attention, run:
  npm audit fix

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 8.11.0 -> 9.1.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.1.2
npm notice Run npm install -g npm@9.1.2 to update!
```

3) Command: -mkdir blockchain-toolkit-8

```
C:\Users\Philip>mkdir blockchain-toolkit-8

C:\Users\Philip>
```

4) Command: -cd blockchain-toolkit-8

```
C:\Users\Philip>cd blockchain-toolkit-8

C:\Users\Philip\blockchain-toolkit-8>
```

5) Command: -truffle init

```
C:\Users\Philip\blockchain-toolkit-8>truffle init

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

> Copying project files to C:\Users\Philip\blockchain-toolkit-8

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


C:\Users\Philip\blockchain-toolkit-8>
```

6) Command: -npm install touch-cli -g

```
C:\Users\Philip\blockchain-toolkit-8>npm install touch-cli -g
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

added 2 packages, and audited 3 packages in 5s

found 0 vulnerabilities
```

7) Command: -touch package.json

```
C:\Users\Philip\blockchain-toolkit-8>touch package.json
Touching package.json

C:\Users\Philip\blockchain-toolkit-8>
```

copy-and-pasting the below code into package.json file
package.json

```
{
  "name": "blockchain-toolkit-8",
  "version": "1.0.0",
  "description": "The Complete Blockchain Developer Toolkit for 2019 & Beyond",
  "main": "truffle-config.js",
```

```
  "directories": {
  "test": "test"
},
"scripts": {
"dev": "lite-server",
"test": "echo \"Error: no test specified\" && exit 1"
},
"author": "christyphilip1234@gmail.com",
"license": "ISC",
"devDependencies": {
"bootstrap": "4.1.3",
"chai": "^4.1.2",
"chai-as-promised": "^7.1.1",
"chai-bignumber": "^2.0.2",
"dotenv": "^4.0.0",
"ganache-cli": "^6.1.8",
"lite-server": "^2.3.0",
"nodemon": "^1.17.3",
"solidity-coverage": "^0.4.15",
"truffle": "5.0.0-beta.0",
"truffle-contract": "3.0.6",
"truffle-hdwallet-provider": "^1.0.0-web3one.0"
}
}
```

Save package.json file

8) Command: -npm install

```
PS C:\Users\Philip\blockchain-toolkit-8> npm install
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
npm WARN deprecated ganache-cli@6.12.2: ganache-cli is now ganache; visit https://trfl.io/g7 for details
npm WARN skipping integrity check for git dependency ssh://git@github.com/debris/bignumber.js.git
npm WARN deprecated truffle-blockchain-utils@0.0.5: WARNING: This package has been renamed to @truffle/blockchain-utils.
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promises now, please switch to t
hat.
npm WARN deprecated babel-preset-es2015@6.24.1: 🙌  Thanks for using Babel: we recommend using babel-preset-env now: please read https://babeljs.io/
env to update!
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated truffle-error@0.0.3: WARNING: This package has been renamed to @truffle/error.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated to-iso-string@0.0.2: to-iso-string has been deprecated, use @segment/to-iso-string instead.
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated chokidar@2.1.8: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated minimatch@0.3.0: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated truffle-contract-schema@2.0.3: WARNING: This package has been renamed to @truffle/contract-schema.
npm WARN deprecated nomnom@1.8.1: Package no longer supported. Contact support@npmjs.com for more info.
npm WARN deprecated uuid@2.0.1: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, which is kno
wn to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, which is kno
wn to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated mkdirp@0.3.0: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has cha
nged to use Promises in 1.x.)
npm WARN deprecated uuid@3.3.2: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain circumstances, which is kno
wn to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has cha
nged to use Promises in 1.x.)
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has cha
nged to use Promises in 1.x.)
npm WARN deprecated truffle-contract@3.0.6: WARNING: This package has been renamed to @truffle/contract.
```

```
npm WARN deprecated truffle-contract@3.0.6: WARNING: This package has been renamed to @truffle/contract.
npm WARN deprecated jade@0.26.3: Jade has been renamed to pug, please install the latest version of pug instead of jade
npm WARN deprecated istanbul@0.4.5: This module is no longer maintained, try this instead:
npm WARN deprecated   npm i nyc
npm WARN deprecated Visit https://istanbul.js.org/integrations for other alternatives.
npm WARN deprecated truffle-hdwallet-provider@1.0.17: WARNING: This package has been renamed to @truffle/hdwallet-provider.
npm WARN deprecated core-js@2.6.12: core-js@<3.23.3 is no longer maintained and not recommended for usage due to the number of issues. Because of th
e V8 engine whims, feature detection in old core-js versions could cause a slowdown up to 100x even if nothing is polyfilled. Some versions have web
 compatibility issues. Please, upgrade your dependencies to the actual version of core-js.

added 1447 packages, changed 2 packages, and audited 1550 packages in 3m

46 packages are looking for funding
  run `npm fund` for details

70 vulnerabilities (4 low, 18 moderate, 17 high, 31 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS C:\Users\Philip\blockchain-toolkit-8> 
```

Start developing a smart contract using solidity

9) Command: -touch ./contracts/MyContract.so**l**

```
C:\Users\Philip\blockchain-toolkit-8>touch ./contracts/MyContract.sol
Touching ./contracts/MyContract.sol

C:\Users\Philip\blockchain-toolkit-8>
```

Copy the below contract code and save in Mycontract.sol
MyContact.sol

```
pragma solidity >=0.4.2 <=0.8.17;
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;
}
}
```

10) Command: - truffle compile

```
C:\Users\Philip\blockchain-toolkit-8>truffle compile
Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol-bin. Attempt #1
> Compilation warnings encountered:

    Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier:
<SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more inform
ation.
--> project:/contracts/MyContract.sol

,Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
 --> project:/contracts/MyContract.sol:4:1:
  |
4 | constructor() public {
  | ^ (Relevant source part starts here and spans across multiple lines).


> Artifacts written to C:\Users\Philip\blockchain-toolkit-8\build\contracts
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang
⠸ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
C:\Users\Philip\blockchain-toolkit-8>
```

Update the project configuration file (Find the file truffle-config.js and paste the following code:)
truffle-config.js

```
module.exports = {
networks: {
development: {
host: "127.0.0.1",
port: 7545,
network_id: "*" // Match any network id
}
```

},
solc: {
optimizer: {
enabled: true,
runs: 200
}
}
}

Create a migration script inside the migrations directory to deploy the smart contract to the personal blockchain network.

11) Command: - touch migrations/2_deploy_contracts.js

```
C:\Users\Philip\blockchain-toolkit-8>touch migrations/2_deploy_contracts.js
Touching migrations/2_deploy_contracts.js

C:\Users\Philip\blockchain-toolkit-8>
```

(Copy the below script into 2_depoy_contracts.js)

2_depoy_contracts.js:

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

Start the Ganache in your PC

12) Command: - truffle migrate

```
C:\Users\Philip\blockchain-toolkit-8>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\MyContract.sol
> Artifacts written to C:\Users\Philip\blockchain-toolkit-8\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)


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

   Deploying 'MyContract'
   ----------------------
   > transaction hash:    0x9883bcc628bb95d4e9f6903ee885b8e8accb9315f7137ae367143cb9f225cce7
   > Blocks: 0            Seconds: 0
   > contract address:    0xCe4448dfe622cEd3fFBe81899b26755B9f3B1e19
   > block number:        1
   > block timestamp:     1668862742
   > account:             0x103f47c86d90bE8ECF3780FC7A39203b8950e3d0
   > balance:             99.99529344
   > gas used:            235328 (0x39740)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00470656 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00470656 ETH
```

```
Summary
=======
> Total deployments:   1
> Final cost:          0.00470656 ETH


C:\Users\Philip\blockchain-toolkit-8>
```

13) Command: - truffle console

```
C:\Users\Philip\blockchain-toolkit-8>truffle console
```

14) Now we are going into the truffle console. Below commands are for the truffle

Command: - MyContract.deployed().then((instance) => { app = instance })

```
truffle(development)> MyContract.deployed().then((instance) => { app = instance } )
undefined
```

Command: -app.get()

```
truffle(development)> app.get()
'myValue'
```

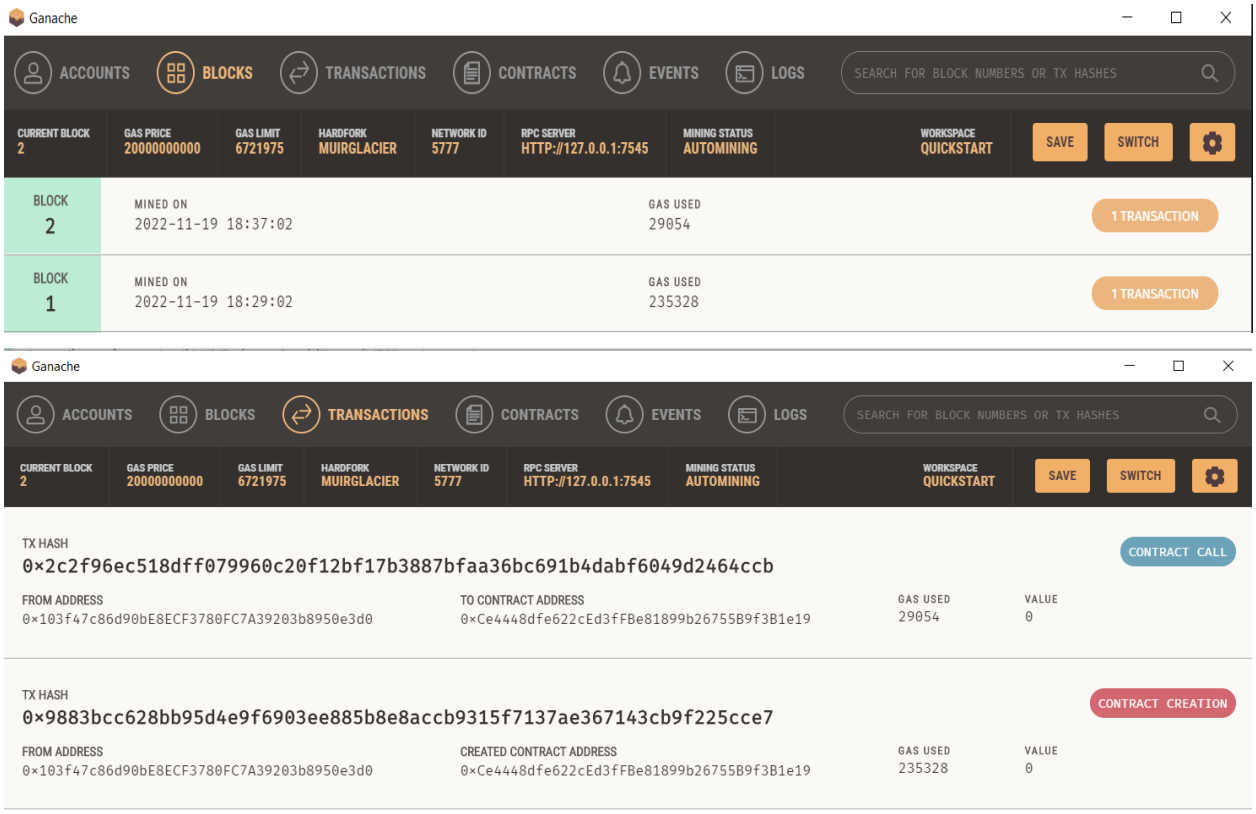
Command: -app.set('New Value')

```
truffle(development)> app.set('New Value')
{
  tx: '0x2c2f96ec518dff079960c20f12bf17b3887bfaa36bc691b4dabf6049d2464ccb',
  receipt: {
    transactionHash: '0x2c2f96ec518dff079960c20f12bf17b3887bfaa36bc691b4dabf6049d2464ccb',
    transactionIndex: 0,
    blockHash: '0xecddec5d3b5cdab952c82fd779e544b0b680b5d31c91103e1433d758f0eb6df8',
    blockNumber: 2,
    from: '0x103f47c86d90be8ecf3780fc7a39203b8950e3d0',
    to: '0xce4448dfe622ced3ffbe81899b26755b9f3b1e19',
    gasUsed: 29054,
    cumulativeGasUsed: 29054,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
```

Command: -app.get()

```
truffle(development)> app.get()
'New Value'
truffle(development)>
```

Goto Ganache→Click on Blocks



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

1) Command: - mkdir blockchain-toolkit-overloading

```
C:\Users\Philip>mkdir blockchain-toolkit-overloading
```

2) Command: - cd blockchain-toolkit-overloading

```
C:\Users\Philip>cd blockchain-toolkit-overloading

C:\Users\Philip\blockchain-toolkit-overloading>
```

3) Command: - truffle init

```
C:\Users\Philip\blockchain-toolkit-overloading>truffle init

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

> Copying project files to C:\Users\Philip\blockchain-toolkit-overloading

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
```

4) Command: -touch package.json

```
C:\Users\Philip\blockchain-toolkit-overloading>touch package.json
Touching package.json

C:\Users\Philip\blockchain-toolkit-overloading>
```

Below code should be kept in package.json file

package. json

```
{
  "name": "blockchain-toolkit-overloading",
  "version": "1.0.0",
  "description": "The Complete Blockchain Developer Toolkit for 2019 & Beyond",
  "main": "truffle-config.js",
  "directories": {
  "test": "test"
  },
  "scripts": {
  "dev": "lite-server",
  "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "christyphilip1234@gmail.com",
  "license": "ISC",
  "devDependencies": {
  "bootstrap": "4.1.3",
  "chai": "^4.1.2",
```

```
    "chai-as-promised": "^7.1.1",
    "chai-bignumber": "^2.0.2",
    "dotenv": "^4.0.0",
    "ganache-cli": "^6.1.8",
    "lite-server": "^2.3.0",
"nodemon": "^1.17.3",
"solidity-coverage": "^0.4.15",
"truffle": "5.0.0-beta.0",
"truffle-contract": "3.0.6",
"truffle-hdwallet-provider": "^1.0.0-web3one.0"
}
}
```

5) Command: - npm install

```
C:\Users\Philip\blockchain-toolkit-overloading>npm install
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

added 1606 packages, and audited 1707 packages in 3m

48 packages are looking for funding
  run `npm fund` for details

73 vulnerabilities (4 low, 18 moderate, 20 high, 31 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\Philip\blockchain-toolkit-overloading>
```

Open Ganache network
Start developing a smart contract using solidity:

6) Command: -touch ./contracts/overloading.sol

```
C:\Users\Philip\blockchain-toolkit-overloading>touch ./contracts/overloading.sol
Touching ./contracts/overloading.sol
```

overloading.sol

```
pragma solidity >=0.4.2 <=0.8.17;
contract overloading {
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() public pure returns (uint) {
return getSum(1, 2);
}
function callSumWithThreeArguments() public pure returns (uint) {
return getSum(1, 2, 3);
}
}
```

7) Command: -truffle compile

```
C:\Users\Philip\blockchain-toolkit-overloading>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\overloading.sol
> Compilation warnings encountered:

    Warning: SPDX license identifier not provided in source file. Before publishing, consider adding a comment containing "SPDX-License-Identifier:
<SPDX-License>" to each source file. Use "SPDX-License-Identifier: UNLICENSED" for non-open-source code. Please see https://spdx.org for more inform
ation.
--> project:/contracts/overloading.sol


> Artifacts written to C:\Users\Philip\blockchain-toolkit-overloading\build\contracts
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang

C:\Users\Philip\blockchain-toolkit-overloading>
```

Update config file

truffle-config.js

```
module.exports = {
  networks: {
  development: {
  host: "127.0.0.1",
  port: 7545,
  network_id: "*", // Match any network id
  },
  },
  solc: {
```

```
optimizer: {
enabled: true,
runs: 200,
},
},
};
```

8) Command: - touch migrations/2_deploy_contracts.js

```
C:\Users\Philip\blockchain-toolkit-overloading>touch migrations/2_deploy_contracts.j
Touching migrations/2_deploy_contracts.js

C:\Users\Philip\blockchain-toolkit-overloading>
```

2_deploy_contracts.js

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

9) Command: -truffle migrate

```
C:\Users\Philip\blockchain-toolkit-overloading>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\overloading.sol
> Artifacts written to C:\Users\Philip\blockchain-toolkit-overloading\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)


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

   Deploying 'overloading'
   -----------------------
   > transaction hash:    0xd585ac4705f79aa0d053565f2f365773d5ca997b22f8cfe8d0cf471d3c615d10
   > Blocks: 0            Seconds: 0
   > contract address:    0x0f8D01a49Cbb7e2eF2031fCe77E354ec63c04375
   > block number:        1
   > block timestamp:     1668877381
   > account:             0x1b25bA685c0Dd504C50ac1ec1536B52435A06b82
   > balance:             99.99776894
   > gas used:            111553 (0x1b3c1)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00223106 ETH

   > Saving artifacts
   -------------------------------------
```

```
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00223106 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.00223106 ETH
```

10) Command: - truffle console

```
C:\Users\Philip\blockchain-toolkit-overloading>truffle console
truffle(development)>
```

11) Commands to run in truffle console: -

Command: -truffle console

Command: -overloading.deployed().then((instance) => {app = instance})

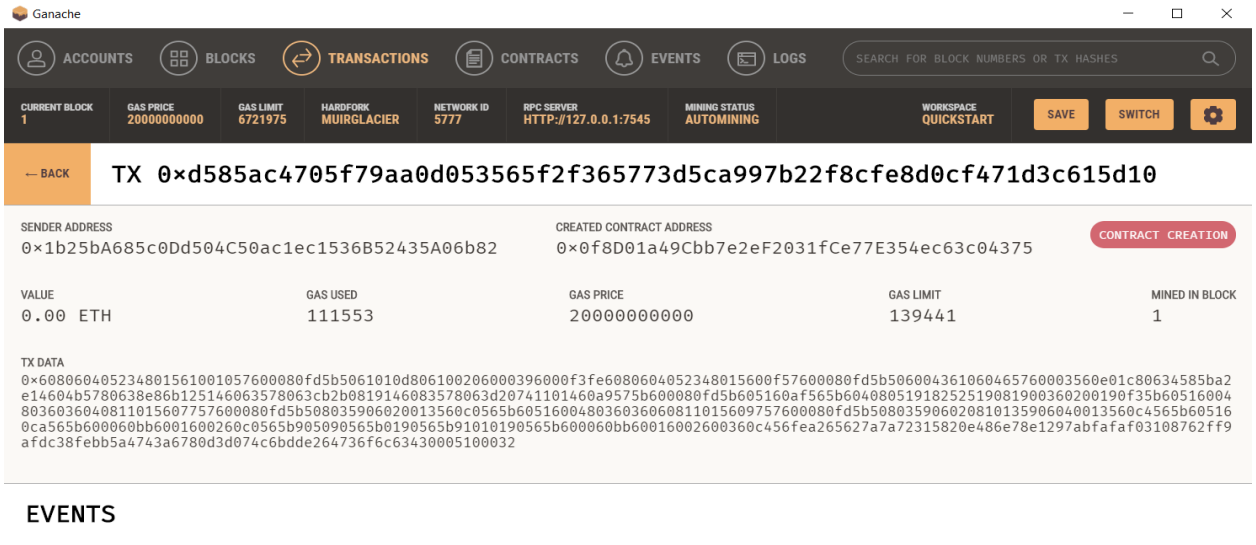
Command: -app.getSum(5,6)

Command: -app.callSumWithTwoArguments()

Command: -app.callSumWithThreeArguments()

```
truffle(development)> overloading.deployed().then((instance) => {app = instance} )
undefined
truffle(development)> app.getSum(5,6)
BN { negative: 0, words: [ 11, <1 empty item> ], length: 1, red: null }
truffle(development)> app.callSumWithTwoArguments()
BN { negative: 0, words: [ 3, <1 empty item> ], length: 1, red: null }
truffle(development)> app.callSumWithThreeArguments()
BN { negative: 0, words: [ 6, <1 empty item> ], length: 1, red: null }
truffle(development)>
```

GANACHE TRANSACTION LOGS



CONCLUSION: -

From this practical I have learned Smart Contract using Truffle Framework.

PRACTICAL 9

AIM: - Create Dapps 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 compute 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.

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

Description of the file directory structure:

● contracts directory- holds all smart contracts

● migrations directory: Whenever we deploy smart contracts to the blockchain, we are updating the blockchain's state, and therefore need a migration.

● node_modules directory: this is the home of all of our Node dependencies.

● src directory: this is where we'll develop our client-side application.

● test directory: this is where we'll write our tests for our smart contracts.

● truffle.js file: this is the main configuration file for our Truffle project

Steps to follow:

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

$ mkdir election

$ cd election

2. Install truffle:

npm install -g truffle

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

$ truffle unbox pet-shop

4. create a new contract file in the contracts directory like this:

$ touch contracts/Election.sol

$ touch migrations/2_deploy_contracts.js

5. Migrate:

truffle migrate

6. open the console to interact with the smart contract.

$ truffle console

7. inside the console, let's get an instance of our deployed smart contract and see if we can read the candidate's name from the contract. From the console, run this code:

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

Here Election is the name of the variable that we created in the migration file. We retrieved a deployed instance of the contract with the deployed() function, and assigned it to an app variable inside the promise's callback function.

8. run the tests from the command line like this:

$ truffle test

9. Client-Side Application

$ truffle migrate --reset

10. Next, start your development server from the command line like this:

$ npm run dev or $ npm run start

Goto Metamask→Setting→Show test networks to be made ON

Networks RPC URL port number should be of Ganache

A) E-VOTING APPLICATION BASED ON THE BLOCKCHAIN

Commands: -

mkdir election

cd election

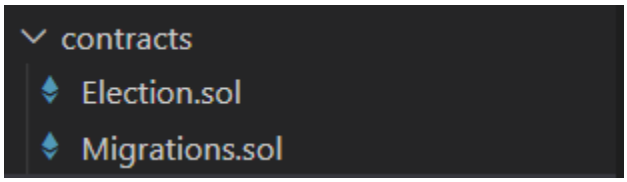
```
PS C:\Users\Exam> mkdir Christy


    Directory: C:\Users\Exam


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       11/21/2022  12:03 PM                Christy


PS C:\Users\Exam> cd Christy
PS C:\Users\Exam\Christy> mkdir election


    Directory: C:\Users\Exam\Christy


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----       11/21/2022  12:03 PM                election


PS C:\Users\Exam\Christy> cd election
PS C:\Users\Exam\Christy\election>
```

Command: - truffle unbox pet-shop

```
C:\Users\Exam\Christy\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 set-value@0.4.3: Critical bug fixed in v3.0.1, please upgrade to the latest version.
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated chokidar@2.0.4: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with 15x fewer dependencies
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://github.com/axios/axios/pull/3410
√ 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


C:\Users\Exam\Christy\election>
```

Command: - touch contracts/Election.sol

```
C:\Users\Exam\Christy\election>touch contracts/Election.sol
Touching contracts/Election.sol

C:\Users\Exam\Christy\election>
```



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("Candidate 1");
addCandidate("Candidate 2");
}
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);
}
}
```

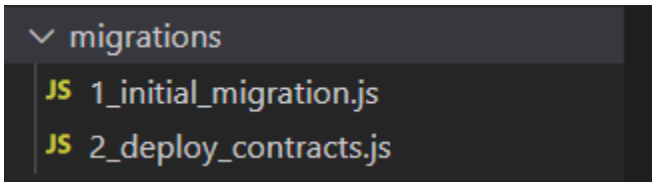
Migrations.sol

```
pragma solidity >=0.4.22 <0.8.0;
contract Migrations {
address public owner = msg.sender;
uint public last_completed_migration;
modifier restricted() {
require(
msg.sender == owner,
"This function is restricted to the contract's owner"
);
_;
}
function setCompleted(uint completed) public restricted {
last_completed_migration = completed;
```

}
}

Command: - touch migrations/2_deploy_contracts.js

```
C:\Users\Exam\Christy\election>touch migrations/2_deploy_contracts.js
Touching migrations/2_deploy_contracts.js

C:\Users\Exam\Christy\election>
```



1_initial_migration.js

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

2_deploy_contracts.js

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

truffle-config.js

```
module.exports = {
  // See <http://truffleframework.com/docs/advanced/configuration>
  // for more about customizing your Truffle configuration!
  networks: {
  development: {
  host: "127.0.0.1",
  port: 7545,
  network_id: "*" // Match any network id
  },
  develop: {
  port: 7545
  }
}
};
```

Command: - truffle migrate

```
C:\Users\Exam\Christy\election> truffle migrate
This version of µWS is not compatible with your Node.js build:

Error: node-loader:
Error: The specified module could not be found.
C:\Users\Exam\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node/1RGFZdPM.node
Falling back to a NodeJS implementation; performance may be degraded.


Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\Exam\Christy\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: 6721975000 (0x190a932d8)


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

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x20c2c0f4e4814bdb2386d3d98f19403dba3da463c11143675f649d5ca1b4318f
   > Blocks: 0            Seconds: 0
   > contract address:    0xe05d834a79691B985EDf246407BfF14A74AE1831
   > block number:        22
   > block timestamp:     1669012946
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.90673476
   > gas used:            191943 (0x2edc7)
   > 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:    0x46a0204b3079723e99f60371191f1d1755d4d5f12be9f7e267113bbfb67c0ddb
   > Blocks: 0            Seconds: 0
   > contract address:    0x2FB63E4Cf80D8898a4419035d95D58a44d375c1d
   > block number:        24
   > block timestamp:     1669012947
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.89817462
   > 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


C:\Users\Exam\Christy\election>
```

Command: - truffle console

```
C:\Users\Exam\Christy\election>truffle console
This version of µWS is not compatible with your Node.js build:

Error: node-loader:
Error: The specified module could not be found.
C:\Users\Exam\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node/1RGFZdPM.node
Falling back to a NodeJS implementation; performance may be degraded.


truffle(development)>
```

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

```
truffle(development)> Election.deployed().then(function(instance) { app = instance })
undefined
truffle(development)>
```

Right click on test folder and create file with name election.js

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 candiates", 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");
});
});
});
```

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">
   <div class="row">
    <div class="col-xs-12 col-sm-8 col-sm-push-2">
     <h1 class="text-center">Pete's Pet Shop</h1>
     <hr/>
     <br/>
    </div>
   </div>

   <div id="petsRow" class="row">
    <!-- PETS LOAD HERE -->
   </div>
  </div>

  <div id="petTemplate" style="display: none;">
   <div class="col-sm-6 col-md-4 col-lg-3">
    <div class="panel panel-default panel-pet">
     <div class="panel-heading">
      <h3 class="panel-title">Scrappy</h3>
     </div>
     <div class="panel-body">
      <img alt="140x140" data-src="holder.js/140x140" class="img-rounded img-center"
style="width: 100%;"
src="https://animalso.com/wp-content/uploads/2017/01/Golden-Retriever_6.jpg"
data-holder-rendered="true">
      <br/><br/>
      <strong>Breed</strong>: <span class="pet-breed">Golden Retriever</span><br/>
      <strong>Age</strong>: <span class="pet-age">3</span><br/>
      <strong>Location</strong>: <span class="pet-location">Warren, MI</span><br/><br/>
      <button class="btn btn-default btn-adopt" type="button" data-id="0">Adopt</button>
     </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>
```

Command: - truffle test

```
C:\Users\Exam\Christy\election>truffle test
This version of µWS is not compatible with your Node.js build:

Error: node-loader:
Error: The specified module could not be found.
C:\Users\Exam\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node/1RGFZdPM.node
Falling back to a NodeJS implementation; performance may be degraded.


Using network 'development'.


Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\Exam\AppData\Local\Temp\test--7864-Xd80ZBORl48x
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


  Contract: Election
    √ initializes with two candidates (63ms)
    √ it initializes the candidates with the correct values (155ms)
    √ allows a voter to cast a vote (238ms)
    √ throws an exception for invalid candiates (395ms)
    1) throws an exception for double voting
    > No events were emitted


  4 passing (1s)
  1 failing

  1) Contract: Election
       throws an exception for double voting:
     AssertionError: error message must contain revert
      at C:\Users\Exam\Christy\election\test\election.js:73:1
      at processTicksAndRejections (node:internal/process/task_queues:96:5)



C:\Users\Exam\Christy\election>
```

Command: - truffle migrate –reset

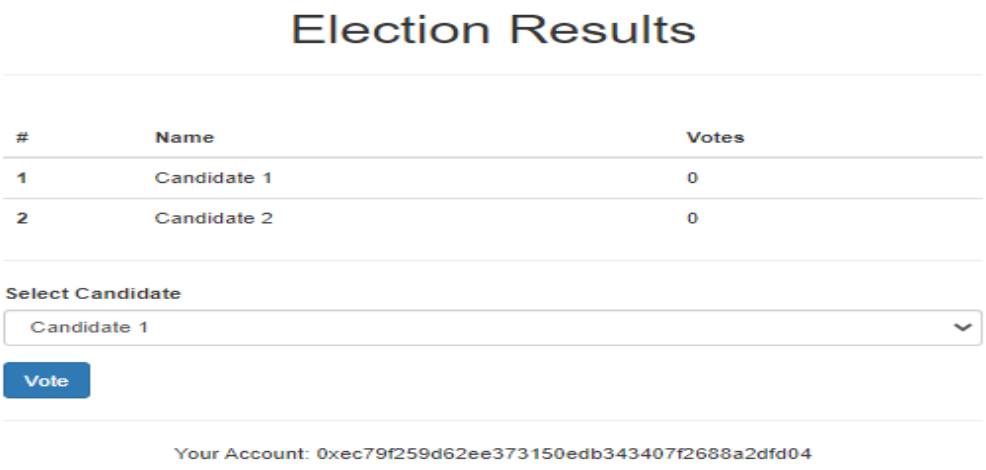
```
C:\Users\Exam\Christy\election>truffle migrate --reset
This version of µWS is not compatible with your Node.js build:

Error: node-loader:
Error: The specified module could not be found.
C:\Users\Exam\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node/1RGFZdPM.node
Falling back to a NodeJS implementation; performance may be degraded.


Compiling your contracts...
===========================
> Compiling .\contracts\Election.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\Users\Exam\Christy\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: 6721975000 (0x190a932d8)


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

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x97b4d84ce1a23dd5509a7234a941514133455aa2113547559e373c8864b6044b
   > Blocks: 0            Seconds: 0
   > contract address:    0x2Cf357B0AA66925Aaff316C035E40AfD0860b3bd
   > block number:        45
   > block timestamp:     1669014642
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.8393562
   > gas used:            191943 (0x2edc7)
   > 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
=====================

   Replacing 'Election'
   --------------------
   > transaction hash:    0x918939bcc1b7b661f7ebdcae510f76ad3fc6f40155b0b1460a379f3d077c54d1
   > Blocks: 0            Seconds: 0
   > contract address:    0x7BD9Db8e3842dB981f86F3D14dC06c6579CD2C71
   > block number:        47
   > block timestamp:     1669014643
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.83079606
   > 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


C:\Users\Exam\Christy\election>
```

Command: - npm run dev

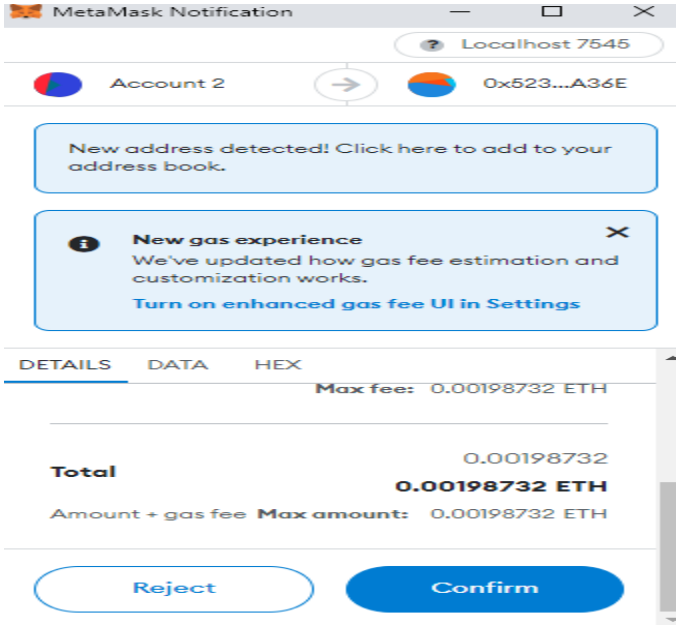
```
C:\Users\Exam\Christy\election>npm run dev

> 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:3002
    External: http://192.168.37.194:3002
 --------------------------------------
          UI: http://localhost:3003
 UI External: http://localhost:3003
 --------------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
```

OUTPUT: -

OUTPUT LOADS IN THE BROWSER

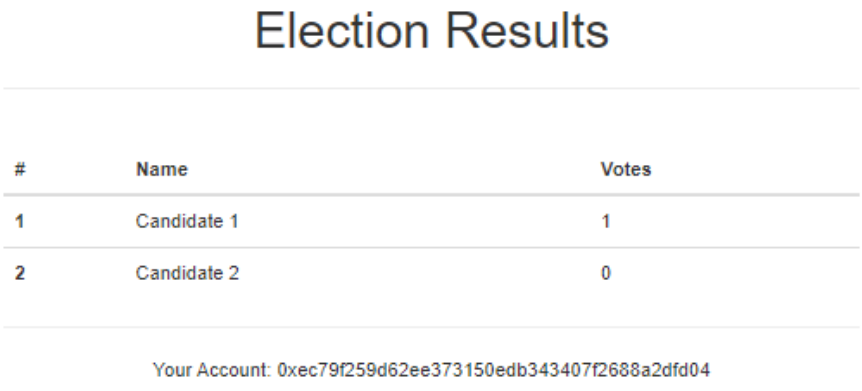


CLICK ON CONFIRM IN BELOW IMAGE PROMPT APPEARING



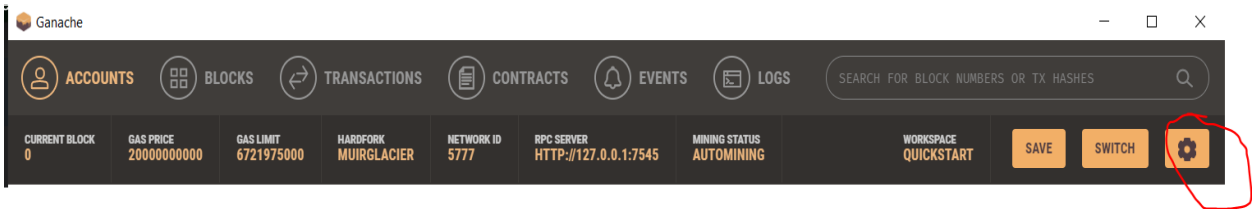
CAST VOTE:

COUNT OF VOTES:

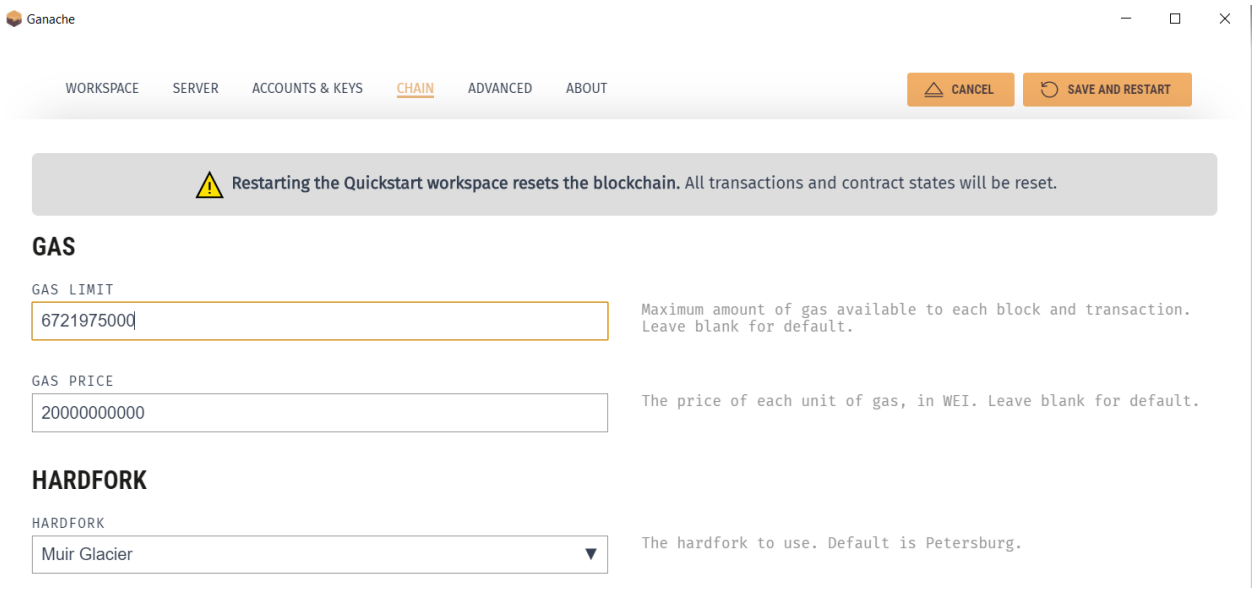


B) SUPPLY CHAIN MANAGEMENT IN ETHEREUM:

Click on the icon of settings which is circled



Changing gas in ganache:



Save and Restart

Download below project from Github

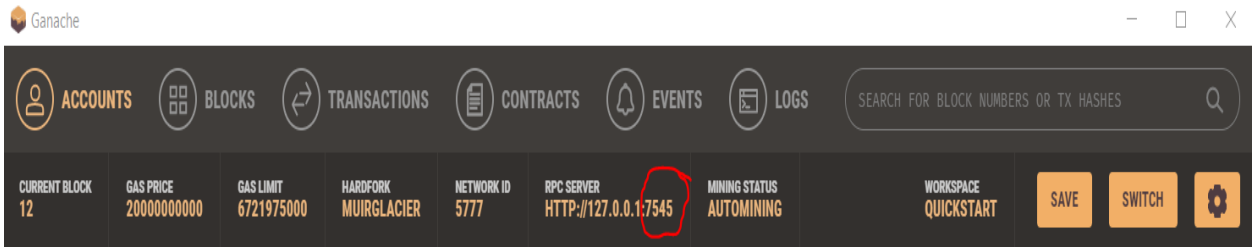
https://github.com/rishav4101/eth-supplychain-dapp

Open project in the VScode

Command: - npm i

```
C:\Users\Exam\Downloads\eth-supplychain-dapp-main>npm i
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.

82 vulnerabilities (8 low, 33 moderate, 31 high, 10 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.

C:\Users\Exam\Downloads\eth-supplychain-dapp-main>
```

Open truffle.config.js and set the port number that is in the Ganache




```
networks: {
  develop: {
    host: "127.0.0.1",
    port: 7545,
    network_id: "*", // match any network
    websockets: true
```

Command: -truffle migrate --network=develop --reset

```
C:\Users\Exam\Downloads\eth-supplychain-dapp-main>truffle migrate --network=develop --reset
This version of µWS is not compatible with your Node.js build:

Error: node-loader:
Error: The specified module could not be found.
C:\Users\Exam\AppData\Roaming\npm\node_modules\truffle\node_modules\ganache\dist\node/1RGFZdPM.node
Falling back to a NodeJS implementation; performance may be degraded.


Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\Structure.sol
> Compiling .\contracts\SupplyChain.sol
> Compiling .\contracts\rolesUtils[deprecated]\Customer.sol
> Compiling .\contracts\rolesUtils[deprecated]\DeliveryHub.sol
> Compiling .\contracts\rolesUtils[deprecated]\Manufacturer.sol
> Compiling .\contracts\rolesUtils[deprecated]\Roles.sol
> Compiling .\contracts\rolesUtils[deprecated]\SortationHub.sol
> Compiling .\contracts\rolesUtils[deprecated]\Thirdparty.sol
> Artifacts written to C:\Users\Exam\Downloads\eth-supplychain-dapp-main\client\src\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang


Starting migrations...
======================
> Network name:    'develop'
> Network id:      5777
> Block gas limit: 6721975000 (0x190a932d8)
```

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

   Deploying 'Migrations'
   ----------------------
   > transaction hash:    0x70a4a3f715972eec6cd990b42f59bf2b68d84ead61ca5f03f314621f9df5ef70
   > Blocks: 0            Seconds: 0
   > contract address:    0x2D3597d33702186e004B042c6f96823314B16DfE
   > block number:        1
   > block timestamp:     1669003917
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.9974206
   > gas used:            128970 (0x1f7ca)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0025794 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0025794 ETH


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

   Deploying 'Manufacturer'
   ------------------------
   > transaction hash:    0x0c312942ae0b3d0f68c3b9523c43ee7e257d242d596c3c3bb3a452539e6fd170
   > Blocks: 0            Seconds: 0
   > contract address:    0x857f27c957833c41Fb2434698F325c0198193EbF
   > block number:        3
   > block timestamp:     1669003918
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.99244462
   > gas used:            206533 (0x326c5)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00413066 ETH



   Deploying 'Roles'
   -----------------
   > transaction hash:    0xae28ef771bbf60f2311daf8dcae5ce039c22261fc9eecb7a15b50a4923d7baf5
   > Blocks: 0            Seconds: 0
   > contract address:    0x830ef690432Fc56Dd5f565BAD8F8CA0Ef3699A66
   > block number:        4
   > block timestamp:     1669003919
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.99101034
   > gas used:            71714 (0x11822)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00143428 ETH


   Deploying 'SupplyChain'
   -----------------------
   > transaction hash:    0x794ad36ae41cd7c51e17e0dedc4845402250139b79db2835cbee3fa533d02426
   > Blocks: 0            Seconds: 0
   > contract address:    0xcF4fEAd5C5d7125FACd2B80BACB184A6D7C95047
   > block number:        5
   > block timestamp:     1669003919
   > account:             0x76D41FB06d4FAd737f14631f0B60dE62cFEA53f7
   > balance:             99.91111894
   > gas used:            3994570 (0x3cf3ca)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0798914 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.08545634 ETH

Summary
=======
> Total deployments:   4
> Final cost:          0.08803574 ETH



C:\Users\Exam\Downloads\eth-supplychain-dapp-main>
```

Command: - cd client

```
C:\Users\Exam\Downloads\eth-supplychain-dapp-main>cd client

C:\Users\Exam\Downloads\eth-supplychain-dapp-main\client>
```

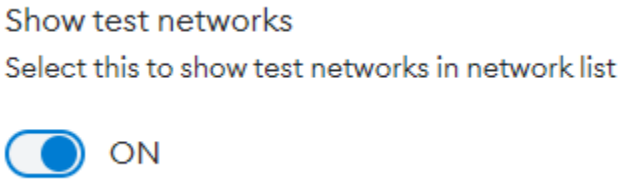
Command: - npm install

```
C:\Users\Exam\Downloads\eth-supplychain-dapp-main\client>npm install
[                ] \ idealTree: timing idealTree Completed in 1909ms


116 packages are looking for funding
  run `npm fund` for details

101 vulnerabilities (4 low, 26 moderate, 38 high, 33 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\Users\Exam\Downloads\eth-supplychain-dapp-main\client>
```

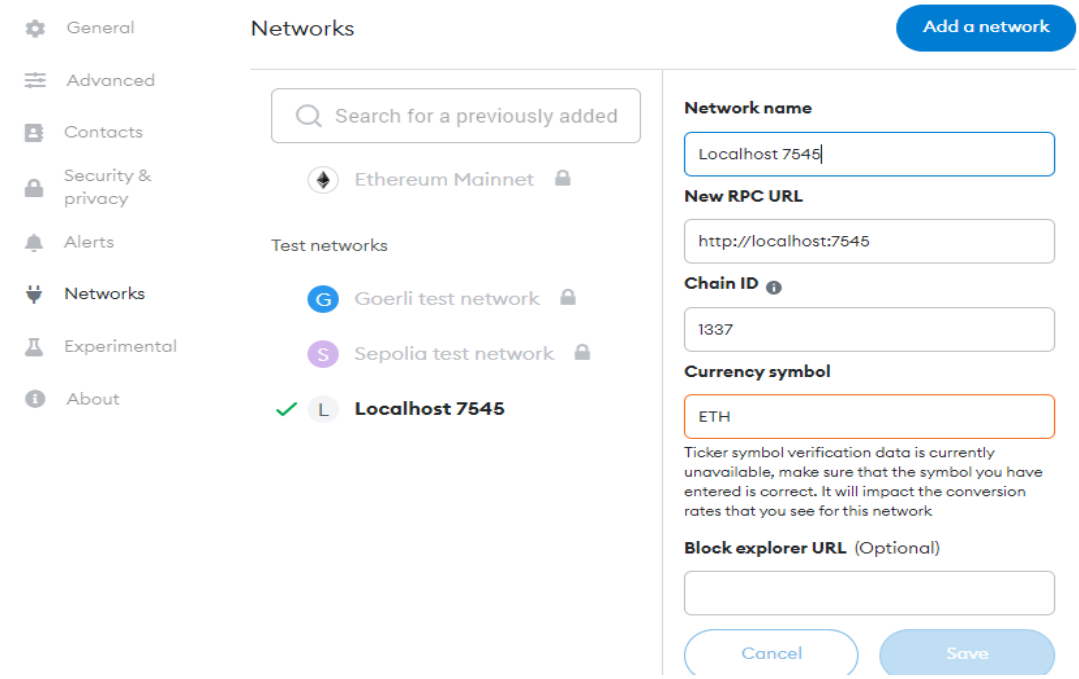
Command: - npm start

```
C:\Users\Exam\Downloads\eth-supplychain-dapp-main\client>npm start
```

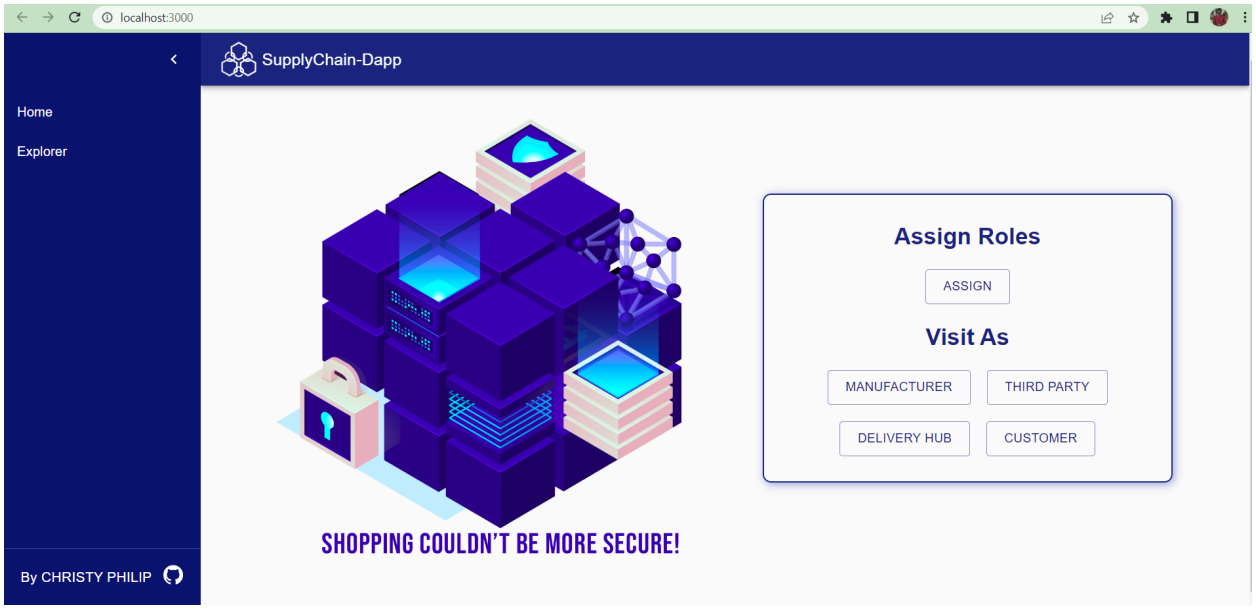
Goto Metamask→Setting→Show test networks to be made ON



Networks as shown in below image change RPC URL port number it should be set of the Ganache



OUTPUT: -



CREATE 4 ACCOUNTS ON METAMASK [IMPORT FROM GANACHE]:

AND ASSIGN THE PRIVATE KEY ADDRESS TO MANUFACTURER, THIRD PARTY, DELIVERY, CUSTOMER:

MANUFACTURER: ACCOUNT 2

THIRD PARTY: ACCOUNT 3

DELIVERY HUB: ACCOUNT 4

CUSTOMER: ACCOUNT 5

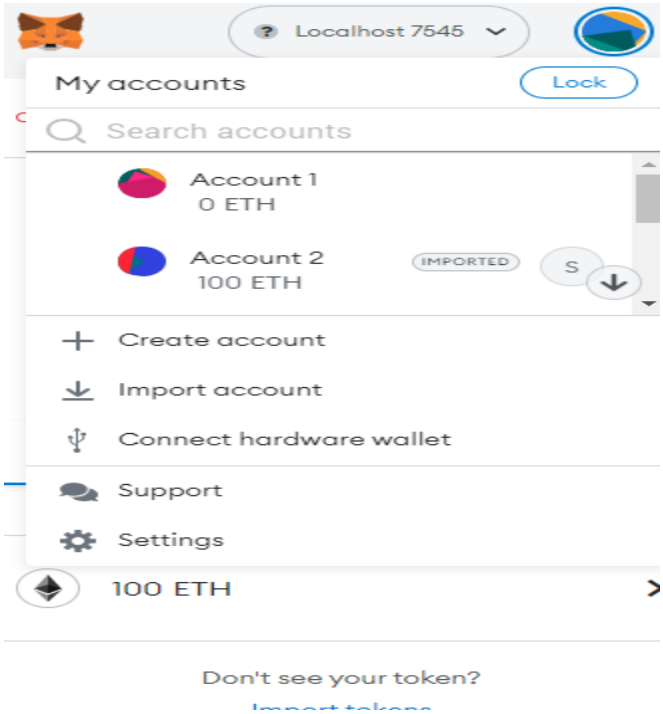
Note: -

1) First choose the account of who's entering address and all the account should be connected to website

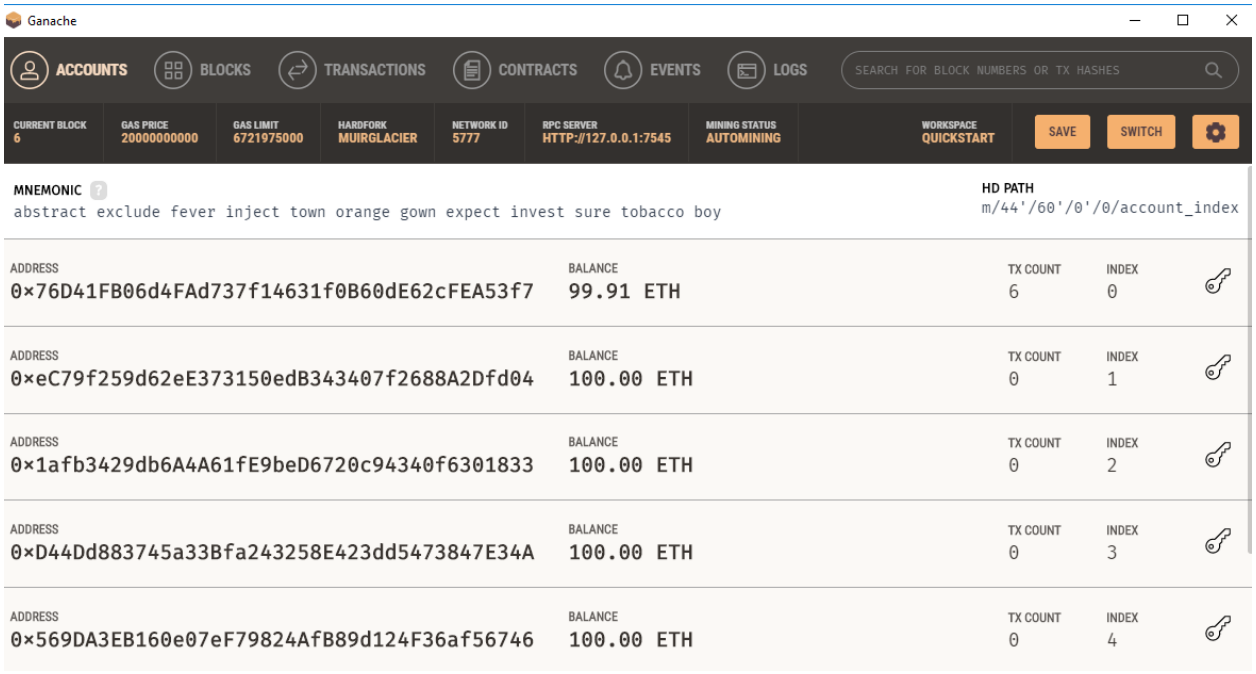
2) Each time while switching the account Browser needs to be refreshed

3) Whose account is selected his address should be entered and corresponding button should be clicked at that time metmask prompts click on confirm.

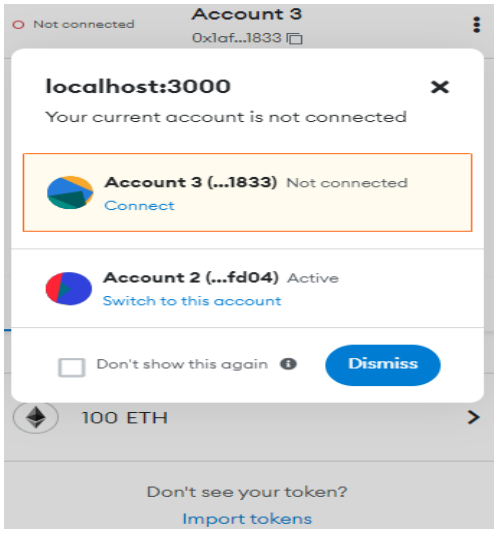
ACCOUNTS:



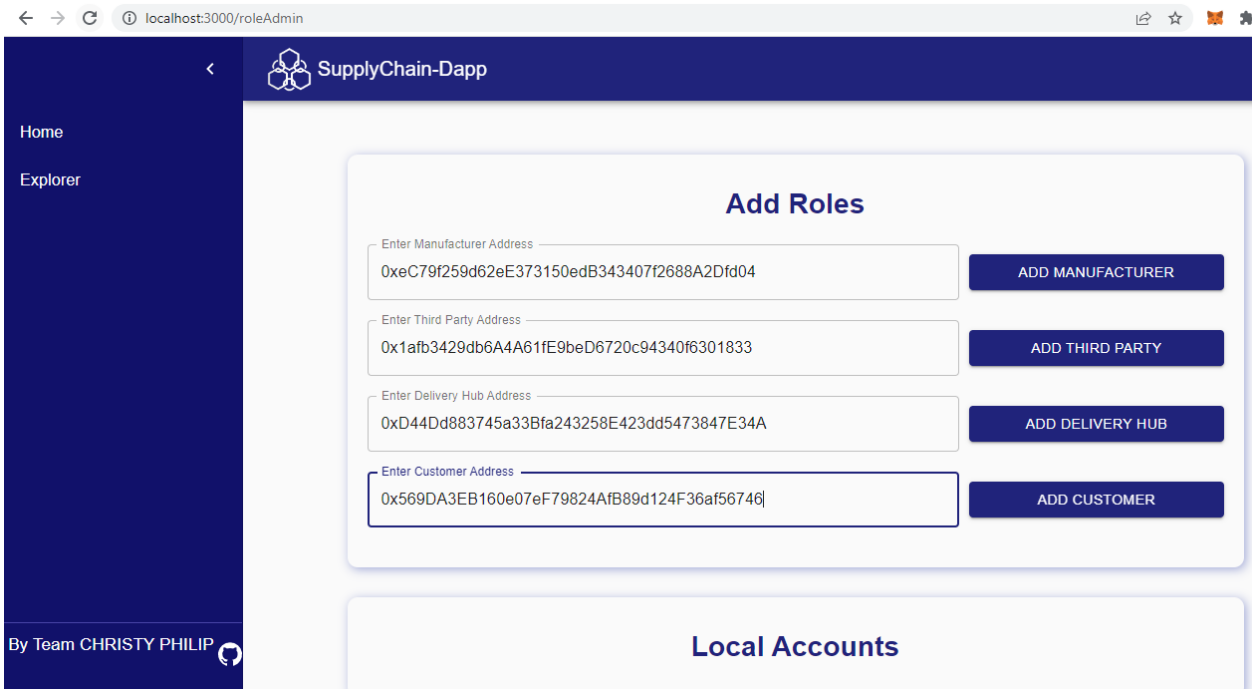
GANACHE ADDRESSES FOR IMPORTING INTO METMASK(99.91 NOT TO CHOOSE)



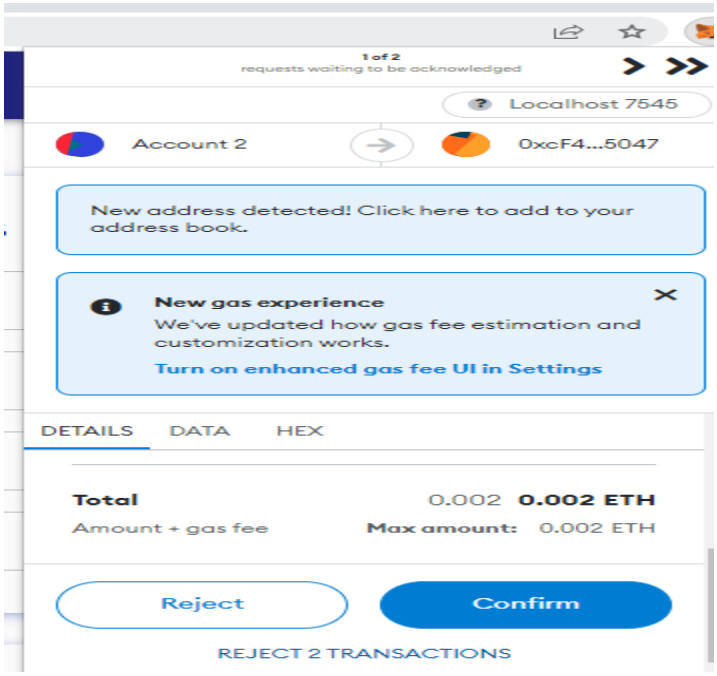
Click on connect to connect website. Do it for each account



ADD ROLES:

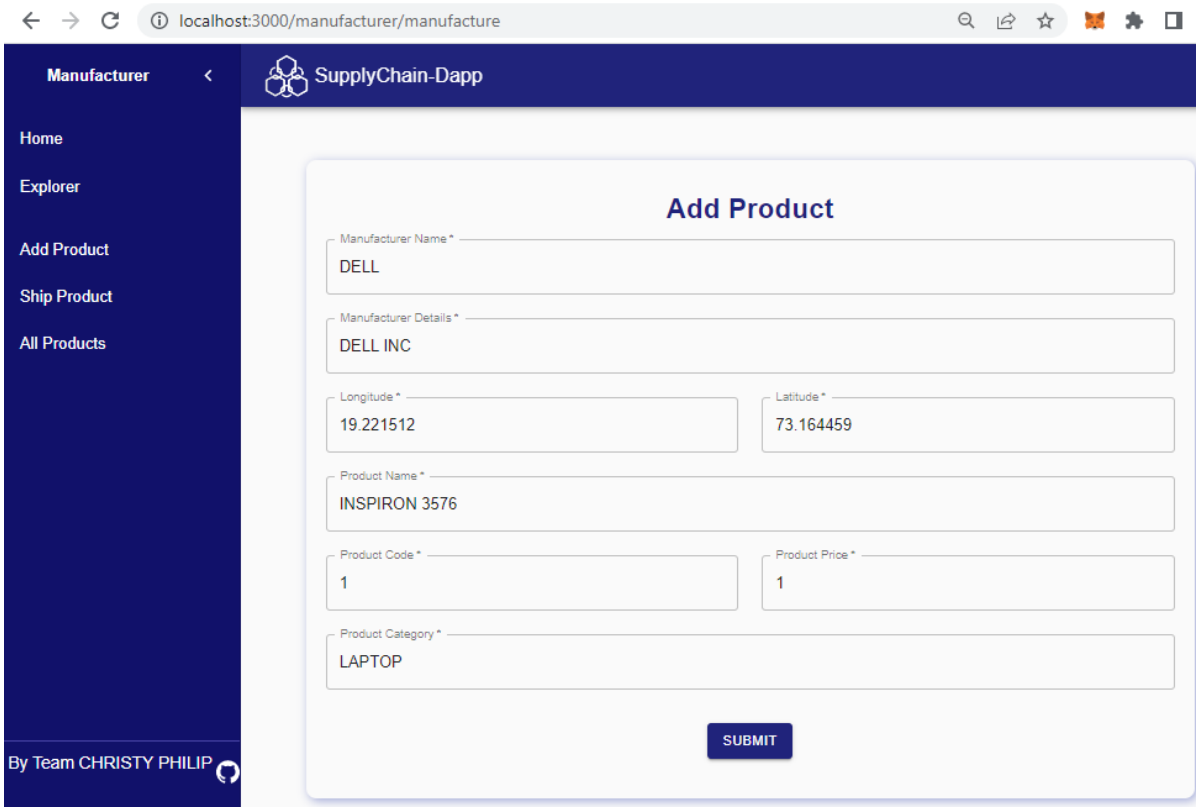


Set The account in metamask which we are adding as
Suppose manufacture then account of manufacture should be chosen in metamask
SET ACCOUNT(MANUFACTURER) → RELOAD SITE→ENTER THE ADDRESS→Click on ADD MANUFACTURER→GOTO METMASK→CLICK CONFIRM AS SHOWN BELOW

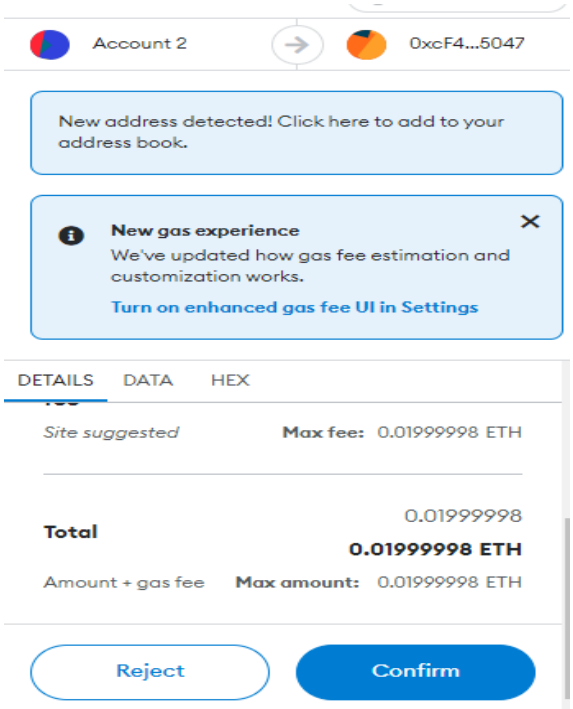


MANUFACTURER:

ADD PRODUCT:

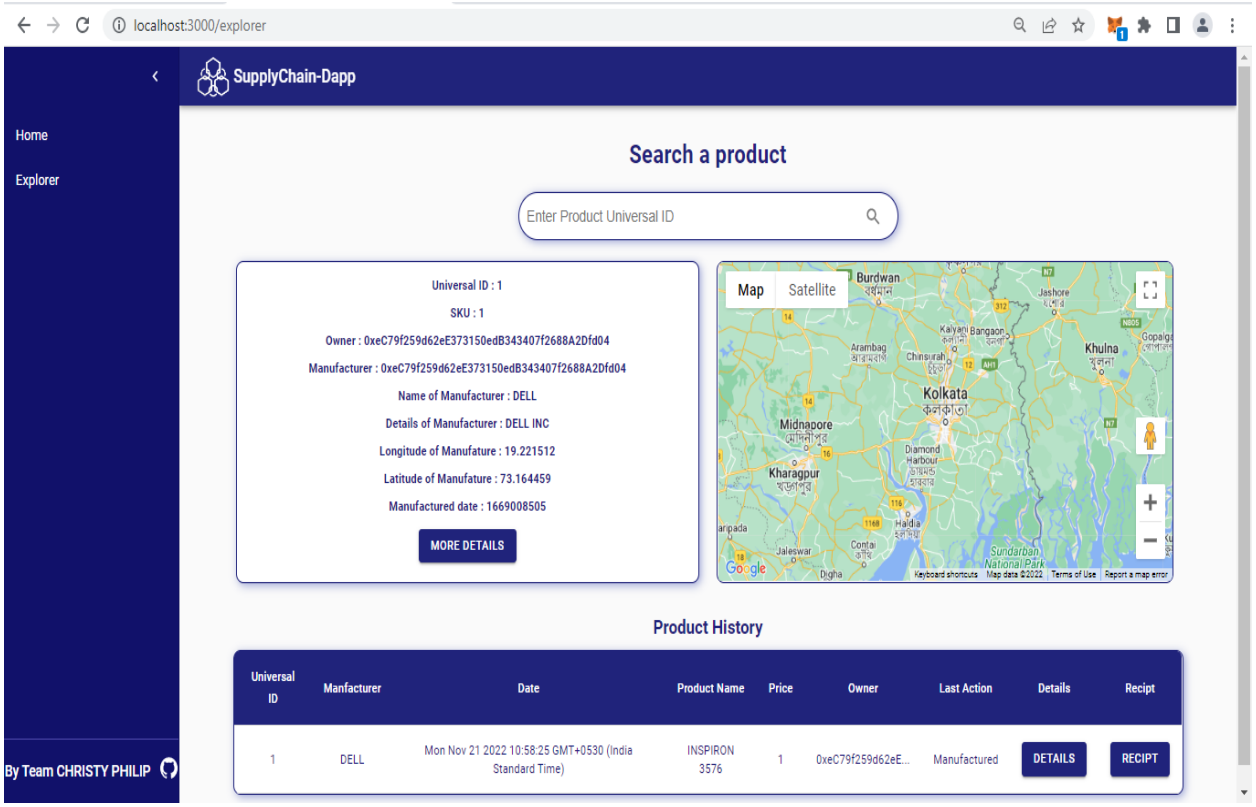


CONFIRM TRANSACTION:

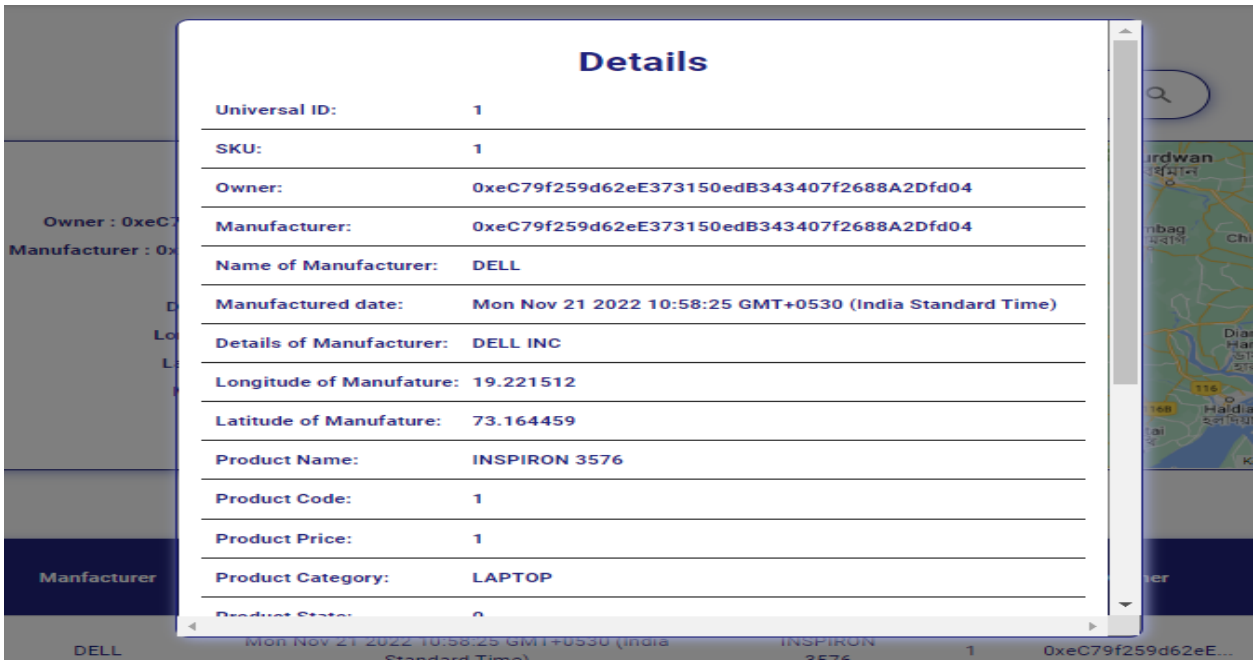


EXPLORER:

CHECK WHETHER THE PRODUCT IS ADDED SUCCESSFULLY OR NOT:



Click on MORE DETAILS

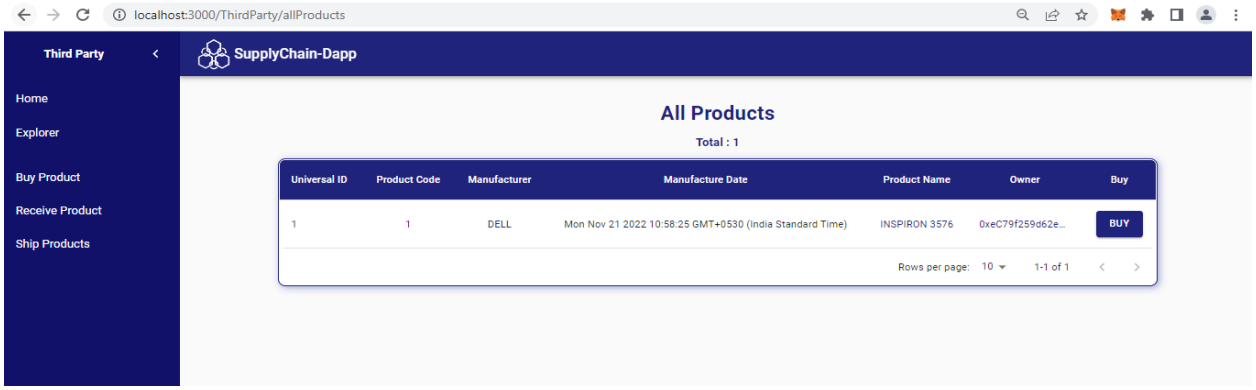


Click on RECEIPT

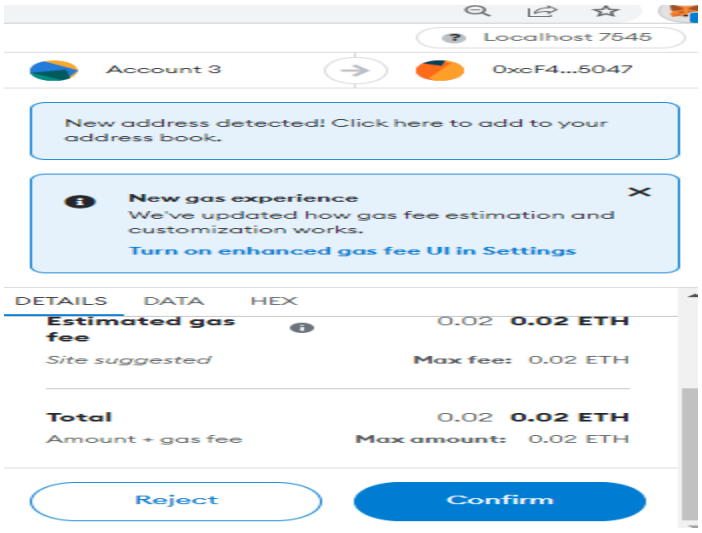
Recipt

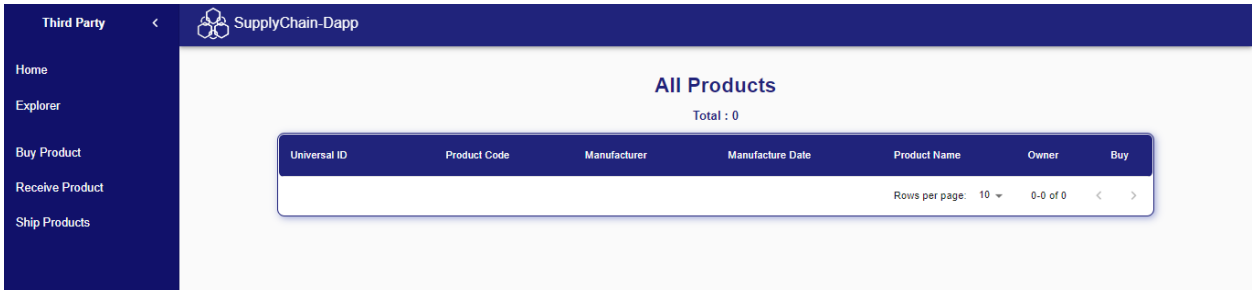
Tx hash:	0x33a34f32a6780cbecf1480e96d86c3e9a67e69c3d59faa9023e9deba6d3e7afb
Block hash:	0xe731aa898e22ec0072bfa2e5b56e6cd1f9e522d91a69b675723c8aa7cc481b57
Block Number:	11
From:	0xeC79f259d62eE373150edB343407f2688A2Dfd04
To:	0xcF4fEAd5C5d7125FACd2B80BACB184A6D7C95047
Nonce:	1
Input:	0x003375b700000000000000000000000000000000000000000000000000000000000000010...
Gas:	999999
Value:	0

THIRD PARTY:
BUY PRODUCT:

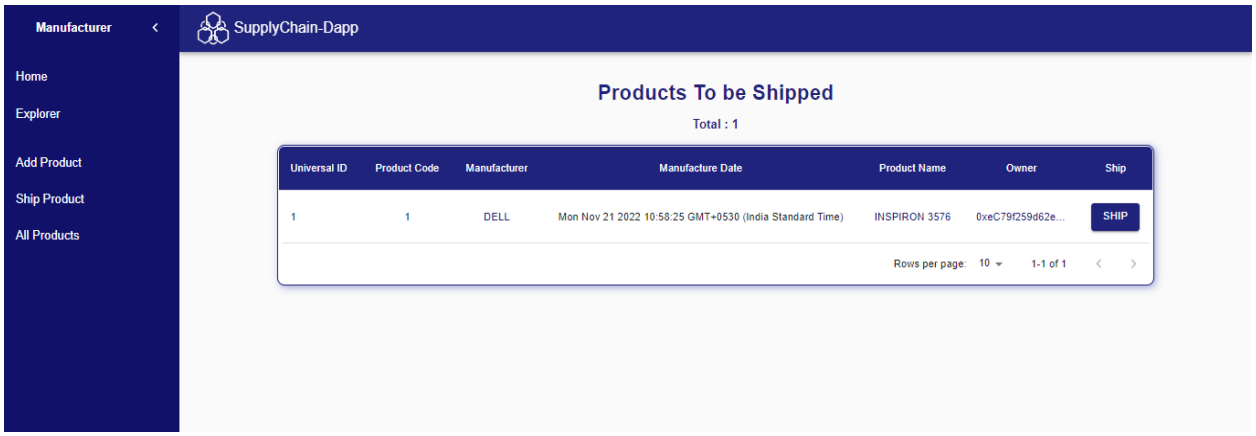


CONFIRM TRANSACTION:

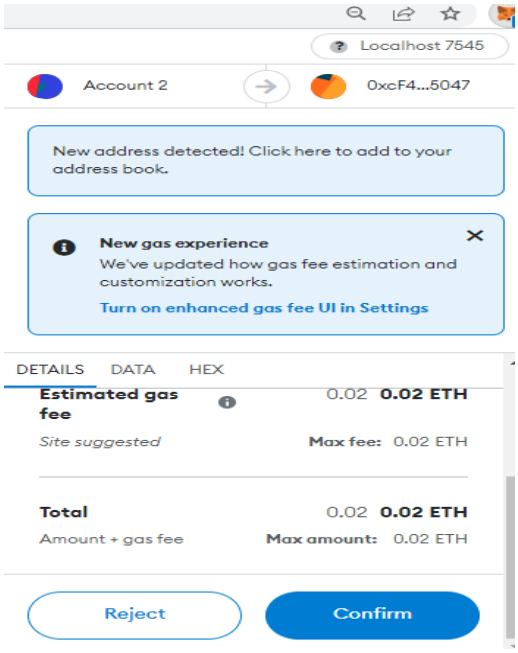




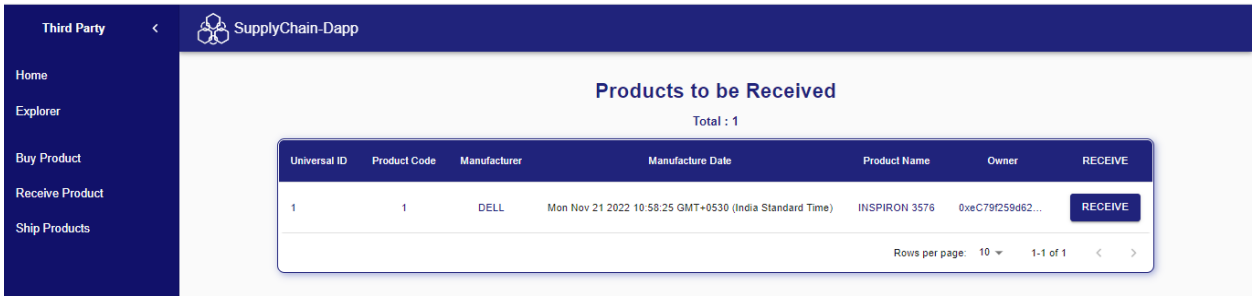
SHIP PRODUCT:



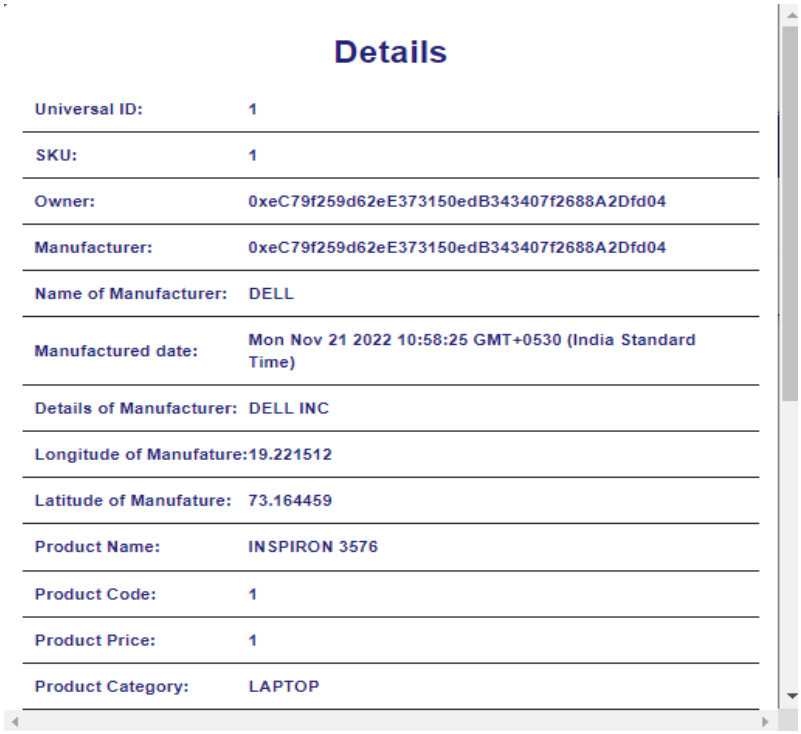
CONFIRM TRANSACTION:

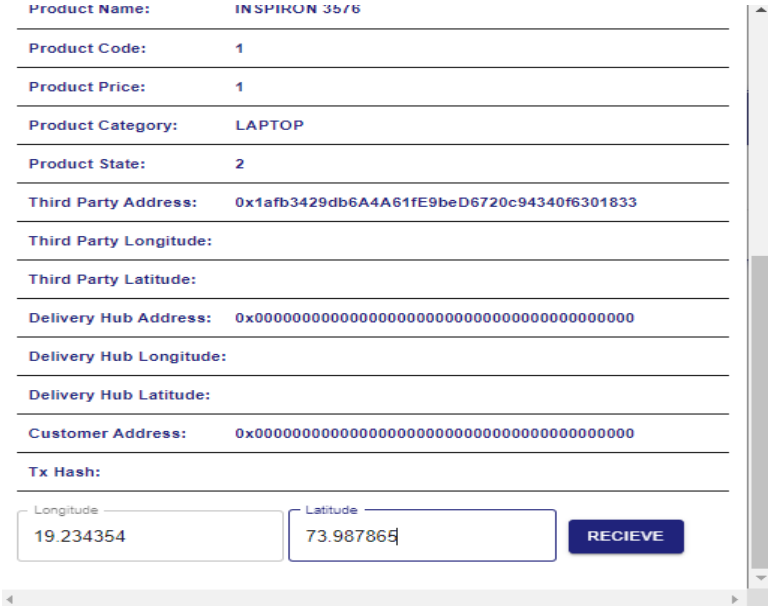


RECEIVE PRODUCT:

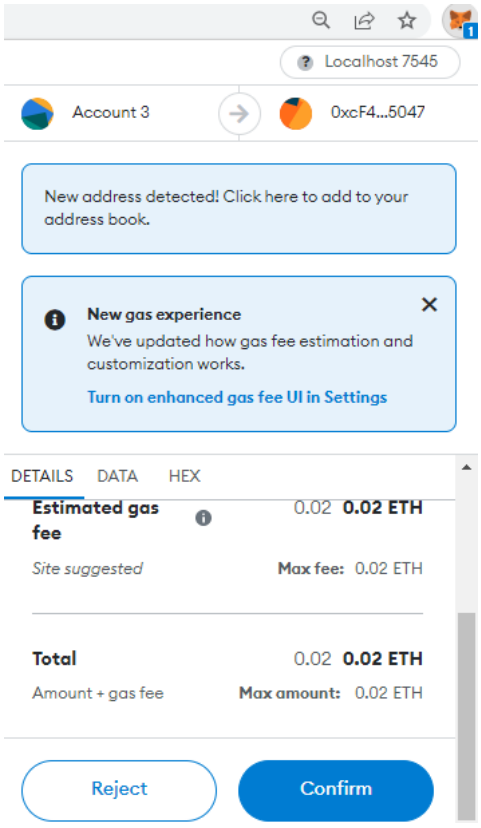


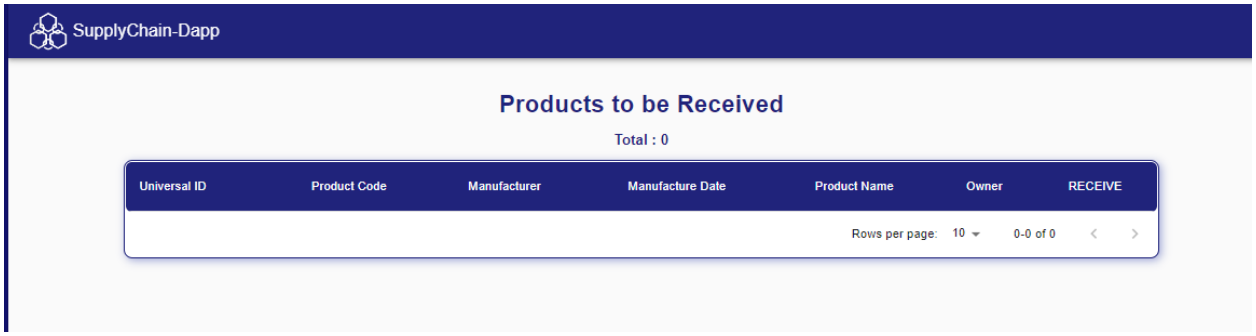
Click on RECEIVE:





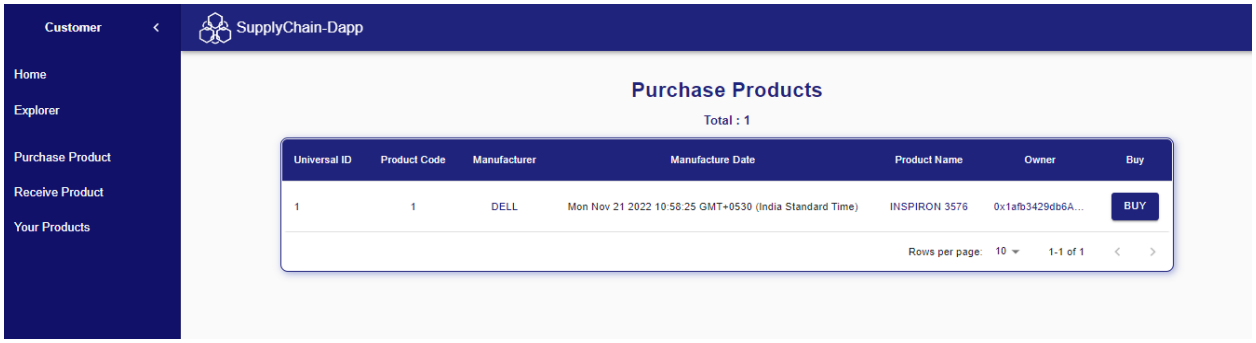
CONFIRM TRANSACTION:



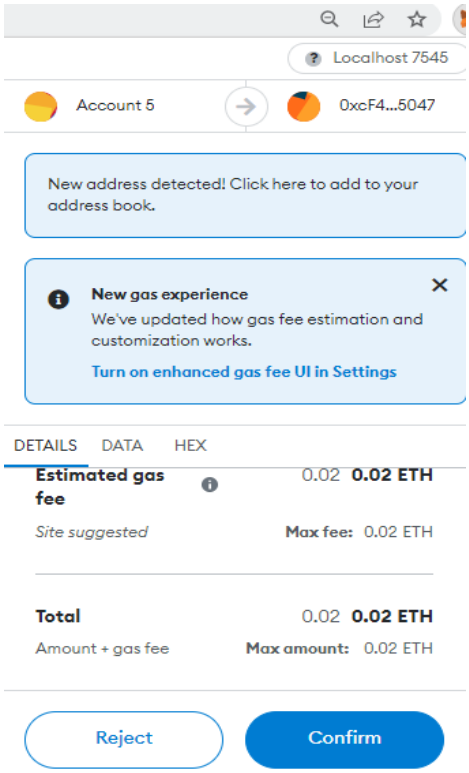


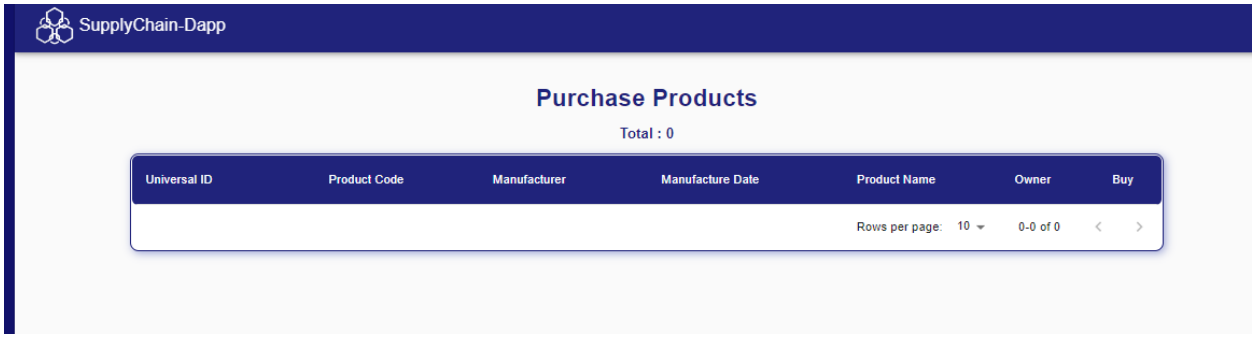
CUSTOMER:

PURCHASE PRODUCT:

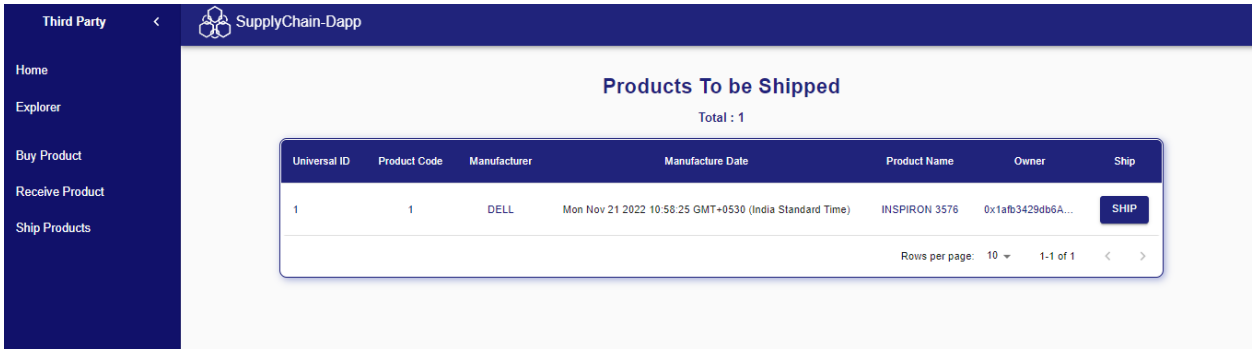


CONFIRM TRANSACTION:

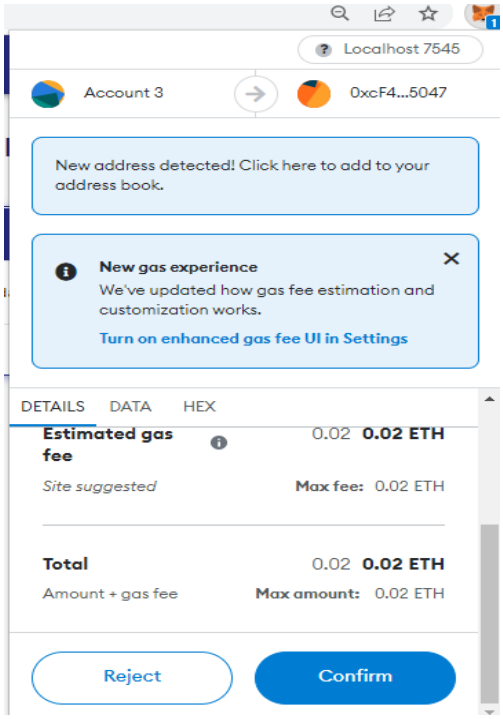


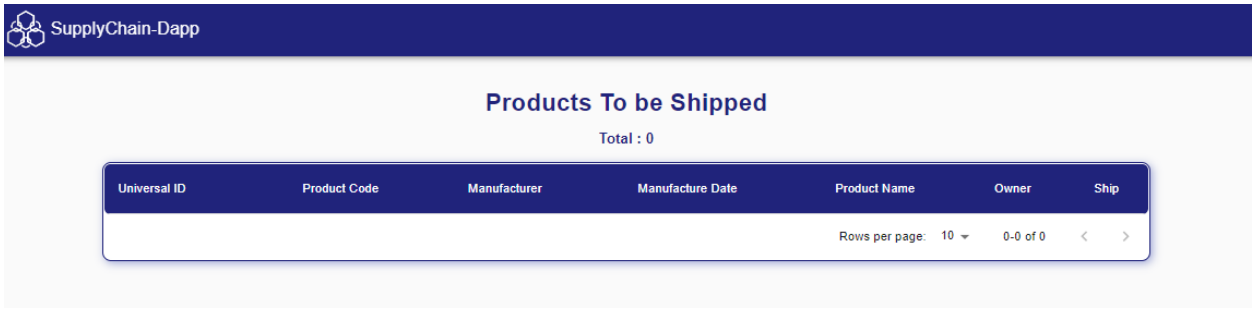


THIRD PARTY:
PRODUCTS TO BE SHIPPED:



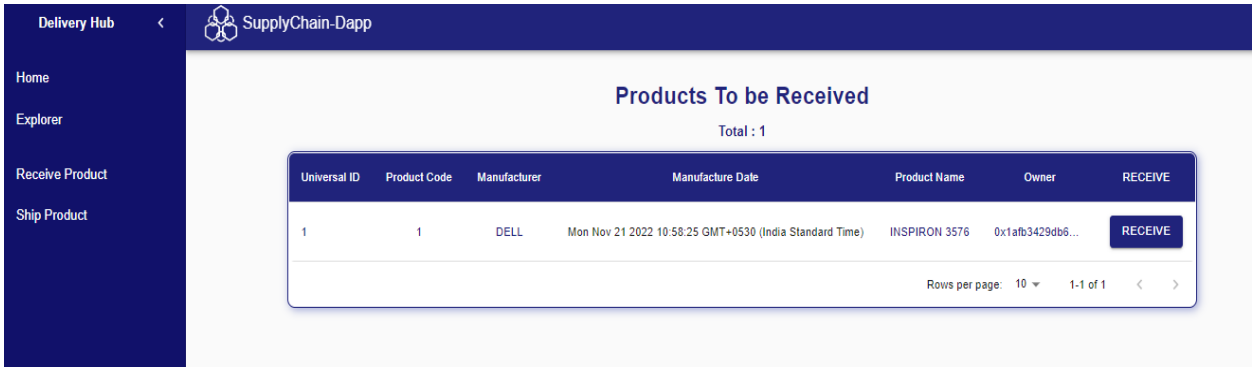
CONFIRM TRANSACTIONS:



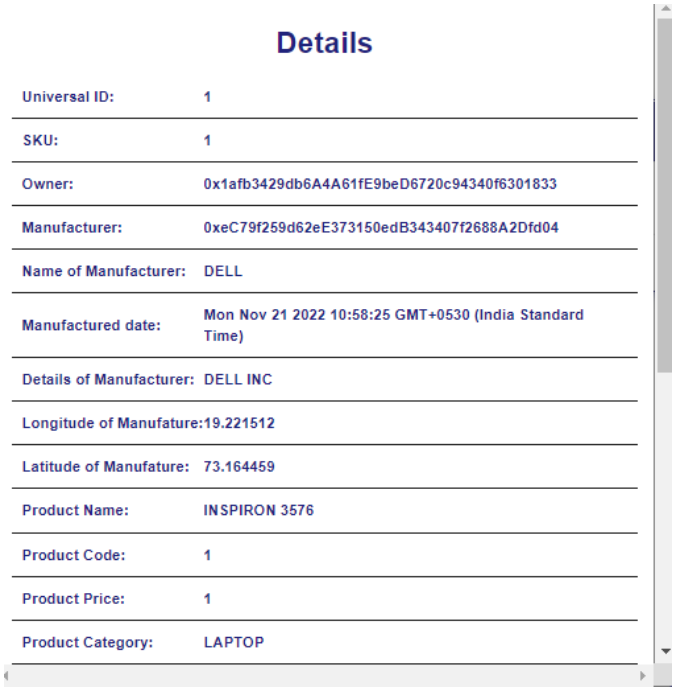


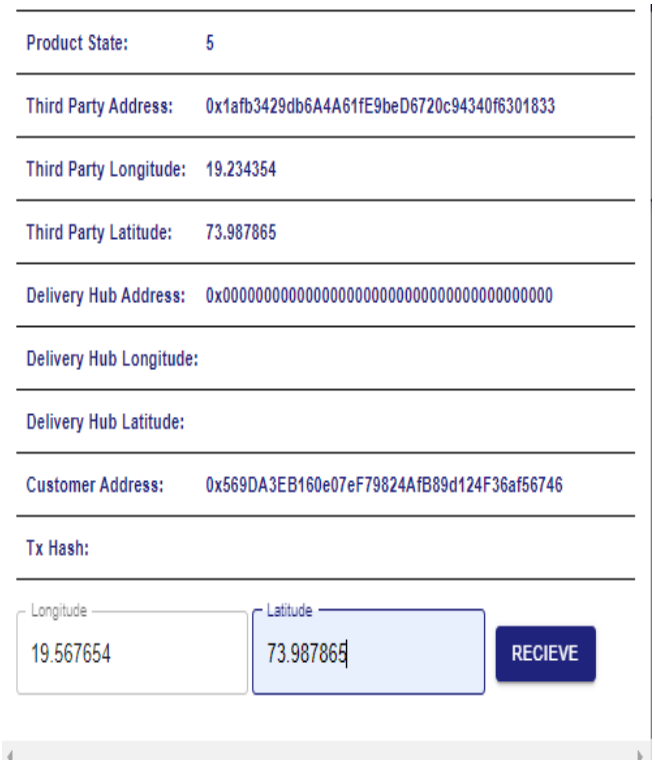
DELIVERY HUB:

Products To be Received:

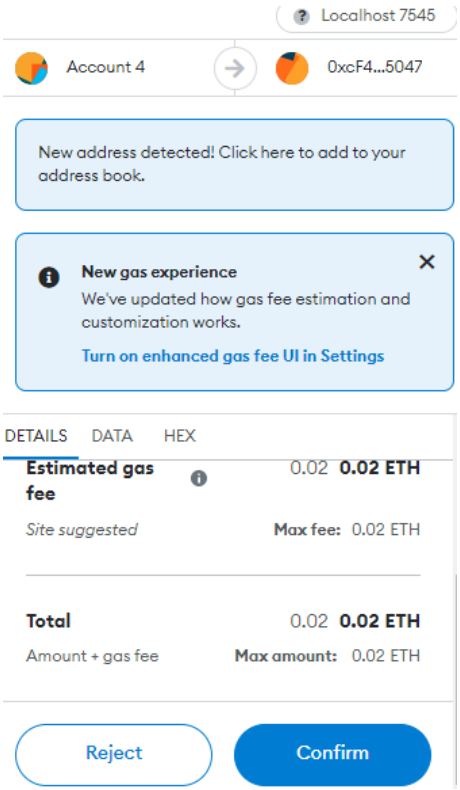


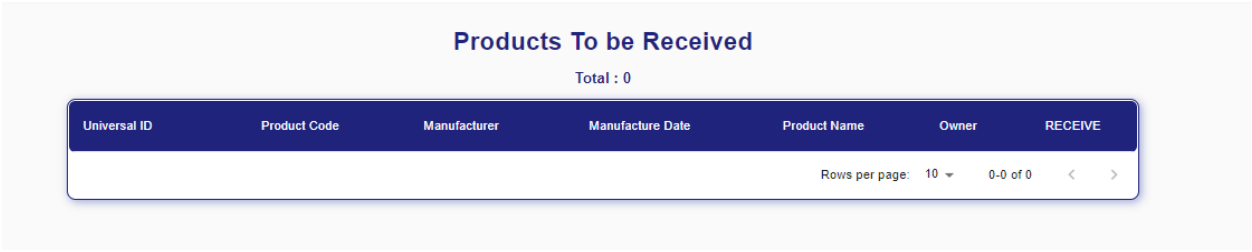
CLICK ON RECEIVE:



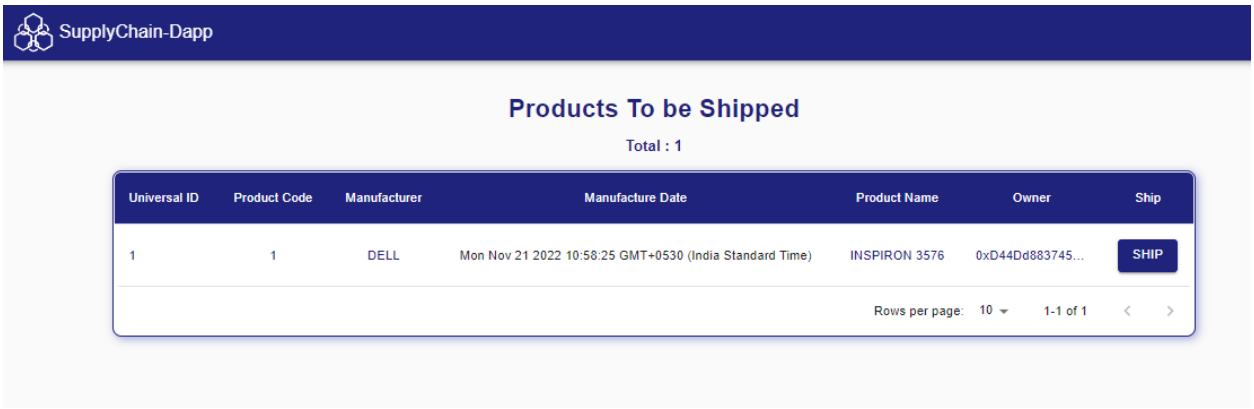


CONFIRM TRANSACTIONS:

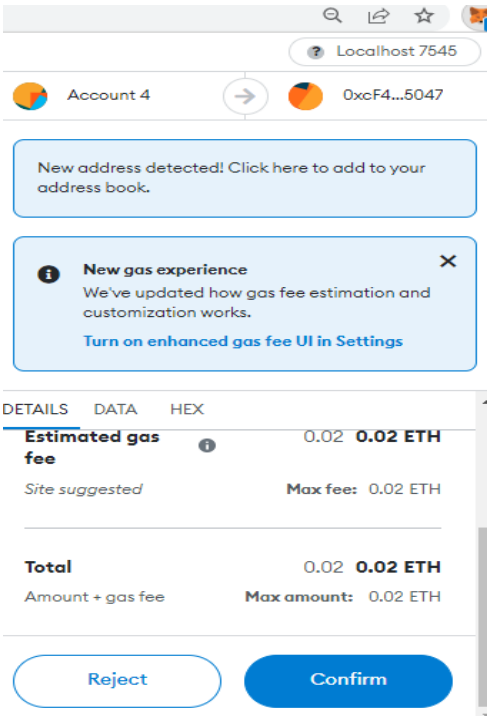


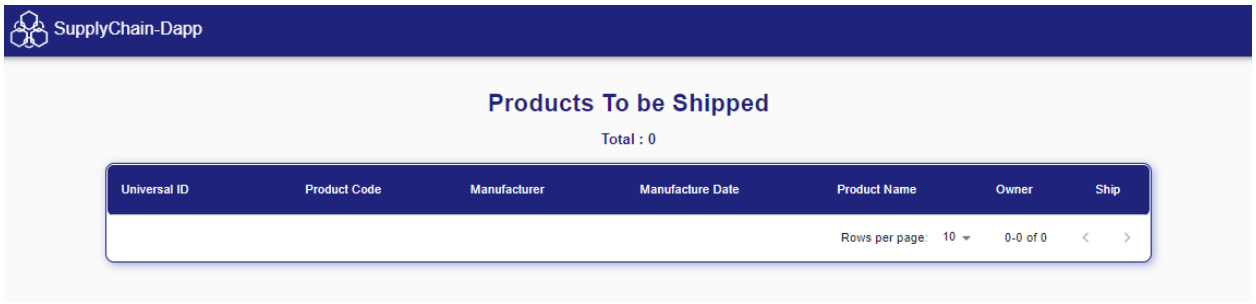


DELIVERY HUB:
SHIP PRODUCT:



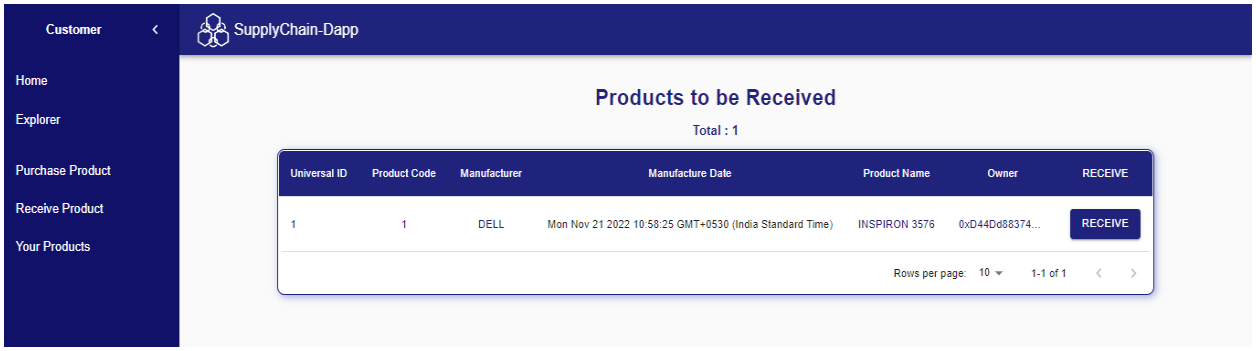
CONFIRM TRANSACTIONS:



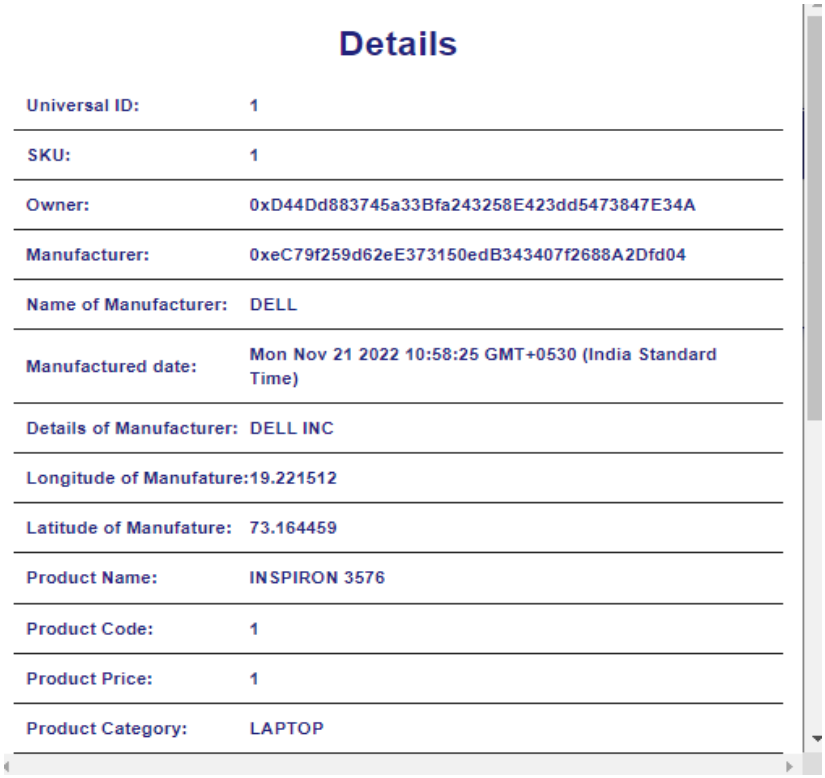


CUSTOMER:

RECEIVE:



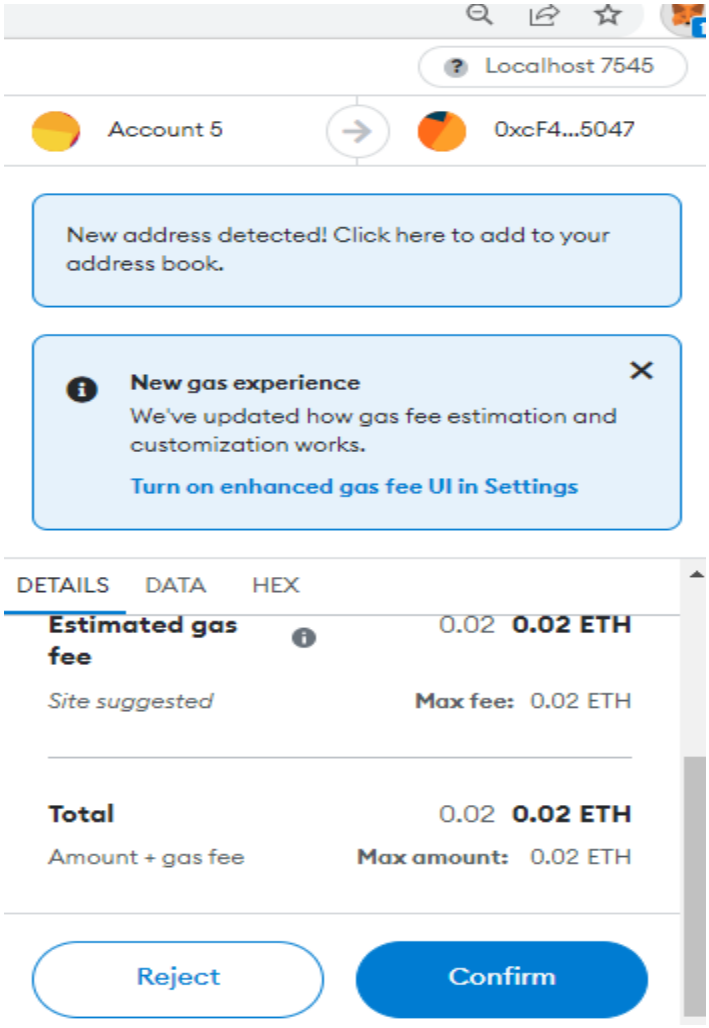
CLICK ON RECEIVE:

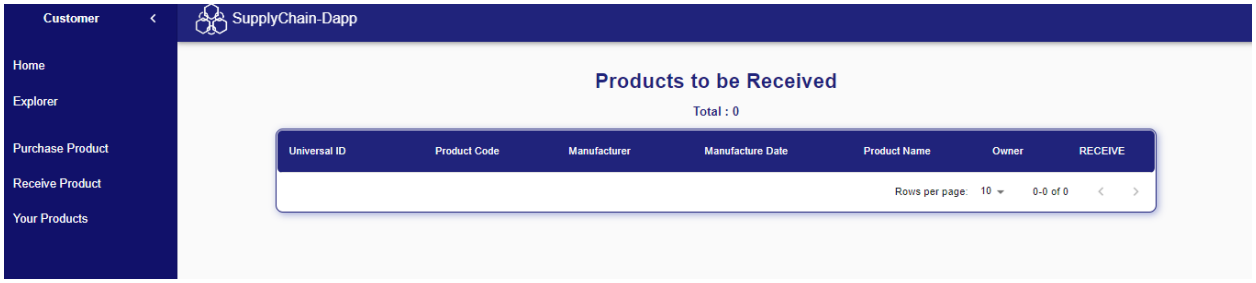


Product State:	7
Third Party Address:	0x1afb3429db6A4A61fE9beD6720c94340f6301833
Third Party Longitude:	19.234354
Third Party Latitude:	73.987865
Delivery Hub Address:	0xD44Dd883745a33Bfa243258E423dd5473847E34A
Delivery Hub Longitude:	19.567654
Delivery Hub Latitude:	73.987865
Customer Address:	0x569DA3EB160e07eF79824AfB89d124F36af56746
Tx Hash:	

RECIEVE

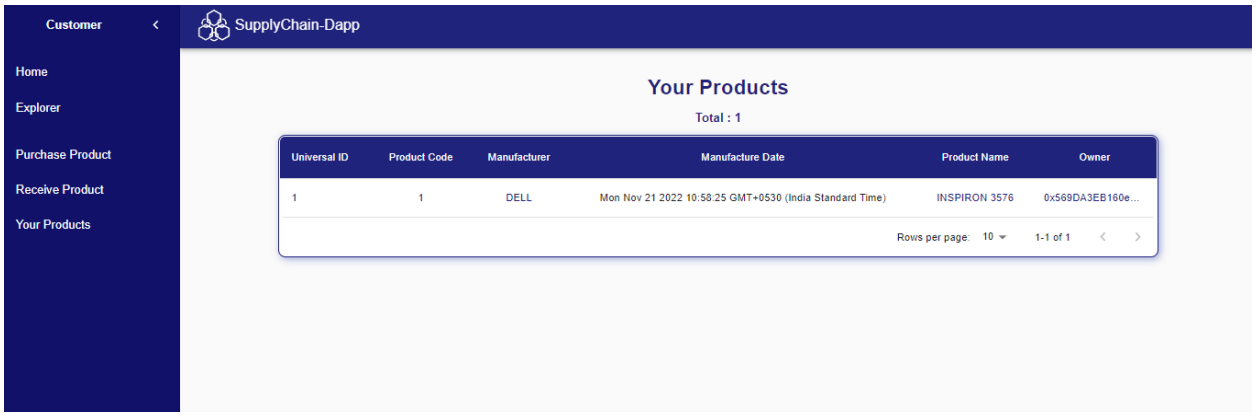
CONFIRM TRANSACTIONS:





CUSTOMER:

PRODUCT:



EXPLORER:

PRODUCT HISTORY: [ALL TRANSACTIONS]

Universal ID	Manfacturer	Date	Product Name	Price	Owner	Last Action	Details	Recipt
1	DELL	Mon Nov 21 2022 10:58:25 GMT+0530 (India Standard Time)	INSPIRON 3576	1	0xeC79f259d62eE...	Manufactured	DETAILS	RECIPT
1	DELL	Mon Nov 21 2022 10:58:25 GMT+0530 (India Standard Time)	INSPIRON 3576	1	0xeC79f259d62eE...	Bought By Third Party	DETAILS	RECIPT
1	DELL	Mon Nov 21 2022 10:58:25 GMT+0530 (India Standard Time)	INSPIRON 3576	1	0xeC79f259d62eE...	Shipped From Manufacturer	DETAILS	RECIPT
1	DELL	Mon Nov 21 2022 10:58:25 GMT+0530 (India Standard Time)	INSPIRON 3576	1	0x1afb3429db6A4...	Received By Third Party	DETAILS	RECIPT
1	DELL	Mon Nov 21 2022 10:58:25 GMT+0530 (India Standard Time)	INSPIRON 3576	1	0x1afb3429db6A4...	Bought By Customer	DETAILS	RECIPT
1	DELL	Mon Nov 21 2022 10:58:25 GMT+0530 (India Standard Time)	INSPIRON 3576	1	0x1afb3429db6A4...	Shipped By Third Party	DETAILS	RECIPT
1	DELL	Mon Nov 21 2022 10:58:25 GMT+0530 (India Standard Time)	INSPIRON 3576	1	0xD44Dd883745a3...	Received at DeliveHub	DETAILS	RECIPT

CONCLUSION: -

From this practical I have learned to create Dapps in Ethereum.

BLOCKCHAIN MINI PROJECT

KYC Verification Using Blockchain

By

Md Abuzer Ansari (04)

Shammas Khan (26)

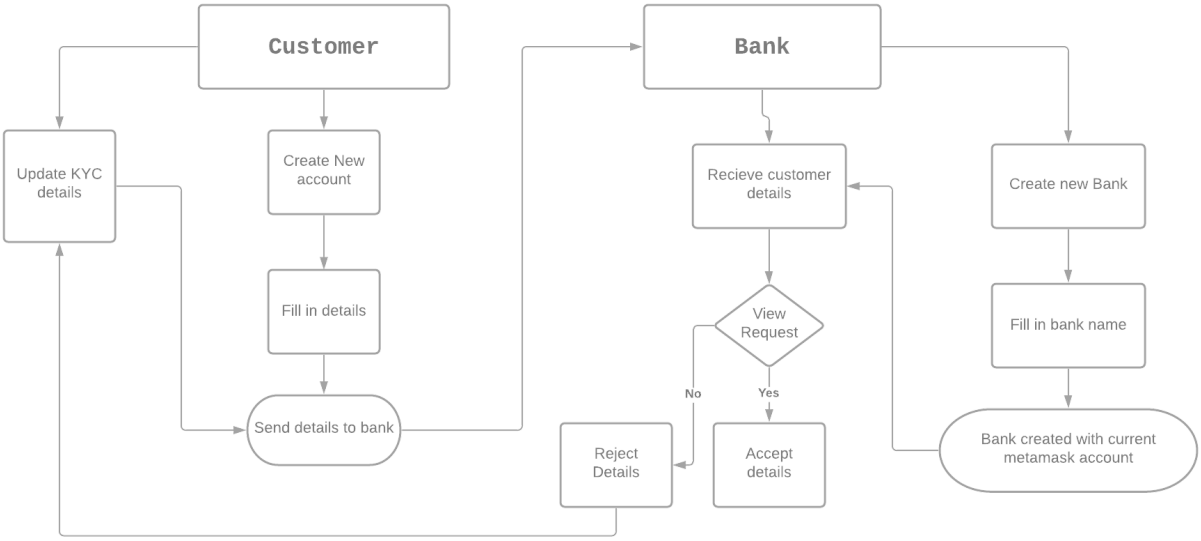
Christy Philip (42)

Introduction:

Since the evaluation of Bitcoin, blockchain has been able to be used in a real-time environment. Banks gather details on the customers' names and addresses through a process known as KYC. This process by which banks obtain information about the identity and address of the customers. KYC is a process that acquire a better understanding of the activities of their potential customers and verify their legality. Due diligence is a process that is supervised by regulators that is used to confirm the legitimacy of clients.

This procedure aids in preventing the abuse of banks' services. The KYC process must be completed by the banks when opening new accounts. Additionally, banks are expected to routinely update the KYC information for their clients. KYC may be laborious, repetitive, and manual across institutions. Financial institutions would be able to achieve better compliance outcomes, boost efficiency, and enhance customer experience by sharing KYC information on Blockchain. Blockchain technology is completely decentralised, immutable, tamper-resistant, and secure. The data within the blockchain network will be accessible only to authorised users, ensuring transparency. This blockchain, in conjunction with the KYC chain, could be used to provide decentralised data storage and transparency. Sharing KYC information on Blockchain would enable financial institutions to deliver better compliance outcomes, increasing efficiency and also improves the experience of the customer. Because of managing the same customer information across multiple banks and financial sectors leads to data redundancy and high maintenance costs for sensitive data. There is a lack of security in the conventional system as well. All of the issues with the conventional technique of KYC verification can be resolved with a blockchain-based approach. Even by providing the user control over the data, we even eliminate third party intervention.

Workflow of the application



Code

Contracts

KycBlockchain.sol

```
pragma solidity >=0.4.21 <0.7.0;

interface KYC_Functions{
    enum Status {Accepted, Rejected, Pending}
    // Checks if the current address is an Organisation(Bank) or not
    function isOrg() external view returns(bool);
    // Checks if the current address is an Customer or not
    function isCus() external view returns(bool);
    // A function to register as a new customer to get your KYC checked by
a bank
    function newCustomer(string calldata _name, string calldata _hash,
address _bank) external payable returns(bool);
    // A function that allows you to be a bank and audit KYC data of
customers
    function newOrganisation(string calldata _name) external payable
returns(bool);
    // A function which is only visible to the bankers so they can verify
the data
    function viewCustomerData(address _address) external view
returns(string memory);
```

```
    // Customers also can change the their data if the KYC request gets
rejected
    function modifyCustomerData(string calldata _name,string calldata
_hash, address _bank)
    external payable returns(bool);
    // Checks the status of a customers KYC Request (Approved or Rejected
or Pending)
    function checkStatus() external returns(Status);
    // Function that can change the status of a request (Only for banks)
    function changeStatusToAccepted(address _custaddress) external
payable;
    // Function that can change the status of a request (Only for banks)
    function changeStatusToRejected(address _custaddress) external
payable;
    // A function that enables the bank to lookup at all the KYC requests
pointed at it
    function viewRequests() external view returns(address[] memory);
    // A function that returns the name of a customer
    function viewName(address _address) external view returns(string
memory);

}

contract KycBlockChain is KYC_Functions{

    address[] public Banks;
    address[] public Requests;
    uint public bankslength=0;

    enum Entity { Customer, Organisation }


    struct Customer{
        string c_name;
        string data_hash;
        address bank_address;
        bool exists;
        Entity entity;
    }
```

```
    struct Organisation{
        string b_name;
        bool exists;
        Entity entity;
        mapping(address => Status) requests;
        address[] allrequests;
    }

    mapping(address => Customer) allCustomers;
    mapping(address => Organisation) allOrganisations;

    function isOrg() public view returns(bool){
        if(allOrganisations[msg.sender].exists){
            return true;
        }
        return false;
    }

    function isCus() public view returns(bool){
        if(allCustomers[msg.sender].exists){
            return true;
        }
        return false;
    }

    function newCustomer(string memory _name, string memory _hash, address
_bank) public payable returns(bool){
        require(!isCus(),"Customer Already Exists!");
        require(allOrganisations[_bank].exists,"No such Bank!");
        allCustomers[msg.sender].c_name = _name;
        allCustomers[msg.sender].data_hash = _hash;
        allCustomers[msg.sender].bank_address = _bank;
        // allCustomers[msg.sender].access[msg.sender] = true;
        allCustomers[msg.sender].exists = true;
        allCustomers[msg.sender].entity = Entity.Customer;
        notifyBank(_bank);
        return true;

    }
```

```
    function newOrganisation(string memory _name) public payable
returns(bool){
        require(!isOrg(),"Organisation already exists with the same
address!");
        allOrganisations[msg.sender].b_name = _name;
        allOrganisations[msg.sender].exists = true;
        allOrganisations[msg.sender].entity = Entity.Organisation;
        Banks.push(msg.sender);
        bankslength++;
        return true;
    }

    function viewCustomerData(address _address) public view returns(string
memory){
        require(isOrg(),"Access Denied");
        if(allCustomers[_address].exists){
            return allCustomers[_address].data_hash;
        }
        return "No such Customer in the database";
    }

    function modifyCustomerData(string memory _name,string memory _hash,
address _bank) public payable returns(bool){
        require(isCus(),"You are not a customer");
        allCustomers[msg.sender].c_name = _name;
        allCustomers[msg.sender].data_hash = _hash;
        allCustomers[msg.sender].bank_address = _bank;
        return true;
    }

    function notifyBank(address _bankaddress) internal {
        allOrganisations[_bankaddress].requests[msg.sender] =
Status.Pending;
        allOrganisations[_bankaddress].allrequests.push(msg.sender);
    }

    function checkStatus() public returns(Status) {
        require(isCus(),"You are not a customer");
        address _presbank = allCustomers[msg.sender].bank_address;
        return allOrganisations[_presbank].requests[msg.sender];
```

```
    }

    function changeStatusToAccepted(address _custaddress) public payable{
        require(isOrg(),"You are not permitted to use this function");
        address _bank = allCustomers[_custaddress].bank_address;
        require(_bank == msg.sender,"You dont have access to verify this
data");
        allOrganisations[msg.sender].requests[_custaddress] =
Status.Accepted;
    }

    function changeStatusToRejected(address _custaddress) public payable{
        require(isOrg(),"You are not permitted to use this function");
        address _bank = allCustomers[_custaddress].bank_address;
        require(_bank == msg.sender,"You dont have access to verify this
data");
        allOrganisations[msg.sender].requests[_custaddress] =
Status.Rejected;
    }

    function viewRequests() public view returns(address[] memory){
        require(isOrg(),"You are not Permitted");
        return allOrganisations[msg.sender].allrequests;
    }

    function viewName(address _address) public view returns(string
memory){
        require(isOrg(),"Not an Organisation");
        return allCustomers[_address].c_name;
    }

}
```

Migrations.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.21 <0.7.0;

contract Migrations {
  address public owner;
```

```
  uint public last_completed_migration;

  modifier restricted() {
    if (msg.sender == owner) _;
  }

  constructor() public {
    owner = msg.sender;
  }

  function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
  }
}
```

Migrations

1_initial_migration.js

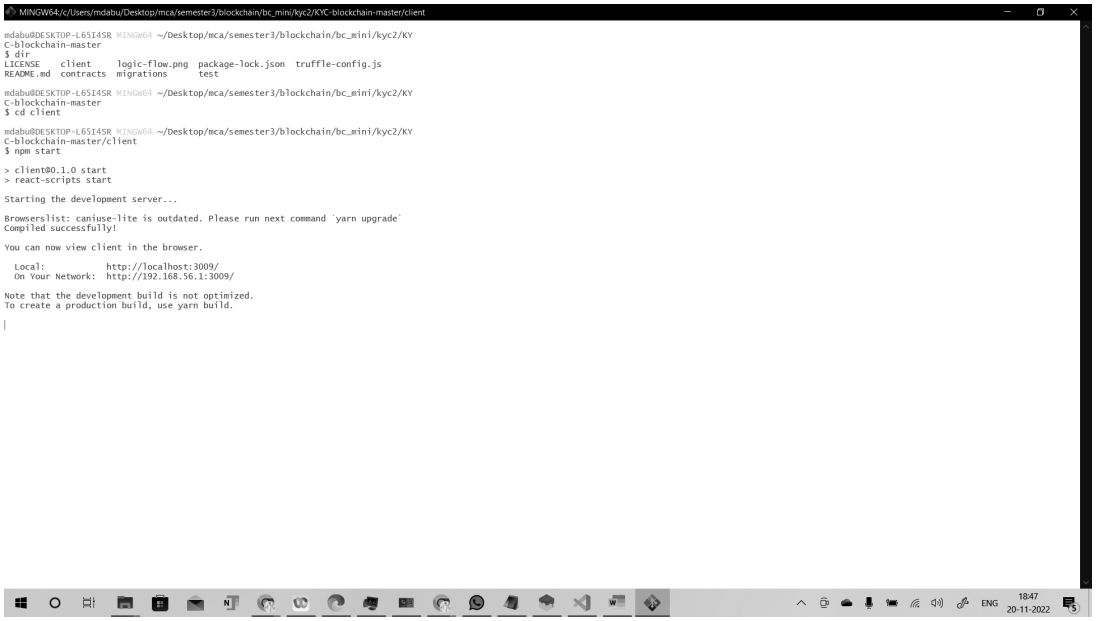
```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.21 <0.7.0;

contract Migrations {
  address public owner;
  uint public last_completed_migration;

  modifier restricted() {
    if (msg.sender == owner) _;
  }

  constructor() public {
    owner = msg.sender;
  }

  function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
  }
}
```

2_deploy_contracts.js

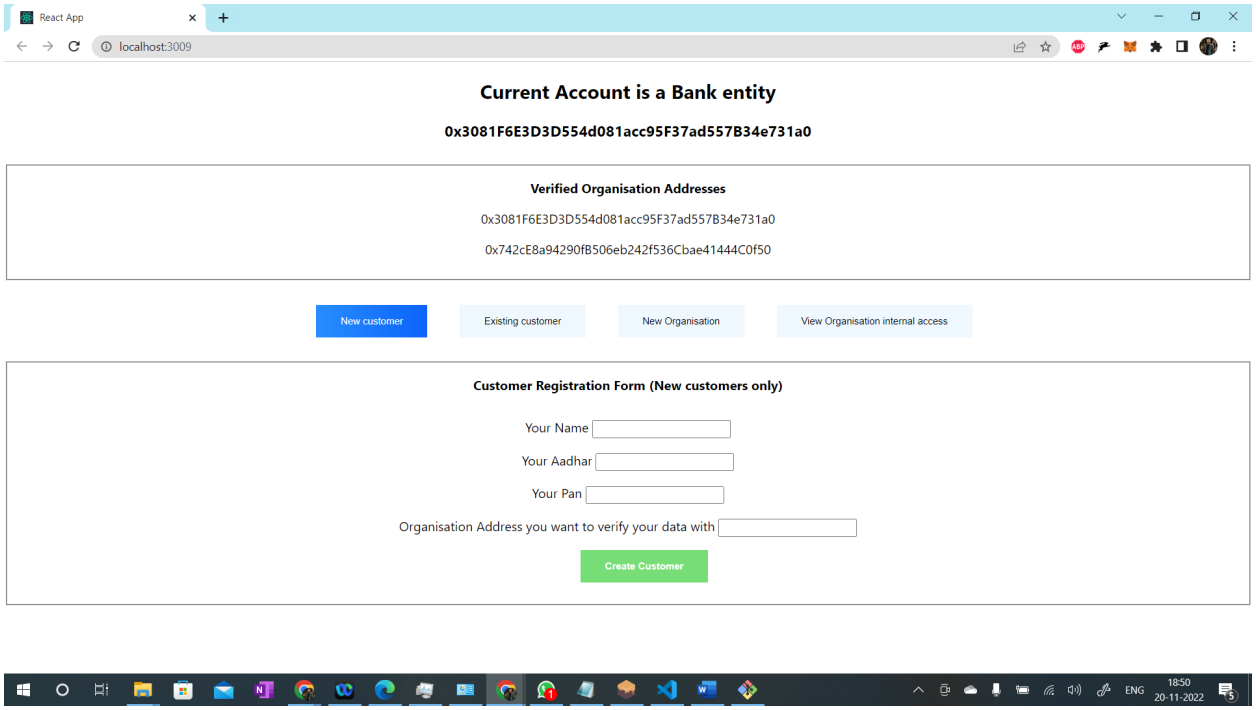
```
const KycBlockChain = artifacts.require("./KycBlockChain.sol");
```

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

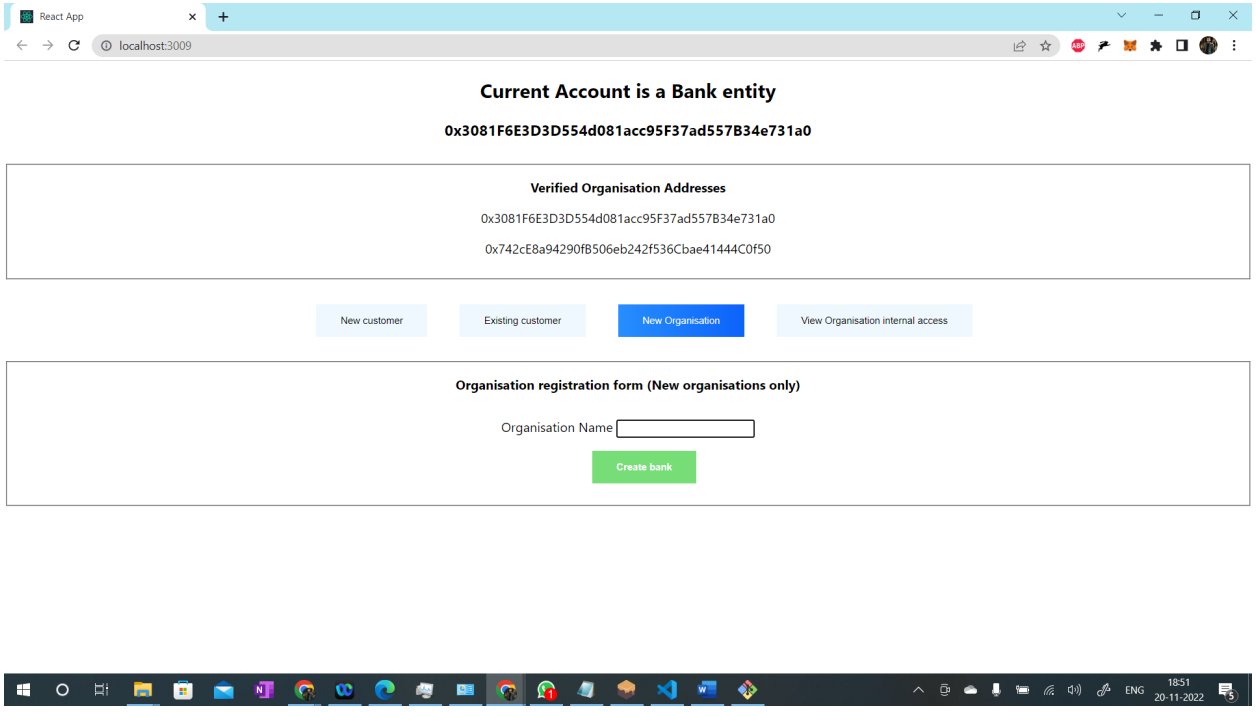
Steps for output:
Move to client folder and
npm start



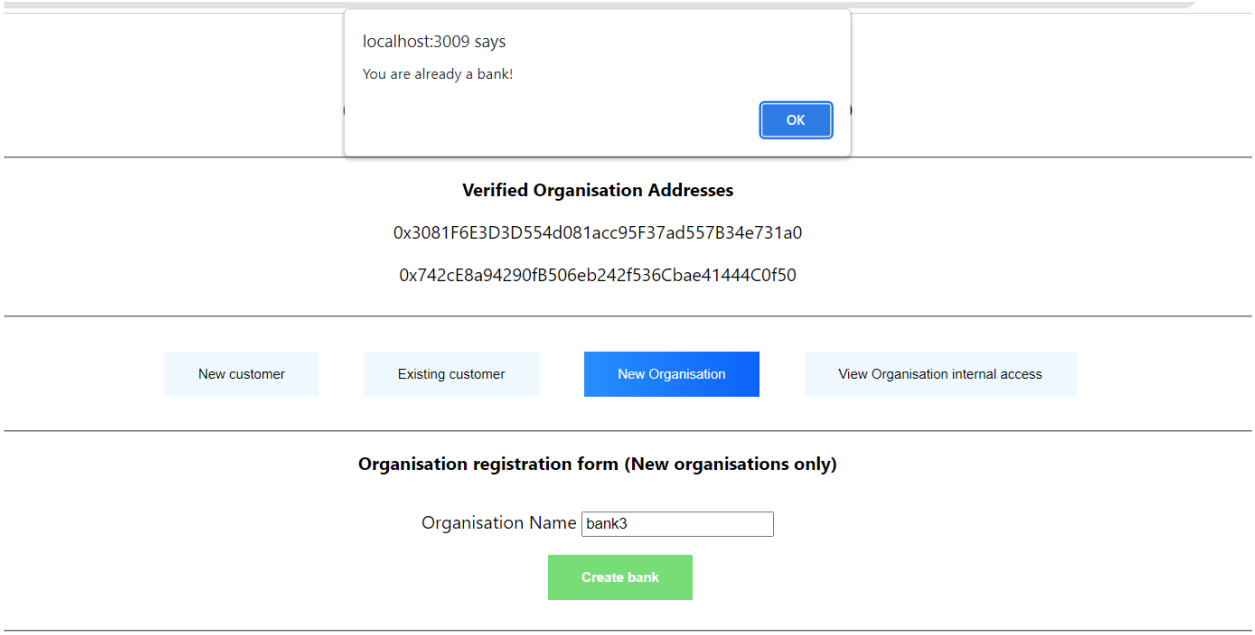
After connecting ganache accounts to metamask



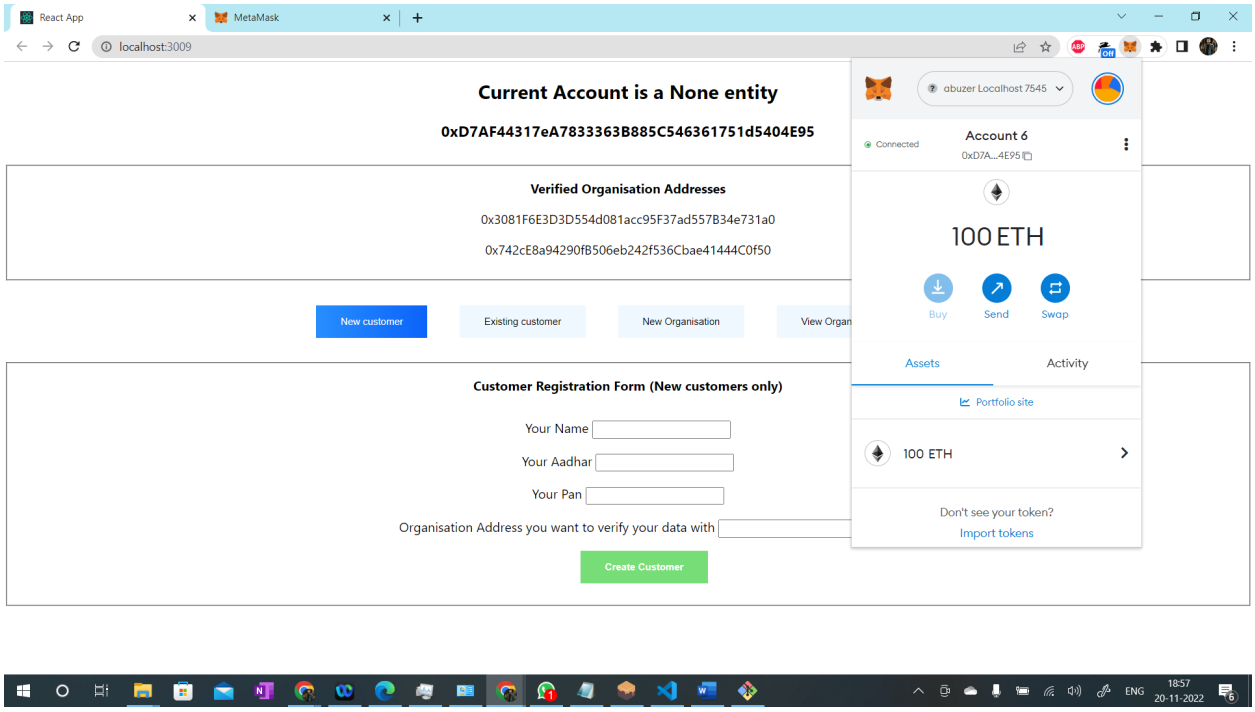
Add new bank account
Goto new organization and give it a name



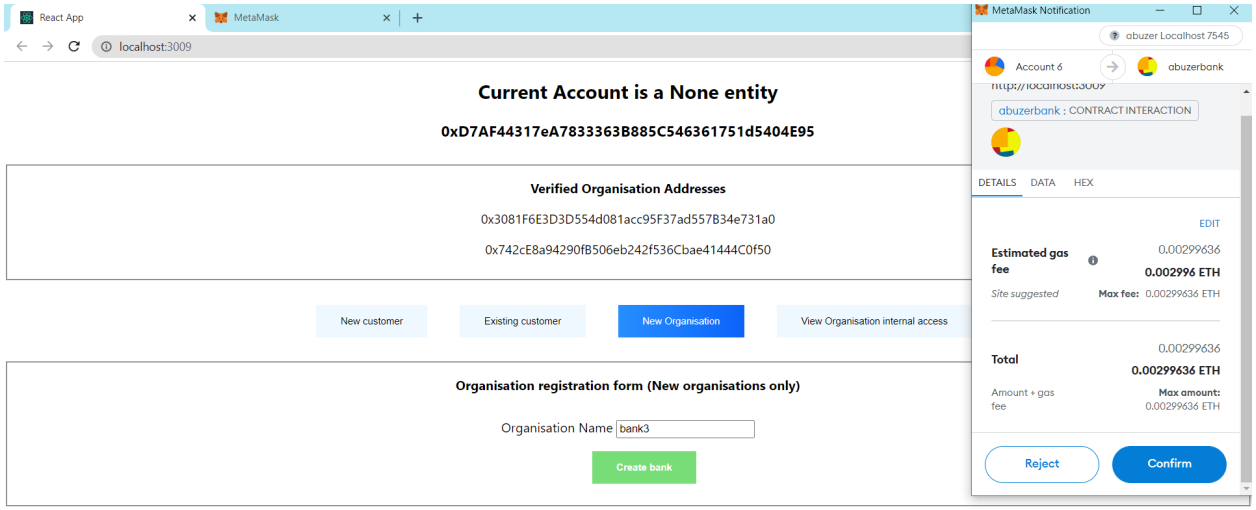
If we create a new bank on the same metamask account we get error, so we need a new metamask account



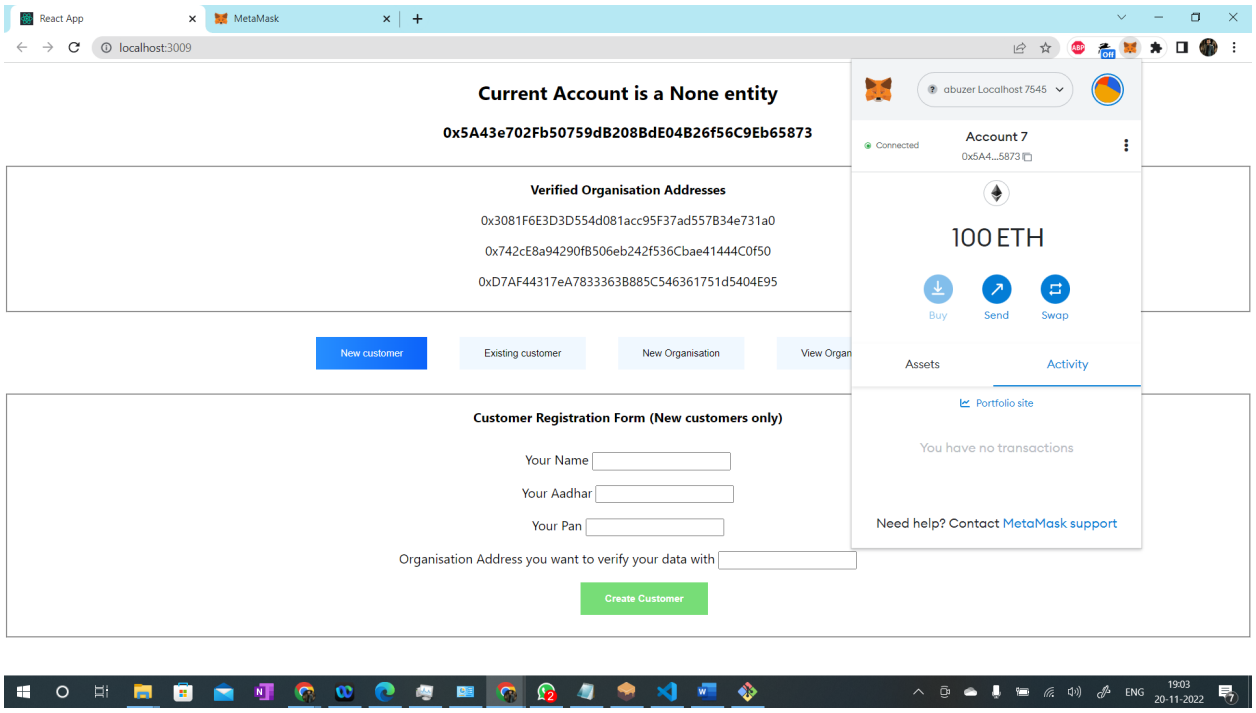
After connecting new account it says None entity, it means that we can create a new bank account and similar is the case for a new user account



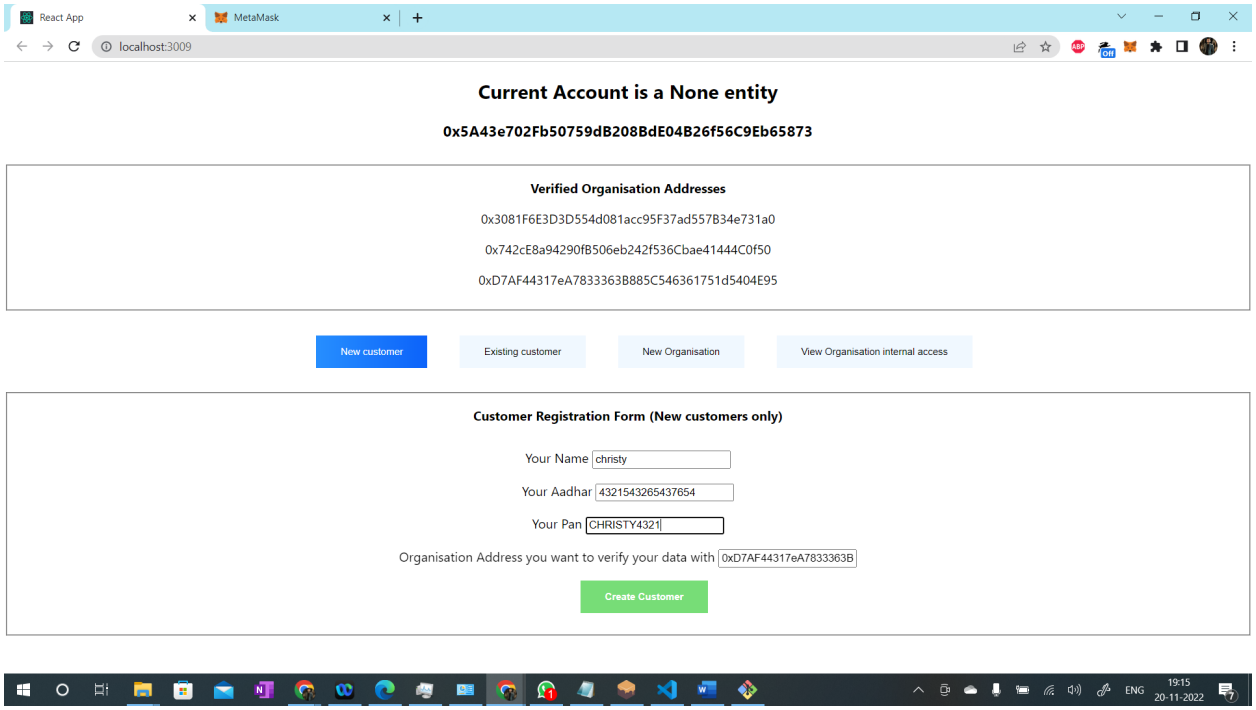
Transaction is done and new bank is created



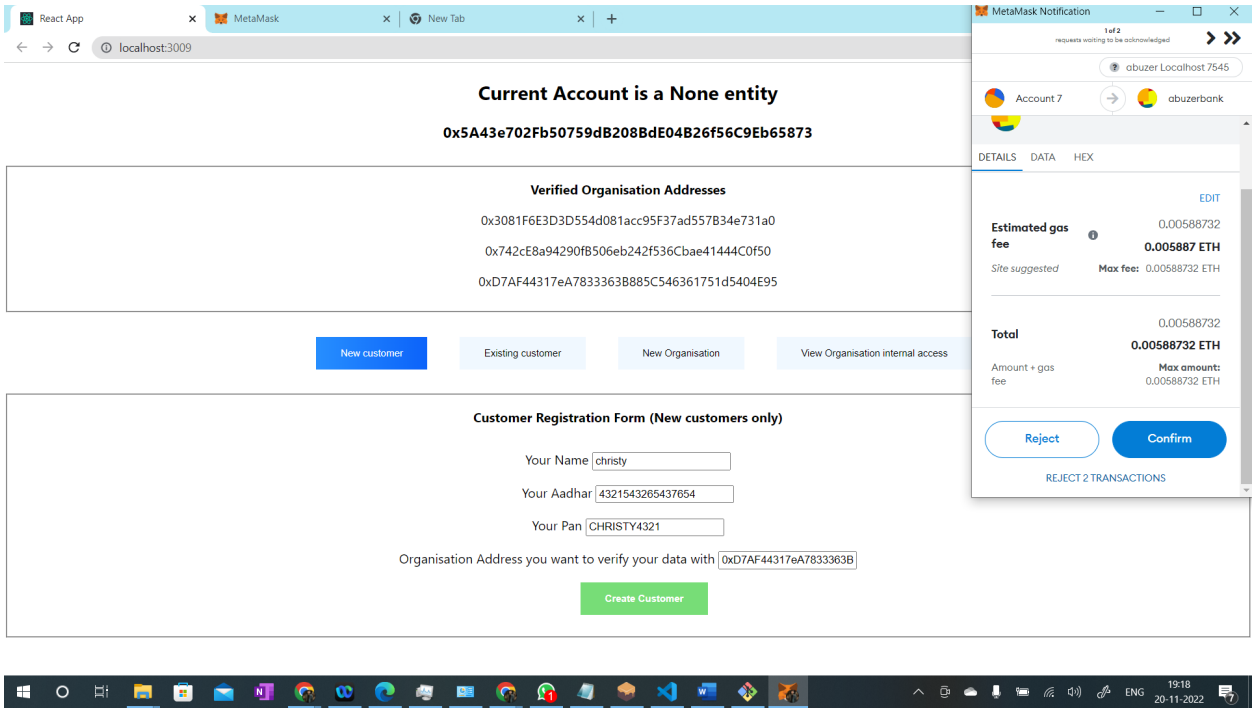
Adding new user account



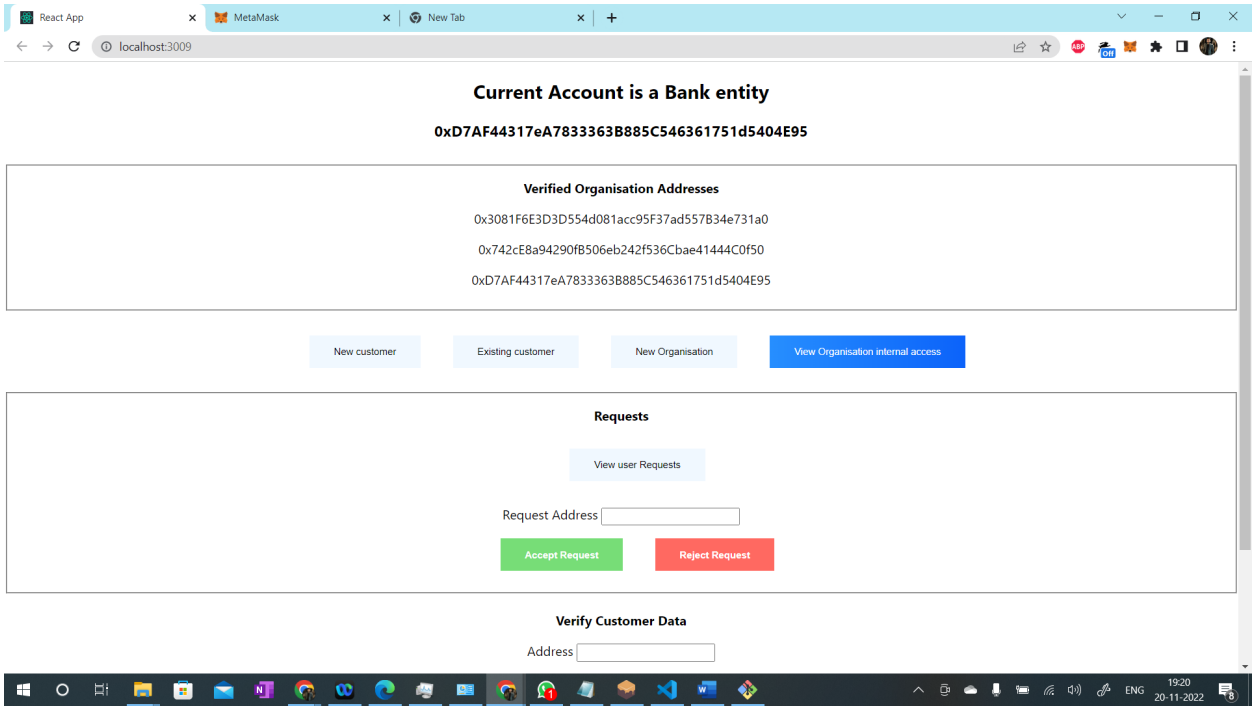
Enter all the details along with the address of the bank you want to verify details with
In this example we have address of bank3 that is 0xD7AF44317eA7833363B885C546361751d5404E95



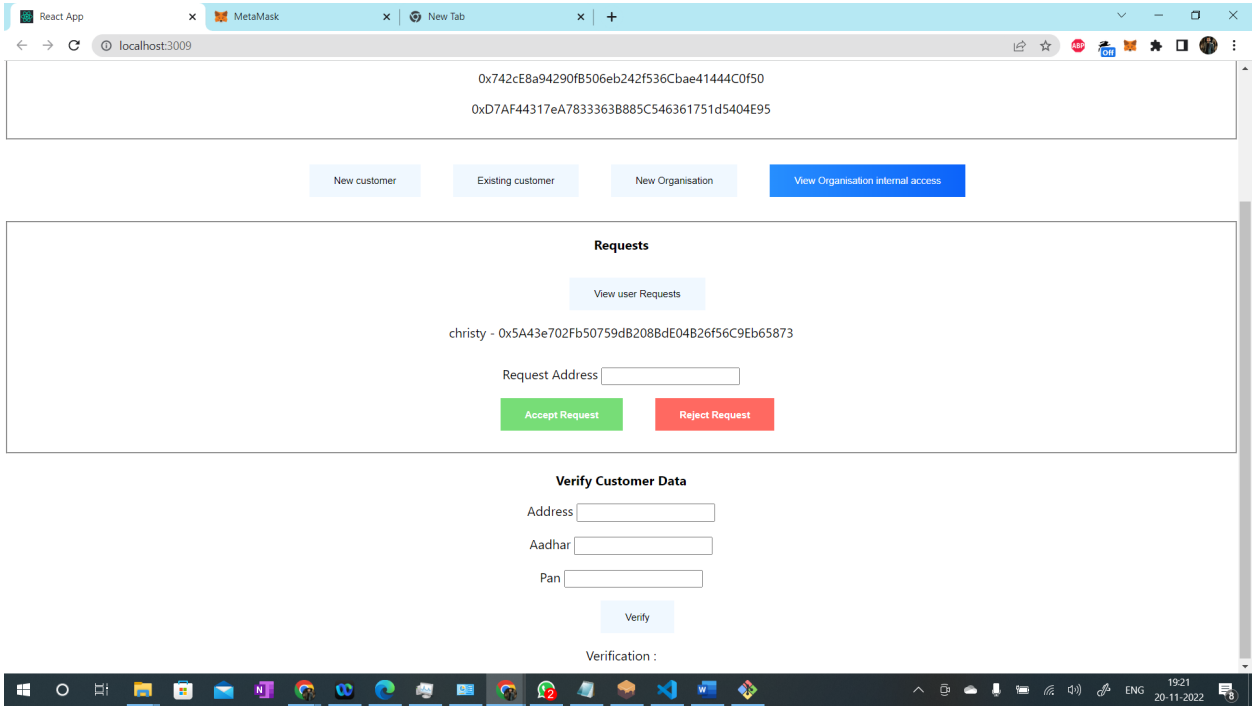
Transaction is done and customer account is added



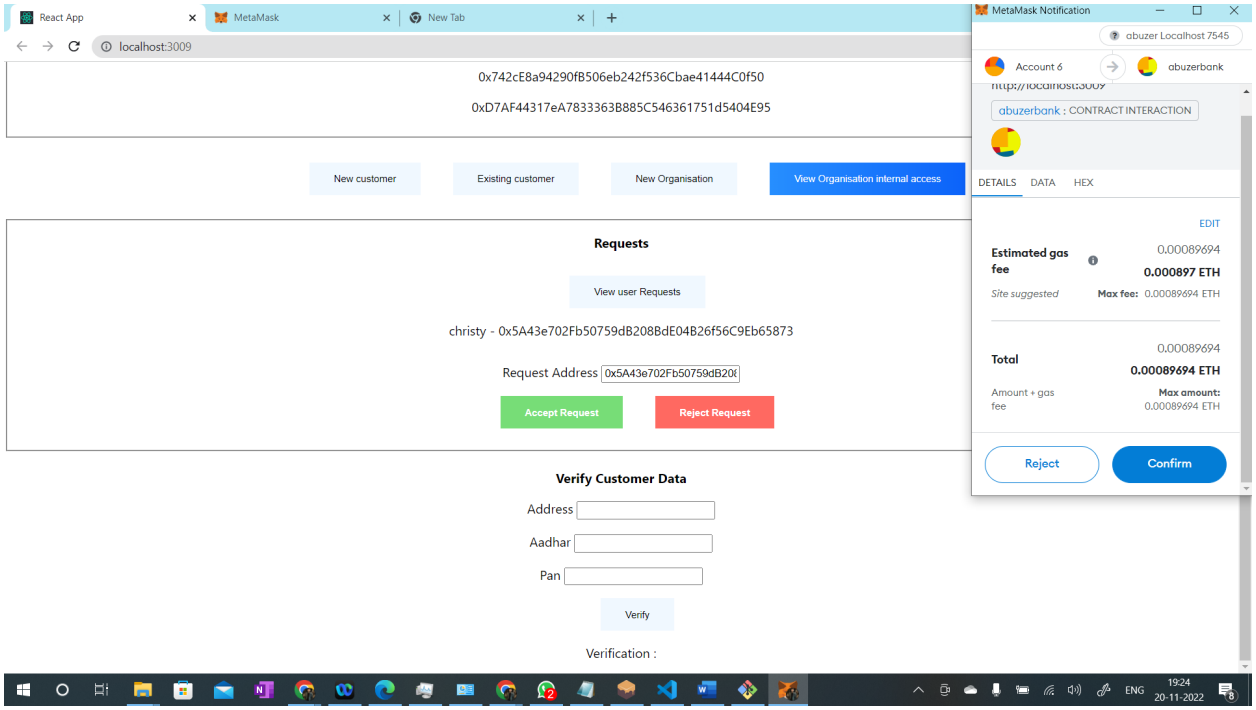
We head on to bank3 account -> view organization internal access to verify the user

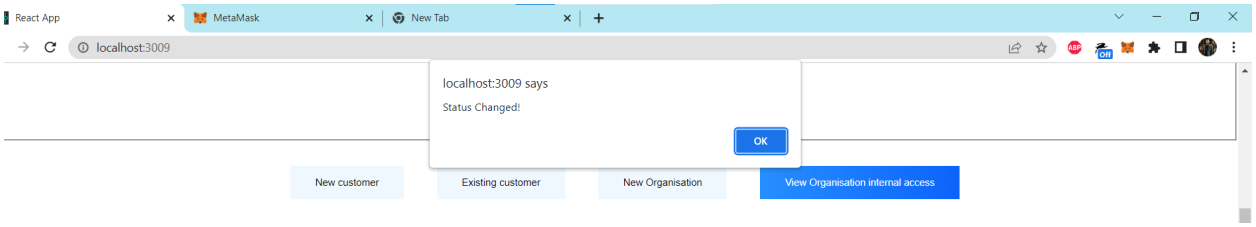


Click on View User Requests to see which user is pending approval Enter the address and accept to add user to blockchain database



Confirm transaction

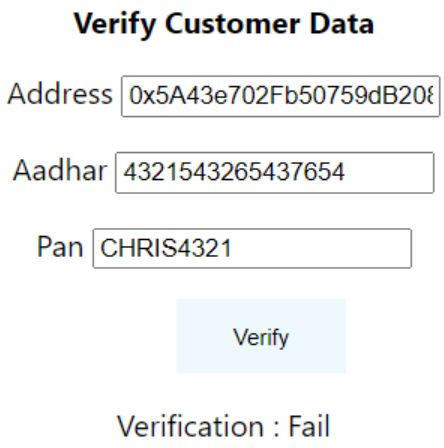




In the same window we can verify if the details is correct, thereby verifying the originally entered data

---

Wrong information will result in failure of verification



Conclusion

Any industry's future lies in complete digital transformation, which can only be achieved through infrastructure changes. To improve operational efficiency, core processes must be modified. This can only be accomplished by being open to new and disruptive technologies. The main goal of our proposed solution was to reimagine the traditional KYC process. This proposed paper gives a solution to the problem of redundancy and inefficiency in the current KYC process, drastically lowering the system\'s operational costs. We also eliminate the presence of a single point of failure by utilizing a blockchain-based approach. Blockchain is a game-changing technology, and its applications are expanding all the time. Implementing a blockchain application for kyc document verification provides proof of identity of a customer on bank and transparent access to all or any of the banks that are connected into the blockchain network, ensuring faster access to the kyc document while also providing security. By doing so, we can lower the cost of maintaining the document from the centralized organization.