

Roll No. 42

Exam Seat No. _____

VIVEKANANDEDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

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



Since 1962

CERTIFICATE

Certified that Mr./Miss Ronak Bipin Pathare

of SYMCA 2nd Shift

has satisfactorily completed a course of the necessary experiments in

Blockchain Lab under my supervision

in the Institute of Technology in the academic year 2020 - 2022.

Principal

Head of Department

Faculty In-charge

External Examiner



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

INDEX

Sr. No	Contents	Page Number	Marks	Sign
1	Implementation of Caesar Cipher (Symmetric Encryption)	2		
2	Implementation of RSA Algorithm (Asymmetric Encryption)	5		
3	Implementation of SHA-256	8		
4	Implementation of Binary Tree	11		
5	Implement the creation of a Genesis Block & Blockchain (Adding the blocks to the chain and validating)	18		
6	Implement the creation of a public/private Blockchain	28		
7	Simple Experiments using Solidity Program Constructs (if-then, while etc...)	45		
8	Creation of smart contract in Ethereum	53		
9	Creation of Dapp in Ethereum	62		
10	Mini project	65		

Practical No – 1

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

Theory:

What is Cryptography?

Cryptography is the study of securing communications from outside observers. Encryption algorithms take the original message, or plaintext, and convert it into cipher text, which is not understandable. The key allows the user to decrypt the message, thus ensuring they can read the message. The strength of the randomness of an encryption is also studied, which makes it harder for anyone to guess the key or input of the algorithm.

Cryptography is how we can achieve more secure and robust connections to elevate our privacy. Advancements in cryptography makes it harder to break encryptions so that encrypted files, folders, or network connections are only accessible to authorized users.

Cryptography focuses on four different objectives:

1. **Confidentiality:** Confidentiality ensures that only the intended recipient can decrypt the message and read its contents.
2. **Non-repudiation:** Non-repudiation means the sender of the message cannot backtrack in the future and deny their reasons for sending or creating the message.
3. **Integrity:** Integrity focuses on the ability to be certain that the information contained within the message cannot be modified while in storage or transit.
4. **Authenticity:** Authenticity ensures the sender and recipient can verify each other's identities and the destination of the message.

What is Symmetric Encryption?

Symmetric Key Cryptography also known as Symmetric Encryption is when a secret key is leveraged for both encryption and decryption functions. This method is the opposite of Asymmetric Encryption where one key is used to encrypt and another is used to decrypt. During this process, data is converted to a format that cannot be read or inspected by anyone who does not have the secret key that was used to encrypt it. The success of this approach depends on the strength of the random number generator that is used to create the secret key.

Symmetric Key Cryptography is widely used in today's Internet and primarily consists of two types of algorithms, Block and Stream. Some common encryption algorithms include the Advanced Encryption Standard (AES) and the Data Encryption Standard (DES). This form of encryption is traditionally much faster than Asymmetric however it requires both the sender and the recipient of the data to have the secret key.

Explanation about Caesar cipher The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places 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 named after Julius Caesar, who apparently used it to communicate with his generals. More complex encryption schemes such as the Vigenère cipher employ the Caesar cipher as one element of the encryption process. The widely known ROT13 'encryption' is simply a Caesar cipher with an offset of 13. The Caesar cipher offers essentially no communication security, and it will be shown that it can be easily broken even by hand.

Example: To pass an encrypted message from one person to another, it is first necessary that both parties have the 'key' for the cipher, so that the sender may encrypt it and the receiver may decrypt it. For the Caesar cipher, the key is the number of characters to shift the cipher alphabet. Here is a quick example of the encryption and decryption steps involved with the Caesar cipher. The text we will encrypt is 'defend the east wall of the castle', with a shift (key) of 1.

Plain-text: defend the east wall of the castle

Cipher-text: efgfoe uif fbtu xbm m pg uif dbtumf

It is easy to see how each character in the plaintext is shifted up the alphabet. Decryption is just as easy, by using an offset of -1.

Plain-text: abcdefghijklmnopqrstuvwxyz

Cipher-text: bcdefghijklmnopqrstuvwxyz a

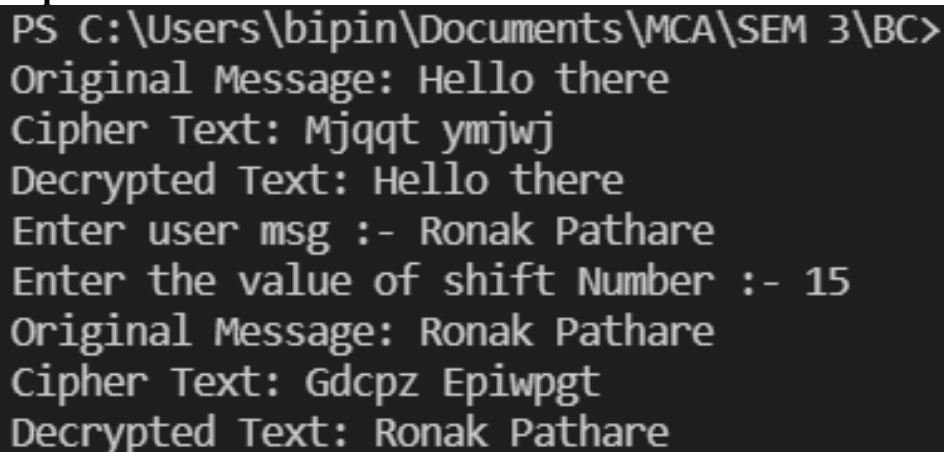
Obviously, if a different key is used, the cipher alphabet will be shifted a different amount.

Code:

```
def cryptography(msg, shift):
    result = ""
    for i in range(len(msg)):
        char = msg[i]
        if (ord(char) == 32):
            result += " "
        elif (char.isupper()):
            result += chr((ord(char) + shift - 65) % 26 + 65)
        else:
            result += chr((ord(char) + shift - 97) % 26 + 97)
    return result

msg = "May the Force be with you"
shift = 5
encryptedMsg = cryptography(msg, shift)
decryptedMsg = cryptography(encryptedMsg, -shift)
print("Original Message: " + msg)
print("Cipher Text: " + encryptedMsg)
print("Decrypted Text: " + decryptedMsg)

msg2 = input("Enter user msg")
shift2 = int(input("Enter the value of shift Number"))
encryptedMsg = cryptography(msg2, shift2)
decryptedMsg = cryptography(encryptedMsg, -shift2)
print("Original Message: " + msg2)
print("Cipher Text: " + encryptedMsg)
print("Decrypted Text: " + decryptedMsg)
```

Ouput:

```
PS C:\Users\bipin\Documents\MCA\SEM 3\BC>
Original Message: Hello there
Cipher Text: Mjqqt ymjwj
Decrypted Text: Hello there
Enter user msg :- Ronak Pathare
Enter the value of shift Number :- 15
Original Message: Ronak Pathare
Cipher Text: Gdcpz Epiwpgt
Decrypted Text: Ronak Pathare
```

Conclusion:

Hence, we have successfully implemented Caesar Cipher (Symmetric Encryption) and we have shown the encryption as well as the decryption process.

Practical No – 2

Aim: Implement RSA Algorithm (Asymmetric Encryption), Encrypt and decrypt a string.

Theory:

What is Asymmetric Encryption?

Asymmetric encryption is a type of encryption that uses two separate yet mathematically related keys to encrypt and decrypt data. The public key encrypts data while its corresponding private key decrypts it. This is why it's also known as public key encryption, public key cryptography, and asymmetric key encryption.

The public key is open to everyone. Anyone can access it and encrypt data with it. However, once encrypted, the data can only be unlocked by using the corresponding private key. As you can imagine, the private key must be kept secret to keep it from becoming compromised. So, only the authorized person, server, machine, or instrument has access to the private key.

Explanation about RSA (steps and example) RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in the year 1978 and hence name RSA algorithm.

Algorithm

The RSA algorithm holds the following features:

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

You will have to go through the following steps to work on RSA 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:

$$ed = 1 \bmod (p-1) (q-1)$$

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:

$$C = P^e \bmod n$$

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:

$$\text{Plaintext} = C^d \bmod n$$

Code:

```
import rsa
public_key, private_key = rsa.newkeys(512)

def encrypt_text(plain_text):
    plain_text = plain_text.encode('utf8')
    encrypted_text = rsa.encrypt(plain_text, public_key)
    return encrypted_text

def decrypt_text(encrypted_text):
    decrypted_text = rsa.decrypt(encrypted_text, private_key)
    return decrypted_text.decode("utf-8")

plain_text = "Luminous beings we are, not this crude matter"
encrypted_text = encrypt_text(plain_text)
print("Encrypted text is = %s" %(encrypted_text))
decrypted_text = decrypt_text(encrypted_text)
print("Decrypted text is = %s" %(decrypted_text))
```

Ouput:

```
PS C:\Users\bipin\Documents\MCA\SEM 3\BC> & C:/Python310/python.exe "C:/Users/bipin/Documents/MCA/SEM 3/BC/Practical 2.py"
Encrypted text is = b"\x1flg\xc3\xd3w\xa7$\xa0\xeb\x93wx\xc6Z2\x98\x03\xb7t\x89\xc6\xee&\x82,\x03j0\x0f\|^ \xc4\x86_\xd9\xd8;'
a8?lvD\xec\x9c\xeex|\xa7~k\x88\xaa\x1d"
Decrypted text is = Luminous beings we are, not this crude matter
```

Conclusion:

Hence, we have successfully implemented RSA Algorithm (Asymmetric Encryption) and we have encrypted and decrypted a string.

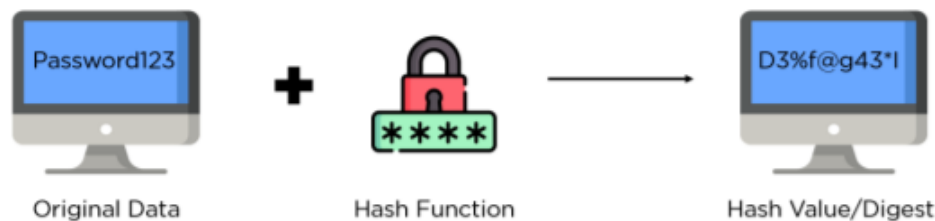
Practical No – 3

Aim: To implement SHA 256 Algorithm.

Theory:

Hashing:

Hashing is the process of scrambling raw information to the extent that it cannot reproduce it back to its original form. It takes a piece of information and passes it through a function that performs mathematical operations on the plaintext. This function is called the hash function, and the output is called the hash value/digest.



As seen from the above image, the hash function is responsible for converting the plaintext to its respective hash digest. They are designed to be irreversible, which means your digest should not provide you with the original plaintext by any means necessary. Hash functions also provide the same output value if the input remains unchanged, irrespective of the number of iterations.

There are two primary applications of hashing:

- **Password Hashes:** In most website servers, it converts user passwords into a hash value before being stored on the server. It compares the hash value recalculated during login to the one stored in the database for validation.
- **Integrity Verification:** When it uploads a file to a website, it also shared its hash as a bundle. When a user downloads it, it can recalculate the hash and compare it to establish data integrity.

It is one of the algorithms which calculates a string value from a file, which is of a fixed size. Basically, it contains blocks of data, which is transformed into a short fixed-length key or value from the original string. Additionally, the term “hash” is used for describing both the hash function as well as the hash value.

Hash functions were created to compress data to reduce the amount of memory required for storing large files. The hashes they create can be stored in a special data structure called hash tables, which enables quicker data lookups. The core reason for hash functions arose from the need to compress content, but the unique identifiers of hash values soon became a staple of simplicity in database management. No two hash inputs should ever return the same hash, but instead create singularly unique identifiers for each hash input. When two different hash inputs return the same output hash, it is called a collision. A more extensive family of hash functions were created with privacy, security and transparency in mind.

SHA256:

SHA 256 is a part of the SHA 2 family of algorithms, where SHA stands for Secure Hash Algorithm. Published in 2001, it was a joint effort between the NSA and NIST to introduce a successor to the SHA 1 family, which was slowly losing strength against brute force attacks. The significance of the 256 in the name stands for the final hash digest value, i.e., irrespective of the size of plaintext/cleartext, the hash value will always be 256 bits.

The other algorithms in the SHA family are more or less similar to SHA 256. Now, look into knowing a little more about their guidelines. The Secure Hash Algorithm is one of a number of cryptographic hash functions. A cryptographic hash is like a signature for a data set. If you would like to compare two sets of raw data it is always better to hash it and compare SHA256 values. It is like the fingerprints of the data. Even if only one symbol is changed the algorithm will produce a different hash value.

SHA256 algorithm generates an almost-unique, fixed size 256-bit (32-byte) hash. Hash is so called a one-way function. This makes it suitable for checking integrity of your data, challenge hash authentication, anti-tamper, digital signatures, blockchain. With the newest hardware improvements, it has become Name: VINOD NAYAK Blockchain Lab Roll No.: 41 3 possible to decrypt SHA256 algorithm back. So, it is no longer recommended to use it for password protection or other similar use cases. Some years ago, you would protect your passwords from hackers by storing SHA256 encrypted passwords in your database. This is no longer a case. SHA256 algorithm can be still used for making sure you acquired the same data as the original one.

For example, if you download something you can easily check if data has not changed due to network errors or malware injection. You can compare hashes of your file and original one which is usually provided in the website you are getting data or the file from. SHA-256 is one of the successor hash functions to SHA-1, and is one of the strongest hash functions available.

Hashlib module in python:

The hashlib module of Python is used to implement a common interface to many different secure hash and message digest algorithms. The hash algorithms included in this module are:

- **SHA1:** a 160-bit hash function that resembles MD5 hash • **SHA224:** internal block size of 32 bits (truncated version).
- **SHA256:** internal block size of 32 bits.
- **SHA384:** internal block size of 32 bits (truncated version).
- **SHA512:** internal block size of 64 bits.
- **MD5** algorithm.

Code:

```
import hashlib
string="Choices are what truly ever matter"
encoded=string.encode()
result = hashlib.sha256(encoded)
print("String : ", end = "")
print(string)
print("Hash Value : ", end = "")
print(result)
print("Hexadecimal equivalent: ",result.hexdigest())
print("Digest Size : ", end = "")
print(result.digest_size)
print("Block Size : ", end = "")
print(result.block_size)
```

Ouput:

```
PS C:\Users\bipin\Documents\MCA\SEM 3\BC> & C:/Python310/python.exe "c:/Users/bipin/Documents/MCA/SEM 3/BC/Practical 3.py"
String : Choices are what truly ever matter
Hash Value : <sha256_hashlib.HASH object @ 0x000001CDF5AF3CB0>
Hexadecimal equivalent: cb0ee3ab138d8fac6a48cdd5a2be09e6075a085b119fe4e48eafbdb7ca02c450
Digest Size : 32
Block Size : 64
```

Conclusion:

Hence, we have successfully implemented RSA algorithm (asymmetric encryption), encrypt and decrypt a string

Practical No – 4

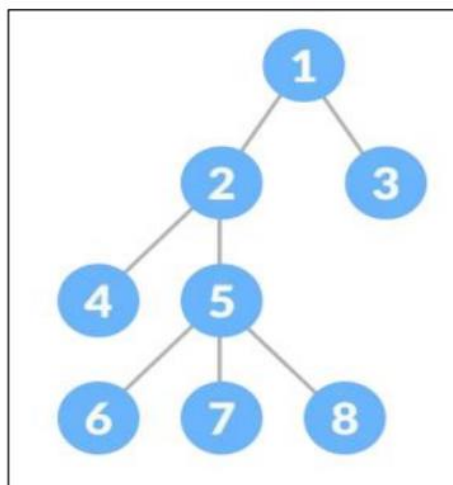
Aim: To Implementation of Binary Tree and to show all operations (Insert, Delete, Traversals, Display).

Theory:

Tree:

A tree is a nonlinear hierarchical data structure that consists of nodes connected by edges. It connects each node in the tree data structure using "edges", both directed and undirected. It can also be defined as a collection of objects or entities known as nodes that are linked together to represent or simulate hierarchy. It is a non-linear data structure because it does not store in a sequential manner. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.

In the Tree data structure, the topmost node is known as a root node. Each node contains some data, and data can be of any type. In the above tree structure, the node contains the name of the employee, so the type of data would be a string. Each node contains some data and the link or reference of other nodes that can be called children.



Tree Terminologies:

- **Root:** In a tree data structure, the root is the first node of the tree. The root node is the initial node of the tree in data structures. In the tree data structure, there must be only one root node.
- **Edge:** In a tree in data structures, the connecting link of any two nodes is called the edge of the tree data structure. In the tree data structure, N number of nodes connecting with N -1 number of edges.
- **Parent:** In the tree in data structures, the node that is the predecessor of any node is known as a parent node, or a node with a branch from itself to any other successive node is called the parent node.

- **Child:** The node, a descendant of any node, is known as child nodes in data structures. In a tree, any number of parent nodes can have any number of child nodes. In a tree, every node except the root node is a child node.
- **Siblings:** In trees in the data structure, nodes that belong to the same parent are called siblings.
- **Leaf:** Trees in the data structure, the node with no child, is known as a leaf node. In trees, leaf nodes are also called external nodes or terminal nodes.
- **Internal nodes:** Trees in the data structure have at least one child node known as internal nodes. In trees, nodes other than leaf nodes are internal nodes. Sometimes root nodes are also called internal nodes if the tree has more than one node.
- **Degree:** In the tree data structure, the total number of children of a node is called the degree of the node. The highest degree of the node among all the nodes in a tree is called the Degree of tree.
- **Level:** In tree data structures, the root node is said to be at level 0, and the root node's children are at level 1, and the children of that node at level 1 will be level 2, and so on.
- **Height:** In a tree data structure, the number of edges from the leaf node to the particular node in the longest path is known as the height of that node. In the tree, the height of the root node is called "Height of Tree". The tree height of all leaf nodes is 0.
- **Depth:** In a tree, many edges from the root node to the particular node are called the depth of the tree. In the tree, the total number of edges from the root node to the leaf node in the longest path is known as "Depth of Tree". In the tree data structures, the depth of the root node is 0.
- **Path:** In the tree in data structures, the sequence of nodes and edges from one node to another node is called the path between those two nodes. The length of a path is the total number of nodes in a path.
- **Subtree:** In the tree in data structures, each child from a node shapes a sub-tree recursively and every child in the tree will form a sub-tree on its parent node.

Binary Tree:

A binary tree is a tree data structure in which each parent node can have at most two children. Each node has a key and an associated value. The value of the key of the left sub-tree is less than the value of its parent (root) node's key. The value of the key of the right sub-tree is greater than or equal to the value of its parent (root) node's key.

A parent node has two child nodes: the left child and right child. Hashing, routing data for network traffic, data compression, preparing binary heaps, and binary search trees are some of the applications that use a binary tree.

Basic Operations:

Following are the basic operations of a tree –

- **Create Root:** we just create a Node class and assign a value to the node. This becomes a tree with only a root node.
- **Search:** Searches an element in a tree.
- **Insert:** (Inserts an element in a tree.) To insert into a tree, we use the same node class created above and add an insert class to it. The insert class compares the value of the node to the parent node and decides to add it as a left node or a right node. Finally, the PrintTree class is used to print the tree.
- **Traversing a Tree:** The tree can be traversed by deciding on a sequence to visit each node. As we can clearly see we can start at a node then visit the left sub-tree first and right sub-tree next. We can also visit the right sub-tree first and left sub-tree next. Accordingly, there are different names for these tree traversal methods. Traversal is a process to visit all the nodes of a tree and may print their values too. Because all nodes are connected via edges, we always start from the root node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree:
 1. **Pre-order Traversal:** In this traversal method, the root node is visited first, then the left sub-tree and finally the right sub-tree. We use the Node class to create placeholders for the root node as well as the left and right nodes. Then, we create an insert function to add data to the tree. Finally, the Pre-order traversal logic is implemented by creating an empty list and adding the root node first followed by the left node. At last, the right node is added to complete the Pre-order traversal.
 2. **In-order Traversal:** In this traversal method, the left sub-tree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself. We use the Node class to create placeholders for the root node as well as the left and right nodes. Then, we create an insert function to add data to the tree. Finally, the In-order traversal logic is implemented by creating an empty list and adding the left node first followed by the root or parent node. At last, the left node is added to complete the In-order traversal.
 3. **Post-order Traversal:** In this traversal method, the root node is visited last, hence the name. First, we traverse the left sub-tree, then the right sub-tree and finally the root node. We use the Node class to create place holders for the root node as well as the left and right nodes. Then, we create an insert function to add data to the tree. Finally, the Post-order traversal logic is implemented by creating an empty list and adding the left node first followed by the right node. At last, the root or parent node is added to complete the post-order traversal.

- **Delete** – Given a binary tree, delete a node from it by making sure that tree shrinks from the bottom. This is different from BST deletion. Here we do not have any order among elements, so we replace it with the last element.

Algorithm:

1. Starting at the root, find the deepest and rightmost node in the binary tree and node which we want to delete.
2. Replace the deepest rightmost node's data with the node to be deleted.
3. Then delete the deepest rightmost node.

Code:

```
class Node:
    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data
# Insert Node
    def insert(self, data):
        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()
# Preorder traversal
# Root -> Left ->Right
    def PreorderTraversal(self, root):
        res = []
        if root:
            res.append(root.data)
            res = res + self.PreorderTraversal(root.left)
```

```
        res = res + self.PreorderTraversal(root.right)
    return res
# Postorder traversal
# Left ->Right -> Root
def PostorderTraversal(self, root):
    res = []
    if root:
        res = self.PostorderTraversal(root.left)
        res = res + self.PostorderTraversal(root.right)
        res.append(root.data)
    return res
# Inorder traversal of a binary tree
def inorder(temp):
    if(not temp):
        return
    inorder(temp.left)
    print(temp.data, end = " ")
    inorder(temp.right)
# function to delete the given deepest node (d_node) in binary tree
def deleteDeepest(root,d_node):
    q = []
    q.append(root)
    while(len(q)):
        temp = q.pop(0)
        if temp is d_node:
            temp = None
            return
        if temp.right:
            if temp.right is d_node:
                temp.right = None
                return
            else:
                q.append(temp.right)
        if temp.left:
            if temp.left is d_node:
                temp.left = None
                return
            else:
                q.append(temp.left)
# function to delete element in binary tree
def deletion(root, key):
    if root == None :
        return None
    if root.left == None and root.right == None:
        if root.key == key :
            return None
        else :
            return root
```



```
key_node = None
q = []
q.append(root)
temp = None
while(len(q)):
    temp = q.pop(0)
    if temp.data == key:
        key_node = temp
    if temp.left:
        q.append(temp.left)
    if temp.right:
        q.append(temp.right)
    if key_node :
        x = temp.data
        deleteDeepest(root,temp)
        key_node.data = x
return root
print("Binary Tree:")
root = Node(18)
root.insert(97)
root.insert(21)
root.insert(15)
root.insert(9)
root.insert(1)
root.insert(100)
root.PrintTree()
print("Inorder Traversal before the deletion:")
inorder(root)
key = 14
root = deletion(root, key)
print()
print("Inorder Traversal after the deletion:")
inorder(root)
print("\nPreorder Traversal :",root.PreorderTraversal(root))
print("Postorder Traversal :",root.PostorderTraversal(root))
```

Output:

```
PS C:\Users\bipin\Documents\MCA\SEM 3\BC> & C:/Python310/python.exe
```

```
Binary Tree:
```

```
4
14
36
55
68
74
96
97
```

```
Inorder Traversal : [4, 14, 36, 55, 68, 74, 96, 97]
```

```
Preorder Traversal : [74, 55, 4, 14, 36, 68, 97, 96]
```

```
Postorder Traversal : [36, 14, 4, 68, 55, 96, 97, 74]
```

Conclusion:

Hence, we have successfully implemented Binary Tree and showed all operations (Insert, Delete, Traversals, Display).

Practical No – 5

Aim: 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.

Theory:

What is a node?

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

Geth:

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

Genesis Block:

A genesis block refers to the first block in a blockchain and is usually hardcoded into its application’s software. A blockchain has multiple “blocks” (containing validated transactions and recorded activity data) linked together by a metaphorical chain.

Each “block” of a cryptoasset contains referential data for the previous one and derives its value/legitimacy from its predecessor. The genesis block, thus, refers to the first block (Block 0 or Block 1) of a new blockchain, to which all other subsequent blocks are attached.

A genesis block is unique as it is the only block in a blockchain that does not reference a predecessor block, and in almost all cases, the first mining rewards it unlocks are unspendable.

Genesis blocks have special significance, as they form the very foundation of a blockchain and often contain interesting stories or hidden meanings. For instance, Bitcoin’s genesis block contains the now-famous message "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks" — a reference to the deteriorating financial conditions of that time and the rationale for creating cryptocurrencies like Bitcoin and Ethereum. Bitcoin’s genesis block in 2009 contained 50 BTC.

The Bitcoin genesis block is very intriguing not just for its included message, but also due to the fact that the next block was timestamped nearly six days later (the average time is 10 minutes). The present hypothesis is that the catchy Times headline enticed Satoshi Nakamoto to release Bitcoin in public, but that he actually created the genesis block earlier and altered the timestamp accordingly. After testing his software from Jan. 3, 2009, Satoshi possibly deleted all the test blocks and used the genesis block for his mainnet launch.

Commands used in this practical:

- **geth --datadir chaindata init genesis.json:** Initializes geth into chaindata.
- **geth --datadir=./chaindata/:** Used to run geth.
- **geth attach ipc:\\.\pipe\geth.ipc:** IPC to interact with geth.
- **personal.newAccount():** Create a new account.
- **eth.accounts:** Get information about all the accounts present.
- **eth.coinbase:** Get information about the coinbase account.
- **eth.getBalance(eth.accounts[0]):** Get the current balance of an account.
- **miner.start() / miner.stop():** Start/Stop the mining process.
- **eth.blockNumber:** Get the blockNumber.
- **personal.unlockAccount(eth.accounts[0]):** Unlock the coinbase account for transaction.
- **eth.sendTransaction({from: eth.coinbase, to: eth.accounts[1], value: web3.toWei(10, "ether")}):** Command for transaction. Enter the account number to which we want to transfer the ethers to.
- **eth.getTransaction(txHash):** get the information about the transaction. Enter the hash instead of “txHash”.
- **web3.fromWei(eth.getBalance(eth.accounts[1]), "ether"):** get balance of the second account in ethers.
- **eth.getBlock("latest"):** Get information about the latest block.
- **eth.getBlock(388):** Get information about a specific block.

Steps:

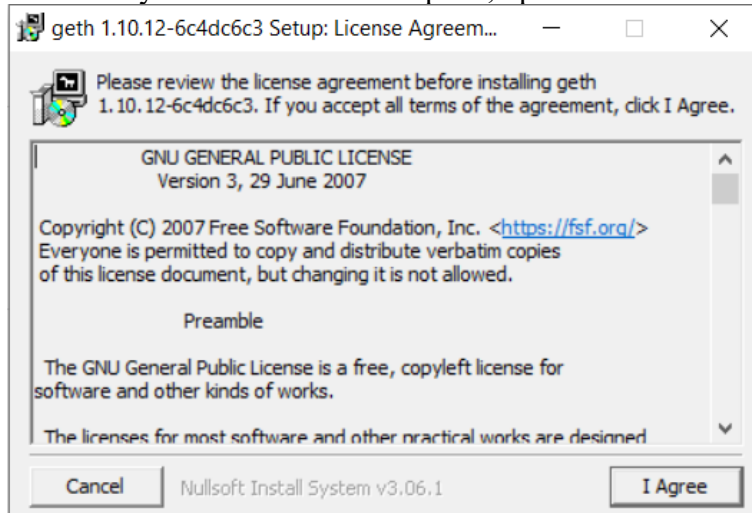
- Steps to install Go-Ethereum(Geth):
- Visit the Go Ethereum website and install Geth.
- Visit here: <https://geth.ethereum.org/downloads/>
- Download the latest release of Geth for Windows, make sure you download the 64-bit version.

Stable releases

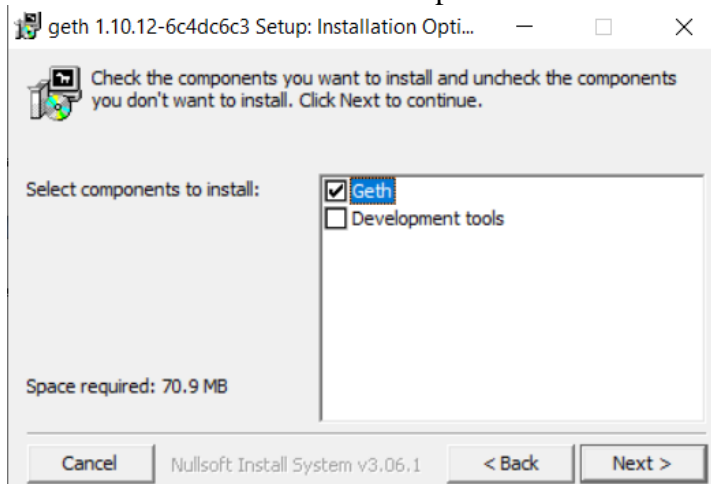
These are the current and previous stable releases of go-ethereum, updated automatically when a new version is tagged in our [GitHub repository](#).

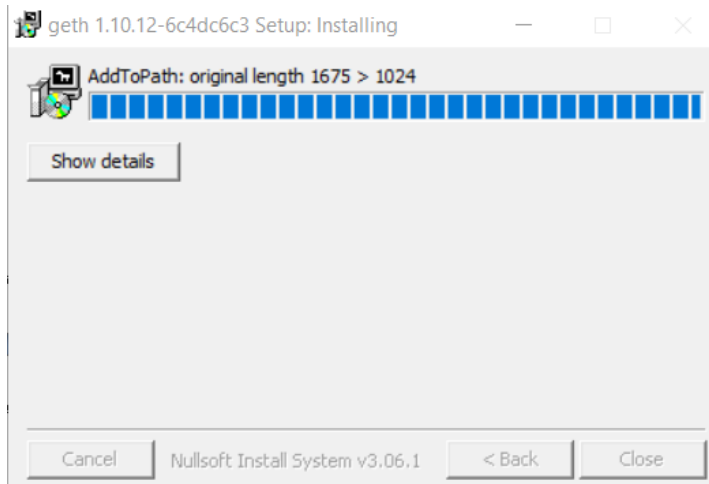
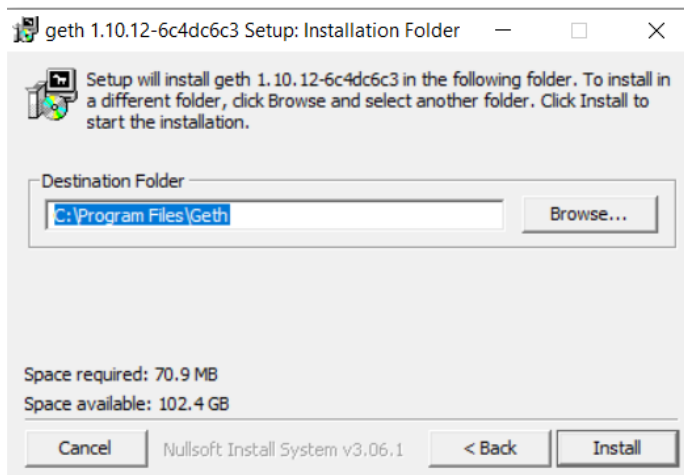
Android iOS Linux macOS Windows								
Release	Commit	Kind	Arch	Size	Published	Signature	Checksum (MD5)	
Geth 1.10.12	6c4dc6c3...	Installer	32-bit	53.81 MB	Last Monday at 8:14 PM	Signature	b0ce5ef1ade2080ba1c27f8b72ab6479	
Geth 1.10.12	6c4dc6c3...	Archive	32-bit	20.36 MB	Last Monday at 8:10 PM	Signature	ba129c26eff2ab7d00433a04d7ef8cb4	
Geth 1.10.12	6c4dc6c3...	Installer	64-bit	55.56 MB	Last Monday at 8:06 PM	Signature	5818053124edfe130fcd123910668f0b	
Geth 1.10.12	6c4dc6c3...	Archive	64-bit	21.04 MB	Last Monday at 8:03 PM	Signature	aac8d5c6a20c331788c44944eb0ff1fa	

- Once your download is complete, open the installer and click “I Agree”

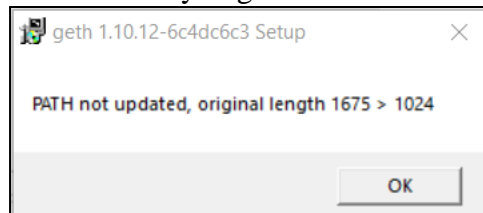


- Follow all the installation steps.

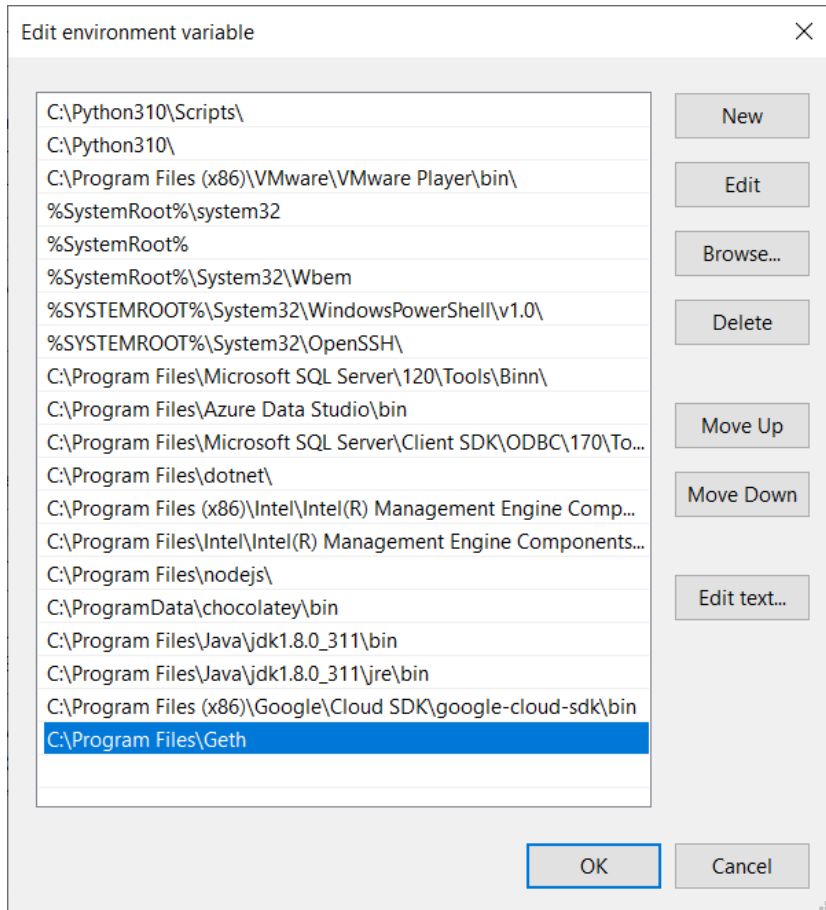
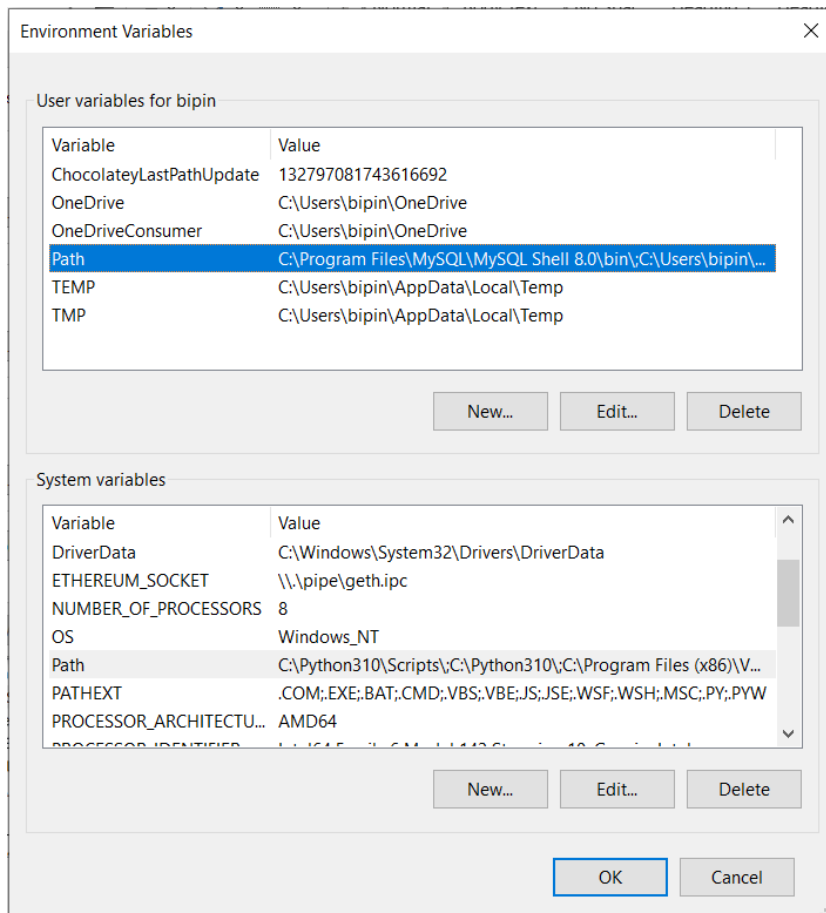




- Check if you get this error after installation.



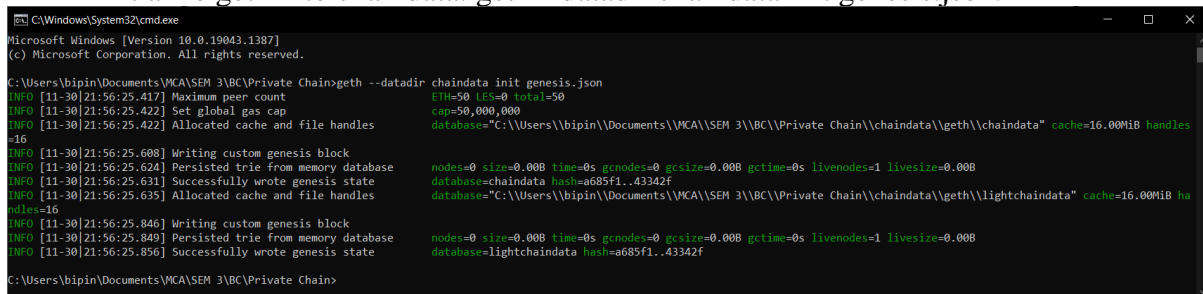
- If you do, then go to the installation path of geth and copy the path of the folder. Then open up environment variables and add the path to the “Path” section there.



- Click OK. The installation of Geth is now done.
 - Steps to create a Genesis block and a private chain:
 - 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.

```
{
  "config": {
    "chainId": 4777,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0, "eip158Block": 0
  },
  "alloc" : {},
  "difficulty" : "0x400",
  "extraData" : "",
  "gasLimit" : "0x7A1200",
  "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp" : "0x00"
}
```

- Now, we have to initialize our Ethereum Node. For that, open the command prompt and change your current directory to the Private Chain folder.
- Initialize geth into chaindata: `geth --datadir chaindata init genesis.json`.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1387]
(c) Microsoft Corporation. All rights reserved.

C:\Users\bipin\Documents\VMCA\SEM 3\BC\Private Chain>geth --datadir chaindata init genesis.json
INFO [11-30]21:56:25.417] Maximum peer count      ETH=50 (ES=0 total=50)
INFO [11-30]21:56:25.422] Set global gas cap      cap=50,000,000
INFO [11-30]21:56:25.422] Allocated cache and file handles database="C:\Users\bipin\Documents\VMCA\SEM 3\BC\Private Chain\chaindata" cache=16.00MiB handles=16
INFO [11-30]21:56:25.608] Writing custom genesis block
INFO [11-30]21:56:25.624] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsz=0.00B gctime=0s livenodes=1 liveness=0.00B
INFO [11-30]21:56:25.631] Successfully wrote genesis state database=chaindata hash=a685f1..43342f
INFO [11-30]21:56:25.635] Allocated cache and file handles database="C:\Users\bipin\Documents\VMCA\SEM 3\BC\Private Chain\chaindata\geth\lightchaindata" cache=16.00MiB handles=16
INFO [11-30]21:56:25.846] Writing custom genesis block
INFO [11-30]21:56:25.849] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsz=0.00B gctime=0s livenodes=1 liveness=0.00B
INFO [11-30]21:56:25.856] Successfully wrote genesis state database=lightchaindata hash=a685f1..43342f

C:\Users\bipin\Documents\VMCA\SEM 3\BC\Private Chain>
```


➤ run geth: `geth --datadir=./chaindata/`

```
C:\Windows\System32\cmd.exe - geth --datadir=./chaindata/
C:\Users\bipin\Documents\MCA\SEM 3\BC\Private Chain>geth --datadir=./chaindata/
INFO [11-30 21:56:53.156] Starting Geth on Ethereum mainnet...
INFO [11-30 21:56:53.161] Dumping default cache on mainnet
INFO [11-30 21:56:53.161] Maximum peer count      provided=1024 updated=4096
INFO [11-30 21:56:53.173] Set global gas cap      ETH=50 lGas=0 total=50
INFO [11-30 21:56:53.175] Allocated trie memory caches  cap=50,000,000
INFO [11-30 21:56:53.180] Allocated cache and file handles clean=614.00MiB dirty=1024.00MiB
INFO [11-30 21:56:53.795] Opened ancient database  database="C:\Users\bipin\Documents\MCA\SEM 3\BC\Private Chain\chaindata\geth\chaindata" cache=2.00GiB handles=
INFO [11-30 21:56:53.804] Initialized chain configuration  config="(ChainID: 4777 Homestead; 0 DAO; <nil> DAOsupport: false EIP150: 0 EIP155: 0 EIP158: 0 Byzantium; <nil> Constant
inople: <nil> Petersburg; <nil> Istanbul; <nil> Muir Glacier; <nil> Berlin; <nil> London; <nil> Arrow Glacier; <nil> Engine: unknown)"
INFO [11-30 21:56:53.816] Disk storage enabled for ethash caches  dir="C:\Users\bipin\Documents\MCA\SEM 3\BC\Private Chain\chaindata\geth\ethash" count=3
INFO [11-30 21:56:53.822] Disk storage enabled for ethash DAGs    dir="C:\Users\bipin\AppData\Local\Ethash" count=2
INFO [11-30 21:56:53.827] Initialising Ethereum protocol  network=1 &#Version=<nil>
INFO [11-30 21:56:53.838] Loaded most recent local header  number=0 hash=a685f1..43342f td=1024 age=52y8mo1d
INFO [11-30 21:56:53.838] Loaded most recent local full block  number=0 hash=a685f1..43342f td=1024 age=52y8mo1d
INFO [11-30 21:56:53.844] Loaded most recent local fast block  number=0 hash=a685f1..43342f td=1024 age=52y8mo1d
WARN [11-30 21:56:53.849] Failed to load snapshot, regenerating  err="missing or corrupted snapshot"
INFO [11-30 21:56:53.849] Rebuilding state snapshot
INFO [11-30 21:56:53.851] Resuming state snapshot generation  root=56e81f..63b421 accounts=0 slots=0 storage=0.008 elapsed="564µs"
INFO [11-30 21:56:53.853] Regenerated local transaction journal  transactions=0 accounts=0
INFO [11-30 21:56:53.856] Generated state snapshot  accounts=0 slots=0 storage=0.008 elapsed=5.969ms
INFO [11-30 21:56:53.862] Gasprice oracle is ignoring threshold set threshold=2
WARN [11-30 21:56:53.869] Error reading unclean shutdown markers  error="leveldb: not found"
INFO [11-30 21:56:53.870] Starting peer-to-peer node  instance=Geth/v1.10.13-stable-7a0c19f8/windows-amd64/go1.17.2
INFO [11-30 21:56:54.041] New local node record  seq=1,638,289,614,040 id=110c5458d2e011a4 ip=127.0.0.1 udp=30303 tcp=30303
INFO [11-30 21:56:54.046] Started P2P networking  self=enode://54ebfbc5afe1f275709ad7bc0bb7fd844da2bb8f87dcf71193eb17162791a1361457757f9e9039599ffc75ac8f7768dd1e6bf4866
INFO [11-30 21:56:54.046] IPC endpoint opened  url=\\.\pipe\geth.ipc
INFO [11-30 21:56:57.382] New local node record  seq=1,638,289,614,041 id=110c5458d2e011a4 ip=103.251.51.42 udp=43945 tcp=30303
INFO [11-30 21:57:04.189] Looking for peers  peercount=1 tried=2 static=0
WARN [11-30 21:58:15.631] Served eth_coinbase  reqid=3 t=0s err="etherbase must be explicitly specified"
WARN [11-30 22:01:53.171] Served eth_coinbase  reqid=3 t=0s err="etherbase must be explicitly specified"
INFO [11-30 22:04:52.812] Looking for peers  peercount=1 tried=0 static=0
INFO [11-30 22:04:53.007] New local node record  seq=1,638,289,614,042 id=110c5458d2e011a4 ip=127.0.0.1 udp=30303 tcp=30303
INFO [11-30 22:04:55.989] New local node record  seq=1,638,289,614,043 id=110c5458d2e011a4 ip=103.251.51.42 udp=30948 tcp=30303
INFO [11-30 22:05:03.041] Looking for peers  peercount=0 tried=25 static=0
INFO [11-30 22:05:13.317] Looking for peers  peercount=0 tried=5 static=0
WARN [11-30 22:05:19.084] Served eth_coinbase  reqid=3 t=0s err="etherbase must be explicitly specified"
```

➤ Now minimize this and open up another command prompt.

➤ IPC to interact with Geth: `geth attach ipc:\\.pipe\geth.ipc`

```
Command Prompt - geth attach ipc:\\.pipe\geth.ipc
Microsoft Windows [Version 10.0.19043.1387]
(c) Microsoft Corporation. All rights reserved.

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

instance: Geth/v1.10.13-stable-7a0c19f8/windows-amd64/go1.17.2
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
datadir: C:\Users\bipin\Documents\MCA\SEM 3\BC\Private Chain\chaindata
modules: admin:1.0 debug: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
>
```

➤ In this command prompt you can execute geth commands

Output:

➤ Create new account, check pre-existing accounts, check coinbase account hash and balance:

```
Command Prompt - geth attach ipc:\\.pipe\geth.ipc
(c) Microsoft Corporation. All rights reserved.

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

instance: Geth/v1.10.13-stable-7a0c19f8/windows-amd64/go1.17.2
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
datadir: C:\Users\bipin\Documents\MCA\SEM 3\BC\Private Chain\chaindata
modules: admin:1.0 debug: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
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x29dc6a0299204487c139d531c2ff557357abce4e"
> eth.accounts
["0x29dc6a0299204487c139d531c2ff557357abce4e"]
> eth.coinbase
"0x29dc6a0299204487c139d531c2ff557357abce4e"
> eth.getBalance(eth.accounts[0])
0
>
```

➤ Start mining:

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

```
C:\Windows\System32\cmd.exe - geth --datadir=.chaindata/
INFO [11-30] 22:07:09.627] Looking for peers
INFO [11-30] 22:07:18.409] Your new key was generated
WARN [11-30] 22:07:18.415] Please backup your key file!
07--29dc6a0299204487c139d531c2ff557357abc4e"
WARN [11-30] 22:07:18.421] Please remember your password!
INFO [11-30] 22:07:30.715] Looking for peers
INFO [11-30] 22:07:36.765] Etherbase automatically configured
INFO [11-30] 22:07:41.576] Looking for peers
INFO [11-30] 22:07:51.800] Looking for peers
INFO [11-30] 22:08:01.879] Looking for peers
INFO [11-30] 22:08:13.204] Looking for peers
INFO [11-30] 22:08:23.224] Looking for peers
INFO [11-30] 22:08:33.275] Looking for peers
INFO [11-30] 22:08:43.427] Looking for peers
INFO [11-30] 22:08:53.965] Looking for peers
INFO [11-30] 22:08:55.022] Updated mining threads
INFO [11-30] 22:08:55.026] Transaction pool price threshold updated
INFO [11-30] 22:08:55.029] Commit new mining work
INFO [11-30] 22:08:56.232] Generating DAG in progress
INFO [11-30] 22:08:56.833] Generating DAG in progress
INFO [11-30] 22:08:57.418] Generating DAG in progress
INFO [11-30] 22:08:58.013] Generating DAG in progress
INFO [11-30] 22:08:58.620] Generating DAG in progress
INFO [11-30] 22:08:59.200] Generating DAG in progress
INFO [11-30] 22:08:59.809] Generating DAG in progress
INFO [11-30] 22:09:00.384] Generating DAG in progress
INFO [11-30] 22:09:00.966] Generating DAG in progress
INFO [11-30] 22:09:01.529] Generating DAG in progress
INFO [11-30] 22:09:02.072] Generating DAG in progress
INFO [11-30] 22:09:02.649] Generating DAG in progress
INFO [11-30] 22:09:03.195] Generating DAG in progress
INFO [11-30] 22:09:03.768] Generating DAG in progress
INFO [11-30] 22:09:04.318] Generating DAG in progress
INFO [11-30] 22:09:04.632] Looking for peers
INFO [11-30] 22:09:04.870] Generating DAG in progress
INFO [11-30] 22:09:05.445] Generating DAG in progress
INFO [11-30] 22:09:06.010] Generating DAG in progress
INFO [11-30] 22:09:06.593] Generating DAG in progress
peercount=0 tried=11 static=0
address=0x29dc6a0299204487c139d531c2ff557357abc4e
path="C:\Users\bipin\Documents\MCA\SEM 3\BC\Private Chain\chaindata\keystore\UTC--2021-11-30T16-37-16.81584730
peercount=0 tried=26 static=0
peercount=0 tried=39 static=0
address=0x29dc6a0299204487c139d531c2ff557357abc4e
peercount=0 tried=36 static=0
peercount=0 tried=21 static=0
peercount=0 tried=31 static=0
peercount=0 tried=20 static=0
peercount=1 tried=32 static=0
peercount=1 tried=24 static=0
peercount=0 tried=27 static=0
peercount=0 tried=22 static=0
threads=8
price=1,000,000,000
number=1 sealhash=b19ab7.52c826 uncles=0 txs=0 gas=0 fees=0 elapsed=0s
epoch=0 percentage=0 elapsed=631.127ms
epoch=0 percentage=1 elapsed=1.232s
epoch=0 percentage=2 elapsed=1.817s
epoch=0 percentage=3 elapsed=2.412s
epoch=0 percentage=4 elapsed=3.019s
epoch=0 percentage=5 elapsed=3.599s
epoch=0 percentage=6 elapsed=4.208s
epoch=0 percentage=7 elapsed=4.783s
epoch=0 percentage=8 elapsed=5.365s
epoch=0 percentage=9 elapsed=5.928s
epoch=0 percentage=10 elapsed=6.471s
epoch=0 percentage=11 elapsed=7.048s
epoch=0 percentage=12 elapsed=7.503s
epoch=0 percentage=13 elapsed=8.167s
epoch=0 percentage=14 elapsed=8.716s
peercount=0 tried=32 static=0
epoch=0 percentage=15 elapsed=9.269s
epoch=0 percentage=16 elapsed=9.844s
epoch=0 percentage=17 elapsed=10.408s
epoch=0 percentage=18 elapsed=10.991s
```

➤ Stop mining:

```
> miner.stop()
null
> eth.getBalance(eth.accounts[0])
2.02e+21
```

➤ Check blockNumber:

```
> eth.blockNumber
404
```

- Now for transaction, we need 2 accounts. One would be the coinbase account and the other one would be the account which we would be transferring to. After creating the new account, we need to unlock the coinbase account as well.

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x859988cbadf56750cff9f152dfc8ab3a91749956"
> eth.getBalance(eth.accounts[1])
0
```

➤ Transaction:

```
> eth.sendTransaction({from: eth.coinbase, to: eth.accounts[1], value: web3.toWei(10,"ether")})
"0x65c4a7cf1f84343f94be7cbb070aa9048c849a6e3aa7d563269da30bcfe255a2"
```

- Now to add the transaction to the blockchain, we need to start the mining process again.

```
> miner.start()
null
> miner.stop()
null
> eth.getBalance(eth.accounts[1])
100000000000000000000
```

- Check the information about the occurred transaction:

```

> eth.getTransaction("0x65c4a7cf1f84343f94be7ccb070aa9048c849a6e3aa7d563269da30bcfe255a2")
{
  blockHash: "0xf10308cee41a3dc1aa633a79d02b500c4dfae4418adc69afc4738f5aa661d3f9",
  blockNumber: 405,
  from: "0x29dc6a0299204487c139d531c2ff557357abce4e",
  gas: 21000,
  gasPrice: 1000000000,
  hash: "0x65c4a7cf1f84343f94be7ccb070aa9048c849a6e3aa7d563269da30bcfe255a2",
  input: "0x",
  nonce: 0,
  r: "0xa9bb51b5b8e3a1ab3c1b63d1cd6e75b4bfa84e061c751b717f5d4c7b2d0e2407",
  s: "0x7bbe87904c6e2fbf7bc85b4c6c439bd408b40c7d6f94f636d58bbad6ee2ef8a9",
  to: "0x859988cbadf56750cff9f152dfc8ab3a91749956",
  transactionIndex: 0,
  type: "0x0",
  v: "0x2575",
  value: 1000000000000000000
}
>

```

- Check balance in terms of ether:

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

- Get information about the latest block:

[illegible]

- Get information about a specific block by entering it's block number.

[illegible]

Conclusion:

Hence, we have successfully learned how to install Geth, create a genesis block, private chain and commands to create user accounts, mine and transact ethers among accounts.

Practical No – 6

Aim: To implement the installation of Ganache, Metamask and Remix IDE and deploy smart contract using injected web 3 environment.

Theory:

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.

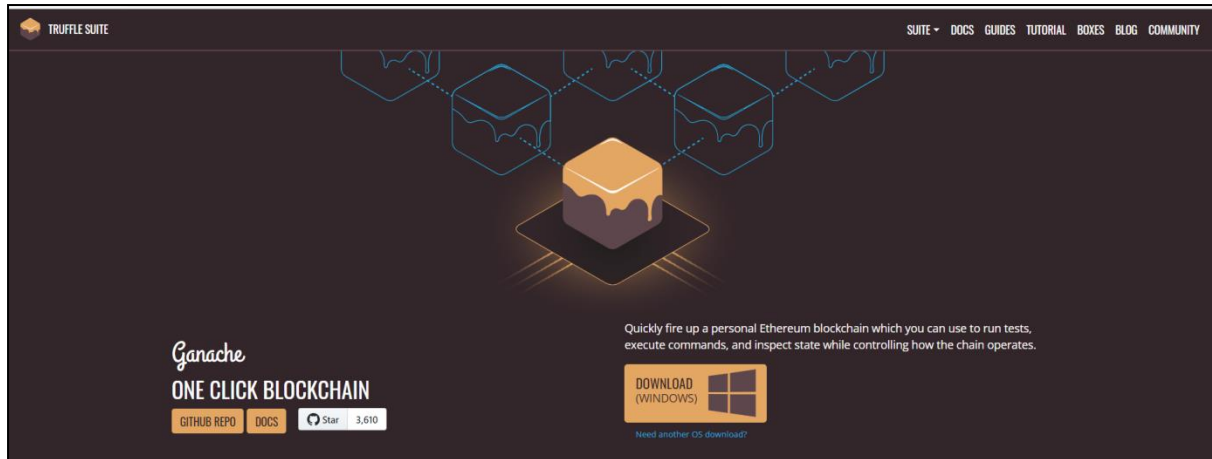
On the other hand, Ganache is a high-end development tool used to run your own local blockchain for both Ethereum and Corda dApp development. Ganache is helpful in all parts of the development process. The local chain allows you to develop, deploy and test your projects and smart contracts in a deterministic 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.

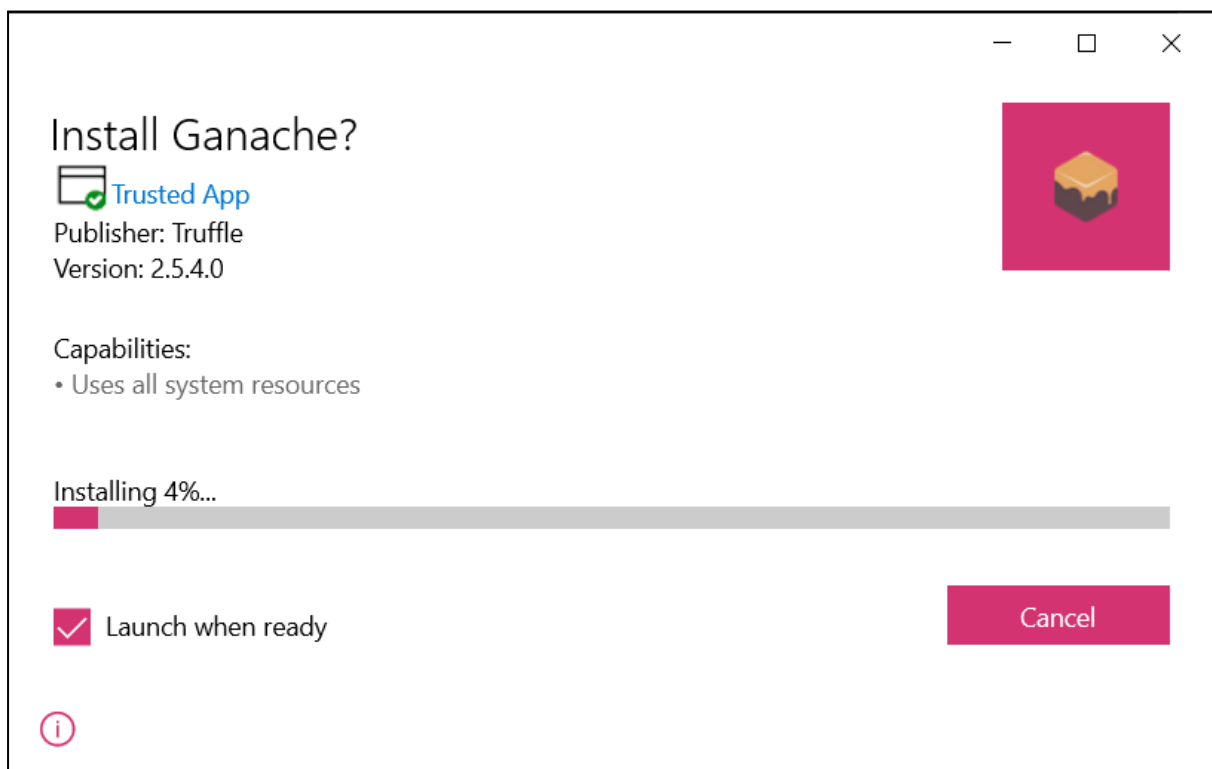
Installation Steps:

➤ Ganache:

1) Download Ganache from <https://www.trufflesuite.com/ganache>



2) Install Ganache



The screenshot shows the Ganache application interface. At the top, there's a navigation bar with icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this is a status bar displaying various network parameters: CURRENT BLOCK (0), GAS PRICE (2000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLACIER), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), MINING STATUS (AUTOMINING), and WORKSPACE (QUICKSTART). There are also buttons for SAVE, SWITCH, and a settings icon.

The main area displays the MNEMONIC: "trash egg quick farm blood notable now name execute tool very ignore" and the HD PATH: "m/44'/60'/0'/0/account_index". Below this, a table lists four accounts:

ADDRESS	BALANCE	TX COUNT	INDEX	
0x9dEE9641d576BF06Ef91CC12f16Dd028f34e7B79	100.00 ETH	0	0	
0xCC3da1aaDDFa3638Fa76D6A750872d7F907e17fA	100.00 ETH	0	1	
0xd945F260Bd207EfcA2003C3A3e85486bDDf0CE39	100.00 ETH	0	2	
0x5d450095a966885AD41b1FE95377bbBEca66512C	100.00 ETH	0	3	

The console in the above screenshot shows user accounts with balance of 100 ETH (Ether - a currency for transaction on Ethereum platform). It shows a transaction count of zero for each account. As the user has not performed any transactions so far, this count is obviously zero.

➤ **Metamask:**

Installation:

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.

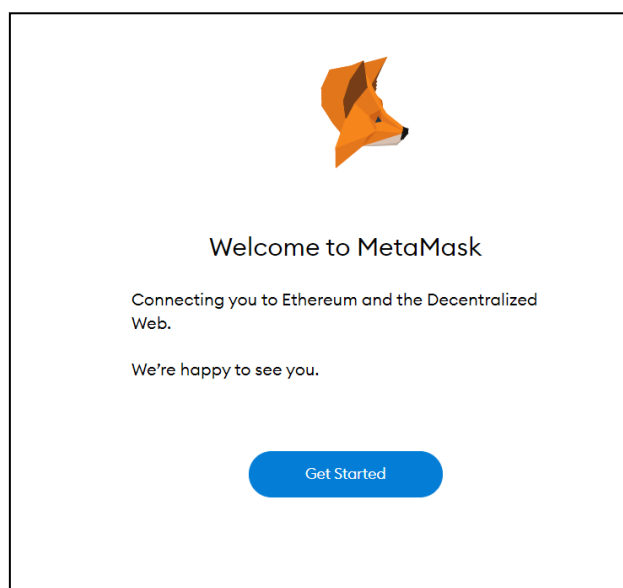
Step 1: Go to Chrome Web Store Extensions Section.

Step 2: Search MetaMask

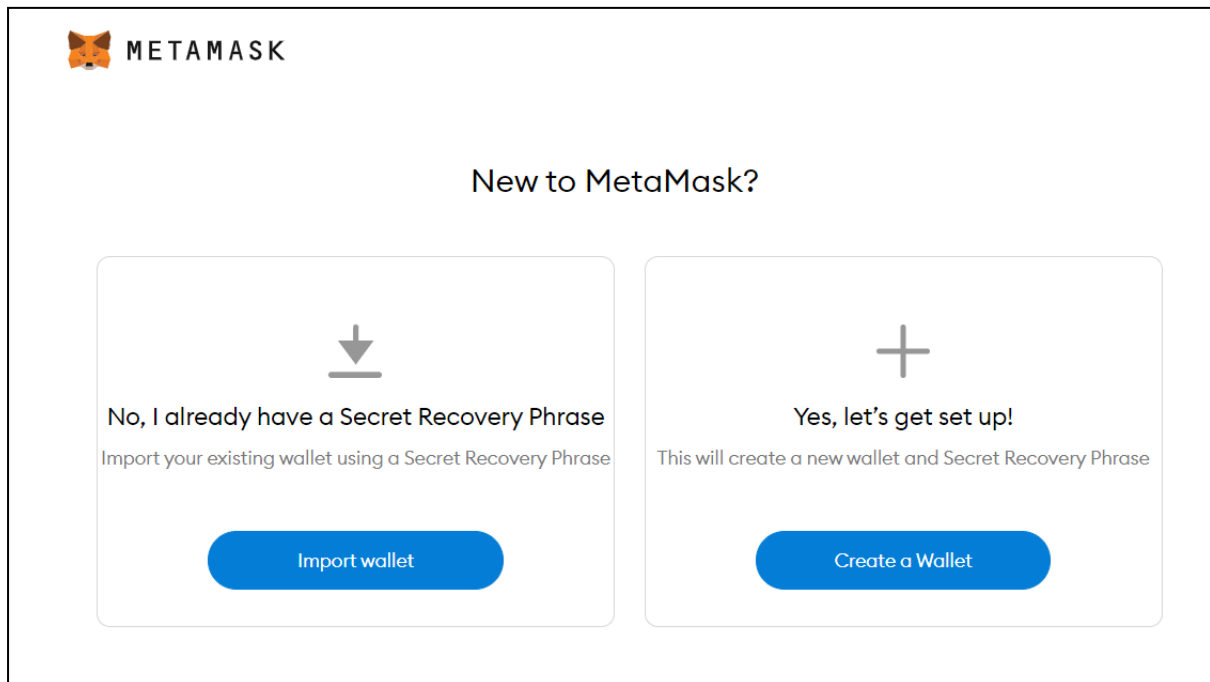


Step 3: Click the Add to Chrome button.

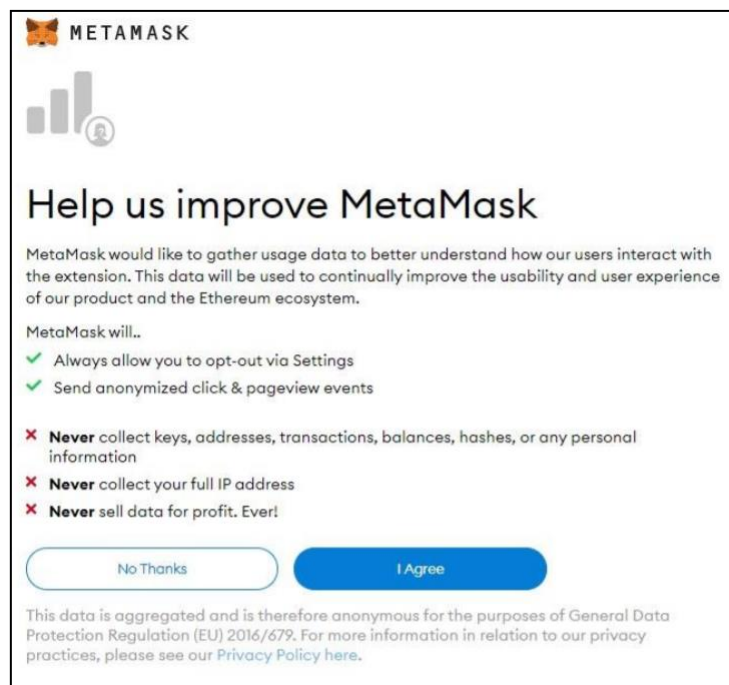
Step 4: Once installation is complete this page will be displayed. Click on the Get Started button.



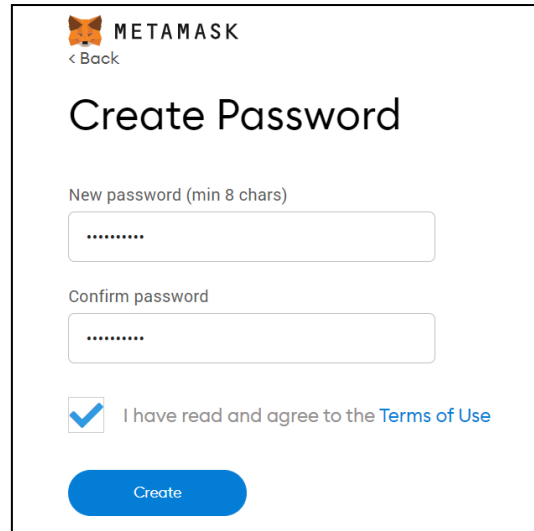
Step 5: This is the first time creating a wallet, so click the Create a Wallet button. If there is already a wallet then import the already created using the Import Wallet button.



Step 6: Click I Agree button to allow data to be collected to help improve MetaMask or else click the No Thanks button. The wallet can still be created even if the user will click on the No Thanks button.



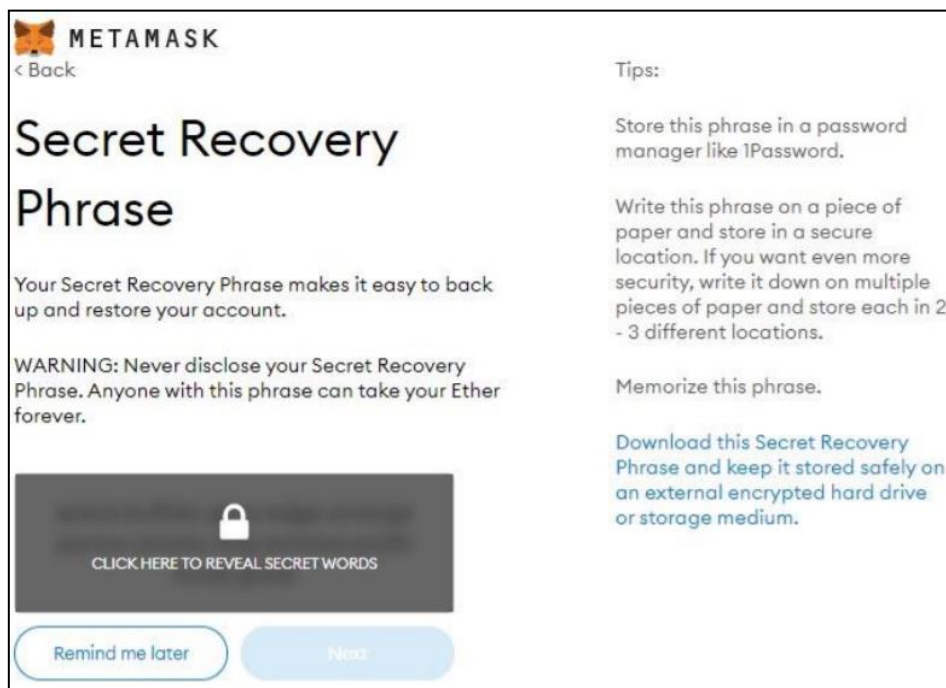
Step 7: Create a password for your wallet. This password is to be entered every time the browser is launched and wants to use MetaMask. A new password needs to be created if chrome is uninstalled or if there is a switching of browsers. In that case, go through the Import Wallet button. This is because MetaMask stores the keys in the browser. Agree to Terms of Use.



The image shows the 'Create Password' screen in the MetaMask mobile app. At the top, there is the MetaMask logo and a '< Back' link. The title 'Create Password' is prominently displayed. Below it, there are two input fields: 'New password (min 8 chars)' and 'Confirm password', both masked with dots. A checkbox with a blue checkmark is checked, followed by the text 'I have read and agree to the [Terms of Use](#)'. At the bottom, there is a blue 'Create' button.

Step 8: Click on the dark area which says Click here to reveal secret words to get your secret phrase.

Step 9: This is the most important step. Back up your secret phrase properly. Do not store your secret phrase on your computer. Please read everything on this screen until you understand it completely before proceeding. The secret phrase is the only way to access your wallet if you forget your password. Once done click the Next button.



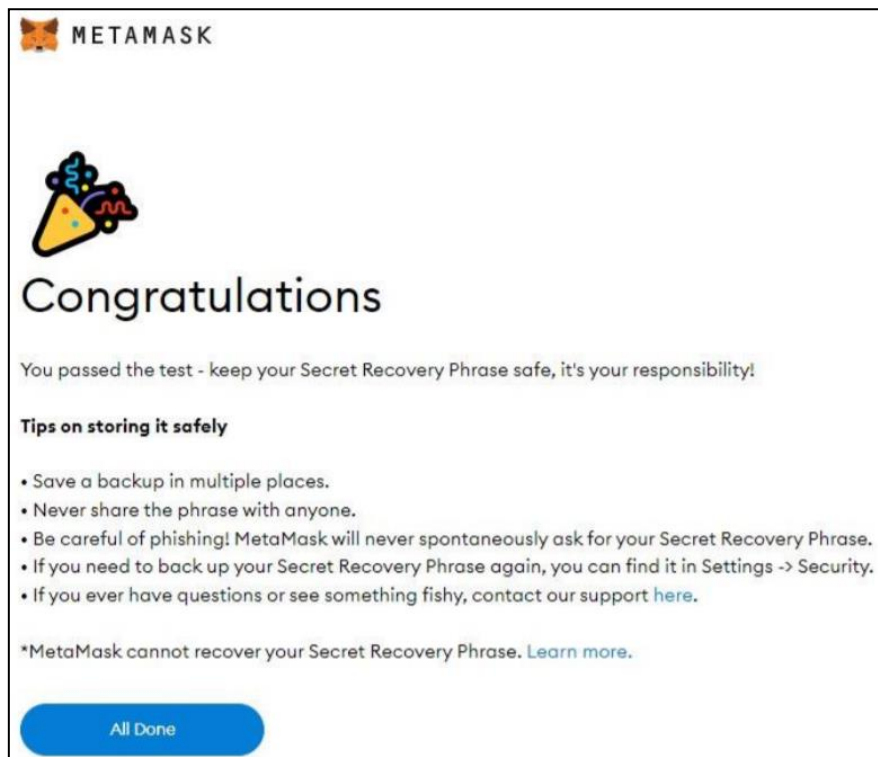
The image shows the 'Secret Recovery Phrase' screen in the MetaMask mobile app. At the top, there is the MetaMask logo and a '< Back' link. The title 'Secret Recovery Phrase' is prominently displayed. Below it, there is a paragraph: 'Your Secret Recovery Phrase makes it easy to back up and restore your account.' followed by a 'WARNING: Never disclose your Secret Recovery Phrase. Anyone with this phrase can take your Ether forever.' A large dark grey button with a white padlock icon and the text 'CLICK HERE TO REVEAL SECRET WORDS' is centered. At the bottom, there are two buttons: 'Remind me later' and 'Next'. On the right side, there is a 'Tips:' section with three bullet points: 'Store this phrase in a password manager like 1Password.', 'Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.', and 'Memorize this phrase.' Below these tips, there is a link: 'Download this Secret Recovery Phrase and keep it stored safely on an external encrypted hard drive or storage medium.'

Step 10: Click the buttons respective to the order of the words in your seed phrase. In other words, type the seed phrase using the button on the screen. If done correctly the Confirm button should turn blue.



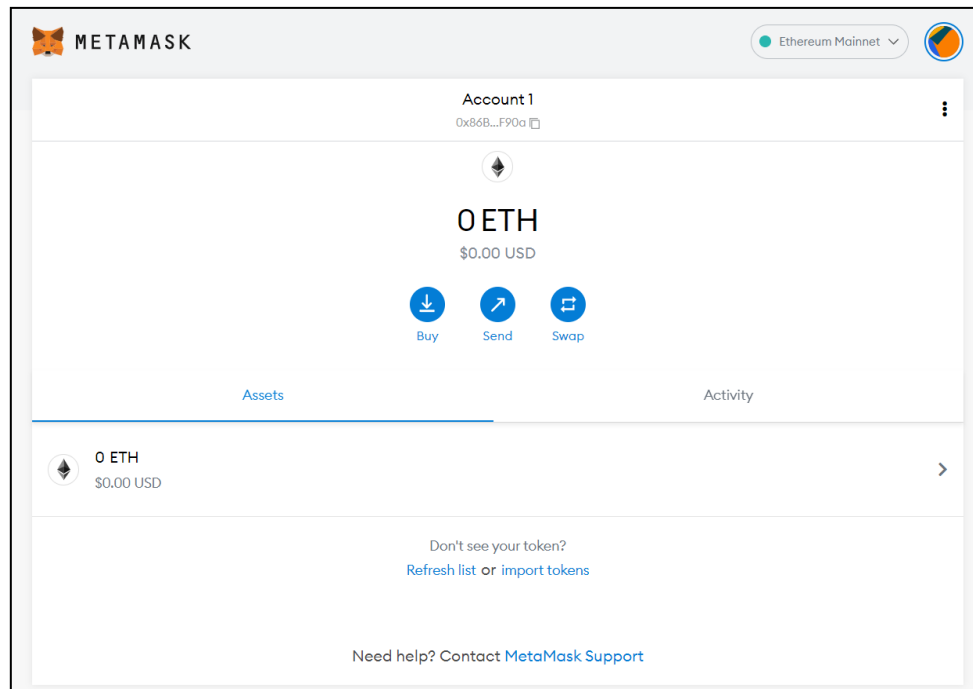
The image shows the Metamask interface for confirming a secret recovery phrase. At the top, there is a Metamask logo and a '< Back' link. The main heading is 'Confirm your Secret Recovery Phrase'. Below this, a instruction says 'Please select each phrase in order to make sure it is correct.' There is a large empty text box for the user to type the phrase. Below the text box, there is a grid of 12 buttons, each containing a word: 'achieve', 'arrange', 'buffalo', 'edge', 'glory', 'glove', 'horse', 'person', 'potato', 'profit', 'sing', and 'wreck'. At the bottom, there is a 'Confirm' button.

Step 11: Click the Confirm button. Please follow the tips mentioned.

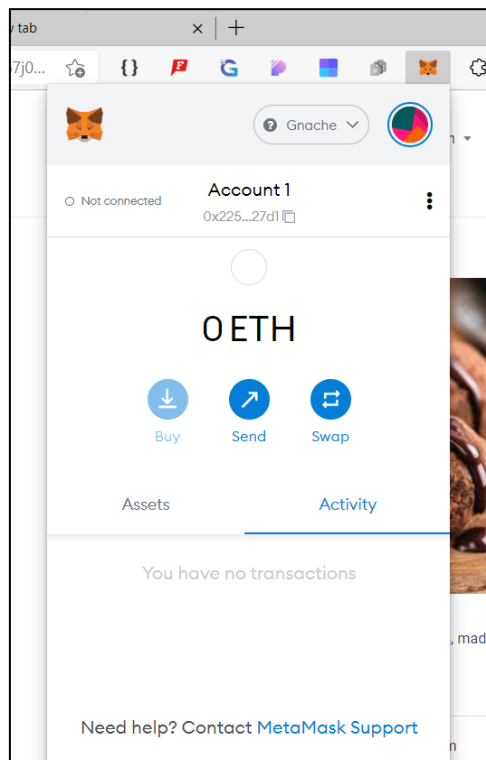


The image shows the Metamask interface for congratulations. At the top, there is a Metamask logo. Below it, there is a colorful party hat icon. The main heading is 'Congratulations'. Below this, a message says 'You passed the test - keep your Secret Recovery Phrase safe, it's your responsibility!'. There is a section titled 'Tips on storing it safely' with a list of tips: 'Save a backup in multiple places.', 'Never share the phrase with anyone.', 'Be careful of phishing! MetaMask will never spontaneously ask for your Secret Recovery Phrase.', 'If you need to back up your Secret Recovery Phrase again, you can find it in Settings -> Security.', and 'If you ever have questions or see something fishy, contact our support [here](#).' Below the tips, there is a note: '*MetaMask cannot recover your Secret Recovery Phrase. [Learn more](#).' At the bottom, there is a blue button labeled 'All Done'.

Step 12: One can see the balance and copy the address of the account by clicking on the Account 1 area.

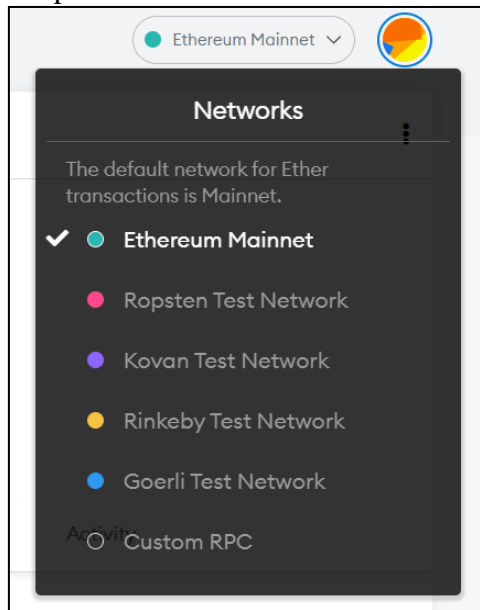


Step 13: One can access MetaMask in the browser by clicking the MetaMask extension icon on the top right.

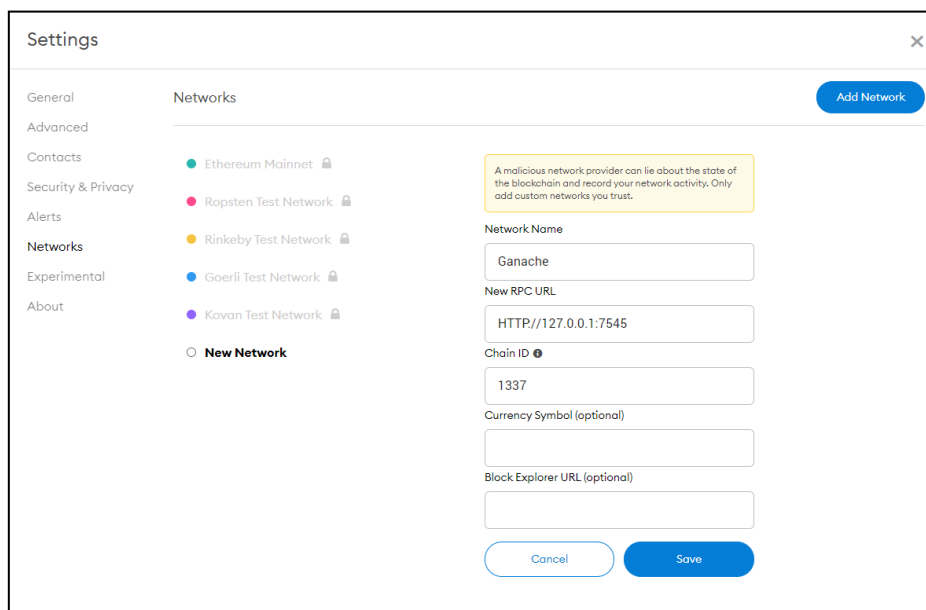
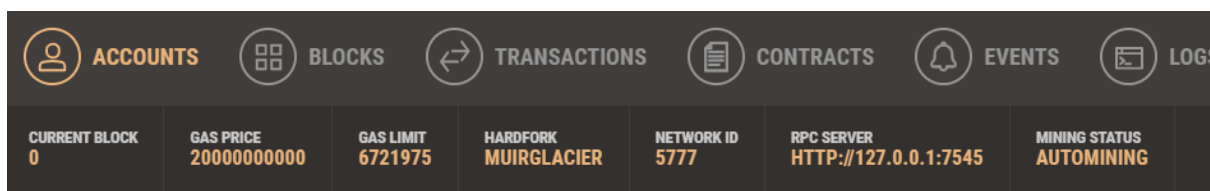


Step 14: Adding Ganache Network to MetaMask.

- a. Click on the Networks Dropdown and Select Custom RPC.



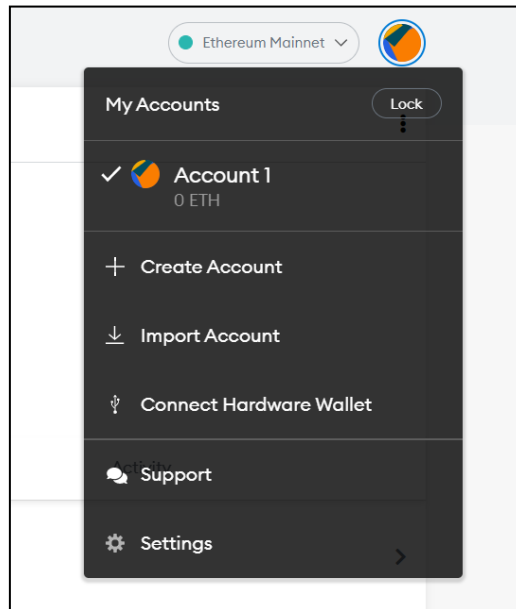
- b. Fill the network details such as Network Name, RPC Url and Chain Id and Save it.
Note: RPC url is mentioned on your ganache console as RPC Server.



Step 14: Importing Accounts.

To Import an account click on the circular icon at the top – right of your MetaMask Extension and select Import accounts.

- Copy the private key of your ganache account by clicking on key icon of particular account.



- You will need to copy the private key of your ganache account by clicking on key icon of particular account.

ADDRESS	BALANCE	TX COUNT	INDEX	
0x9bB4A88af2DcD75F18D8F39d49bDaBd1CF307361	100.00 ETH	0	0	

ACCOUNT INFORMATION

ACCOUNT ADDRESS
0x9bB4A88af2DcD75F18D8F39d49bDaBd1CF307361

PRIVATE KEY
604bc3e904a9652eb31cfafba695314f701fc77dded0de61c6b6e82f3bde4918
Do not use this private key on a public blockchain; use it for development purposes only!

DONE

Import Account

Imported accounts will not be associated with your originally created MetaMask account Secret Recovery Phrase. Learn more about imported accounts [here](#)

Select Type

Private Key ▾

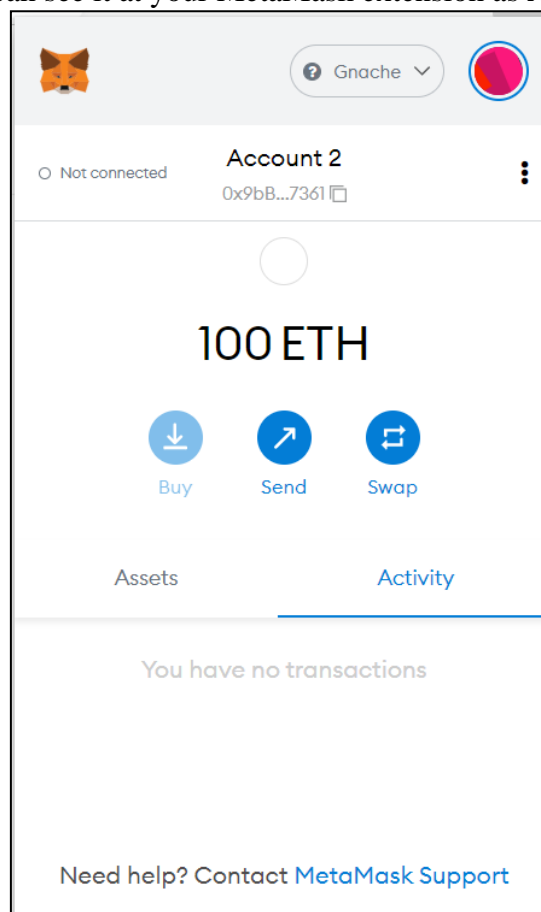
Paste your private key string here:

.....|

Cancel

Import

- c. Click on import button once private key string is pasted.
- d. Once imported you can see it at your MetaMask extension as Account 2.



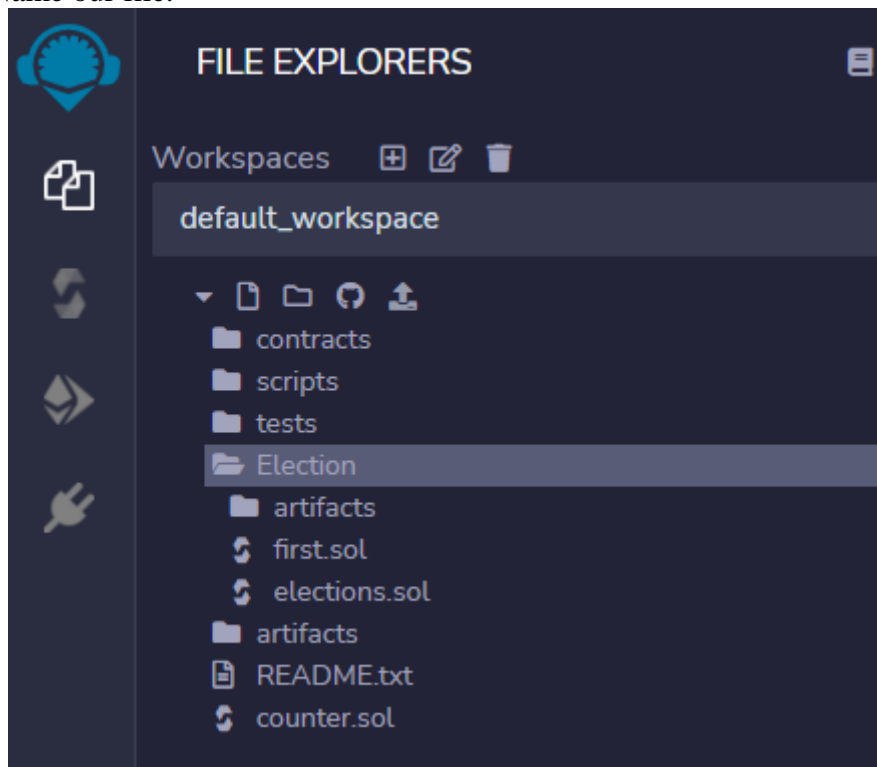
➤ **Remix IDE:**

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.

Step 1 : [Go to https://remix.ethereum.org/](https://remix.ethereum.org/)

- a. As we use Solidity to write our smart contracts, .sol extension is used.
- b. Let's now create a new contract. For that, right-click on the workspace and select New File. Name our file.



Step 2: Select your newly created file and type the following code.

Code:

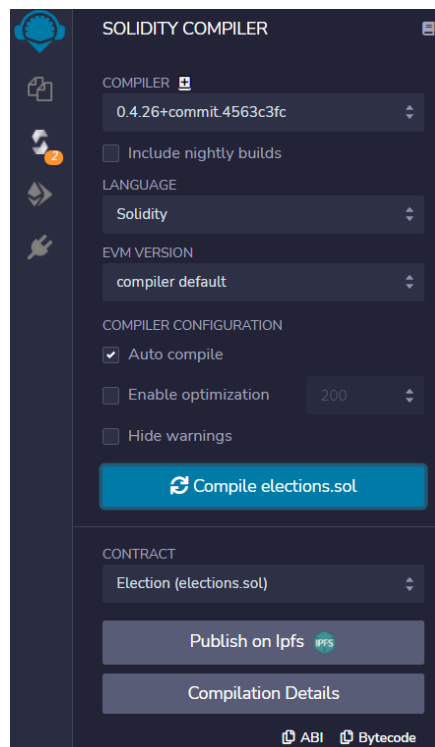
```
pragma solidity ^0.4.2;
contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }
    // Store accounts that have voted
    mapping(address => bool) public voters;
    // Store Candidates
    // Fetch Candidate
    mapping(uint => Candidate) public candidates;
    // Store Candidates Count
    uint public candidatesCount;

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

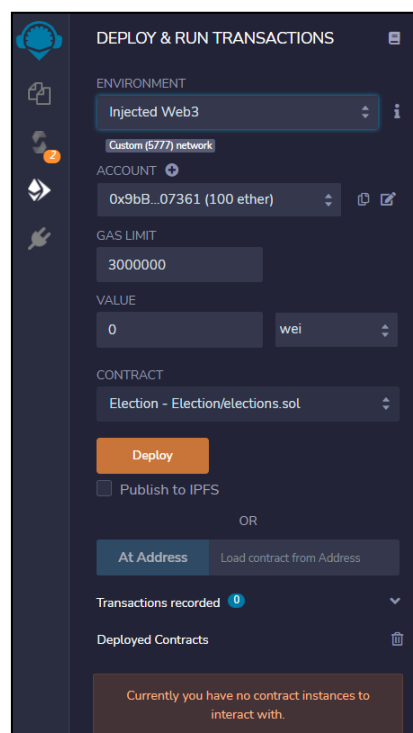
    function addCandidate (string _name) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
    }
    function vote (uint _candidateId) public {
        // require that they haven't voted before
        require(!voters[msg.sender]);
        // 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
        votedEvent(_candidateId);
    }
}
```

Step 3:

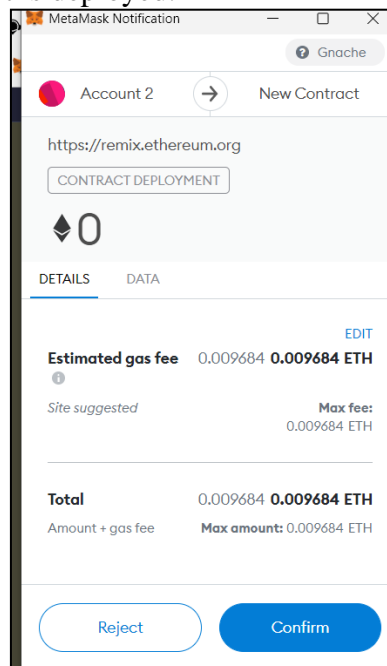
- Click on the solidity compiler present on the left.
- Select Auto-compile so our contract automatically compiles when we do some changes.

**Step 4:**

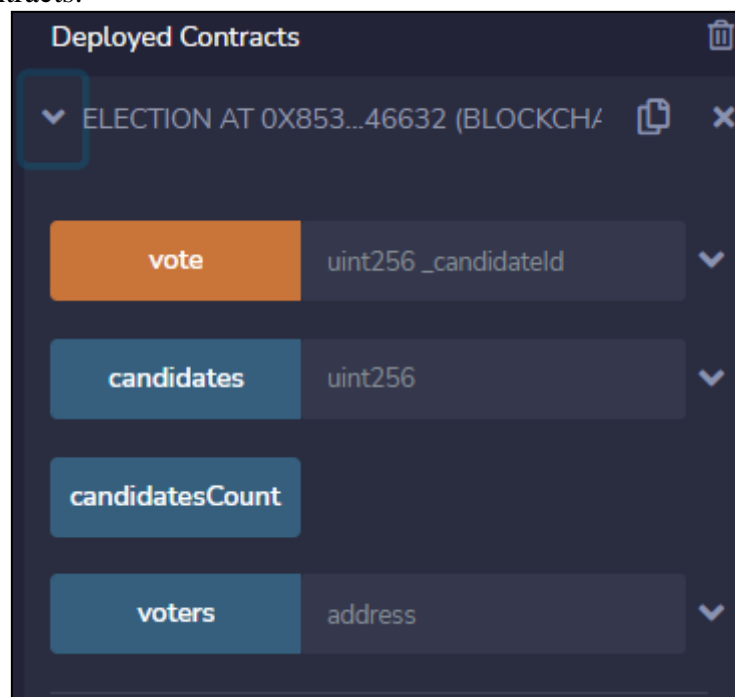
- Click on Deploy and Run Transactions Button.
- Set Environment to “Injected Web3”.(Make sure you are connected to website with Metamask).



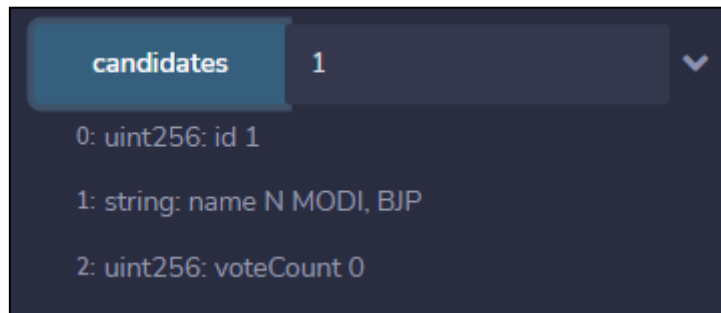
- c. Click on Deploy Button and you will see a pop up for confirmation.
- d. Once confirm your contract is deployed.



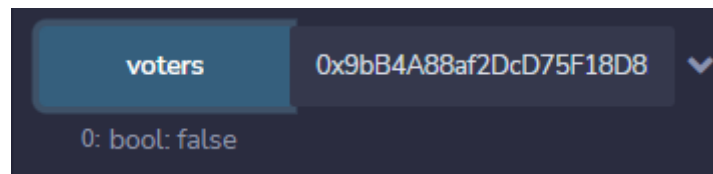
- e. After your contract is successfully deployed, you will be able to see your contract under the deployed contracts.



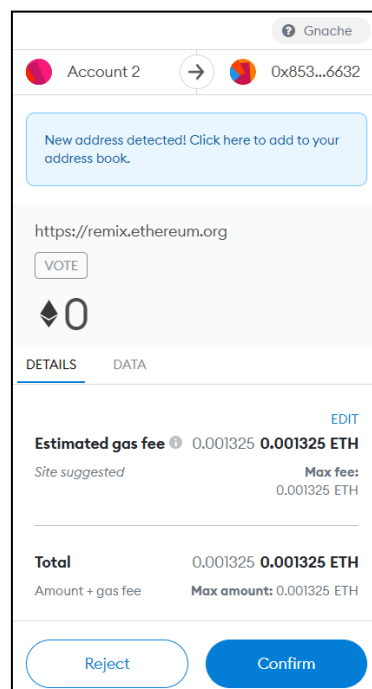
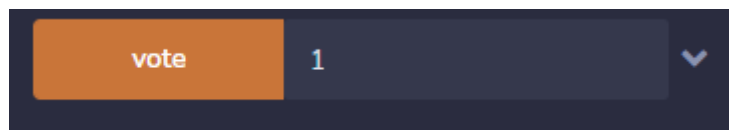
- f. If you give the input as 1 in Candidates column, you will be able to see the details of our first candidate. In our case, the first candidate is Modi.



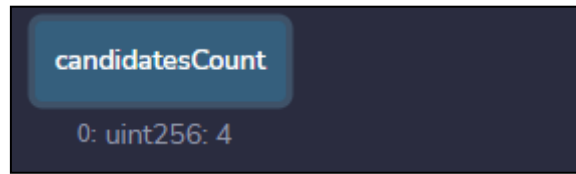
- g. If you click on votes button with any account address as input you can see whether the person has voted or not.
h. It gives the Boolean result meaning false as not voted and true as voted.



- i. If you click on vote button with input as your candidate's id for eg 1, a pop up will appear to confirm the transaction.
j. Once confirmed the vote will be registered.



- k. If you click on candidatesCount Button you will get the count of total candidates standing for election.

**Conclusion:**

Hence, we have successfully completed the installation of Ganache, Metamask and Remix IDE and deployed the smart contract and shown its output.

Practical- 7

Aim: To study solidity and implement some examples with it.

Theory:

Solidity:

Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state.

Solidity is a curly-bracket language. It is influenced by C++, Python and JavaScript, and is designed to target the Ethereum Virtual Machine (EVM). You can find more details about which languages Solidity has been inspired by in the language influences section.

Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features. With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

When deploying contracts, you should use the latest released version of Solidity. Apart from exceptional cases, only the latest version receives security fixes. Furthermore, breaking changes as well as new features are introduced regularly. We currently use a 0.y.z version number to indicate this fast pace of change.

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.

Benefits of smart contracts:

- **Speed, efficiency and accuracy:**

Once a condition is met, the contract is executed immediately. Because smart contracts are digital and automated, there's no paperwork to process and no time spent reconciling errors that often result from manually filling in documents.

- **Trust and transparency:**

Because there's no third party involved, and because encrypted records of transactions are shared across participants, there's no need to question whether information has been altered for personal benefit.

- **Security:**

Blockchain transaction records are encrypted, which makes them very hard to hack. Moreover, because each record is connected to the previous and subsequent records on a distributed ledger, hackers would have to alter the entire chain to change a single record.

- **Savings:**

Smart contracts remove the need for intermediaries to handle transactions and, by extension, their associated time delays and fees.

1) Write a program in solidity to create a structured student with Roll no, Name, Class, Department, Course enrolled as variables.

- a) Add information of 5 students.
- b) Search for a student using Roll no
- c) Display all information

Code:

```
prac7.sol
pragma solidity ^0.5.0;
pragma experimental ABIEncoderV2;
contract pract7 {
    struct Student {
        uint rn;
        string name;
        string class;
        string department;
        string course;
    }
    Student[] student;
    uint count;
    constructor() public{
        count=0;
    }
    function addStudentInfo(uint rollNumber, string memory name,
string memory
    class, string memory dept, string memory course) public {
        student.push(Student(rollNumber, name, class, dept,
course));
    }
    function getStudent(uint rollNumber) public view returns (uint,
string memory) {
        uint i=0;
        for(i=0;i<student.length;i++){
            if(student[i].rn==rollNumber){
                return (student[i].rn,student[i].name);
            }
        }
        return (student[0].rn,student[0].name);
    }
    function displayAllInfo() public view returns(Student[] memory){
        return student;
    }
}
```

Output:

Deploy the code

ENVIRONMENT

JavaScript VM (London)

ACCOUNT

0x5B3...eddC4 (99.9999999)

GAS LIMIT

3000000

VALUE

0

Wei

CONTRACT

pract7 - contracts/prac7.sol

Deploy

☐ Publish to IPFS

OR

At Address

Load contract from Address

Transactions recorded 7

Deployed Contracts

PRACT7 AT 0XD91...39138 (MEMORY)

PRACT7 AT 0XD91...39138 (MEMORY)

addStudentInfo

uint256 rollNumber, string name

displayAllInfo

getStudent

uint256 rollNumber

Low level interactions

CALLDATA

Transact

Page | 47

Add Student Info

▼ PRACT7 AT 0XD91...39138 (MEMORY)

addStudentInfo

rollNumber: 62

name: Akshay Vedak

class: A

dept: IT

course: MCA

transact

Display all information

displayAllInfo

```
0: tuple(uint256,string,string,string,string)[]: 62,Akshay Vedak,A,IT,MCA,22,Prathamesh Indulkar,A,IT,MCA,8,Siddhi Bhabal,A,IT,MCA,37,Ronak Pathare,B,IT,MCA,40,Shruti Naik,A,IT,MCA,45,Shraddha Shinde,A,IT,MCA
```

Get particular student info

getStudent

rollNumber: "62"

call

```
0: uint256: 62
1: string: Akshay Vedak
```

2) Create a structure Consumer with Name , Address, Consumer ID, Units and Amount as members. Write a program in solidity to calculate the total electricity bill according to the given condition:

For first 50 units Rs. 0.50/unit

For next 100 units Rs. 0.75/unit

For next 100 units Rs. 1.20/unit

For unit above 250 Rs. 1.50/unit

An additional surcharge of 20% is added to the bill. Display the information of 5 such consumers along with their units consumed and amount.

Code:

prac7b.sol

```
pragma solidity ^ 0.8 .0;
pragma experimental ABIEncoderV2;
contract pract7b
{
    struct Consumer
    {
        uint units;
        string name;
        string addr;
        uint consumerID;
        uint amount;
    }

    Consumer[] consumer;

    function addConsumerInfo(uint units, string memory name, string
memory addr,
        uint consumerID) public
    {
        consumer.push(Consumer(units, name, addr, consumerID, 0));
    }

    function getBillAmount() public
    {
        uint i = 0;
        uint amount = 0;
        uint surcharge = 0;
        uint 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(uint consumerID) public view
returns(uint, string memory, string memory, uint, uint)
{
    uint 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:

Deploy the code

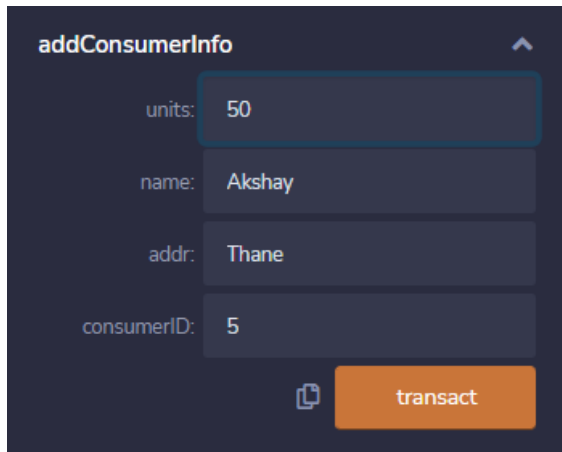
The screenshot displays a web3 deployment interface with the following sections:

- ENVIRONMENT:** JavaScript VM (London)
- ACCOUNT:** 0x5B3...eddC4 (99.99999999)
- GAS LIMIT:** 3000000
- VALUE:** 0 Wei
- CONTRACT:** pract7b - contracts/prac7b.sol
- Deploy** button
- ☐ Publish to IPFS
- OR**
- At Address** button (Load contract from Address)
- Transactions recorded** 3
- Deployed Contracts**
- PRACT7B AT 0X939...78492 (MEMORY)** (highlighted with a red box)

The expanded view of the deployed contract shows the following functions:

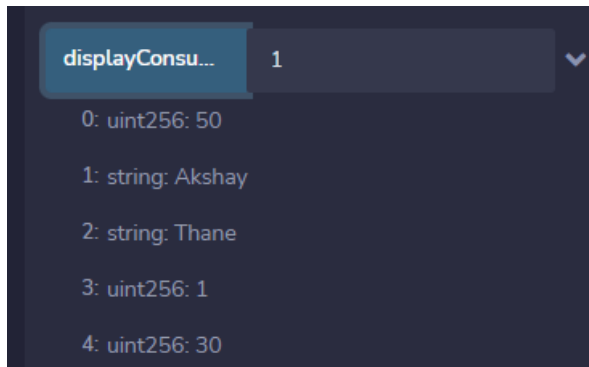
- addConsumerl...** (orange button) with input: "50", "Akshay", "Thane", "1"
- getBillAmount** (orange button)
- displayAllInfo** (blue button)
- displayConsu...** (blue button) with input: "1"

Add consumer information



The screenshot shows a web interface for the `addConsumerInfo` function. It features four input fields: `units` with the value 50, `name` with the value Akshay, `addr` with the value Thane, and `consumerID` with the value 5. Below the inputs is a blue button labeled `transact` and a small icon of a document with a plus sign.

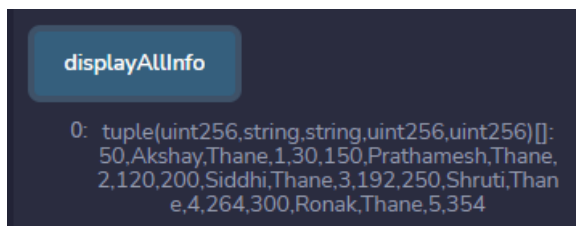
Get bill amount, display all information, display consumer by searching consumerID



The screenshot shows a web interface for the `displayConsumers` function. It has a dropdown menu with the value 1 selected. Below the dropdown, the following data is displayed:

- 0: uint256: 50
- 1: string: Akshay
- 2: string: Thane
- 3: uint256: 1
- 4: uint256: 30

Display information of all consumer



The screenshot shows a web interface for the `displayAllInfo` function. It displays a single line of data representing a tuple of five elements:

```
0: tuple(uint256,string,string,uint256,uint256):  
50,Akshay,Thane,1,30,150,Prathamesh,Thane,  
2,120,200,Siddhi,Thane,3,192,250,Shruti,Thane,  
4,264,300,Ronak,Thane,5,354
```

Conclusion:

Hence, we have successfully studied solidity and implemented some examples with 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:

```
$ 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.

Q1) Create a Bank Account contract and implement the following services:

- 1. Deposit**
- 2. Withdraw (keep a condition that only the owner of the contract can withdraw)**
- 3. Receive Ether**
- 4. Transfer Ether**
- 5. Check Balance**

Code:

1. Open remix ide:

FC1.sol

```
pragma solidity ^0.5.0;
contract FC1
{
    address owner;
    constructor() public
    {
        owner=msg.sender;
    }
    modifier ifOwner()
    {
        if(owner != msg.sender)
        {
            revert();
        }
        else
        {
            _;
        }
    }
    function receiveDeposit() payable public
    {
    }
    function getbalance() public view returns (uint)
    {
        return address(this).balance;
    }
    function withdraw(uint funds) public ifOwner
    {
        msg.sender.transfer(funds);
    }
}
```

Status and Output:

1. Deploy the code

ENVIRONMENT

JavaScript VM (London)

ACCOUNT

0x5B3...eddC4 (99.999999%)

GAS LIMIT

3000000

VALUE

0

Wei

CONTRACT

FC1 - fc.sol

Deploy

☐ Publish to IPFS

OR

At Address

Load contract from Address

Transactions recorded 1

Deployed Contracts

FC1 AT 0XD91...39138 (MEMORY)

FC1 AT 0XD91...39138 (MEMORY)

receiveDeposit

withdraw

uint256 funds

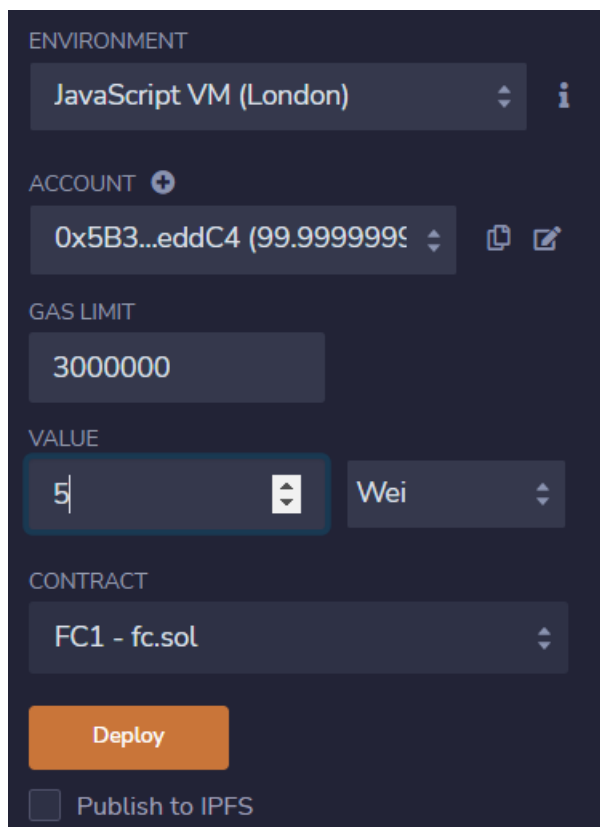
getbalance

Low level interactions

CALLDATA

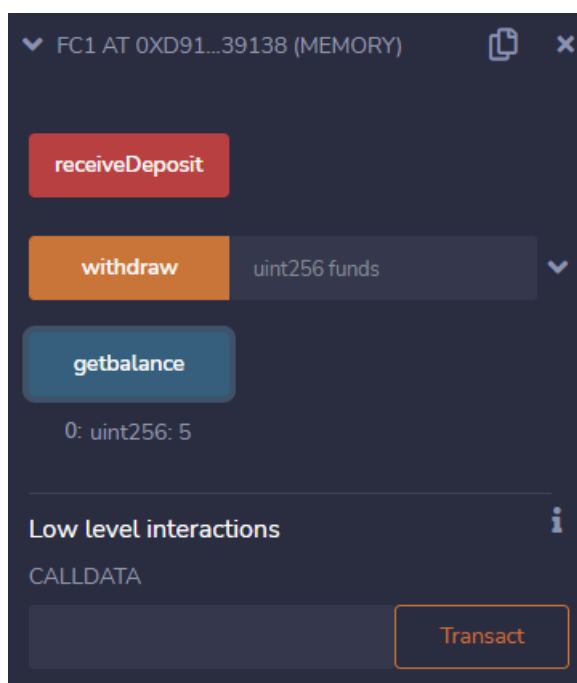
Transact

2. Enter value 5 wei and click on receiveDeposit



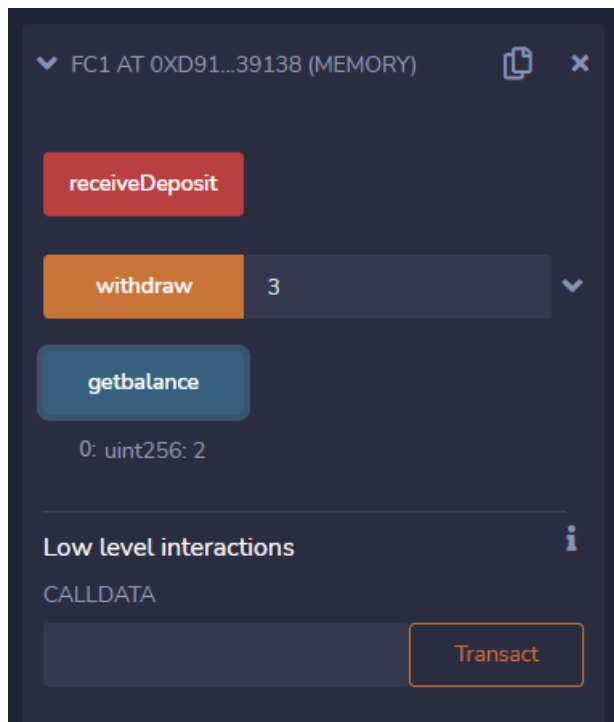
The screenshot shows the 'ENVIRONMENT' configuration panel in the Remix IDE. The 'ENVIRONMENT' dropdown is set to 'JavaScript VM (London)'. The 'ACCOUNT' dropdown shows '0x5B3...eddC4 (99.999999%)'. The 'GAS LIMIT' is set to '3000000'. The 'VALUE' input field contains '5' and the unit is set to 'Wei'. The 'CONTRACT' dropdown is set to 'FC1 - fc.sol'. There is a 'Deploy' button and a checkbox for 'Publish to IPFS'.

3. Click on getbalance



The screenshot shows the transaction panel for the 'FC1 AT 0XD91...39138 (MEMORY)' contract. It features three buttons: 'receiveDeposit' (red), 'withdraw' (orange), and 'getbalance' (blue). Below the buttons, the text '0: uint256: 5' is displayed. The 'withdraw' button has a dropdown menu showing 'uint256 funds'. At the bottom, there is a section for 'Low level interactions' with a 'CALLDATA' input field and a 'Transact' button.

4. Withdraw some amount and check balance



Q2) Create a Smart contract to simulate function overloading. Execute the contract using the truffle framework.

Steps:

1. Go to cmd prompt

```
$ mkdir blockchain-toolkit
$ cd blockchain-toolkit
$ truffle init
$ touch package.json
```

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

```
"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\" && sexit 1"
  },
  "author": "gregory@dappuniversity.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"
  }
}
"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": "gregory@dappuniversity.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"
  }
}
}

```

3. Save package.json file

4. npm install

5. Update the project config file(find truffle-config.js and paste the code)

```

module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*" // Match any network id
    },
    develop: {
      port: 8545
    }
  }
}

```

```
}  
};
```

6. Open Ganache network
7. Start developing a smart contract using solidity:

Code:

In contracts folder > overloading.sol

```
pragma solidity >=0.4.2 <=0.8.0;  
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);  
}  
}
```

In migration folder > 2_deploy_contracts.js

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

Output:

1. truffle compile

```
C:\Users\adved\blockchain-toolkit>truffle compile

Compiling your contracts...
=====
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\overloading.sol
> Artifacts written to C:\Users\adved\blockchain-toolkit\build\contracts
> Compiled successfully using:
  - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

2. truffle migrate

```
C:\Users\adved\blockchain-toolkit>truffle migrate

Compiling your contracts...
=====
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\overloading.sol
> Artifacts written to C:\Users\adved\blockchain-toolkit\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: 0x349660b0a73b04868b98823806e239760c2c00315bed326cfdc5d0e0c6ed8a88
> Blocks: 0        Seconds: 0
> contract address: 0xDBc7c4a25ea2BEA90911587250Bff47c43Bb4Dd5
> block number:     3
> block timestamp:   1639118540
> account:          0xD79C0F986664348E8CD6B38dD99f9BB18c102694
> balance:          99.99241192
> gas used:         145123 (0x236e3)
> gas price:        20 gwei
> value sent:       0 ETH
> total cost:       0.00290246 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:       0.00290246 ETH

Summary
=====
> Total deployments: 2
> Final cost:       0.00674132 ETH
```

3. truffle console

4. `overloading.deployed().then((instance) => {app = instance})`

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

5. `app.getSum(5,6)`

```
truffle(development)> app.getSum(5,6)
BN { negative: 0, words: [ 11, <1 empty item> ], length: 1, red: null }
```

6. `app.callSumWithTwoArguments()`

7. `app.callSumWithThreeArguments()`

```
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 }
```

Conclusion:

Hence, we have successfully executed the 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

Steps to follow:

1. `npm install -g truffle`

2. `truffle unbox react`

3. Start Development Environment

```
truffle develop
```

4. Compile and Migrate.**5. Run the Dapp**

```
npm run start
```

6. Modify the Dapp Code

In `App.js`, we get a reference to the `web3` object in the `React componentWillMount()` life cycle method and store it in the local state. We also instantiate a local version of the contract.

```
getWeb3
.then(results => {
  this.setState({
    web3: results.web3
  })

  // Instantiate contract once web3 provided.
  this.instantiateContract()
})
.catch(() => {
  console.log('Error finding web3.')
})
```

7. Add a Form

```
<form className="pure-form pure-form-stacked">
  <fieldset>
    <label htmlFor="storage">Storage Amount</label>
    <input id="storage" type="number" ref={c => {
      this.storageAmountInput = c }} /> <button
      className="pure-button"
      onClick={ (e) => {
        e.preventDefault();
        this.addToSimpleStorage()
      }}
    >
      Set Storage
    </button>
  </fieldset>
</form>
```


8. Add Action Button Code

```
addToSimpleStorage() {  
  if (this.state.simpleStorageInstance && this.state.accounts) {  
    const value = this.storageAmountInput.value;  
    this.state.simpleStorageInstance.set(value, {from:  
      this.state.accounts[0]}) .then((result) => {  
      return  
        this.state.simpleStorageInstance.get.call(this.state.accounts[0])  
        .then((result) => {  
          this.setState(prevState => ({  
            ...prevState,  
            storageValue: result.c[0]  
          }));  
        }).catch((err) => {  
          console.log('error');  
          console.log(err);  
        });  
    } else {  
      this.setState(prevState => ({  
        ...prevState,  
        error: new Error('simple storage instance not loaded')  
      })))  
    }  
  }  
}
```

9. Run the app

Conclusion:

Hence, we have successfully created Dapp with ethereum.

Mini Project
Title: Marketplace

Developer:
42, Ronak Pathare
60, Siddesh Waman

❖ Introduction:

A decentralized online marketplace is an eCommerce platform operated on a blockchain.

These are the distinguishing characteristics:

- No middleman controls the trade between buyers and sellers
- The terms of trade are determined by buyers and sellers, and no middlemen intervene
- The terms of trade are open, transparent, and immutable
- No tampering of transactions is possible
- Payment between buyers and sellers is direct, no intermediary has any role to play
- Transactions require no third-party payment system
- Transactions are instantaneous, without depending on a third-party
- It's a permanent marketplace that can't be destroyed
- No personal sensitive data is needed

❖ Existing System:

E-commerce provides tremendous benefits to traditional brick and mortar stores by allowing businesses to exponentially expand their reach and reach a much larger market.

However, with such convenience comes a cost; this cost is paid to the third party, which essentially acts as insurance in mediating the payment. eBay, Amazon, Ali Baba, and other well-known centralized e-commerce platforms are examples. The issue with these e-commerce websites is that they charge exorbitant fees.

Furthermore, because the dispute process is painfully slow, it could take weeks or even months to get your money back when a dispute arises.

❖ Problem Statement

There are several problems with existing centralized marketplaces:

1. The sellers are at the mercy of the company: due to their central position in the application infrastructure, the company can decide to block the merchant from transacting on their platform at their own discretion any time which decreases job security.
2. Sellers pay a fee to list their product and also pay commission on sales. This reduces profit margins or leads to increased prices for the buyer.
3. Platform users do not own any of their data. Reviews, purchase history and personal information are all owned by these companies. If a merchant wanted to move their operation to another provider, it may not be possible, as the platform may still own the previous user's data.

Because all of these problems include a centralized service that manages data as well as transactions and has power over both buyers and sellers, a decentralized system like blockchain can replace this middleman and fully automate this service, and in turn, drive down prices and protect user data.

The goal of this application is to build a decentralized marketplace infrastructure using smart contracts on the Ethereum blockchain that would solve all these problems. The application will allow a seller to list an item, multiple users to bid within a given time and the highest bidder to get the item. Once the buyer is satisfied, the bid amount is transferred to the seller's digital wallet. This framework can be specialized in the future for a specific market. The advantage over the traditional model is the lack of a middle man making profits and central points of failure. For example, a middle-man cannot block merchants from selling goods on the platform, and each user keeps their data to themselves, as ensured by the Ethereum framework.

❖ Proposed System:

Marketplace is not a fee-based intermediary platform for buyers and sellers; it simply provides them with a client. Sellers create a listing on the Marketplace, and other Marketplace buyer can view and buy it. Payments are only made in Ether. Once a product is sold it cannot be sold again.

❖ Technologies/Software Used:**▪ Node.js:**

This is used to install all the packages required for the project and also to handle the client-side of the project

▪ Truffle Suite:

It is framework for creating Ethereum smart contracts using solidity programming language write test against them and deploy them to a blockchain

▪ Ganache:

It is blockchain used for this project where you and use it without paying any thing for it.

▪ MetaMask:

It is an Ethereum Wallet that turns our browser into a blockchain browser because most of the do not support blockchain out of the box.

▪ React:

React is a JavaScript library for building user interfaces. React is used to build single-page applications. React allows us to create reusable UI components.

❖ Code:

Marketplace.sol:

```
pragma solidity ^0.5.0;

// contract that handles the entire business logic (buying and selling)
contract Marketplace {
    // state variable
    string public name;

    uint256 public productCount = 0;
    mapping(uint256 => Product) public products;

    struct Product {
        uint256 id;
        string name;
        uint256 price;
        address payable owner;
        // owner - person who owns the product when the product is
        // sold the owner gets changed to new buyer
        bool purchased; //to check if the product is already purchased or
not
    }

    event ProductCreated(
        uint256 id,
        string name,
        uint256 price,
        address payable owner,
        bool purchased
    );

    event ProductPurchased(
        uint256 id,
        string name,
        uint256 price,
        address payable owner,
        bool purchased
    );

    // constructor gets run 1 time only
    constructor() public {
        name = "Shruti Prathamesh Marketplace";
    }
}
```

```
function createProduct(string memory _name, uint256 _price) public {
    //making sure all parameters are correct
    // require valid name
    require(bytes(_name).length > 0);
    // require valid price
    require(_price > 0);

    //increase product count
    productCount++;
    //create product
    products[productCount] = Product(
        productCount,
        _name,
        _price,
        msg.sender,
        false
    );
    //trigger event - to tell blockchain something happened
    emit ProductCreated(productCount, _name, _price, msg.sender,
false);
}

// payable is a modifier
function purchaseProduct(uint256 _id) public payable {
    // fetch product
    Product memory _product = products[_id];
    // fetch owner
    address payable _seller = _product.owner;

    // make sure product is valid
    // 1. product has valid id
    require(_product.id > 0 && _product.id <= productCount);
    // 2. required that there is enough ether intransaction
    require(msg.value >= _product.price);
    // 3. product should not be already purchased (purchased should be
false)
    require(!_product.purchased);
    // 4. required that buyer is not seller
    require(_seller != msg.sender);

    // transfer ownership to buyer
    _product.owner = msg.sender;

    // mark as purchase
    _product.purchased = true;
```

```
// update product
products[_id] = _product;

// pay the seller by sending ether
address(_seller).transfer(msg.value);

// trigger an event
emit ProductPurchased(
    productCount,
    _product.name,
    _product.price,
    msg.sender,
    true
);
}
}
```

2_deploy_contracts.js:

```
const Marketplace = artifacts.require("Marketplace");

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

Marketplace.test.js:

```
const { assert } = require('chai')
const Marketplace = artifacts.require('./Marketplace.sol')
require('chai').use(require('chai-as-promised')).should()
contract('Marketplace', ([deployer, seller, buyer])=>{
    let marketplace
    before(async ()=>{
        marketplace = await Marketplace.deployed()
    })
    describe('deployment', async()=>{
        it('deploys successfully', async()=>{
            const address = await marketplace.address
            assert.notEqual(address, 0x0)
            assert.notEqual(address, '')
            assert.notEqual(address, null)
            assert.notEqual(address, undefined)
        })
    })
})
```



```
    it('has a name', async()=>{
      const name = await marketplace.name()
      assert.equal(name, 'Shruti Prathamesh Marketplace')
    })
  })

  describe('products', async()=>{
    let result, productCount
    before(async ()=>{
      // web3.utils.toWei('2', 'Ether')
      result = await marketplace.createProduct('Pixel 6',
web3.utils.toWei('1', 'Ether'), { from: seller})
      productCount = await marketplace.productCount()
    })

    // creating products
    it('creates products', async()=>{
      // success
      assert.equal(productCount, 1)
      // console.log(result.logs)
      const event = result.logs[0].args
      assert.equal(event.id.toNumber(), productCount.toNumber(), 'id
is Correct')
      assert.equal(event.name , 'Pixel 6' , 'name is correct')
      assert.equal(event.price , '1000000000000000000' , 'price is
correct')
      assert.equal(event.owner , seller , 'owner is correct')
      assert.equal(event.purchased , false , 'purchased is correct')

      // failed
      // 1. product name wrong
      await await marketplace.createProduct('',
web3.utils.toWei('1', 'Ether'), { from: seller}).should.be.rejected;
      // 2. product price wrong
      await await marketplace.createProduct('Pixel 6 ', 0 , { from:
seller}).should.be.rejected;
    })

    // listing our products
    it('lists products', async()=>{
      const product = await marketplace.products(productCount)

      assert.equal(product.id.toNumber(), productCount.toNumber(),
'id is Correct')
      assert.equal(product.name , 'Pixel 6' , 'name is correct')
```

```
    assert.equal(product.price , '100000000000000000' , 'price is
correct')
    assert.equal(product.owner , seller , 'owner is correct')
    assert.equal(product.purchased , false , 'purchased is
correct')
  })

  // selling product
  it('sells product', async()=>{

    // track sellers balance before purchase
    let oldSellerBalance
    oldSellerBalance = await web3.eth.getBalance(seller)
    oldSellerBalance = new web3.utils.BN(oldSellerBalance)

    // success: buyer makes purchase
    result = await marketplace.purchaseProduct(productCount,
{from: buyer, value: web3.utils.toWei('1', 'Ether')}});
    // checking logs
    const event = result.logs[0].args
    assert.equal(event.id.toNumber(), productCount.toNumber(), 'id
is Correct')
    assert.equal(event.name , 'Pixel 6' , 'name is correct')
    assert.equal(event.price , '1000000000000000000' , 'price is correct')
    assert.equal(event.owner , buyer , 'owner is correct')
    assert.equal(event.purchased , true , 'purchased is correct')

    // checking seller receives funds
    let newSellerBalance
    newSellerBalance = await web3.eth.getBalance(seller)
    newSellerBalance = new web3.utils.BN(newSellerBalance)

    let price
    price = web3.utils.toWei('1', 'Ether')
    price = new web3.utils.BN(price)

    // console.log(newSellerBalance)
    // console.log(oldSellerBalance)
    // console.log(price);

    const expectedBalance = oldSellerBalance.add(price)
    // let x = expectedBalance.toString()
    // console.log(x);
    assert.equal(newSellerBalance.toString(),
expectedBalance.toString())
```

```
        // failed
        // 1. trying to buy product that does not exist (product
should have a valid id)
        await marketplace.purchaseProduct(99, { from: buyer, value:
web3.utils.toWei('1', 'Ether') }).should.be.rejected;
        // 2. buyer tries to buy without enough ethers
        await marketplace.purchaseProduct(productCount, { from: buyer,
value: web3.utils.toWei('0.5', 'Ether') }).should.be.rejected;
        // 3. deployer tries to buy the product, i.e product cant be
purchased twice (fake product)
        await marketplace.purchaseProduct(productCount, { from:
deployer, value: web3.utils.toWei('1', 'Ether') }).should.be.rejected;
        // 4. buyer tries to buy again, i.e buyer cant be the seller
        await marketplace.purchaseProduct(productCount, { from: buyer,
value: web3.utils.toWei('1', 'Ether') }).should.be.rejected;

    })
  })
})
```

Client Side:

App.js:

```
import React, { Component } from 'react';
import Web3 from 'web3';
import logo from '../logo.png';
import './App.css';
import Navbar from './Navbar';
import Main from './Main'
import Marketplace from '../abis/Marketplace.json'

class App extends Component {

  // lifecycle component - anytime the component gets called this function
will be executed first
  async componentWillMount(){
    await this.loadWeb3()
    // console.log(window.web3);
    await this.loadBlockchainData()
  }

  async loadWeb3(){
    if (window.ethereum) {
      window.web3 = new Web3(window.ethereum);
      await window.ethereum.enable();
    }
  }
}
```

```
    }
    else if (window.web3) {
        window.web3 = new Web3(window.web3.currentProvider);
    }
    else {
        window.alert('Non-Ethereum browser detected. You should consider
trying MetaMask!');
    }
}

async loadBlockchainData(){
    const web3 = window.web3
    // load acct
    const accounts = await web3.eth.getAccounts()
    // console.log(accounts);
    this.setState({account: accounts[0]})
    // console.log(Marketplace.abi, Marketplace.networks[5777].address);
    const networkId = await web3.eth.net.getId()
    // console.log(networkId);
    const networkData = Marketplace.networks[networkId]
    if(networkData){
        // const abi = Marketplace.abi
        // const address = Marketplace.networks[networkId].address
        // const marketplace = new web3.eth.Contract(abi, address)
        // console.log(marketplace);
        const marketplace = new web3.eth.Contract(Marketplace.abi,
networkData.address)
        // console.log(marketplace);
        this.setState({ marketplace })
        const productCount = await marketplace.methods.productCount().call()
        // console.log(productCount);
        this.setState({productCount})
        // fetching products
        for(var i = 1; i<= productCount; i++)
        {
            const product = await marketplace.methods.products(i).call()
            this.setState({
                products: [...this.state.products, product]
            })
        }
        this.setState({ loading: false })
        // console.log(this.state.products);
    }
    else{
        window.alert("Marketplace Contract not deployed to detected
network")
    }
}
```

```

    }
  }

  constructor(props){
    super(props)
    this.state = {
      account: '',
      productCount: 0,
      products: [],
      loading: true
    }
    this.createProduct = this.createProduct.bind(this)
    this.purchaseProduct = this.purchaseProduct.bind(this)
  }
  createProduct(name, price){
    this.setState({ loading : true })
    this.state.marketplace.methods.createProduct(name, price).send({ from:
this.state.account })
    .once('receipt', (receipt) => {
      this.setState({loading: false})
    })
  }
  purchaseProduct(id, price){
    this.setState({loading: true})
    this.state.marketplace.methods.purchaseProduct(id).send({from:
this.state.account, value: price})
    .once('receipt', (receipt)=>{
      this.setState({loading: false})
    })
  }
  render() {
    return (
      <div>
        <Navbar account={this.state.account}/>
        <div className="container-fluid mt-5">
          <div className="row">
            <main role="main" className="col-lg-12 d-flex justify-content-
center text-center">
              { this.state.loading
                ? <div id="loader" className="text-center"><p
className="text-center">Loading ..</p></div>
                : <Main
                  products = {this.state.products}
                  createProduct = {this.createProduct }
                  purchaseProduct = {this.purchaseProduct }
                />

```

```
    }  
    </main>  
  </div>  
</div>  
</div>  
);  
}}  
export default App;
```

Navbar.js:

```
import React, { Component } from 'react';  
  
class Navbar extends Component {  
  render() {  
    return (  
      <nav className="navbar navbar-dark fixed-top bg-dark flex-md-  
nowrap p-0 shadow">  
        <a  
          className="navbar-brand col-sm-3 col-md-2 mr-0"  
          href="http://localhost:3000/"  
          rel="noopener noreferrer"  
        >  
          Marketplace  
        </a>  
        <ul className="navbar-nav px-3">  
          <li className="nav-item text-nowrap d-none d-sm-none d-sm-  
block">  
            <small className="text-white"><span  
id="account">{this.props.account}</span></small>  
            </li>  
          </ul>  
        </nav>  
      );  
    }  
  }  
}  
  
export default Navbar;
```

Main.js:

```
import React, { Component } from 'react';
class Main extends Component {
  render() {
    return (
      <div id="content">
        <h1>Add Products</h1>

        <form onSubmit = { (event) => {
          event.preventDefault()
          const name = this.productName.value
          const price =
window.web3.utils.toWei(this.productPrice.value.toString(), 'Ether')
          this.props.createProduct(name, price)
        } }>
          <div className="form-group mr-sm-2">
            <input
              id="productName"
              type="text"
              ref = { (input)=> { this.productName = input} }
              className="form-control"
              placeholder="Product Name"
              required />
          </div>
          <div className="form-group mr-sm-2">
            <input
              id="productPrice"
              type="text"
              ref = { (input)=> { this.productPrice = input} }
              className="form-control"
              placeholder="Product Price"
              required />
          </div>
          <button type="submit" className="btn btn-primary">Add
Product</button>
        </form>
        <p> </p>
        <h2>Buy Product</h2>
        <table className="table">
          <thead>
            <tr>
              <th scope="col">Sr.No</th>
              <th scope="col">Name</th>
              <th scope="col">Price</th>
              <th scope="col">Owner</th>
            </tr>
          </thead>
        </table>
      </div>
    )
  }
}
```

```

        <th scope="col"></th>
      </tr>
    </thead>
    <tbody id="productList">
      { this.props.products.map((product,key)=>{
        return(
          <tr key={key}>
            <th scope="row">{product.id.toString()}</th>
            <td>{product.name}</td>
            <td>{window.web3.utils.fromWei(product.price.to
oString(), 'Ether')} Eth</td>
            <td>{product.owner}</td>
            <td>
              {
                !product.purchased
                ?
                  <button
                    name={product.id}
                    value={product.price}
                    onClick={(event)=>{
                      // console.log('Clicked');
                      this.props.purchaseProduct(event.target.name, event.target.value)
                    }}
                  >
                    Buy
                  </button>
                : <strong>Already Purchased</strong>
              }
            </td>
          </tr>
        )
      })}
    </tbody>
  </table>
</div>
);
}
}
export default Main;

```


❖ **Output:**

1. truffle compile:

```
D:\mca\SEM 3\BC\Blockchain-Marketplace>truffle compile
Compiling .\src\contracts\Marketplace.sol...
Writing artifacts to .\src\abis
```

2. Open Ganache and start a new Workspace

The screenshot shows the Ganache application window. The top navigation bar includes tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this, a status bar displays various metrics: CURRENT BLOCK (17), GAS PRICE (2000000000), GAS LIMIT (6721975), HARDFORK (MUIRGLACIER), NETWORK ID (5777), RPC SERVER (http://127.0.0.1:7545), MINING STATUS (AUTOMINING), and WORKSPACE (MARKETPLACE). The main area shows the MNEMONIC 'judge primary noble develop pudding pet three skill owner enter chicken diagram' and the HD PATH 'm/44'/60'/0'/0/account_index'. Below this is a table of accounts.

ADDRESS	BALANCE	TX COUNT	INDEX
0x94E9D4Abe504c04252d5eE75c93E36315931F281	99.94 ETH	7	0
0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01	102.99 ETH	5	1
0x376D31c7D9fEE6C40D937711bFF82ecbFE9D05ED	97.00 ETH	5	2
0x794F362CCddE2425b8a26c5165D87f69c788826d	100.00 ETH	0	3
0x20c74F780a266CD9eDa81E5195a9f5B3a37503B6	100.00 ETH	0	4
0x6d007bfC23436F3A69BEca6412E8D3a9337f31bf	100.00 ETH	0	5
0x04fE6a569e43863DdbBe9688DB8CC90e02C50EeB	100.00 ETH	0	6

3. truffle test:

```
D:\mca\SEM 3\BC\Blockchain-Marketplace>truffle test
Using network 'development'.

Contract: Marketplace
  deployment
    ✓ deploys successfully
    ✓ has a name (217ms)
  products
    ✓ creates products (608ms)
    ✓ lists products (264ms)
    ✓ sells product (2307ms)

5 passing (5s)
```

4. truffle migrate:

```
D:\mca\SEM 3\BC\Blockchain-Marketplace>truffle migrate
ⓂⓂ Important ⓂⓂ
If you're using an HDWalletProvider, it must be Web3 1.0 enabled or your migration will hang.

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

1_initial_migration.js
=====

  Deploying 'Migrations'
  -----
  > transaction hash: 0x300ac03be365bd19a76007af918a1e3f6d2151d3051f3ed57509019a869492fc
  > Blocks: 0        Seconds: 0
  > contract address: 0x94075c1ddf48e17C2C55599aa8c32C43E0c33F61
  > account:          0xcBFaE7796d5BECA55B42285D34698016a1800EfE
  > balance:          99.93450928
  > gas used:          244636
  > gas price:         20 gwei
  > value sent:        0 ETH
  > total cost:        0.00489272 ETH

  > Saving artifacts
  -----
  > Total cost:        0.00489272 ETH

2_deploy_contracts.js
=====

  Replacing 'Marketplace'
  -----
  > transaction hash: 0x4121a5ee3aa849931c9b086c304fbbe43e1bc81a03efe7da253d776a78343118
  > Blocks: 0        Seconds: 0
  > contract address: 0x6Ad2a548910e7F5B0066164d8FFA75161AD25A3f
  > account:          0xcBFaE7796d5BECA55B42285D34698016a1800EfE
  > balance:          99.9195866
  > gas used:          746134
  > gas price:         20 gwei
  > value sent:        0 ETH
  > total cost:        0.01492268 ETH

  > Saving artifacts
  -----
  > Total cost:        0.01492268 ETH

Summary
=====
> Total deployments: 2
> Final cost:        0.0198154 ETH
```

5. Run the server to open frontend:

npm run start:

```
D:\mca\SEM 3\BC\Blockchain-Marketplace>npm run start

> eth-marketplace@0.1.0 start D:\mca\SEM 3\BC\Blockchain-Marketplace
> react-scripts start
Starting the development server...

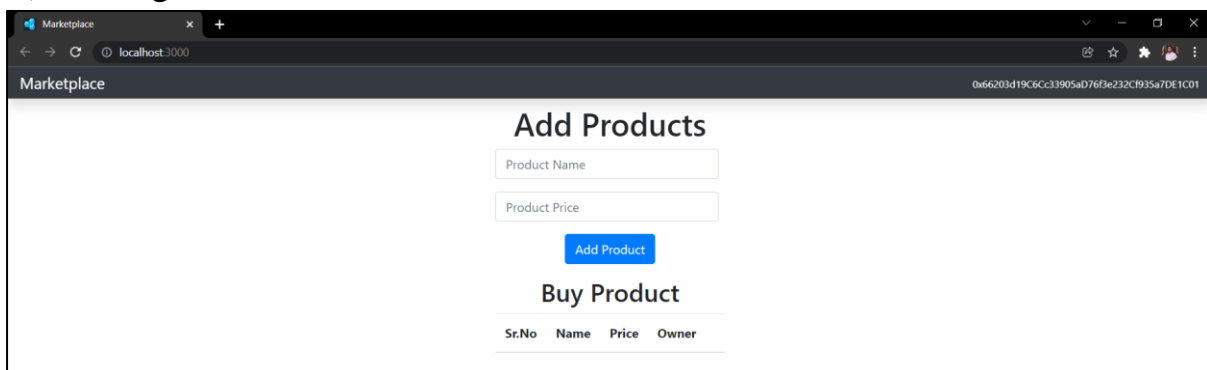
Browserslist: caniuse-lite is outdated. Please run next command `npm update`
Browserslist: caniuse-lite is outdated. Please run next command `npm update`
Compiled with warnings.

./src/components/App.js
  Line 3: 'logo' is defined but never used  no-unused-vars

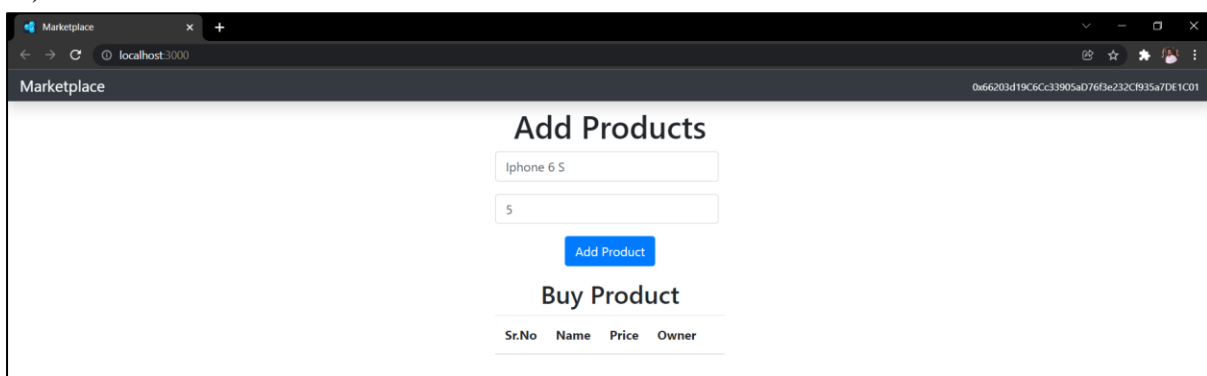
Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

6. On Chrome:

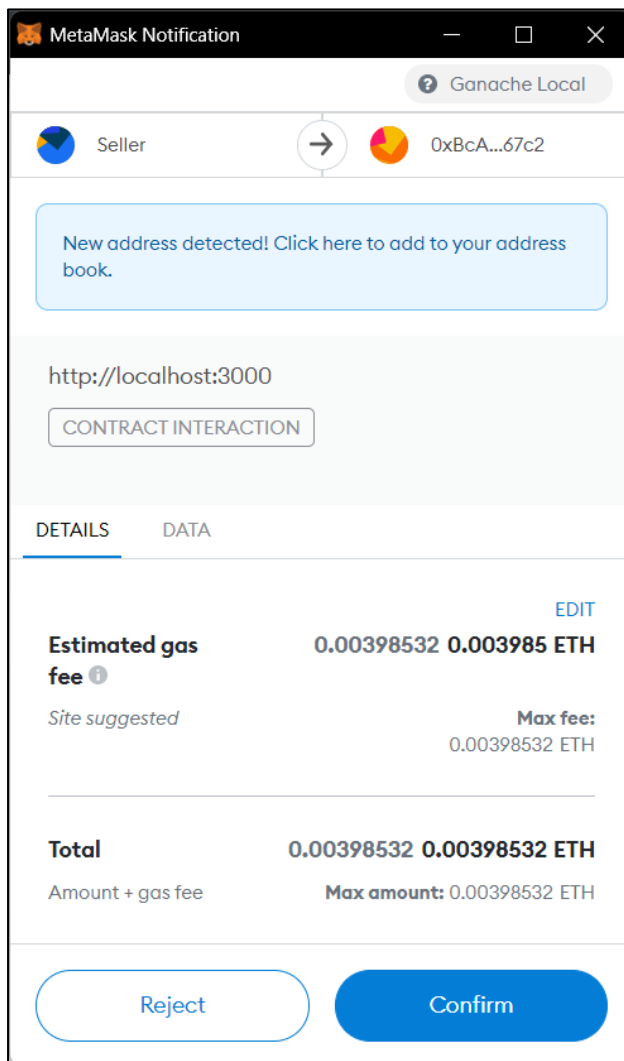
a) Through Sellers Account



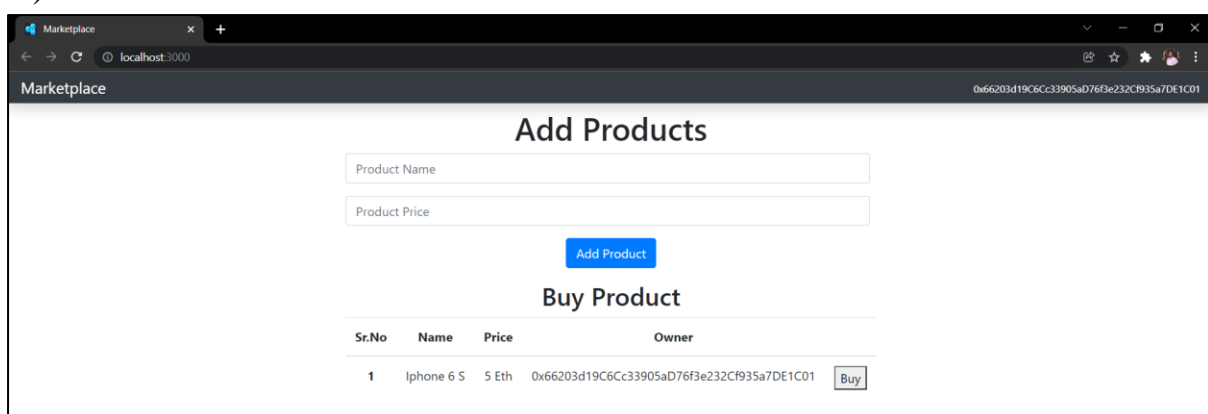
b) Add Product:



c) Click Confirm



d) Product added:



e) Add 2 more products

The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page title is 'Marketplace'. The main content area has a dark header with the title 'Marketplace' and a wallet address '0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01'. Below the header, there is a section titled 'Add Products' with two input fields: 'Product Name' and 'Product Price', and a blue 'Add Product' button. Below this is a section titled 'Buy Product' containing a table with the following data:

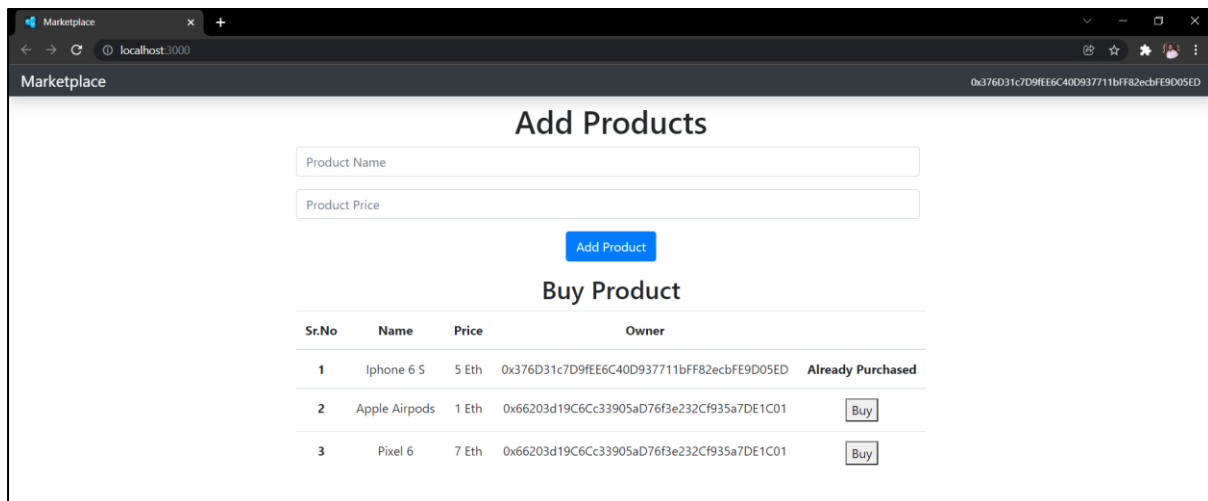
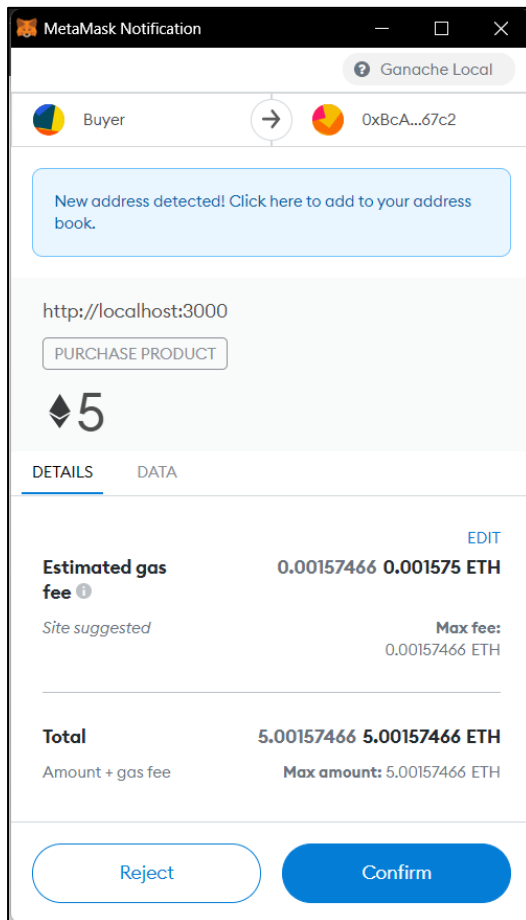
Sr.No	Name	Price	Owner	
1	Iphone 6 S	5 Eth	0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01	<button>Buy</button>
2	Apple Airpods	1 Eth	0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01	<button>Buy</button>
3	Pixel 6	7 Eth	0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01	<button>Buy</button>

f) **Change Metamask account to Buyers:**
On Chrome: Through Buyers Account

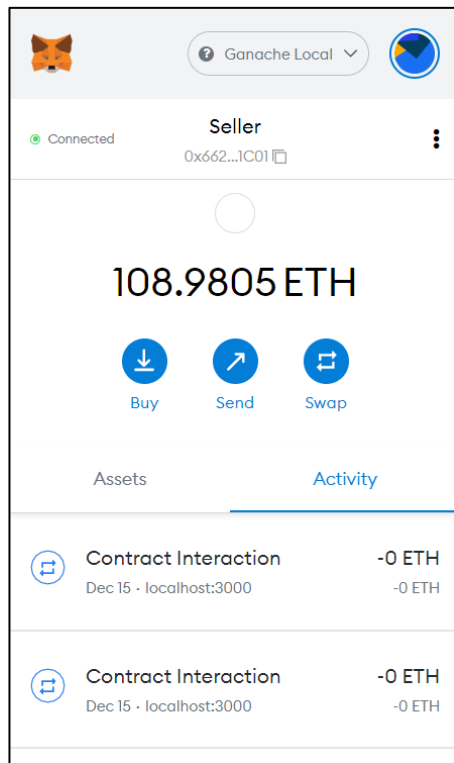
The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page title is 'Marketplace'. The main content area has a dark header with the title 'Marketplace' and a wallet address '0x376D31c7D9fEE6C40D937711bFF82ecbfE9D05ED'. Below the header, there is a section titled 'Add Products' with two input fields: 'Product Name' and 'Product Price', and a blue 'Add Product' button. Below this is a section titled 'Buy Product' containing a table with the following data:

Sr.No	Name	Price	Owner	
1	Iphone 6 S	5 Eth	0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01	<button>Buy</button>
2	Apple Airpods	1 Eth	0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01	<button>Buy</button>
3	Pixel 6	7 Eth	0x66203d19C6Cc33905aD76f3e232Cf935a7DE1C01	<button>Buy</button>

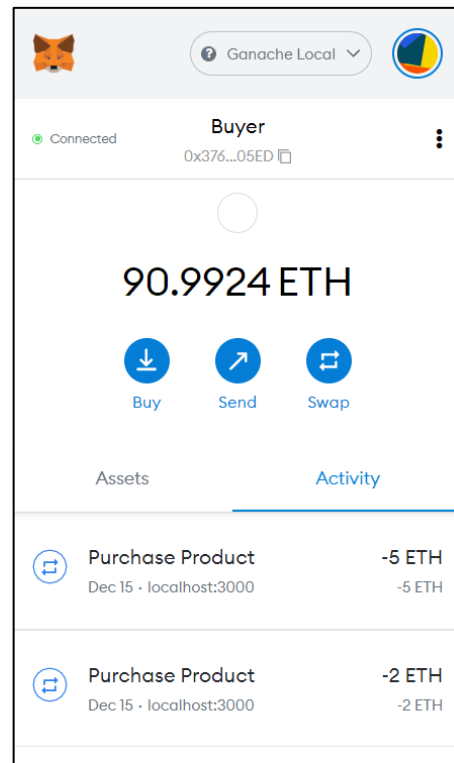
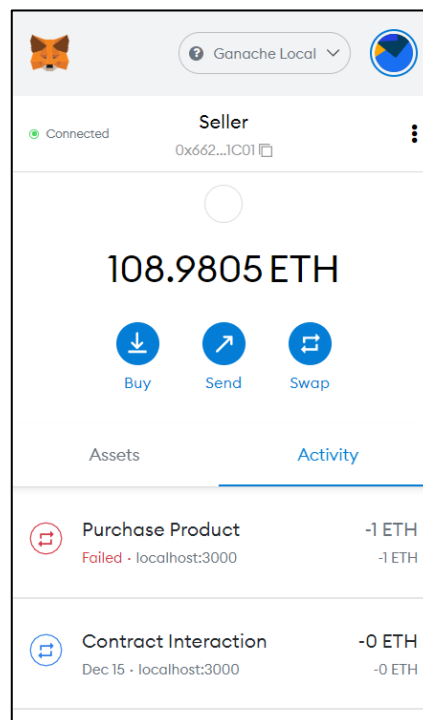
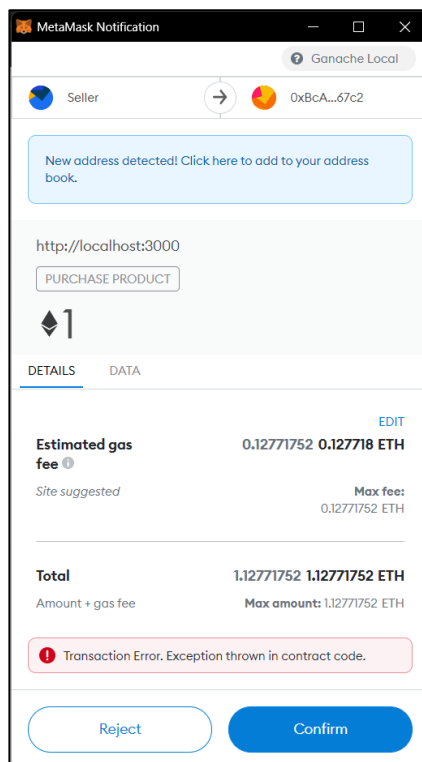
g) Buy Iphone 6s: Owner address changes to Buyers and cannot be purchased again



h) Sellers Metamask Account Balance



i) Buyers Metamask Account Balance

j) Owner trying to buy the product: **Failure**

❖ Conclusion:

We have successfully created Marketplace using Blockchain.

With regards to the problem definition stated earlier, our application can be considered successful as:

- 1) Through the use of smart contracts, sellers cannot be discriminated against by the system, as the rules for the system are set in stone by the smart contract.
- 2) As an added benefit, we have publicly verifiable transparency of the system rules. Moreover, the fact that the transaction data are stored on the blockchain simplifies the audit procedure.
- 3) It is shown in the cost analysis of the system that it is not expensive for a seller to list an item or sell the item, thereby driving prices for buyers down.
- 4) The Ethereum framework preserves the security of user data, ensuring that no users can acquire another user's information

❖ Reference:

- <https://www.devteam.space/blog/how-to-build-online-marketplace-on-blockchain-like-openbazaar/>
- [https://nsl.cse.unt.edu/sites/default/files/biblio/documents/Decentralized Marketplace Ethereum Blockchain.pdf](https://nsl.cse.unt.edu/sites/default/files/biblio/documents/Decentralized_Marketplace_Ethereum_Blockchain.pdf)
- <https://ieeexplore.ieee.org/document/8730733>