Dr. D. Y. Patil Pratishthan's

DR. D. Y. PATIL INSTITUTE OF ENGINEERING, MANAGEMENT & RESEARCH

Approved by A.I.C.T.E, New Delhi , Maharashtra State Government, Affiliated to Savitribai Phule Pune University

Sector No. 29, PCNTDA , Nigidi Pradhikaran, Akurdi, Pune 411044. Phone: 020–27654470, Fax: 020-27656566 Website : www.dypiemr.ac.in Email : principal.dypiemr@gmail.com

---

DEPARTMENT OF COMPUTER ENGINEERING

LAB MANUAL

Laboratory Practice III

Blockchain Technology (Group C)

(Final Year Engineering)

Semester – I

Assignment No: 1

Title of the Assignment: Installation of MetaMask and study spending Ether per transaction

Objective of the Assignment: Students should be able to learn new technology such as metamask.Its application and implementations

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Contents for Theory:

1. **Introduction Blockchain**
2. **Cryptocurrency**
3. **Transaction Wallets**
4. **Ether transaction**
5. **Installation Process of Metamask**

Introduction to Blockchain

- Blockchain can be described as a data structure that holds transactional records and while ensuring security, transparency, and decentralization. You can also think of it as a chain or records stored in the forms of blocks which are controlled by no single authority.
- A blockchain is a distributed ledger that is completely open to any and everyone on the network. Once an information is stored on a blockchain, it is extremely difficult to change or alter it.
- Each transaction on a blockchain is secured with a digital signature that proves its authenticity. Due to the use of encryption and digital signatures, the data stored on the blockchain is tamper-proof and cannot be changed.
- Blockchain technology allows all the network participants to reach an agreement, commonly known as consensus. All the data stored on a blockchain is recorded digitally and has a common history which is available for all the network participants. This way, the chances of any fraudulent activity or duplication of transactions is eliminated without the need of a third-party.

Blockchain Features

The following features make the revolutionary technology of blockchain stand out:

- **Decentralized**

  Blockchains are decentralized in nature meaning that no single person or group holds the authority of the overall network. While everybody in the network has the copy of the distributed ledger with them, no one can modify it on his or her own. This unique feature of blockchain allows transparency and security while giving power to the users.

- **Peer-to-Peer Network**

  With the use of Blockchain, the interaction between two parties through a peer-to-peer model is easily accomplished without the requirement of any third party. Blockchain uses P2P protocol which allows all the network participants to hold an identical copy of transactions, enabling approval through a machine consensus. For example, if you wish to make any transaction from one part of the world to another, you can do that with blockchain all by yourself within a few seconds. Moreover, any interruptions or extra charges will not be deducted in the transfer.

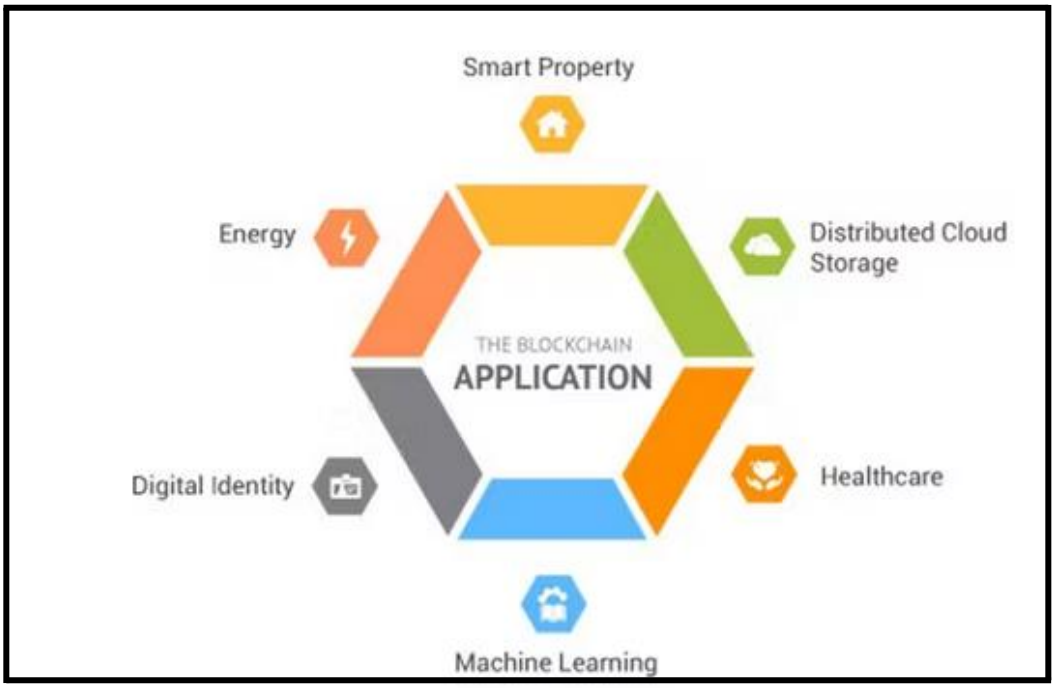
- **Immutable**

  The immutability property of a blockchain refers to the fact that any data once written on the blockchain cannot be changed. To understand immutability, consider sending email as an example. Once you send an email to a bunch of people, you cannot take it back. In order to find a way around, you'll have to ask all the recipients to delete your email which is pretty tedious. This is how immutability works.

- **Tamper-Proof**

  With the property of immutability embedded in blockchains, it becomes easier to detect tampering of any data. Blockchains are considered tamper-proof as any change in even one single block can be detected and addressed smoothly. There are two key ways of detecting tampering namely, hashes and blocks.

Popular Applications of Blockchain Technology



Benefits of Blockchain Technology:

- **Time-saving:** No central Authority verification needed for settlements making the process faster and cheaper.
- **Cost-saving:** A Blockchain network reduces expenses in several ways. No need for third-party verification. Participants can share assets directly. Intermediaries are reduced. Transaction efforts are minimized as every participant has a copy of shared ledger.
- **Tighter security:** No one can temper with Blockchain Data as it is shared among

millions of participants. The system is safe against cybercrimes and Fraud.

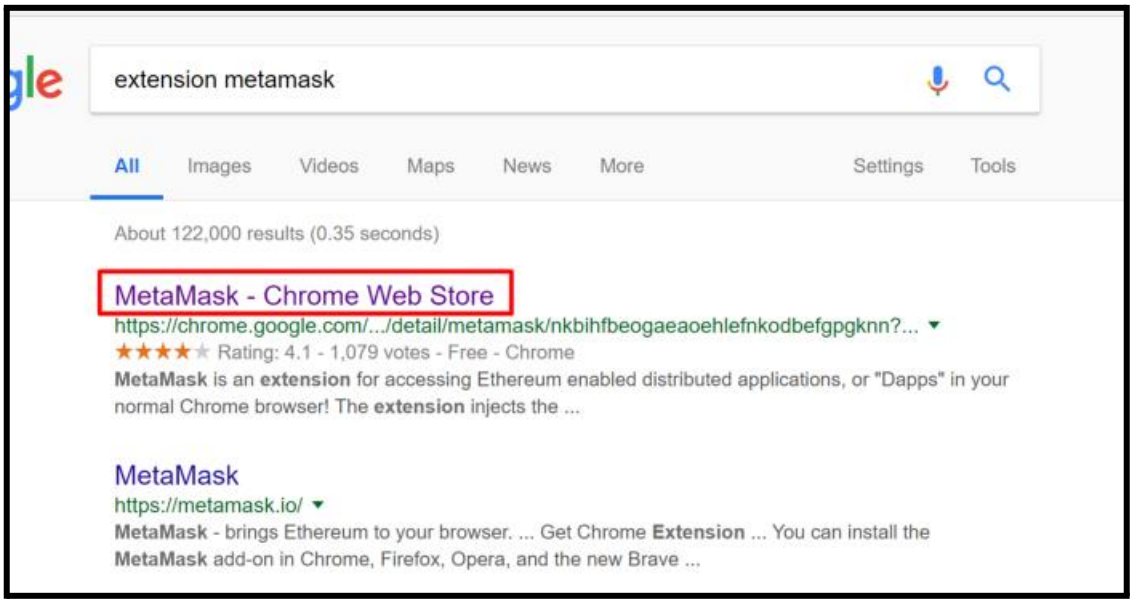
- In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

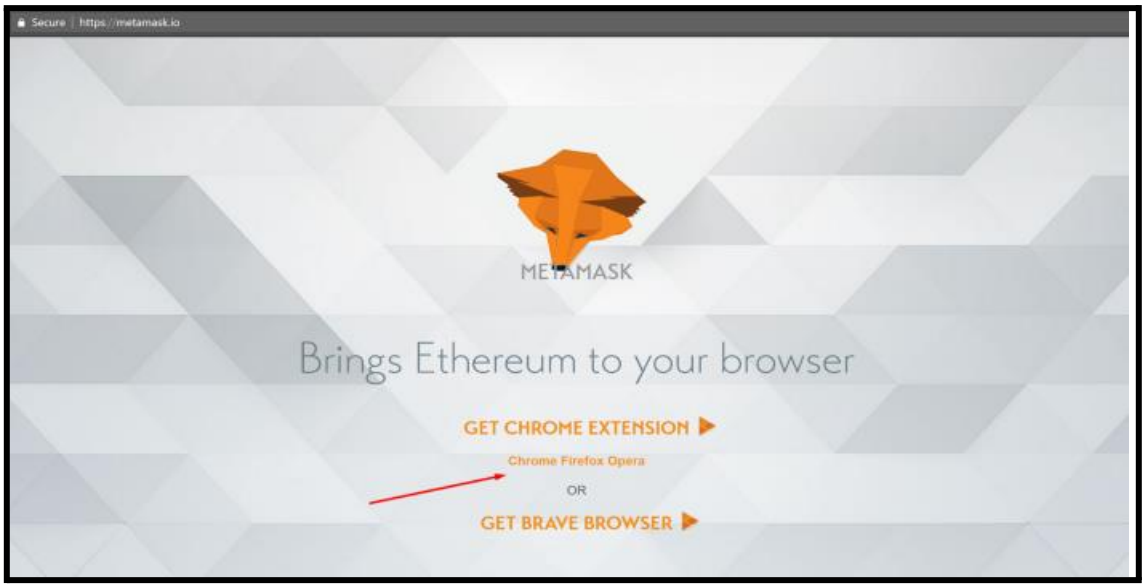
How to use MetaMask: A step by step guide

MetaMask is one of the most popular browser extensions that serves as a way of storing your Ethereum and other ERC-20 Tokens. The extension is free and secure, allowing web applications to read and interact with Ethereum's blockchain.

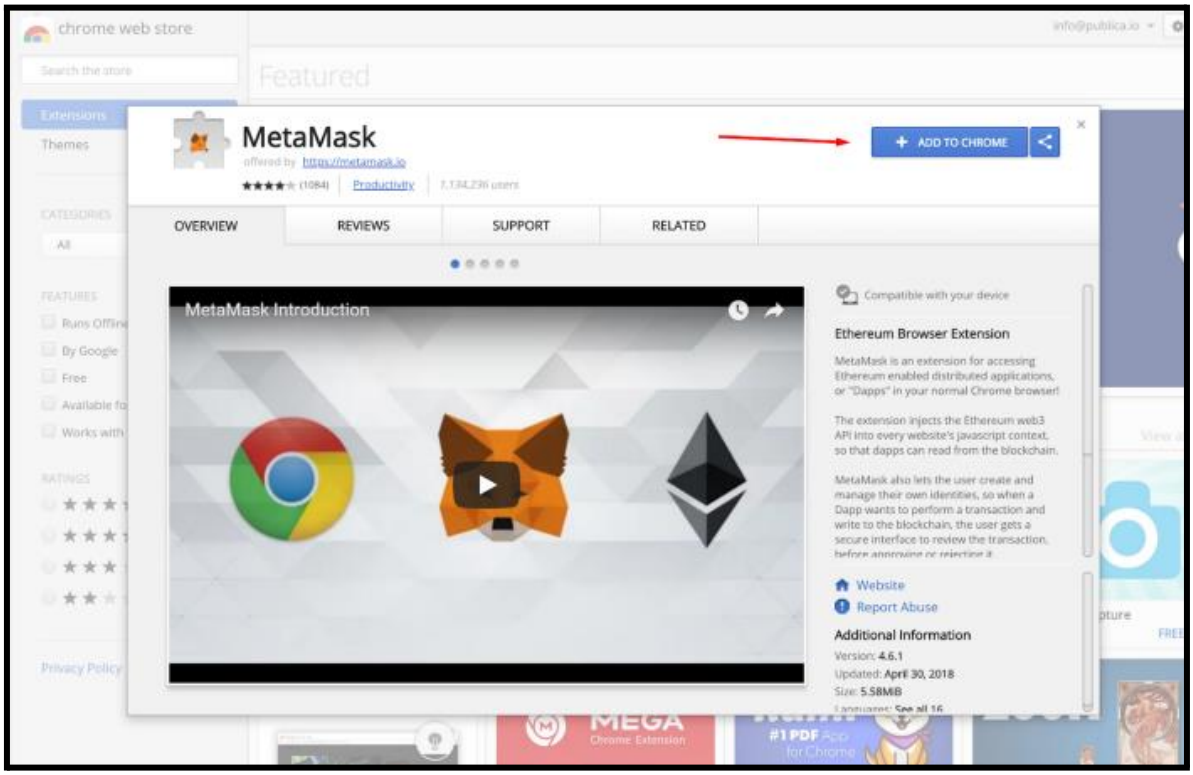
Step 1. Install MetaMask on your browser.

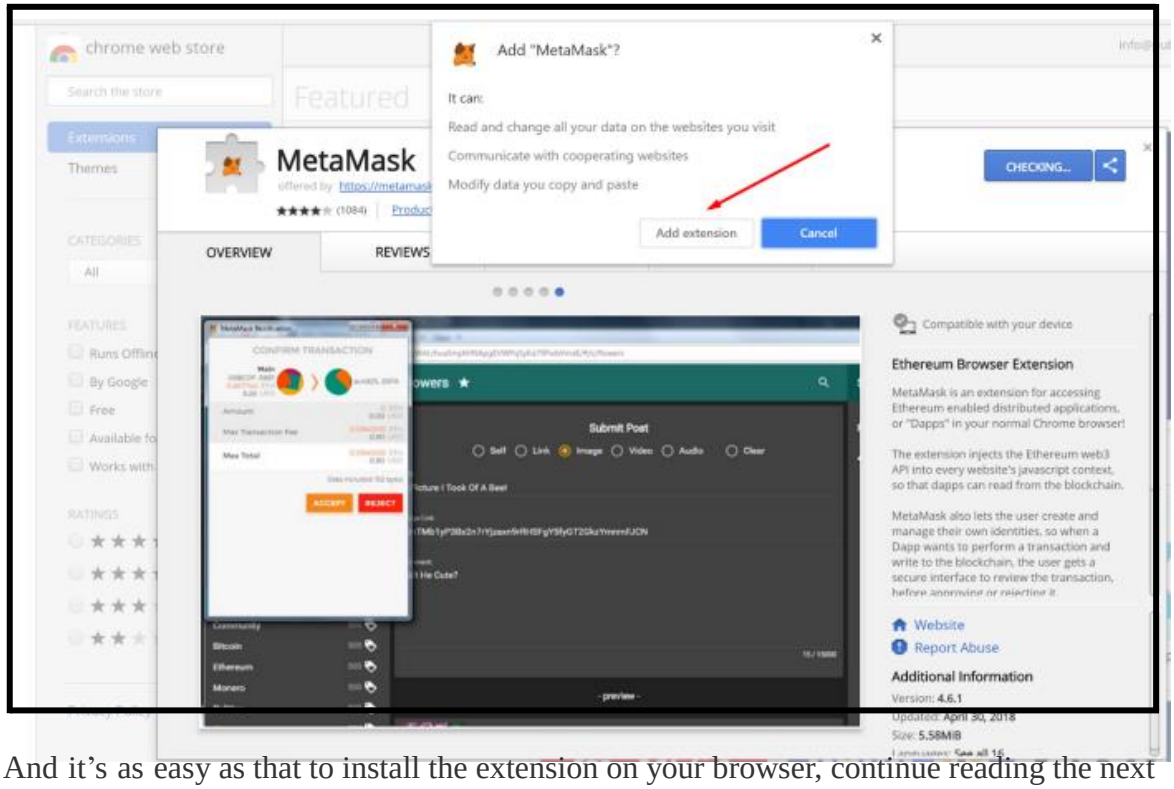
To create a new wallet, you have to install the extension first. Depending on your browser, there are different marketplaces to find it. Most browsers have MetaMask on their stores, so it's not that hard to see it, but either way, here they are Chrome , Firefox, and Opera .





- Click on **Install MetaMask** as a Google Chrome extension.
- Click **Add to Chrome**.
- Click **Add Extension**.

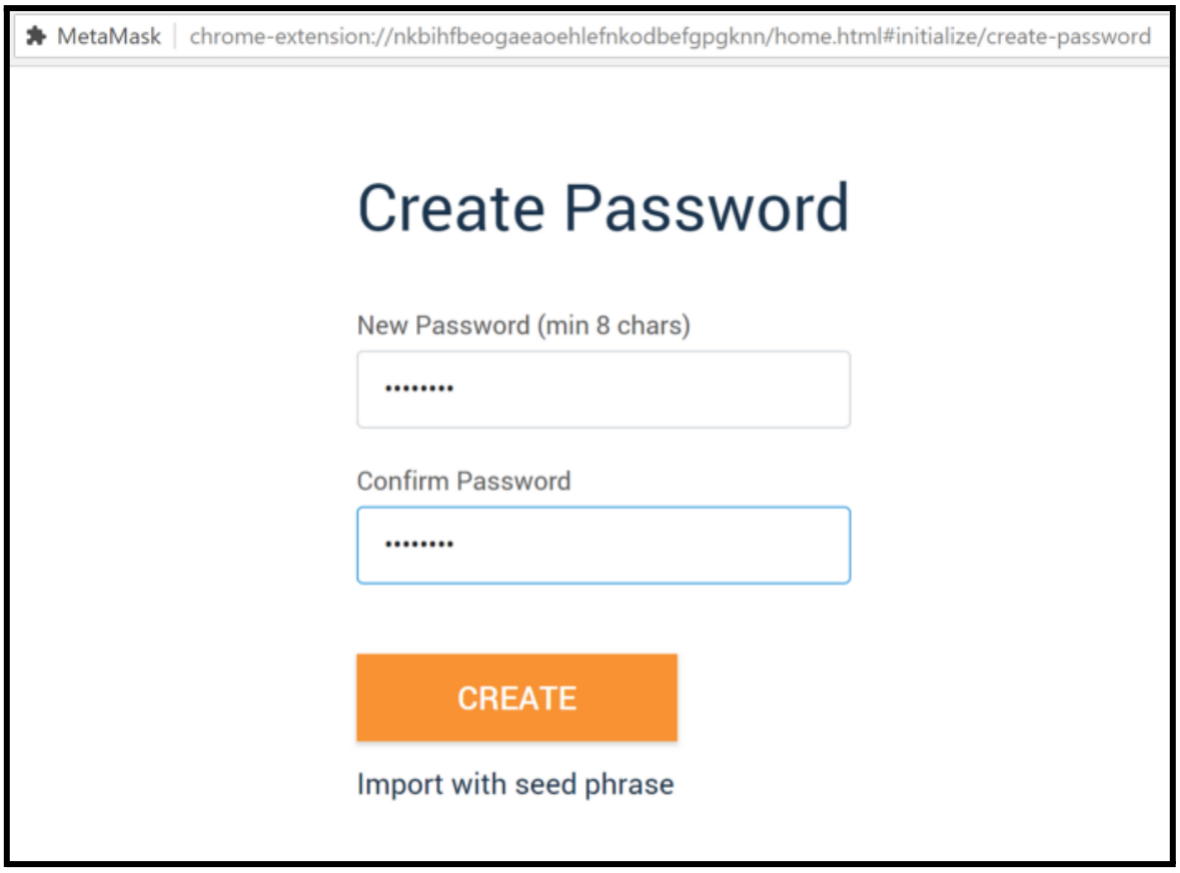




And it's as easy as that to install the extension on your browser, continue reading the next step to figure out how to create an account.

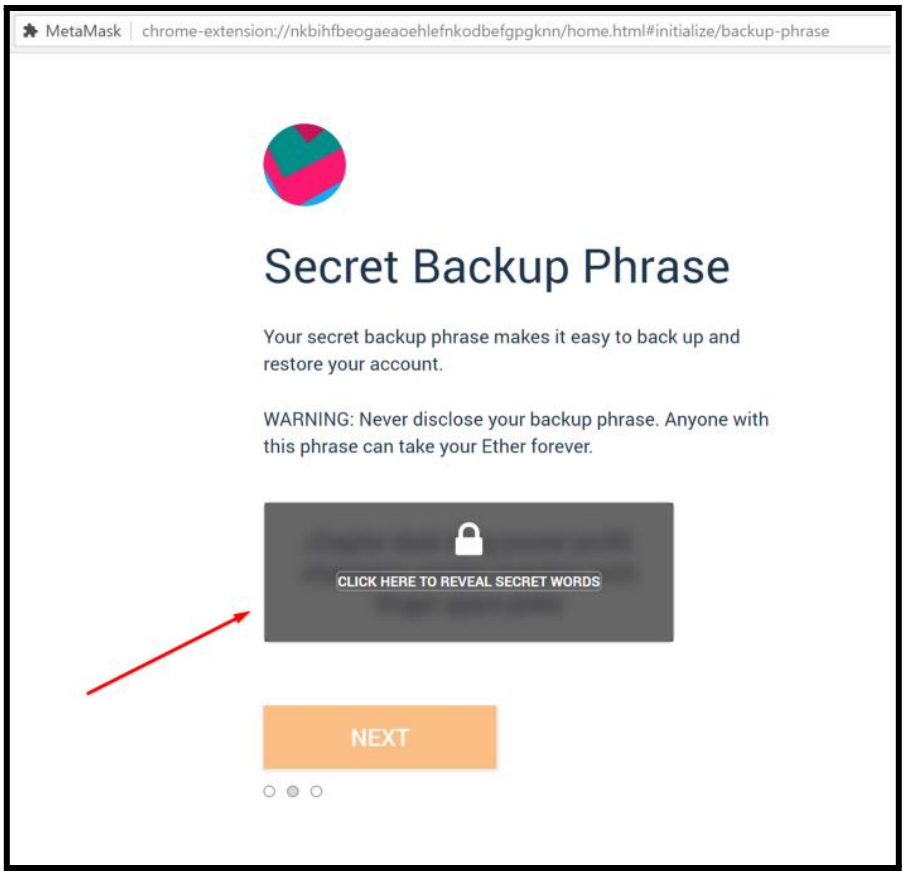
Step 2. Create an account.

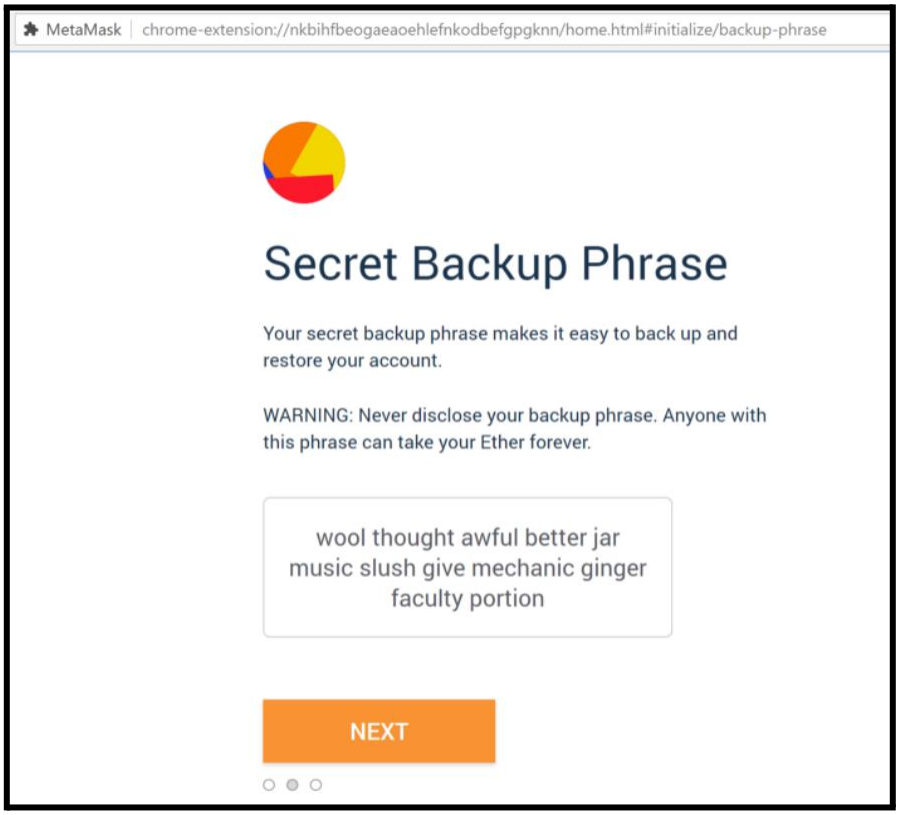
- Click on the extension icon in the upper right corner to open MetaMask.
- To install the latest version and be up to date, **click Try it now**.
- **Click Continue**.
- You will be prompted to create a new password. **Click Create**.



- Proceed by **clicking Next** and accept the Terms of Use.

Click Reveal Secret Words. There you will see a 12 words seed phrase. This is really important and usually not a good idea to store digitally, so take your time and write it down





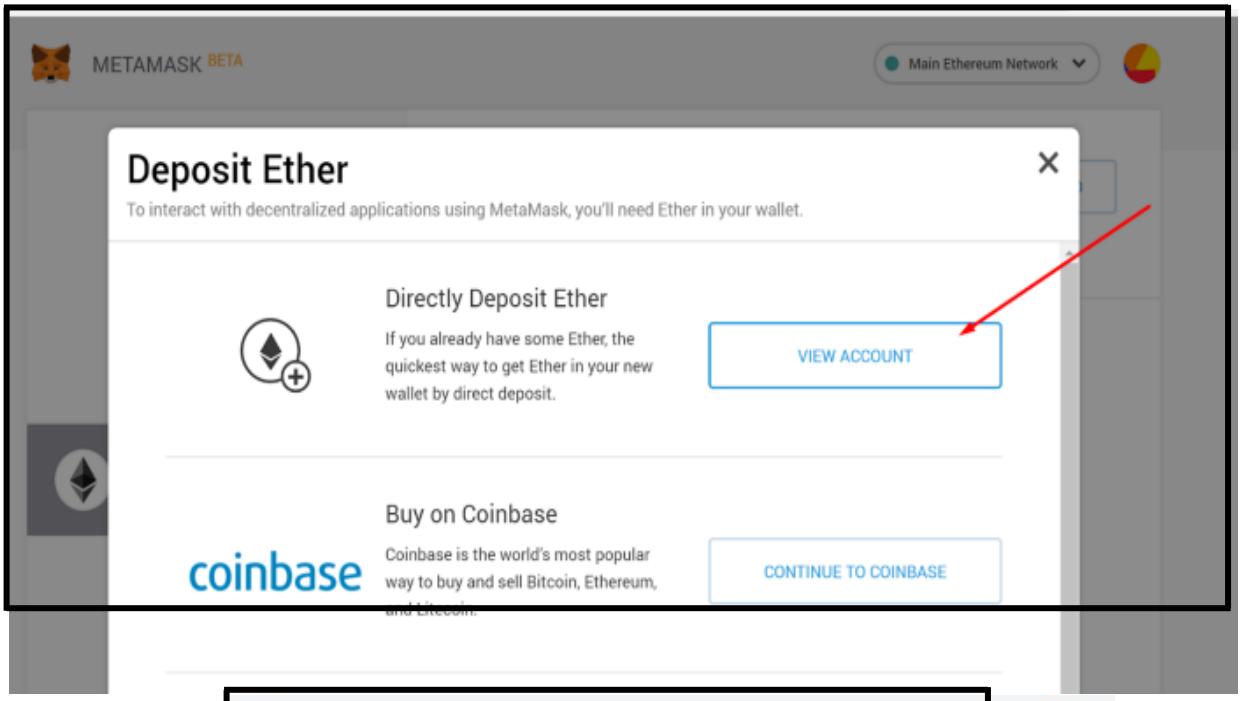
- Verify your secret phrase by selecting the previously generated phrase in order. **Click Confirm**.

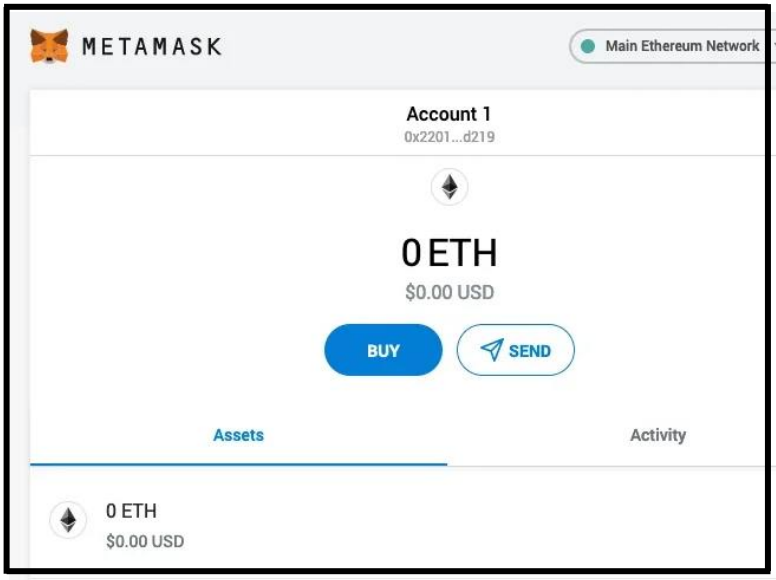
And that's it; now you have created your MetaMask account successfully. A new Ethereum wallet address has just been created for you. It's waiting for you to deposit funds, and if you want to learn how to do that, look at the next step below.

Step 3. Depositing funds.

- Click on **View Account**.

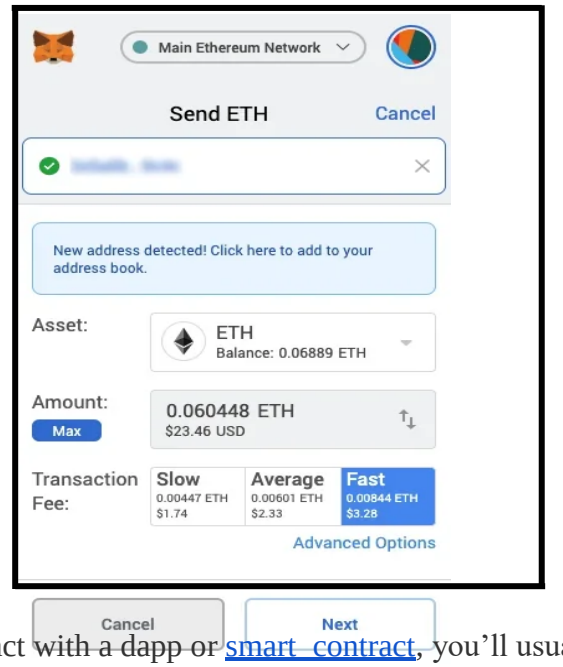
You can now see your public address and share it with other people. There are some methods to buy coins offered by MetaMask, but you can do it differently as well; you just need your address. If you ever get logged out, you'll be able to log back in again by clicking the MetaMask icon, which will have been added to your web browser (usually found next to the URL bar).





You can now access your list of assets in the ‘Assets’ tab and view your transaction history in the ‘Activity’ tab.

- Sending crypto is as simple as clicking the ‘Send’ button, entering the recipient address and amount to send, and selecting a transaction fee. You can also manually adjust the transaction fee using the ‘Advanced Options’ button, using information from ETH Gas Station or similar platforms to choose a more acceptable gas price.
- After clicking ‘Next’, you will then be able to either confirm or reject the transaction on the subsequent page.



- To use MetaMask to interact with a dapp or smart contract, you’ll usually need to find a ‘Connect to Wallet’ button or similar element on the platform you are trying to use. After clicking this, you should then see a prompt asking whether you want to let the dapp connect to your wallet.

What advantages does MetaMask have?

- **Popular** - It is commonly used, so users only need one plugin to access a wide range of dapps.
- **Simple** - Instead of managing private keys, users just need to remember a list of words, and transactions are signed on their behalf.
- **Saves space** - Users don’t have to download the Ethereum blockchain, as MetaMask sends requests to nodes outside of the user’s computer.
- **Integrated** - Dapps are designed to work with MetaMask, so it becomes much easier to send

Ether in and out.

Conclusion- In this way we have explored Concept Blockchain and metamat wallet for transaction of digital currency

Assignment Question

1. **What Are the Different Types of Blockchain Technology?**
2. **What Are the Key Features/Properties of Blockchain?**
3. **What Type of Records You Can Keep in A Blockchain?**
4. **What is the difference between Ethereum and Bitcoin?**
5. **What are Merkle Trees? Explain their concept.**
6. **What is Double Spending in transaction operation**
7. **Give real-life use cases of blockchain.**

Reference link

- https://hackernoon.com/blockchain-technology-explained-introduction-meaning-and-applications-edbd6759a2b2
- https://levelup.gitconnected.com/how-to-use-metamask-a-step-by-step-guide-f380a3943fb1
- https://decrypt.co/resources/metamask

Assignment No: 2

Title of the Assignment: Create your own wallet using Metamask for crypto transactions

Objective of the Assignment: Students should be able to learn about cryptocurrencies and learn how transaction done by using different digital currency

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Contents for Theory:

1. **Cryptocurrency**
2. **Transaction Wallets**
3. **Ether transaction**

Introduction to Cryptocurrency

- Cryptocurrency is a digital payment system that doesn't rely on banks to verify transactions. It's a peer-to-peer system that can enable anyone anywhere to send and receive payments. Instead of being physical money carried around and exchanged in the real world, cryptocurrency payments exist purely as digital entries to an online database describing specific transactions. When you transfer cryptocurrency funds, the transactions are recorded in a public ledger. Cryptocurrency is stored in digital wallets.
- Cryptocurrency received its name because it uses encryption to verify transactions. This means advanced coding is involved in storing and transmitting cryptocurrency data between wallets and to public ledgers. The aim of encryption is to provide security and safety.
- The first cryptocurrency was Bitcoin, which was founded in 2009 and remains the best known today. Much of the interest in cryptocurrencies is to trade for profit, with speculators at times driving prices skyward.

How does cryptocurrency work?

- Cryptocurrencies run on a distributed public ledger called block chain, a record of all transactions updated and held by currency holders.
- Units of cryptocurrency are created through a process called mining, which involves using computer power to solve complicated mathematical problems that generate coins. Users can also buy the currencies from brokers, then store and spend them using cryptographic wallets.
- If you own cryptocurrency, you don't own anything tangible. What you own is a key that allows you to move a record or a unit of measure from one person to another without a trusted third party.
- Although Bitcoin has been around since 2009, cryptocurrencies and applications of blockchain technology are still emerging in financial terms, and more uses are expected in the future. Transactions including bonds, stocks, and other financial assets could eventually be traded using the technology.

Cryptocurrency examples

There are thousands of cryptocurrencies. Some of the best known include:

- **Bitcoin:**
  Founded in 2009, Bitcoin was the first cryptocurrency and is still the most commonly traded. The currency was developed by Satoshi Nakamoto – widely believed to be a pseudonym for an individual or group of people whose precise identity remains unknown.

- **Ethereum:**
  Developed in 2015, Ethereum is a blockchain platform with its own cryptocurrency, called Ether (ETH) or Ethereum. It is the most popular cryptocurrency after Bitcoin.

- **Litecoin:**
  This currency is most similar to bitcoin but has moved more quickly to develop new innovations, including faster payments and processes to allow more transactions.

- **Ripple:**
  Ripple is a distributed ledger system that was founded in 2012. Ripple can be used to track different kinds of transactions, not just cryptocurrency. The company behind it has worked with various banks and financial institutions.

- Non-Bitcoin cryptocurrencies are collectively known as “altcoins” to distinguish them from the original.

How to store cryptocurrency

- Once you have purchased cryptocurrency, you need to store it safely to protect it from hacks or theft. Usually, cryptocurrency is stored in crypto wallets, which are physical devices or online software used to store the private keys to your cryptocurrencies securely. Some exchanges provide wallet services, making it easy for you to store directly through the platform. However, not all exchanges or brokers automatically provide wallet services for you.
- There are different wallet providers to choose from. The terms “hot wallet” and “cold wallet” are used:
- **Hot wallet storage:** "hot wallets" refer to crypto storage that uses online software to protect the private keys to your assets.
- **Cold wallet storage:** Unlike hot wallets, cold wallets (also known as hardware wallets) rely on offline electronic devices to securely store your private keys.

Conclusion- In this way we have explored Concept Cryptocurrency and learn how transactions are done using digital currency

Assignment Question

1. **What is Bitcoin?**
2. **What Are the biggest Four common cryptocurrency scams**
3. **Explain How safe are money e-transfers?**
4. **What is cryptojacking and how does it work?**

Reference link

- https://www.kaspersky.com/resource-center/definitions/what-is-cryptocurrency

Assignment No: 3

Title of the Assignment: Write a smart contract on a test network, for Bank account of a customer for following operations: Deposit money Withdraw Money Show balance

Objective of the Assignment: Students should be able to learn about smart contract for banking application and able to perform some operation like Deposit money Withdraw Money Show balance

Prerequisite:

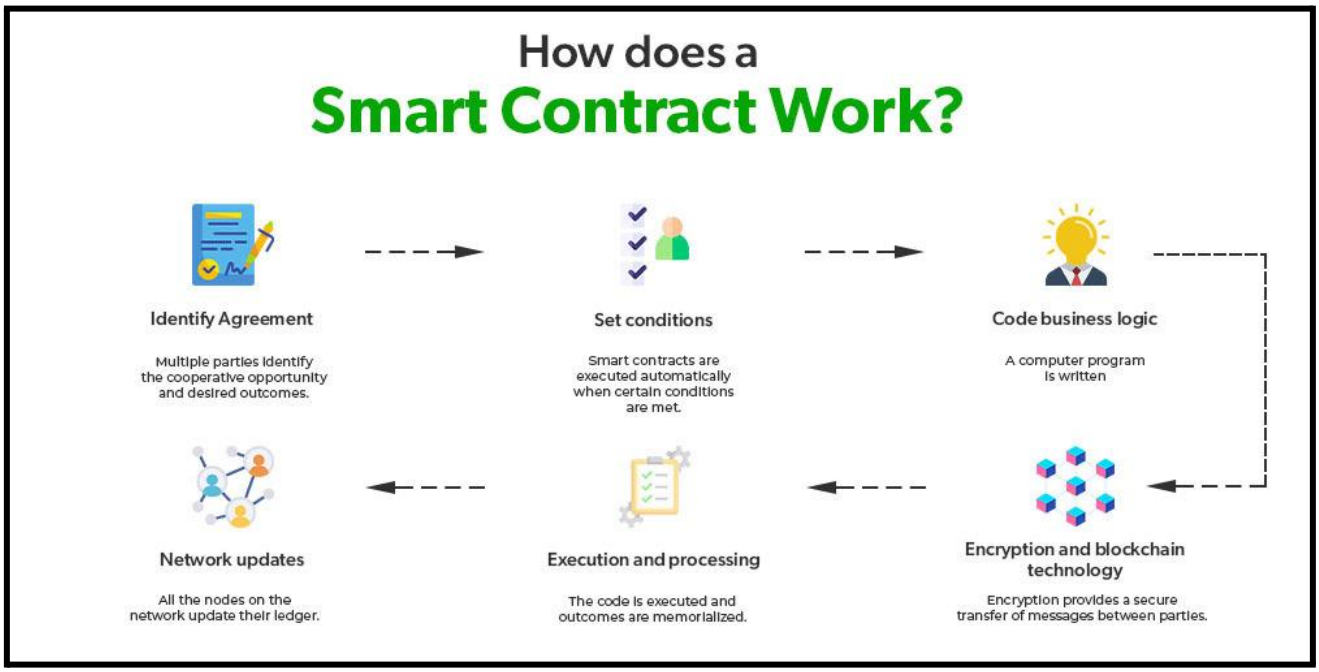
1. Basic knowledge of smart contract
2. Remix IDE
3. Basic knowledge of solidity

Contents for Theory:

1. **Smart contract**
2. **Remix IDE**
3. **Solidity Basics**
4. **Ether transaction**

Introduction to Smart Contract

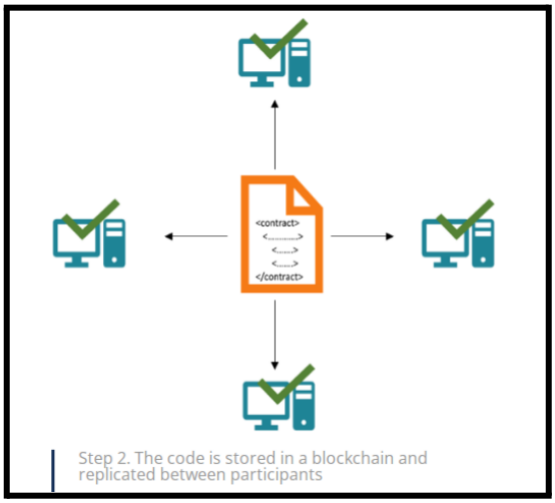
- Smart contracts are self-executing lines of code with the terms of an agreement between buyer and seller automatically verified and executed via a computer network.
- Nick Szabo, an American computer scientist who invented a virtual currency called "Bit Gold" in 1998,1 defined smart contracts as computerized transaction protocols that execute terms of a contract.2
- Smart contracts deployed to blockchains render transactions traceable, transparent, and irreversible.



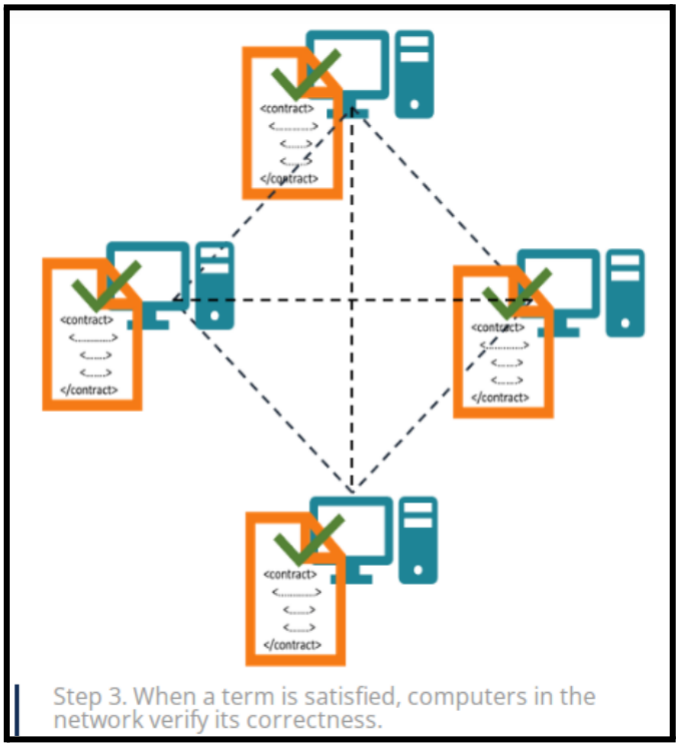
How does Smart Contract work?

- First, the contractual parties should determine the terms of the contract. After the contractual terms are finalized, they are translated into programming code. Basically, the code represents a number of different conditional statements that describe the possible scenarios of a future transaction.

- When the code is created, it is stored in the blockchain network and is replicated among the participants in the blockchain.



- Then, the code is run and executed by all computers in the network. If a term of the contract is satisfied and it is verified by all participants of the blockchain network, then the relevant transaction is executed.

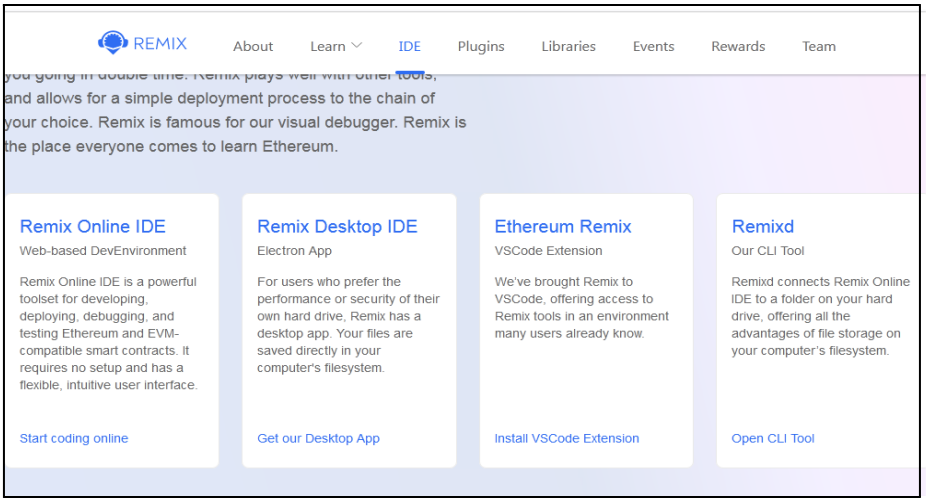


Remix IDE : Installation steps

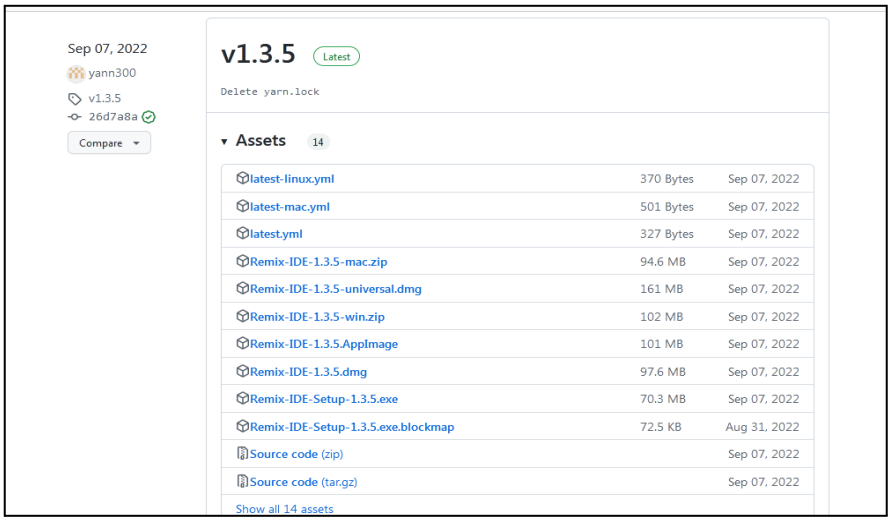
- First and foremost, head on over to the release page of the official Remix Desktop repository and grab the binary suited for your host system. This will be the .exe file for Windows users,

the .dmg for macOS and the .deb for Debian-derivative GNU/Linux systems. For Ubuntu and other AppImage setups, the .AppImage will be what you are looking for.

- Go to https://remix-project.org/
- Go to IDE option .Select Desktop IDE



- Select The file As per your choice



- Install Executable file,by double clicking on file(Applicable to .exe file Windows OS)

What is Solidity?

- Solidity is a rather simple language deliberately created for a simplistic approach to tackle real world solutions. Gavin Wood initially proposed it in August of 2014. Several developers of the Ethereum chain such as Christian Reitwiessner, Alex Beregszaszi, Liana Husikyan,

Yoichi Hirai and many more contributed to creating the language. The Solidity language can be executed on the Ethereum platform, that is a primary Virtual Machine implementing the blockchain network to develop decentralized public ledgers to create smart contract systems.

Solidity Basics

- To get started with the language and learn the basics let's dive into coding. We will begin by understanding the syntax and general data types, along with the variable data types. Solidity supports the generic value types, namely:
- Booleans: Returns value as either true or false. The logical operators returning Boolean data types are as follows:
- ! Logical negation
- && logical conjunction, "and"
- || logical disjunction, "or"
- == equality
- != inequality

Integers: Solidity supports int/unit for both signed and unsigned integers respectively. These storage allocations can be of various sizes. Keywords such as uint8 and uint256 can be used to allocate a storage size of 8 bits to 256 bits respectively. By default, the allocation is 256 bits. That is, uint and int can be used in place of uint256 and int256. The operators compatible with integer data types are:

- Comparisons: <=, <, ==, !=, >=, >. These are used to evaluate to bool.
- Bit operators: &, |, ^ bitwise exclusive 'or', ~ bitwise negation, "not".
- Arithmetic operators: +, -, unary -, unary +, *, /, % remainder, ** exponentiation, << left shift, >> right shift.

The EVM returns a Runtime Exception when the modulus operator is applied to the zero of a "divide by zero" operation.

Address: An address can hold a 20 byte value that is equivalent to the size of an Ethereum address. These address types are backed up with members that serve as the contract base.

String Literals: String literals can be represented using either single or double quotes (for example, "foo" or 'bar'). Unlike in the C language, string literals in Solidity do imply trailing value zeroes. For instance, “bar” will represent a three byte element instead of four. Similarly, in the case of integer literals, the literals are convertible inherently using the corresponding fit, that is, byte or string.

Modifier: In a smart contract, modifiers are used to ensure the coherence of the conditions defined before executing the code.

Solidity provides basic arrays, enums, operators, and hash values to create a data structure known as **“mappings**.” These mappings are used to return values associated with a given storage location. An Array is a contiguous memory allocation of a size defined by the programmer where if the size is initialized as K, and the type of element is instantiated as T, the array can be written as T[k].

Arrays can also be dynamically instantiated using the notation uint[][6]. Here the notation initializes a dynamic array with six contiguous memory allocations. Similarly, a two dimensional array can be initialized as arr[2][4], where the two indices point towards the dimensions of the matrix.

We will begin our programming venture with a simple structure of a contract. Consider the following code:

```
pragma solidity^0.4.0;
contract StorageBasic {
  uint storedValue;
  function set(uint var) {
     storedValue= var;
  }
  function get() constant returns (uint) {
   return storedValue;
  }
}
```

- The word “**Pragma**” refers to the instructions given to a compiler to sequentially execute the source code.
- Solidity is statically typed language. Therefore, each variable type irrespective of their scope can be instantiated at compile time. These elementary types can be further combined to create

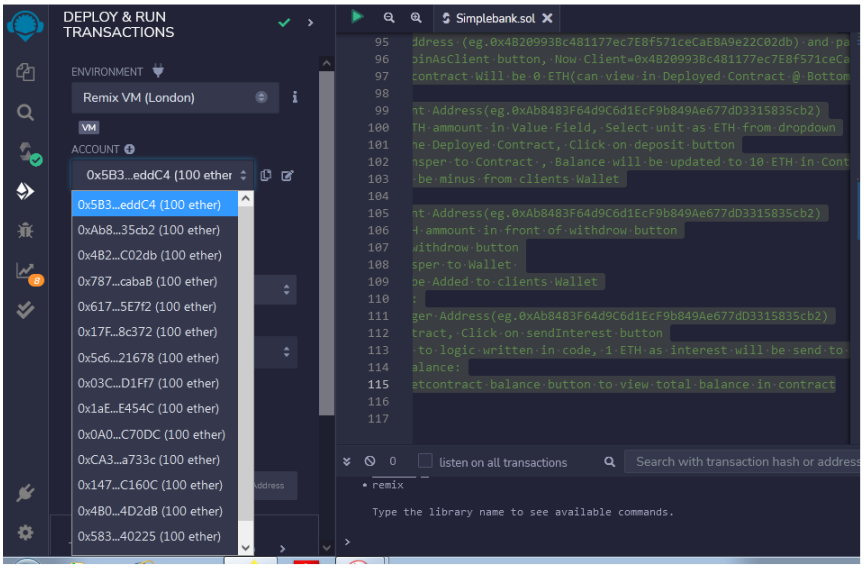
complex data types. These complex data types then synchronize with each other according to their respective preferences.

Steps to Run Banking Application

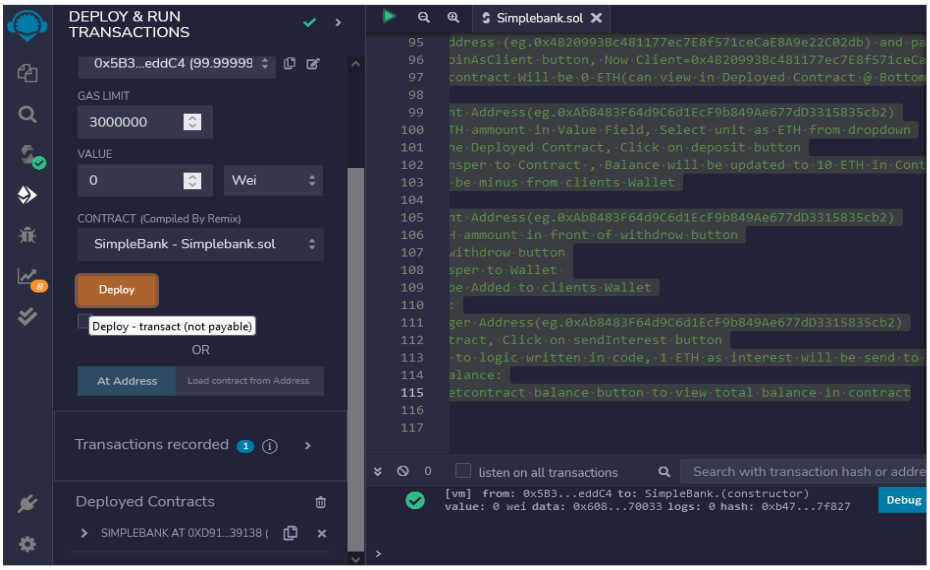
1.//Output steps

//Initially All Account has 100 fake ether

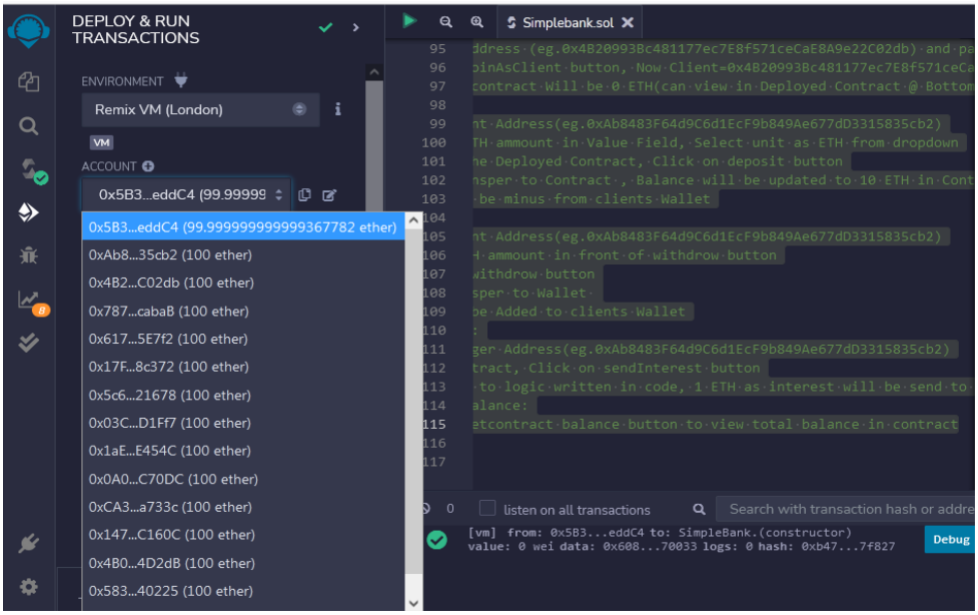
//Step 1: Select first Address (eg.0x5B38Da6a701c568545dCfcB03FcB875f56beddC4)



//Step 2: Click on Deploy button(Contract Created,Can view under Deployed Contract)



//After deploying contract 100 ETH turns to 99.99999.... ETH

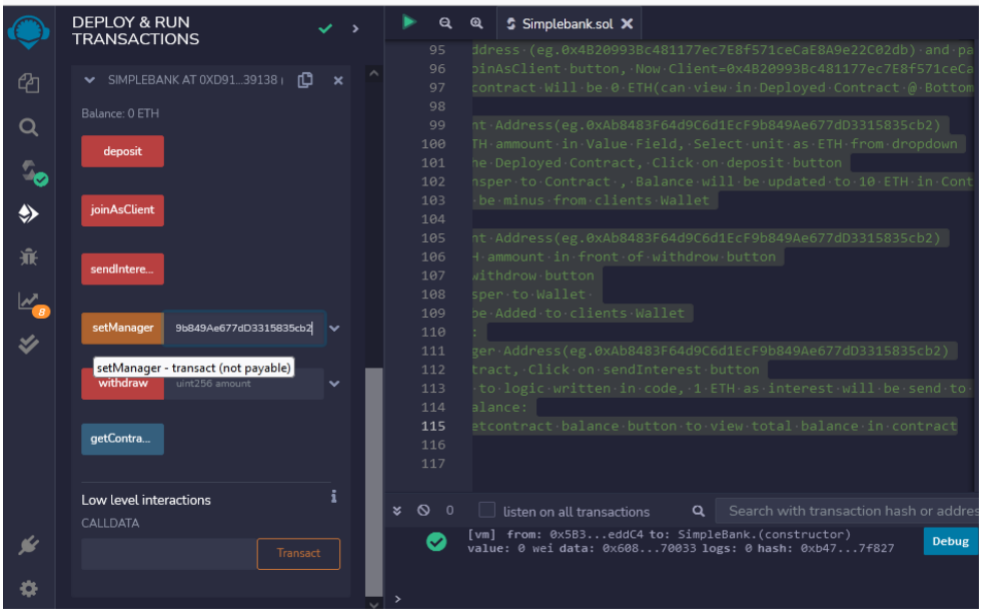


//Step 3: Set Manager: Follow Following instructions

// i.Select Onother Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

// ii.Copy this address (eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2) and paste it in contract, infront of set Manager button

// iii. click on set manager button, Now Manager=0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2



//Step 4: join as Client: Follow Following instructions

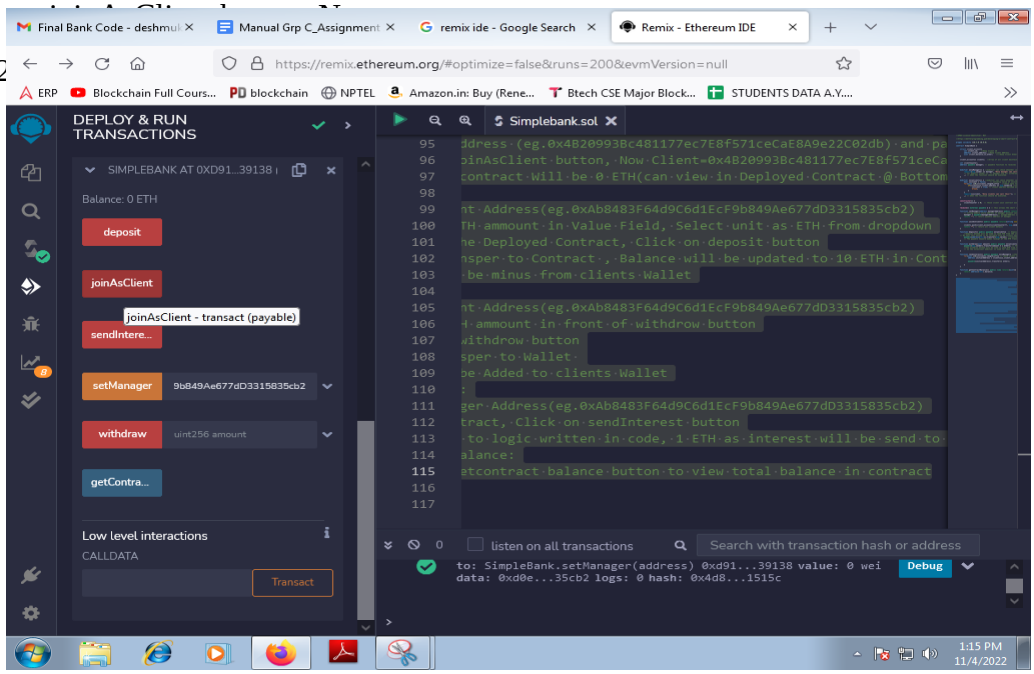
// i.Select Onother Address(eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db)

// ii.Copy this address (eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db) and paste it in

contract, infront of joinAsClient button

// iii.click

Client=0x4B2



//Initially Balance in contract Will be 0 ETH(can view in Deployed Contract @ Bottom)

//Step 5: Deposit:

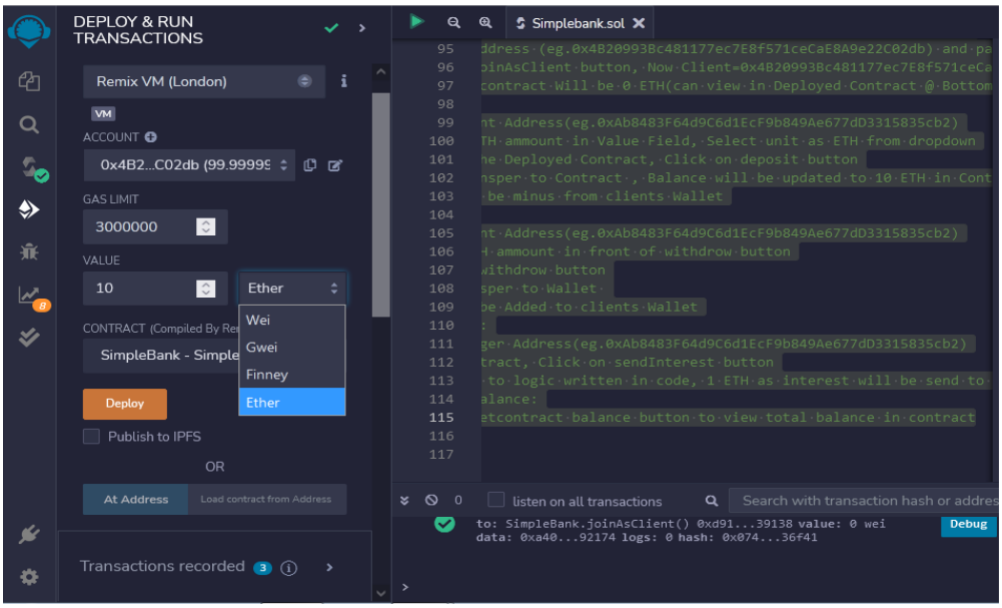
// i.Select Client Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

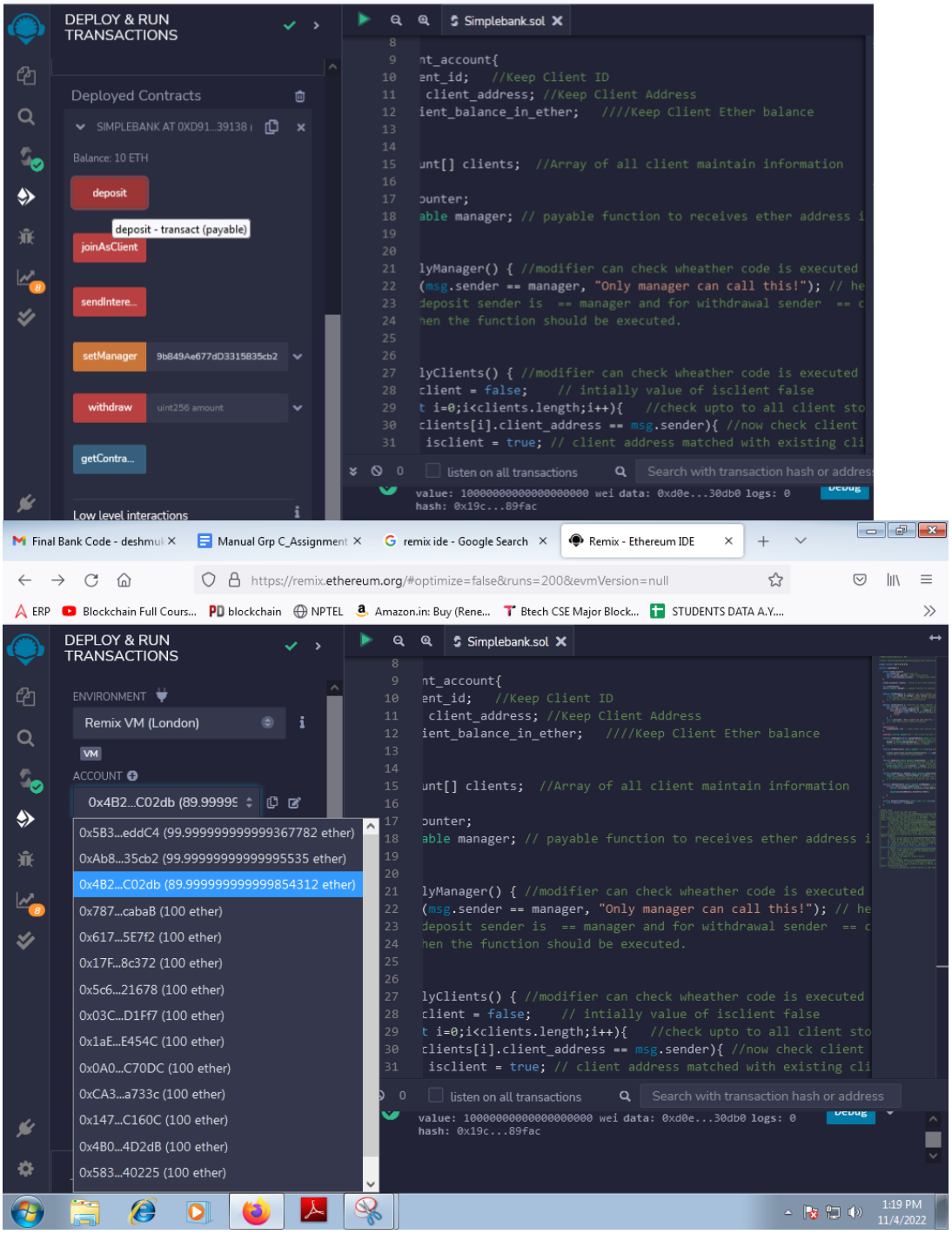
// ii. Enter 10 ETH ammount in Value Field, Select unit as ETH from dropdown

// iii. Come to the Deployed Contract, Click on deposit button

// iv. 10 ETH transper to Contract , Balance will be updated to 10 ETH in Contract

// V. 10 ETH will be minus from clients Wallet





//Step 6: Withdraw:

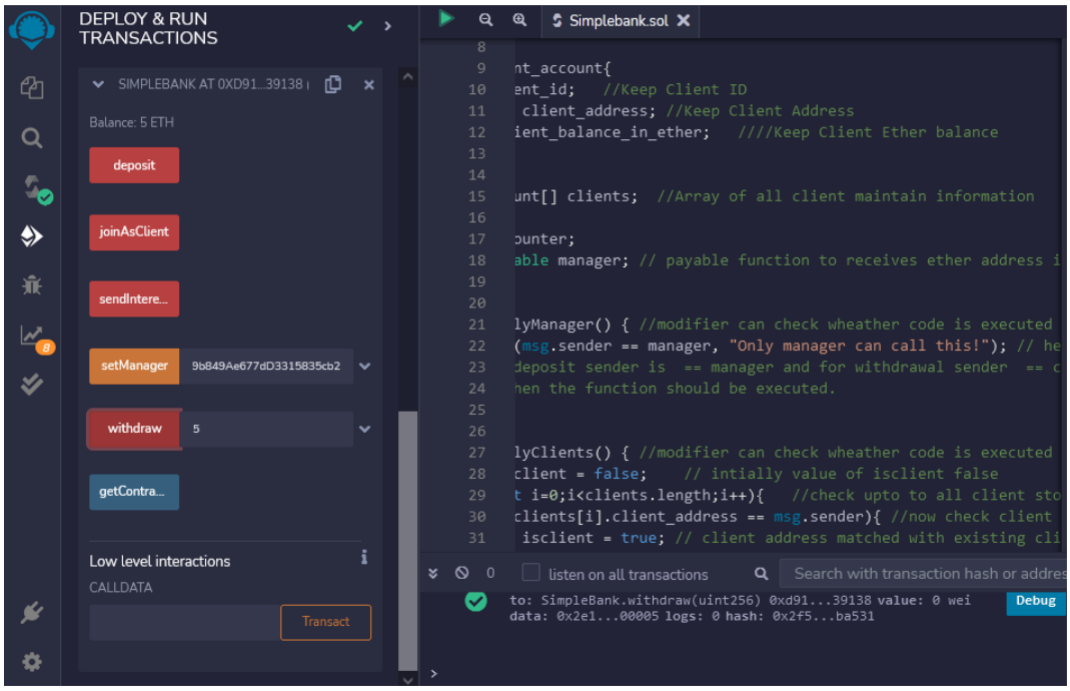
// i.Select Client Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

// ii. Enter 5 ETH ammount in front of withdrow button

// iii. Click on withdrow button

// iv. 5 ETH transper to Wallet

// V. 5 ETH will be Added to clients Wallet

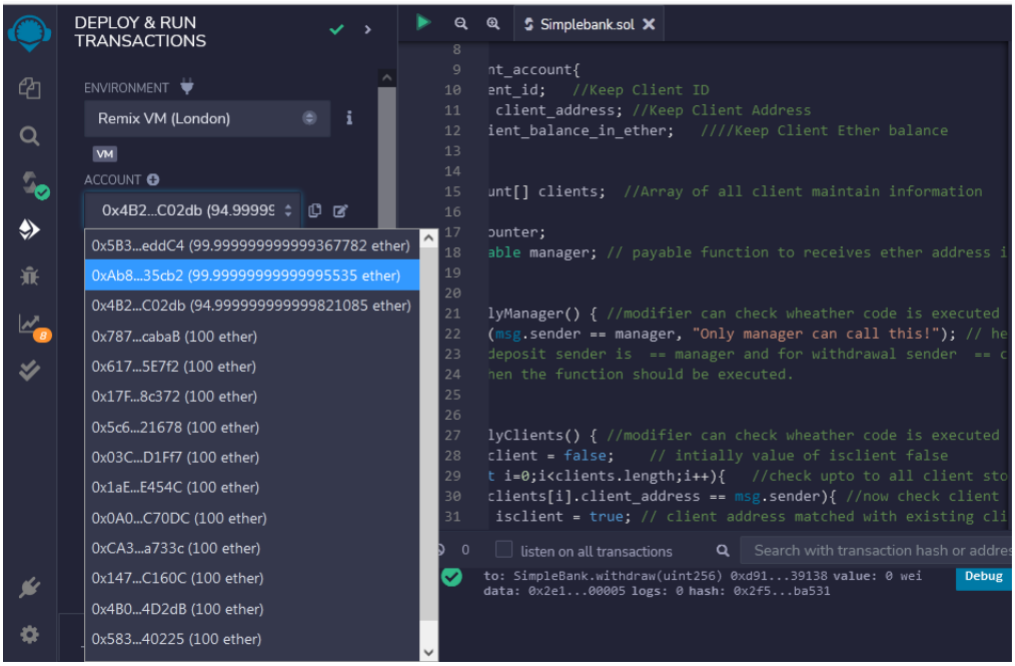


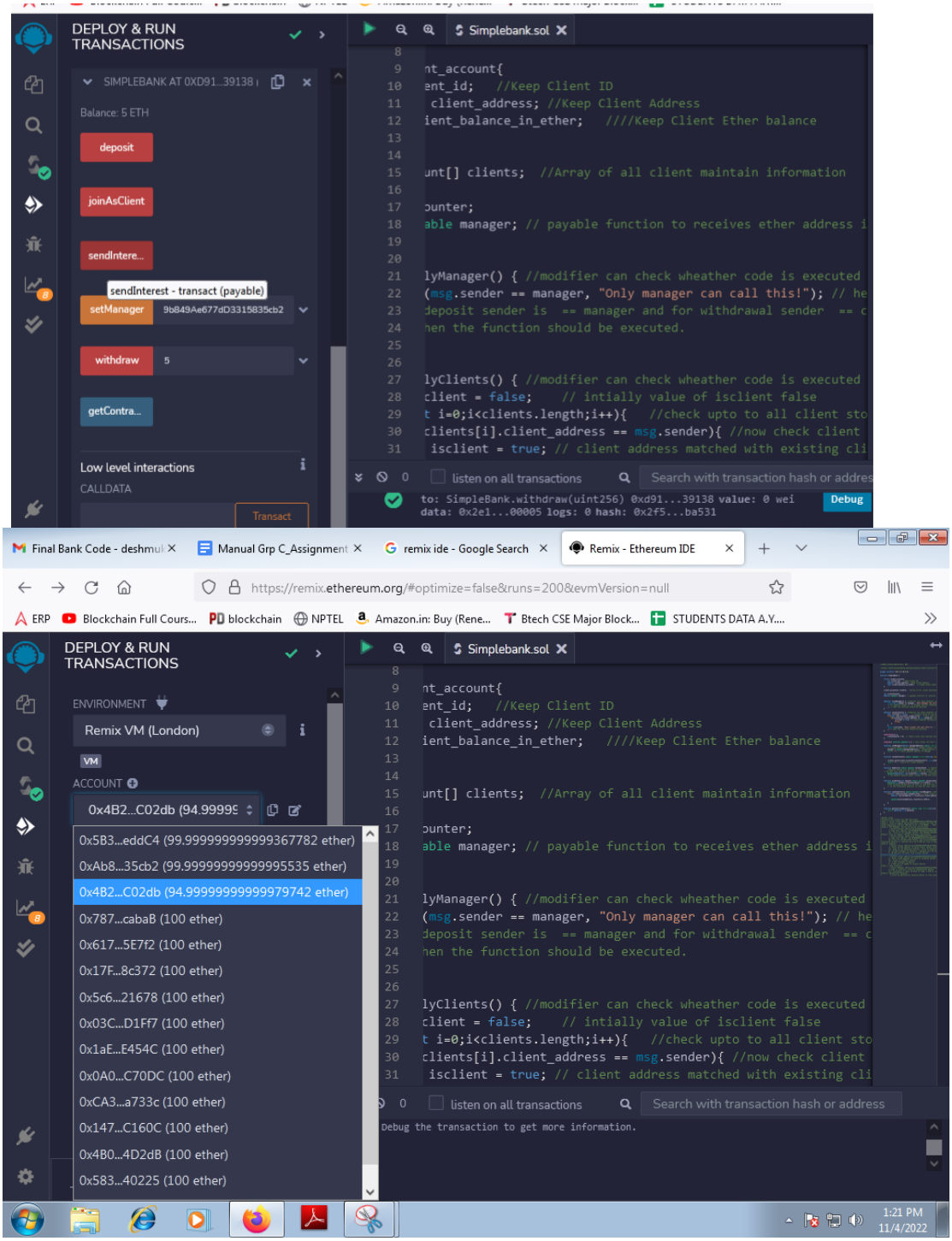
//Step 7: Send Interest:

// i.Select Manager Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)

// ii.Come to contract, Click on sendInterest button

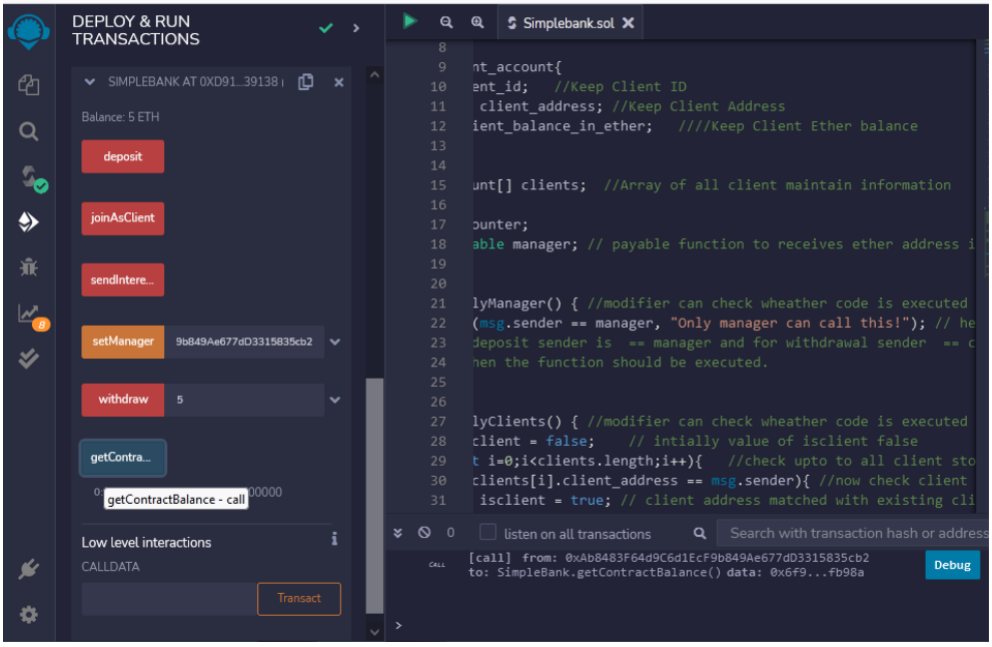
// iii. According to logic written in code, 1 ETH as interest will be send to Client Wallet





//Step 8: getContract Balance:

// i.Click on get contract balance button to view total balance in contract



Assignment Question

1. **What is Solidity?**
2. **What are the main differences between Solidity and other programming languages like Python, Java, or C++?**
3. **What is EVM bytecode?**
4. **What are the differences between Ethereum and blockchain and bitcoin?**

Reference link

- https://www.simplilearn.com/solidity-interview-questions-article

Code :

//SPDX-License-Identifier: MIT

```
//https://betterprogramming.pub/developing-a-smart-contract-by-using-remix-ide-81ff6f44ba2f

pragma solidity >=0.7.0 <0.9.0;

contract SimpleBank {

    struct client_account{
        int client_id;   //Keep Client ID
        address client_address; //Keep Client Address
        uint client_balance_in_ether;   ////Keep Client Ether balance
    }

    client_account[] clients;  //Array of all client maintain information

    int clientCounter;
    address payable manager; // payable function to receives ether address is datatype it is 20 byte hash address public key


    modifier onlyManager() { //modifier can check wheather code is executed accouding to condition for manager side
        require(msg.sender == manager, "Only manager can call this!"); // here sender is manager in this case
        // for deposit sender is  == manager and for withdrawal sender == client
        _; // when the function should be executed.
    }

    modifier onlyClients() { //modifier can check wheather code is executed accouding to condition for client side
        bool isclient = false;    // intially value of isclient false
        for(uint i=0;i<clients.length;i++){   //check upto to all client store in array
            if(clients[i].client_address == msg.sender){ //now check client address matched with sender only that client intiate transaction
                isclient = true; // client address matched with existing client address in bank database isclient value updated true.
                break;
            }
```

```
        }
        require(isclient, "Only clients can call this!"); // isclient
true here so allowed call the transaction.
        _; // when the function should be executed.
    }

    constructor() {
        clientCounter = 0;  // those client join contract assign there
ID intially it set 0
    }

    receive() external payable { } // this allows the smart contract to
receive ether

    function setManager(address managerAddress) public returns(string
memory){ //setManager method will be used to set the manager address to
variables
        // string memory store address of manager account instead of
store data
        manager = payable(managerAddress);// managerAddress is consumed
as a parameter and cast as payable to provide sending ether.
        return ""; // return payable address of manager
    }

    function joinAsClient() public payable returns(string memory){
//joinAsClient method will be used to make sure the client joins the
contract.

        clients.push(client_account(clientCounter++, msg.sender,
address(msg.sender).balance)); // push() array method to add items into
a storage array.
        return ""; // return all client details
    }

    function deposit() public payable onlyClients{  // deposit ==
client to contract by onlyclient
        //deposit method will be used to send ETH from the client
account to the contract.
    // We want this method to be callable only by clients who've joined
the contract, so the onlyClient modifier is used for this restriction.
        payable(address(this)).transfer(msg.value); //transfer methods
belongs to the contract, and it's dedicated to sending an indicated
amount of ETH between addresses.
```

```
        // The payable keyword makes receipt of the ETH transfer
possible so the amount of ETH indicated in the msg.value will be
transferred to the contract address.
    }

    function withdraw(uint amount) public payable onlyClients{ //
withdraw == contract to client by onlyclient
        payable(msg.sender).transfer(amount * 1 ether); // The address
of the sender( ie contract ) is held in the msg.sender variable.
        //The withdraw method will be used to send ETH from the
contract to the client account. It sends the unit of ETH indicated in
the amount parameter, from the contract to the client who sent the
transaction. We want this method to be callable only by clients who've
joined the contract either,
        // so the onlyClient modifier is used for this restriction.
    }

    function sendInterest() public payable onlyManager{ //The
sendInterest method will be used to send ETH as interest from the
contract to all clients. can called by only manager
        for(uint i=0;i<clients.length;i++){ // check client in database
            address initialAddress = clients[i].client_address; //
check client address

            payable(initialAddress).transfer(1 ether);

        }
    }

    function getContractBalance() public view returns(uint){
//getContractBalance method will be used to get the balance of the
contract we deployed.
        return address(this).balance;
    }
}

//Output steps
//Initially All Acount has 100 fake ether
//Step 1: Select first Address
(eg.0x5B38Da6a701c568545dCfcB03FcB875f56beddC4)
//Step 2: Click on Deploy button(Contract Created,Can view under
Deployed Contract)
//After deploying contract 100 ETH turns to 99.99999.... ETH
```

```
 //Step 3: Set Manager: Follow Following instructions
//   i.Select Onother
     Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
 // ii.Copy this address
(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2) and paste it
in     contract, infront of set Manager button
    //      iii. click on set manager button, Now
Manager=0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
//Step 4: join as Client: Follow Following instructions
     //     i.Select Onother
Address(eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db)
//  ii.Copy this address
(eg.0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db) and paste it
in contract, infront of joinAsClient button
    //      iii.click on joinAsClient button, Now
Client=0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
      //Initially Balanve in contract Will be 0 ETH(can view in
 Deployed Contract @ Bottom)
//Step 5: Deposit:
     //     i.Select Client
      Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
  //  ii. Enter 10 ETH ammount in Value Field, Select unit as ETH
from dropdown
//  iii. Come to the Deployed Contract, Click on deposit button
//  iv. 10 ETH transper to Contract , Balance will be updated to
10 ETH in Contract
    //      V. 10 ETH will be minus from clients Wallet
//Step 6: Withdraw:
     //     i.Select Client
Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
    //      ii. Enter 5 ETH ammount in front of withdrow button
    //      iii. Click on withdrow button
    //      iv. 5 ETH transper to Wallet
    //      V. 5 ETH will be Added to clients Wallet
//Step 7: Send Interest:
     //     i.Select Manager
Address(eg.0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2)
    //      ii.Come to contract, Click on sendInterest button
//  iii. According to logic written in code, 1 ETH as interest
will be send to Client Wallet
```

```
//Step 8: getContract Balance:
//         i.Click on getcontract balance button to view total balance in
contract
```

Assignment No: 4

Title of the Assignment: Write a program in solidity to create Student data. Use the following constructs:

- Structures
- Arrays
- Fallback

Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.

Objective of the Assignment: Students should be able to learn about Solidity. Its datatypes and implementations.

Prerequisite:

1. Basic Programming Logic
2. Basic knowledge of Solidity

--- Contents for Theory:

1. **Solidity - Arrays**
2. **Solidity - Structures**
3. **Solidity – Fallback**
4. **Implementation**

---

1. Solidity – Arrays:

Arrays are data structures that store the fixed collection of elements of the same data types in which each and every element has a specific location called index. Instead of creating numerous individual variables of the same type, we just declare one array of the required size and store the elements in the array and can be accessed using the index. In Solidity, an array can be of fixed size or dynamic size. Arrays have a continuous memory location, where the lowest index corresponds to the first element while the highest represents the last.

Creating an Array

To declare an array in Solidity, the data type of the elements and the number of elements should be specified. The size of the array must be a positive integer and data type should be a valid Solidity type

Syntax:

<data type> <array name>[size] = <initialization>

Fixed-size Arrays :

The size of the array should be predefined. The total number of elements should not exceed the size of the array. If the size of the array is not specified then the array of enough size is created which is enough to hold the initialization.

Dynamic Array:

The size of the array is not predefined when it is declared. As the elements are added the size of array changes and at the runtime, the size of the array will be determined.

Array Operations:

a. Accessing Array Elements:

The elements of the array are accessed by using the index. If you want to access ith element then you have to access (i-1)th index.

b. Length of Array:

Length of the array is used to check the number of elements present in an array. The size of the memory array is fixed when they are declared, while in case the dynamic array is defined at runtime so for manipulation length is required.

c. Push:

Push is used when a new element is to be added in a dynamic array. The new element is always added at the last position of the array.

d. Pop:

Pop is used when the last element of the array is to be removed in any dynamic array.

2. Solidity – Structures :

Structs in Solidity allows you to create more complicated data types that have multiple properties. You can define your own type by creating a **struct**.

They are useful for grouping together related data.

Structs can be declared outside of a contract and imported in another contract. Generally, it is used to represent a record. To define a structure *struct* keyword is used, which creates a new data type.

Syntax:

```
struct <structure_name> {
   <data type> variable_1;
   <data type> variable_2;
}
```

For accessing any element of the structure, 'dot operator' is used, which separates the struct variable and the element we wish to access. To define the variable of structure data type structure name is used.

3. Solidity – Fallback :

The solidity fallback function is executed if none of the other functions match the function identifier or no data was provided with the function call. Only one unnamed

function can be assigned to a contract and it is executed whenever the contract receives plain Ether without any data. To receive Ether and add it to the total balance of the contract, the fallback function must be marked payable. **If no such function exists, the contract cannot receive Ether through regular transactions and will throw an exception.**

Properties of a fallback function:

1. Has no name or arguments.
2. If it is not marked **payable**, the contract will throw an exception if it receives plain ether without data.
3. Can not return anything.
4. Can be defined once per contract.
5. It is also executed if the caller meant to call a function that is not available
6. It is mandatory to mark it external.
7. It is limited to 2300 gas when called by another function. It is so for as to make this function call as cheap as possible.

4. Implementation

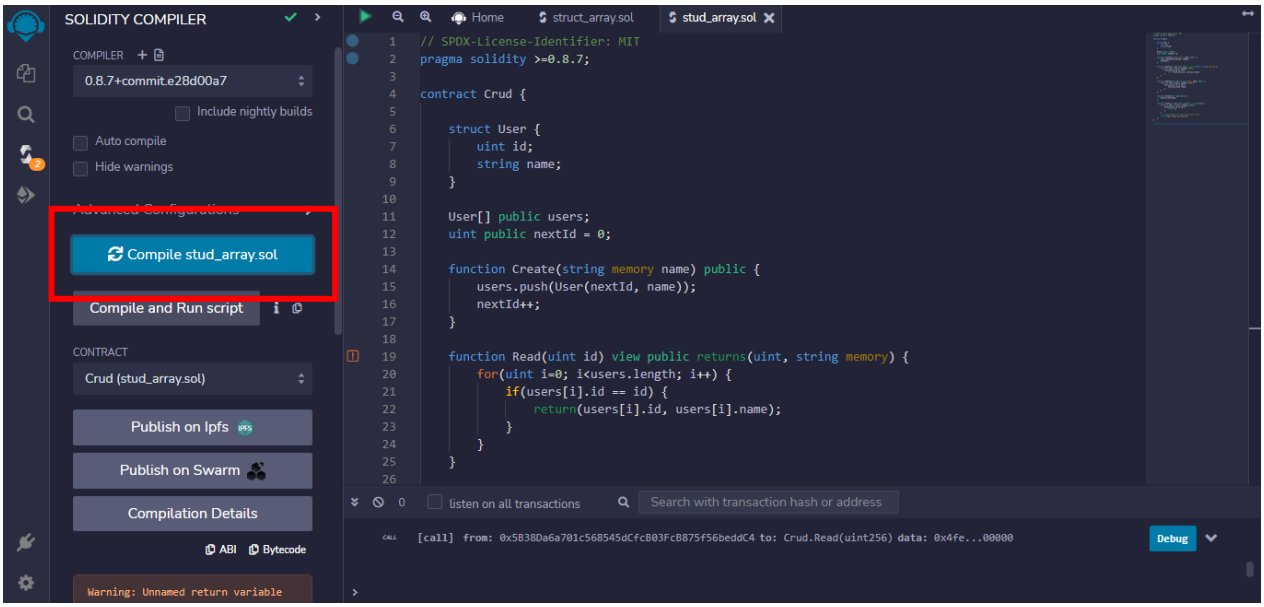
Code -

```
pragma solidity ^0.5.0;
contract Crud {
    struct User {
        uint id;
        string name;
    }
    User[] public users;
    uint public nextId = 0;
    function Create(string memory name) public {
        users.push(User(nextId, name));
        nextId++;
    }
    function Read(uint id) view public returns(uint, string memory) {
        for(uint i=0; i<users.length; i++) {
            if(users[i].id == id) {
                return(users[i].id, users[i].name);
            }
        }
    }
    function Update(uint id, string memory name) public {
        for(uint i=0; i<users.length; i++) {
            if(users[i].id == id) {
```

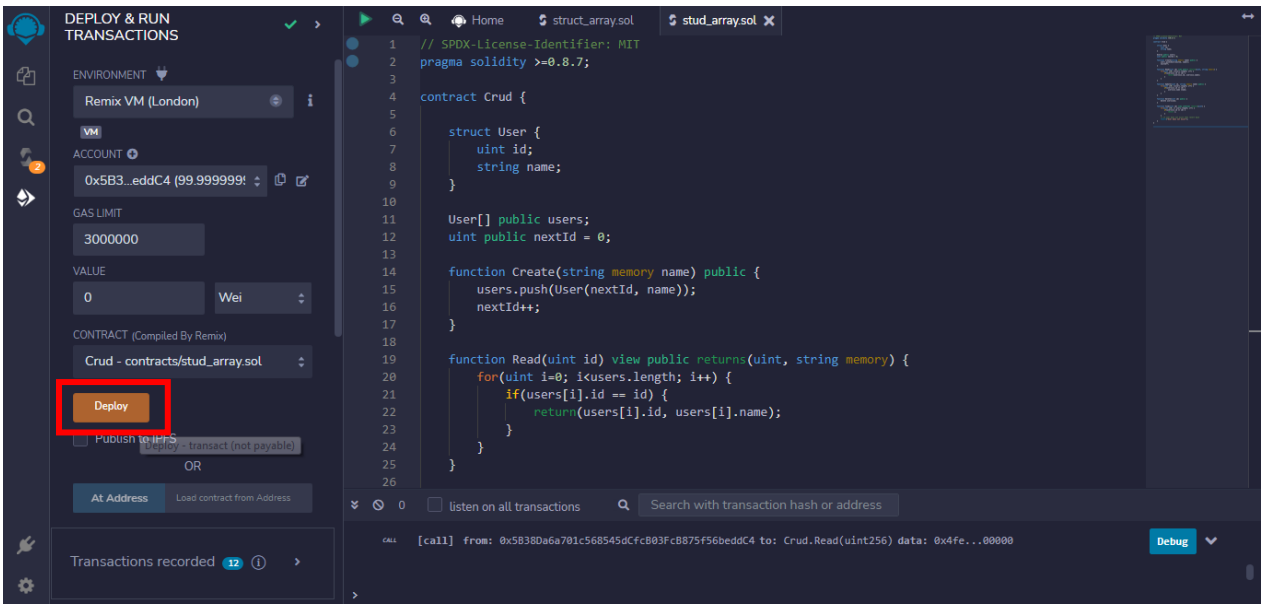
```
                users[i].name =name;
            }
        }
    }
    function Delete(uint id) public {
        delete users[id];
    }
    function find(uint id) view internal returns(uint) {
        for(uint i=0; i< users.length; i++) {
            if(users[i].id == id) {
                return i;
            }
        }
        // if user does not exist then revert back
        revert("User does not exist");
    }
}
```

Output –

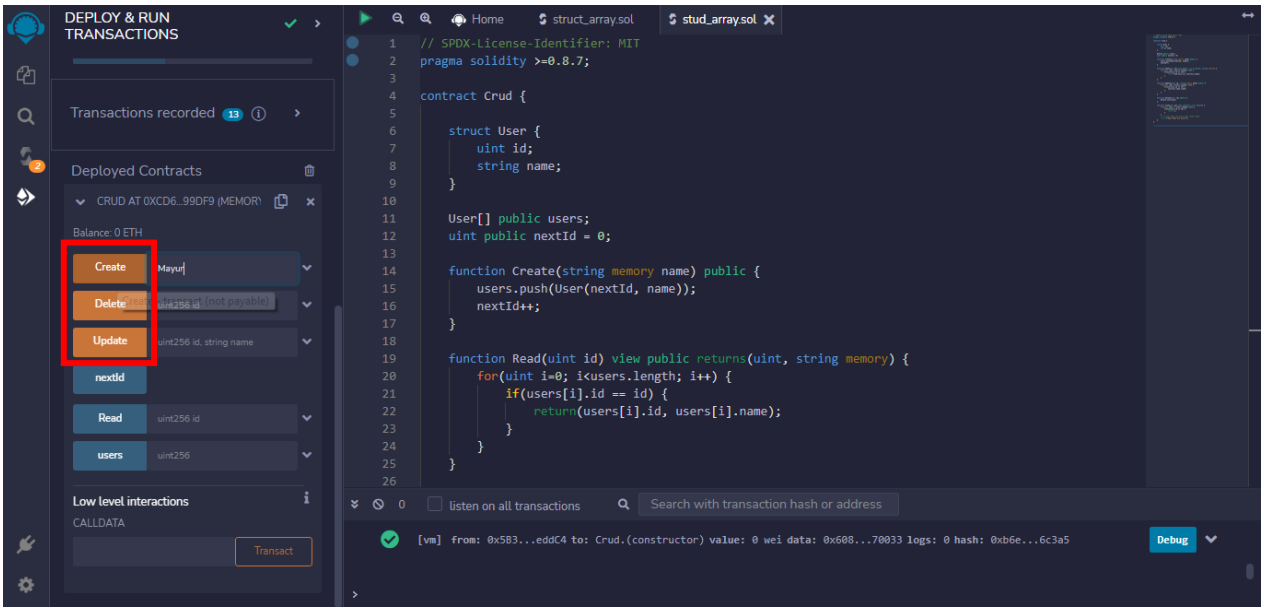
Step 1 – Compile the program by clicking on compile button.



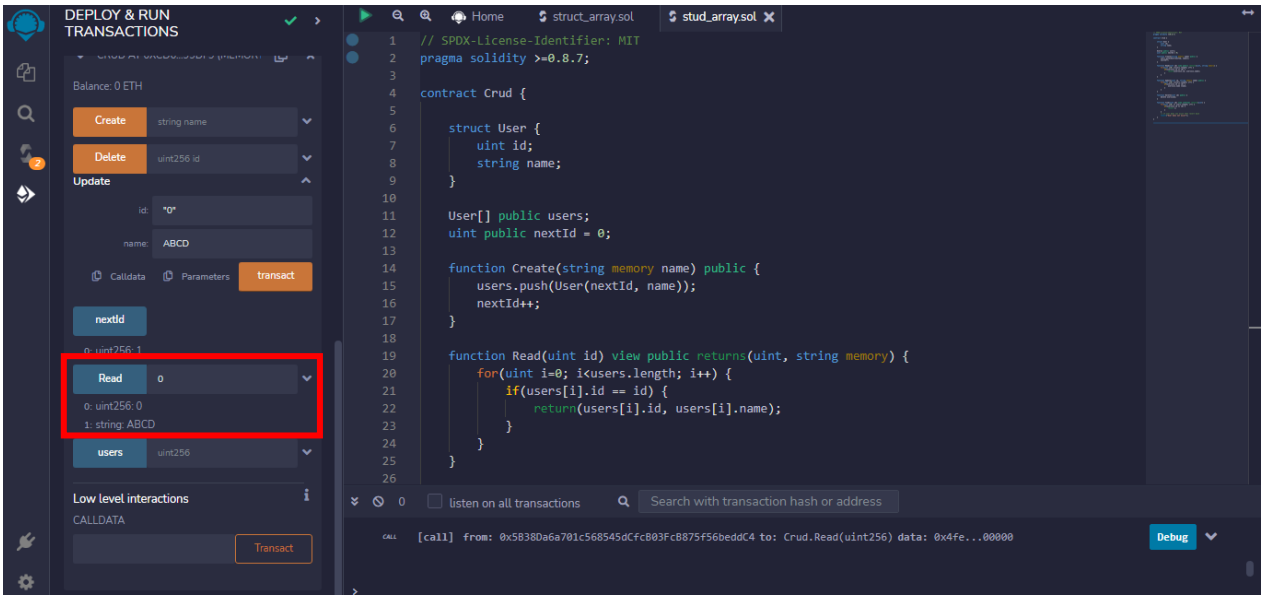
Step 2 – After the Successful compilation, Deploy the contract to see the output.



Step 3 - Now you can check the output. You can insert, update and delete the student data using your smart contract.



Step 4 – After entering your data, you can read the data using ID.



Conclusion-

In this way we have created array, structure and used fallback function in solidity.

Assignment Question:

1. **What is fixed array and dynamic array in solidity?**
2. **What is Array in solidity?**
3. **What is structure in solidity? Define its syntax.**
4. **What is fallback function?**

Reference link

- https://www.geeksforgeeks.org/solidity-arrays/?ref=lbp
- tutorialspoint.com/solidity/solidity_structs.htm

Assignment No: 5

Title of the Assignment: Write a survey report on types of Blockchains and its real time use cases.

Objective of the Assignment: Students should be able to learn about different use cases/ real time application of Blockchain and perform survey on one of the case study

Points Expected in Report:

1. Introduction of Blockchain
2. Types of Blockchain
3. Different Applications of Blockchain
4. Detail survey on one use case