

## **Assignment No.1**

**Aim :-** Installation of Meta Mask and study spending Ether per transaction.

Objective: Student will be able to learn

1. Concept of Meta mask
2. Installation and study of Meta mask and Ether

Theory:-

### **ETAMASK**

- The existing browsers on our systems are centralized. In order to create a de-centralized system we add a plug-in called “Metamask”
- We need it because we need “ether” (currency for blockchain)
- Installed Metamask( Gives a virtual Ethereum wallet)
- Created a simple smart contract through MetaMask(Using fake ether i.e currency of blockchain)
- In-depth study of state-of-art on smart contract implementation.

### **REMIX IDE**

Platform to create and deploy smart contract, supports solidity .

### **SOLIDITY**

A Language to create smart contracts, similar to JavaScript. reating a De-centralized platform for testing a smart contract

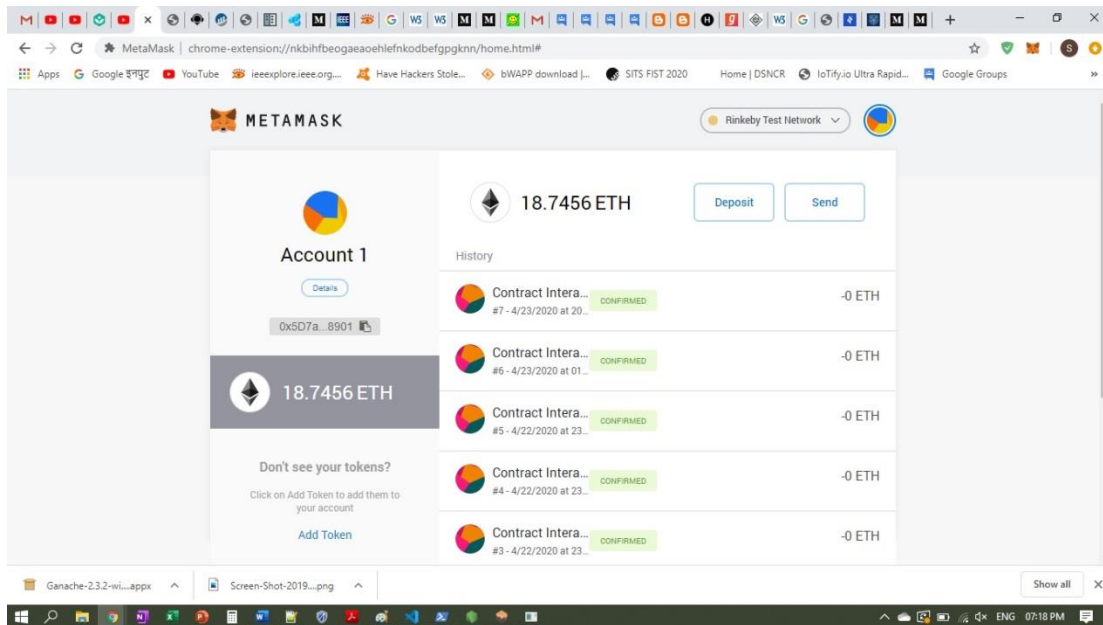
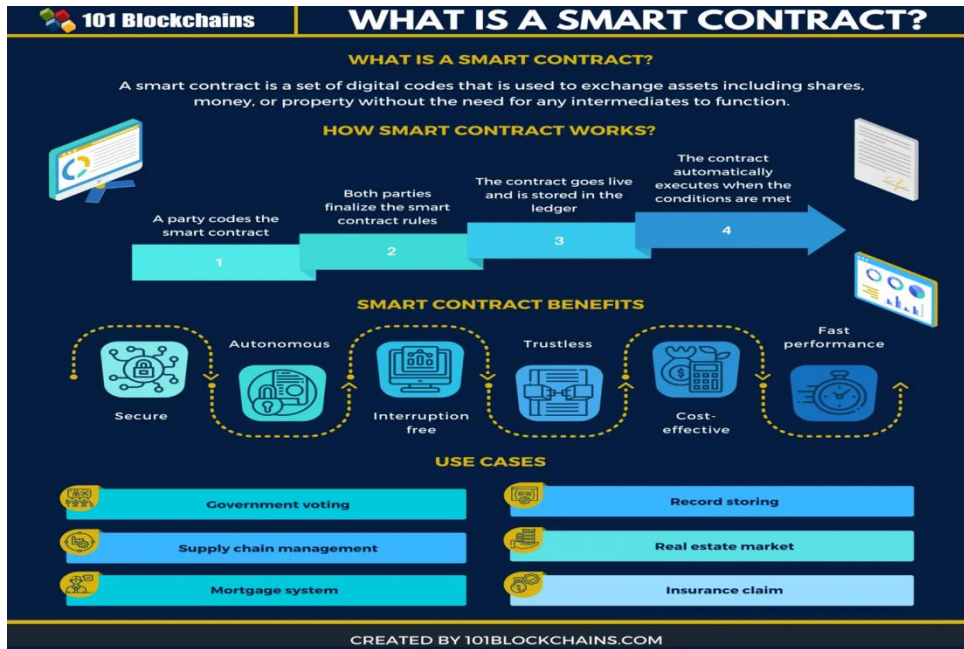
### **Pre-requisites for creating a Block chain environment:**

Adding Metamask extension to default browser Creating a wallet through test network Rinkeby.

Currency needed for block chain transaction – ether

Rinkeby gives this ether – 18.75 ethers / 3 days Smart contract in solidity language on IDE

Remix



## Smart Contract Implementation details

The screenshot shows the Etherscan website interface for a transaction on the Rinkeby Testnet. The transaction hash is 0xad0eeb982fa0a3eab2dcd089ce4860544b2d61f5c9330de9bf685503367f1b6a. The transaction is marked as 'Success' and has 573314 block confirmations. It was sent from 0x5d7a55c5f52d58d71adefc87c7423345e5df8901 to a contract at 0xb4a96735f1fca0f0abb0ee5d83d71f82ac5f05. The value is 0 Ether (\$0.00) and the transaction fee is 0.00027291 Ether (\$0.000000). The gas limit is 27,291. The timestamp is 99 days 12 hrs ago (Apr-23-2020 03:00:07 PM +UTC).

Transaction Hash:	0xad0eeb982fa0a3eab2dcd089ce4860544b2d61f5c9330de9bf685503367f1b6a
Status:	Success
Block:	6366572 573314 Block Confirmations
Timestamp:	99 days 12 hrs ago (Apr-23-2020 03:00:07 PM +UTC)
From:	0x5d7a55c5f52d58d71adefc87c7423345e5df8901
To:	Contract 0xb4a96735f1fca0f0abb0ee5d83d71f82ac5f05
Value:	0 Ether (\$0.00)
Transaction Fee:	0.00027291 Ether (\$0.000000)
Gas Limit:	27,291

Create your Own Wallet Using meta mask for Crypto Transaction

The banner features the MetaMask logo and navigation links: Features, Support, About, Build, and a Download button. Below these are buttons for Chrome, iOS, and Android. The main heading is 'Install MetaMask for your browser'. The central image shows the MetaMask mobile app interface with 'Account1' selected, displaying a balance of 1 ETH (\$300,000.00 USD) and buttons for Deposit and Send. A transaction history shows a recent send of 3 ETH. At the bottom is a large button that says 'Install MetaMask for Chrome'.

**METAMASK** Features Support About Build Download

Chrome iOS Android

### Install MetaMask for your browser

Account1  
1 ETH  
\$300,000.00 USD  
Deposit Send

History  
#3 - 4/1/2019 at 11:30  
Sent Ether - 3 ETH  
-\$600 USD

Install MetaMask for Chrome

## ETHER(TRANSACTION FEE SPENT ON THE SMART CONTRACT

The screenshot shows the Etherscan.io interface for a transaction on the Rinkby testnet. The transaction details are as follows:

Field	Value
Transaction Fee	0.00027291 Ether (\$0.000000)
Gas Limit	27,291
Gas Used by Transaction	27,291 (100%)
Gas Price	0.00000001 Ether (10 Gwei)
Nonce	7
Input Data	Function: deposit(int256 amount) *** MethodID: 0xf04991f0 [0]: 0012c

The interface also shows a "View Input As" button and a "Click to see Less" link. The footer indicates the page is "Powered by Ethereum" and includes the Etherscan logo and version information.


## Create Your Own Wallet Using Meta mask For Crypto Transaction

The screenshot shows the MetaMask website landing page. The header includes the MetaMask logo, navigation links (Features, Support, About, Build), and a "Download" button. Below the header, there are buttons for "Chrome", "iOS", and "Android". The main heading is "Install MetaMask for your browser". Below this, there is a screenshot of the MetaMask mobile app interface showing a wallet with 1 ETH and a transaction history. At the bottom, there is a large blue button that says "Install MetaMask for Chrome".

Navigate to the extension icon in the top right corner of your web browser and find the MetaMask option, once you've successfully downloaded the software. Click the "Get Started" button and you'll be taken to the next page and presented with two options (see below.)




## New to MetaMask?



**No, I already have a Secret Recovery Phrase**  
Import your existing wallet using a Secret Recovery Phrase

Import wallet



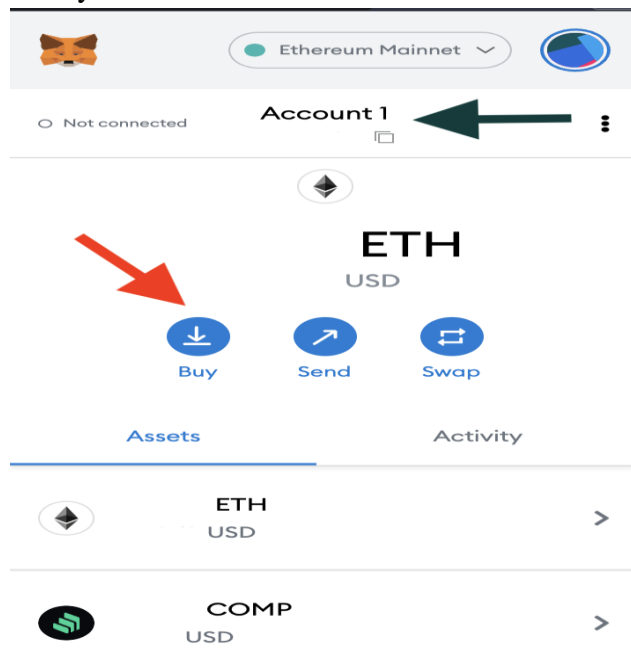
**Yes, let's get set up!**  
This will create a new wallet and Secret Recovery Phrase

Create a Wallet

Once you've completed the above steps, you'll be able to access your new MetaMask wallet. There are two main components you'll need to familiarize yourself with so that you can begin using the software:

**Identifying your public address:** This is the address you can freely share with people or platforms like exchanges in order to receive cryptocurrency into your wallet. Think of it as your home address that you share with people to receive inbound mail. It's always advisable, however, to check to make sure any inbound tokens are compatible with MetaMask first before receiving them, otherwise, they might be lost forever.

**How to fund/buy and send:** These are the core functions of MetaMask.



You can locate your unique MetaMask public address by clicking the “Account 1” button (black arrow).

Finally, in order to begin interacting with any Ethereum platform, you’ll first need to fund your MetaMask wallet with an amount of ether – the native cryptocurrency of Ethereum. All actions on the blockchain cost a fee, whether that’s moving tokens from A to B or creating an NFT collection. This fee, known as a “gas” fee, is denominated in ether

How much you choose to fund your wallet depends on how much you intend to interact with various platforms. For moderate use, \$100 worth of ether is usually a good starting point to cover any initial fees.

**Conclusion: We Have Study Installation of Metamask**

## **Assignment No 02**

**Aim :- Create your own wallet using Metamask for crypto transactions.**

Objective: Student will be able to learn

1. Concept of Meta mask
2. own wallet using Metamask for crypto transactions

### **Theory:-**

#### **A Brief introduction to MetaMask**

MetaMask is an open-source, straightforward, and easy-to-use cryptocurrency wallet. It functions as a web browser extension available for Chrome, Firefox, Brave, or a mobile application for iOS or Android. Initially, this wallet supported only Ether and ERC-20 tokens, and now it is compatible with ERC-721 and ERC-1155 token standards. Furthermore, MetaMask benefits include interaction with websites; hence, it can function as a connection node for various DApps on Ethereum.

Adrian Devis and Dan Finlay are the MetaMask developers. Their idea was revolutionary and straightforward; they intended to create a web browser extension that would allow managing cryptocurrency and using the browser for fast and secure access with DApps. ConsenSys Software Inc. — a development company, focusing on applications that use Ethereum's blockchain, implemented the idea in 2016.

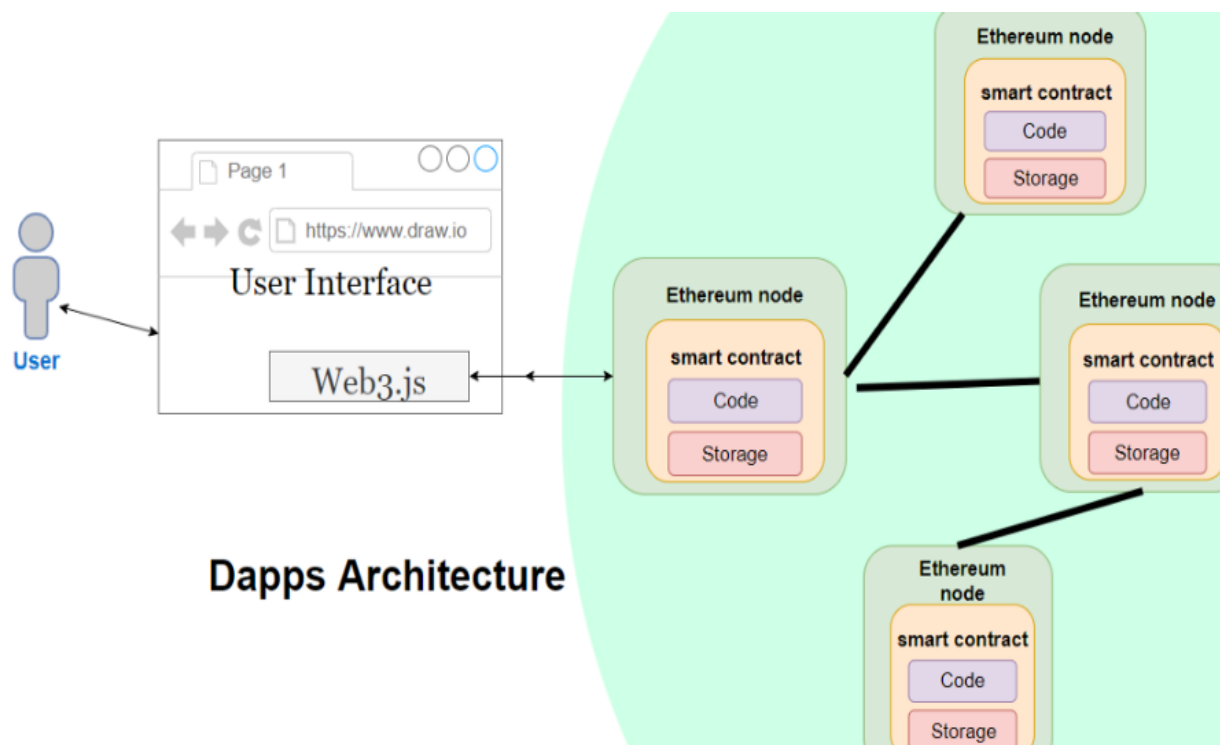
The solution used Ethereum's interface and a web API called web3.js. This Ethereum library is the fundament of MetaMask since it allows the browser to interact with the local or remote blockchain nodes via HTTP, IPC, and WebSocket; also, it gained the ability to record and read data from smart contracts, transfer tokens, etc. In another way, web3.js allowed the blockchain developers to create proxy and communication bridges between MetaMask, DApps, and the user.

Adrian Devis and Dan Finlay admit that their idea was great. Yet, the technical implementation was super complicated, especially in providing security for the users (web wallets are considered the most vulnerable to hacker attacks). Nonetheless, ConsenSys succeeded, and on the 14th of July in 2016, they offered the first version of MetaMask web browser cryptocurrency wallet for Chrome. Later, they presented a version for Firefox, Brave, and other popular browsers. In 2019 they also launched the mobile version of the MetaMask cryptocurrency wallet.

### How does the MetaMask wallet function?

As we mentioned above, the MetaMask cryptocurrency wallet employs the web3.js library to function. This library is a part of the official Ethereum product. The library was developed focusing on the requirements of web applications that could interact with the Ethereum blockchain and take advantage of all blockchain's benefits and functions.

MetaMask is a cryptocurrency wallet for Ethereum and an instrument that helps to interact with DApps. MetaMask connects the extension to the DApp so that to fulfill both tasks. When the application identifies the MetaMask, it creates a connection, and the user can start using all the features of a specific application.



For instance, it can assets trading, access to resources or services, or any other task within the capability of a DApp. Each action has its cost (transaction fee) that must be paid in Ethereum or any specified token. MetaMask wallet has all instruments and protocols for this purpose.

Hence, we can state that Metamask also controls the interaction of the user and DApp, and processes the operations required for specific actions, besides the function of a wallet. Reliable and secure cryptography and safe internet connection are the environments for these operations.



Furthermore, MetaMask can generate asymmetric keys, store them on a local device, and manage access to the keys. To sum up, MetaMask is a super-safe extension

### **Extended functions set for MetaMask clone**

To help your MetaMask wallet clone become famous, you should add some advantages that highlight it from the competitors and improve the user experience.

These can be the following:

**Linking an account.** Your users will find it useful to be able to buy a cryptocurrency and exchange it for fiat within the wallet. This will be possible if you develop a wallet like MetaMask and add the feature of linking bank accounts, credit/debit cards, PayPal, or other online payment systems.

**eCommerce integrations.** We mean integrating the wallet with exchanges, NFT marketplaces, decentralized applications, shops, and other services that the users might find useful.

**Multilingual interface.** If you focus on a market where all people speak the same language, you might neglect this aspect. However, your intentions are global, and you should add as many languages as possible to increase the target audience.

**Push notifications.** The notifications will inform the users of receiving payments, ending transactions, rapid exchange rate changes in the investment account, system updates, suspicious activity, etc.

**VIP support.** Numerous cryptocurrency trading platforms offer support for an additional fee. This may include 24/7 support, communication with a personal specialist, etc.

**QR scanner.** This is another useful feature that allows your users to make payments even faster. Moreover, it will decrease the number of transfers done by mistake.

**Conclusion: We Have Study own wallet using Metamask for crypto transactions**

## Assignment No 03

**Aim :- Write a smart contract on a test network, for Bank account of a customer for following operations**

- **Deposit money**
- **Withdraw Money**
- **Show balance**

**Objective :**

1. Concept of Smart Contract
2. for Bank account of a customer

**Theory :-**

### Writing a Banking Contract

This article will demonstrate how to write a simple, but complete, smart contract in Solidity that acts like a bank that stores ether on behalf of its clients. The contract will allow deposits from any account, and can be trusted to allow withdrawals only by accounts that have sufficient funds to cover the requested withdrawal. This post assumes that you are comfortable with the ether-handling concepts introduced in our post, [Writing a Contract That Handles Ether](#).

That post demonstrated how to restrict ether withdrawals to an “owner’s” account. It did this by persistently storing the owner account’s address, and then comparing it to the `msg.sender` value for any withdrawal attempt. Here’s a slightly simplified version of that smart contract, which allows anybody to deposit money, but only allows the owner to make withdrawals:

```
pragma solidity ^0.4.19;

contract TipJar {

    address owner;  // current owner of the contract

    function TipJar() public {
        owner = msg.sender;
    }

    function withdraw() public {
        require(owner == msg.sender);
        msg.sender.transfer(address(this).balance);
    }
}
```

```

function deposit(uint256 amount) public payable {
    require(msg.value == amount);
}

function getBalance() public view returns (uint256) {
    return address(this).balance;
}
}

```

## Maintaining Individual Account Balances

I am going to generalize this contract to keep track of ether deposits based on the account address of the depositor, and then only allow that same account to make withdrawals of that ether. To do this, we need a way keep track of account balances for each depositing account—a mapping from accounts to balances. Fortunately, Solidity provides a ready-made mapping data type that can map account addresses to integers, which will make this bookkeeping job quite simple. (This mapping structure is much more general key/value mapping than just addresses to integers, but that’s all we need here.)

Here’s the code to accept deposits and track account balances:

```

pragma solidity ^0.4.19;

contract Bank {

    mapping(address => uint256) public balanceOf; // balances, indexed by addresses

    function deposit(uint256 amount) public payable {
        require(msg.value == amount);

        balanceOf[msg.sender] += amount; // adjust the account's balance
    }
}

```

### Here are the new concepts in the code above:

- `mapping(address => uint256) public balanceOf;` declares a persistent public variable, `balanceOf`, that is a mapping from account addresses to 256-bit unsigned integers. Those integers will represent the current balance of ether stored by the contract on behalf of the corresponding address.
- Mappings can be indexed just like arrays/lists/dictionaries/tables in most modern programming languages.
- The value of a missing mapping value is 0. Therefore, we can trust that the beginning balance for all account addresses will effectively be zero prior to the first deposit.

It's important to note that `balanceOf` keeps track of the ether balances assigned to each account, but it does not actually move any ether anywhere. The bank contract's ether balance is the sum of all the balances of all accounts—only `balanceOf` tracks how much of that is assigned to each account. Note also that this contract doesn't need a constructor. There is no persistent state to initialize other than the `balanceOf` mapping, which already provides default values of 0.

### Withdrawals and Account Balances

Given the `balanceOf` mapping from account addresses to ether amounts, the remaining code for a fully-functional bank contract is pretty small. I'll simply add a withdrawal function:

#### bank.sol

```
pragma solidity ^0.4.19;

contract Bank {

    mapping(address => uint256) public balanceOf; // balances, indexed by addresses

    function deposit(uint256 amount) public payable {
        require(msg.value == amount);
        balanceOf[msg.sender] += amount; // adjust the account's balance
    }

    function withdraw(uint256 amount) public {
        require(amount <= balanceOf[msg.sender]);
        balanceOf[msg.sender] -= amount;
        msg.sender.transfer(amount);
    }
}
```

The code above demonstrates the following:

- The `require(amount <= balances[msg.sender])` checks to make sure the sender has sufficient funds to cover the requested withdrawal. If not, then the transaction aborts without making any state changes or ether transfers.
- The `balanceOf` mapping must be updated to reflect the lowered residual amount after the withdrawal.
- The funds must be sent to the sender requesting the withdrawal.

### **Important: Avoiding the Reentrancy Vulnerability**

In the `withdraw()` function above, it is very important to adjust `balanceOf[msg.sender]` **before** transferring ether to avoid an exploitable vulnerability. The reason is specific to smart contracts and the fact that a transfer to a smart contract executes code in that smart contract. (The essentials of Ethereum transactions are discussed in [How Ethereum Transactions Work](#).)

Now, suppose that the code in `withdraw()` did not adjust `balanceOf[msg.sender]` before making the transfer *and* suppose that `msg.sender` was a malicious smart contract. Upon receiving the transfer—handled by `msg.sender`'s fallback function—that malicious contract could initiate *another* withdrawal from the banking contract. When the banking contract handles this second withdrawal request, it would have already transferred ether for the original withdrawal, but it would not have an updated balance, so it would allow this second withdrawal!

This vulnerability is called a “reentrancy” bug because it happens when a smart contract invokes code in a different smart contract that then calls back into the original, thereby reentering the exploitable contract. For this reason, it's essential to always make sure a contract's internal state is fully updated before it potentially invokes code in another smart contract. (And, it's essential to remember that every transfer to a smart contract executes that contract's code.)

**Conclusion: Conclusion: We Have Study a smart contract on a test network, for Bank account of a customer operations.**

## Assignment No 04

**Aim :-** 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 :**

1. Concept of solidity
2. smart contract on Ethereum and Observe the transaction fee and Gas values

**Theory :-**

### Introduction

Blockchain is a decentralized, distributed public ledger that lets us collaborate and coordinate the members that do not trust each other to make a secure transaction. Many of you understand blockchain as a bitcoin, but bitcoin is a cryptocurrency that takes the help of blockchain technology to operate.

### *Blockchain Technology*

First, when Windows was launched to create reports and work on any project, we used MS word, where only one person could edit at a time and then send to another, and the process goes on, one after the other method. After some technological evolution, we have seen Google docs, google sheets in the market where online multiple people can work on a single document simultaneously. But the problem here is that it works on centralized architecture where a single server maintains google docs and various nodes are connected. Still, if the significant server crashes or gets corrupt, all the nodes get disconnected, and all the work gets destroyed. So the solution to this is decentralized blockchain technology. Decentralized means that no single server or single node is managing the network. Data is replicated to multiple nodes so that if any node goes down, other nodes operate as it is, and original data can quickly be recovered.

## ***What is Ethereum?***

Ethereum is a decentralized blockchain designed to be highly secure, fault-tolerant, and programmable. Ethereum blockchain is a choice for many developers and businesses. As said programmable, the main task of Ethereum is to securely execute and verify the application code known as smart contracts. Ethereum helps to build native scripting language(solidity) and EVM. Ethereum consensus mechanism is proof of work to operate to verify the new transaction. Now we will learn about smart contracts and how it runs on the Ethereum platform.

## **Overview of Smart Contracts**

A smart contract is a small program that runs on an Ethereum blockchain. Once the smart contract is deployed on the Ethereum blockchain, it cannot be changed. To deploy the smart contract to Ethereum, you must pay the ether (ETH) cost. Understand it as a digital agreement that builds trust and allows both parties to agree on a particular set of conditions that cannot be tampered with.

### **Earn Rewards by Writing and Sharing Data Science Knowledge**



Learn | Write | Earn

To understand the need for a smart contract, suppose there was one grocery shop, and ram went to buy some groceries. He purchased the groceries for 500 rupees and kept on debt that would pay the money next month when he returned, so the shopkeeper jotted down his purchase in his ledger. In between the period somehow shopkeeper changed 500 to 600 and when next month ram went to pay the money, the shopkeeper has demanded 600 INR and ram has no proof to show that he has only bought 500 INR so in this case, smart contracts play an essential role which prevents both the parties to tamper the agreement and only gets terminate when all the

conditions satisfy after the deal. There are a couple of languages to write smart contracts, but the most popular is solidity.

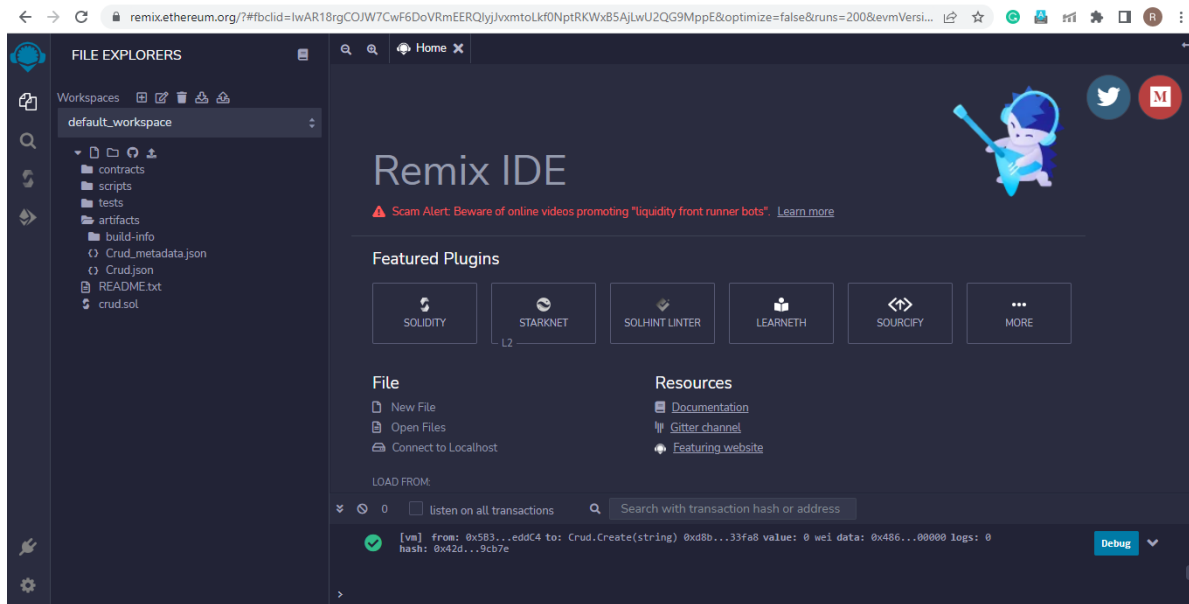
### **Introduction to Solidity Programming**

Solidity is object-oriented, high-level statically-typed programming language used to create smart contracts. Solidity programming looks similar to Javascript, but there are a lot of differences between both languages. In solidity, you need to compile the program first, while in Javascript, you can run the program directly in your browser or by using Node JS. With solidity, you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets. It is also a case-sensitive programming language. Visit the official solidity documentation to read more and be updated about any new functionality release.

### **What is Remix IDE?**

It is an online IDE for creating solid, smart contracts, so you do not need to install or download anything to do any setup. You can develop, deploy, and Administer your solidity smart contract using Remix IDE. Visit [this](#) link to access the Remix IDE where you will find multiple options and a window shown below. The window is a little bit similar to VS code where on the left-hand side, you will find some icons to terminate to other options like a compiler, file explorer, search files, deploy, etc.





Using the file explorer, you can create and open any file. We can set the compiler version, run our smart contract, and observe the output using the compiler. Each compiler type provides a different amount of fake ethers used for practicing purposes. **Solidity Compilation Process**

Smart contract compilation is a critical process to understand how a smart contract runs when created using solidity. We will understand the process using the below flow chart. We can see that the smart contract written in solidity with sol extension first gets the compiler version. After it goes under the compiler, It gets split into two parts where one is Byte code, and the other is ABI (Abstract Binary Interface) key. Byte code is only executed and deployed on the Ethereum blockchain, not the complete smart contract. Whenever any smart contract wants to communicate with this smart contract, they need the ABI key to call functions and variables.

- To observe how ABI and Byte code is generated on Remix IDE, visit IDE, open any contract in the contracts folder, and compile and run it. Scroll down, and you will find two options: ABI and Byte code, where you can copy and paste them into any notepad and observe how your code gets converted to Byte code.

To create a smart contract, the first thing is to define the compiler version to use using the Pragma keyword (you can also determine whether the program supports multiple versions or the version in a particular range); after this, you define the contract using the contract keyword which is same as creating a class in object-oriented programming.

## Important points related to smart contract Compilation

1. Contract Bytecode is public in readable form – It means It does not get encrypted because It will run on different nodes of Ethereum. For then, It needs to decrypt again and again not to increase computation time. It is kept in a readable form.
2. The contract doesn't have to be public – It does not need to keep contracts public, but most organizations keep them public to maintain the trust.
3. Bytecode is immutable
4. ABI act as a bridge between application and smart contract
5. ABI and bytecode cannot be generated without source code

## State and Local Variables in Solidity

Any variable declared on the contract level is known as a state variable. The critical property of the state variable is that it is permanently stored in the blockchain, so you have to pay some amount of gas and use the state variable with care. Solidity does not have a concept of Null or None; indeed, each data type has one default value which on declaration is assigned to that variable. To define Public before any variable or function, automatically, one get function is set with that variable, and you can access its value. Storage to state variable is not dynamically allocated (To initialize state variable with the value, you need to assign a value at declaration time, use constructor, use getter and setter functions). An instance of a contract variable cannot have another state variable besides those already declared. Local variables are those variables that are declared in the function body and are stored in a stack, not in contract storage. Local variables don't cost gas; some types reference the storage by default. Memory keywords cannot be used at the contract level.

## Functions in Solidity

Functions are an essential part of any programming language for the reusability of a particular code. We will see the getter and setter function in solidity to learn how to create a function in solidity. The getter function is a function from which we can access the value of our variables. It is a view-only function, so we can define it as a view or Pure, which states that the value of the state variable cannot be changed it returns the variable's value, so we define the return type of value. On the other side, the setter function changes the value, so it is a simple public function.

```
pragma solidity >= 0.5.0 < 0.9.0;

contract local {

    uint age = 10;

    function getter() public view returns(uint) {

        return age;

    }

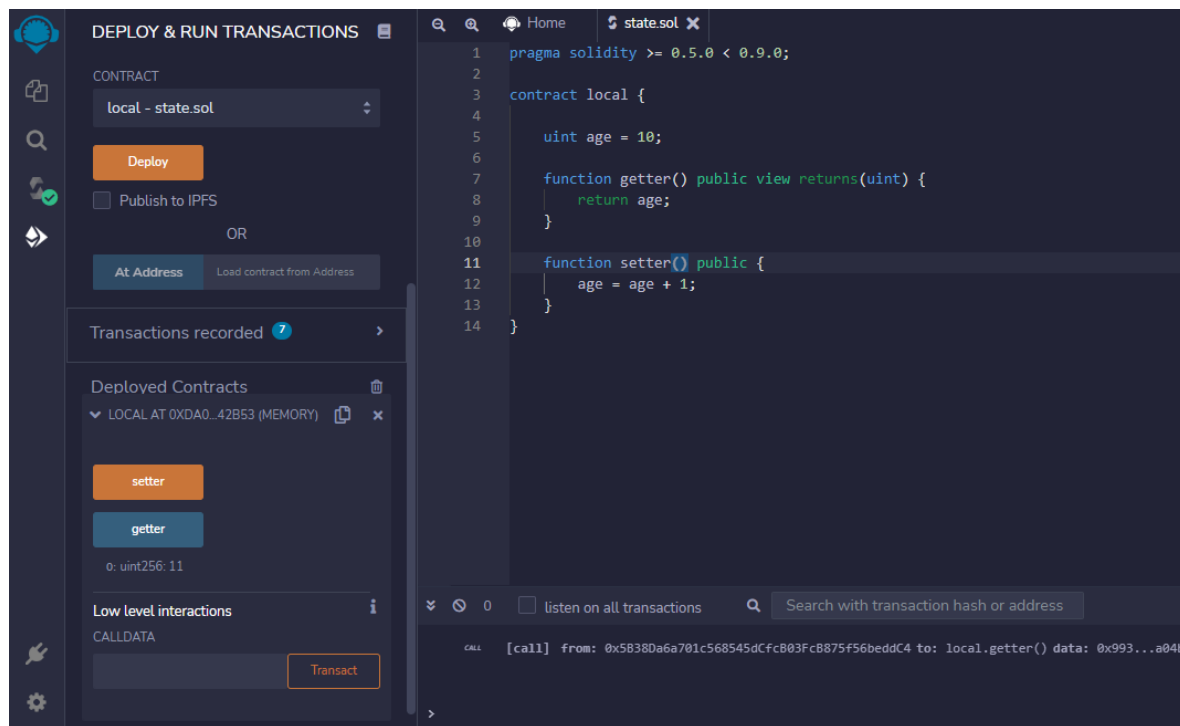
    function setter() public {

        age = age + 1;

    }

}
```

After writing the above code on Remix IDE in a new file with sol extension, you can compile the code, visit the deploy section, and deploy the code to observe the deploy section output as shown below. The value will increase as you click the setter and getter function buttons..



Suppose you want to implement the setter function to set the new value of age so we can pass the parameter in the setter function and set the value of age. Thus in the setter function, we change the matter, so we need to pay a certain amount of gas in the setter function, but the getter function is view-only and does not require any amount of gas to be paid. By default, the visibility of a function is private, so to make it public, we define a function as Public.

### ***Pure and View in Solidity***

We have seen that we use to view and pure where we are not updating the state variable. But pure, you cannot use where it is also reading the state variable. Pure is used where both reading and writing are not performed. Indeed in View, reading is allowed, but writing is not permitted. When we do not define any one of the following to a function, it simply warns that we can provide one restriction of pure or View to function.

### ***Constructor in Solidity***

A constructor is a particular type of function which executes only once when you create your contract. Constructor is used to work with state variables and define smart contracts' owners.

You can create only one constructor, and it is optional to create. The compiler creates a default constructor if there is no explicitly defined constructor.

```
pragma solidity >= 0.5.0 < 0.9.0;
contract local {
    uint public count;
    constructor(uint new_count) {
        count = new_count;
    }
}
```

In the above code, we have created a constructor, and before clicking on deploy, you need to define the value of the constructor before it is called only once, so enter the value and click on deploy, and on scrolling, you can observe the value of count.

### **Control Statements in Solidity**

All programming languages have control statements that help us check multiple conditions using loops and an if-else ladder, and solidity also supports loops and if-else statements.

#### *Loops in Solidity*

Solidity also supports three loops: a while loop, a Do while loop, and a loop. If you are familiar with any other programming language, you must know about control statements and using loops to run particular code multiple times with different values. In solidity, you cannot write the loops directly in contract storage; instead, you must declare them in any function.

- While the loop runs a code snippet multiple times until the condition is proper, the loop terminates when the condition is false. In contrast, the loop runs 0 or multiple times.
- Do while loop is a loop that runs even one time when the condition in the while loop is false. So it is used when you need to run a particular code at least once and if certain conditions meet, then run it multiple times.

- For loop is a loop that is used when you know the start and end time of the loop and how many intervals you need to take. For loop, the initialization and iterator updating are part of loop syntax.

So let us look after the syntax of each type of loop using a sample program.

```
pragma solidity >= 0.5.0 < 0.9.0;
```

```
contract Loops {  
    uint [3] public arr;  
    uint public count;  
    function Whileloop() public {  
        while(count < arr.length) {  
            arr[count] = count;  
            count++;  
        }  
    }  
    function Forloop() public {  
        for(uint i=count; i<arr.length; i++) {  
            arr[count] = count;  
            count++;  
        }  
    }  
    function doWhileLoop() public {  
        do {  
            arr[count] = count;  
            count++;  
        }while(count < arr.length);  
    }  
}
```

### ***If-else Statements in Solidity***

If-else statements are an essential part of any programming language that helps compare two or more two types of values to make a particular decision. Below is the sample code snippet denoting the use of if-else in the solidity that you should try and deploy the contract. After deploying, check by entering the different values.

```
pragma solidity >= 0.5.0 < 0.9.0;
contract Array {
    function check(int a) public pure returns(string memory) {
        string memory value;
        if(a > 0) {
            value = "Greater Than zero";
        }
        else if(a == 0) {
            value = "Equal to zero";
        }
        else {
            value = "Less than zero";
        }
        return value;
    }
}
```

### **Arrays in Solidity**

The array is a special data structure used to create a list of similar type values. The array can be of fixed size and dynamic-sized. With the help of index elements can be accessed easily. below is a sample code to create, and access a fixed-sized array in solidity.

```
pragma solidity >= 0.5.0 < 0.9.0;
contract Array {
```

```

uint [4] public arr = [10, 20, 30, 40];
function setter(uint index, uint value) public {
    arr[index] = value;
}
function length() public view returns(uint) {
    return arr.length;
}
}

```

You can compile and deploy the code to try changing the array elements with an index and printing the array length.

### ***Creating Dynamic Array***

A dynamic array is an array where we can insert any number of elements and delete the details easily using an index. So solidity has functions like push and pops like python, making it easy to create a dynamic array. Below is a code using which you can create a dynamic array. After writing code, compile and deploy the code by visiting the deploy section in the left-side navigation bar. After that, try inserting and deleting some elements from an array.

```

pragma solidity >= 0.5.0 < 0.9.0;
contract Array {
    uint [] public arr;
    function PushElement(uint item) public {
        arr.push(item);
    }
    function Length() public view returns(uint) {
        return arr.length;
    }
    function PopElement() public {
        arr.pop();
    }
}

```



```
}
```

## Structure in Solidity

The structure is a user-defined data type that stores more than one data member of different data types. As in array, we can only store elements of the same data type, but in structure, you can keep elements of different data types used to create multiple collections. The structure can be made outside and inside the contract storage, and the Structure keyword can be used to declare the form. The structure is storage type, meaning we use it in-store only, and if we want to use it in function, then we need to use the memory keyword as we do in the case of a string.

```
pragma solidity >= 0.5.0 < 0.9.0;
```

```
struct Student {
```

```
    uint rollNo;
```

```
    string name;
```

```
}
```

```
contract Demo {
```

```
    Student public s1;
```

```
    constructor(uint _rollNo, string memory _name) {
```

```
        s1.rollNo = _rollNo;
```

```
        s1.name = _name;
```

```
    }
```

```
// to change the value we have to implement a setter function
```

```
function changeValue(uint _rollNo, string memory _name) public {
```

```
    Student memory new_student = Student( {
```

```
        rollNo : _rollNo,
```

```
        name : _name
```

```
    });
```

```
    s1 = new_student;
```

```
}
```

```
}
```

## Create a Smart Contract with CRUD Functionality

We have excellent theoretical and hands-on practical knowledge about solidity, and now you can create a primary smart contract like hello world, getter, and setter contracts. So it's a great time to try making some functional smart contracts, and the best way to try all the things in one code is to create one program that performs all CRUD operations.

```
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");
}
}

```

## Conclusion

Solidity is an object-oriented high-level programming language for creating a smart contract that runs on the Ethereum blockchain. We have learned about the smart contract and its creation using solidity programming. Let us conclude the article with the main critical points through learning from an article.

1. The smart contract is a digital agreement containing some condition on which both signing parties agree, and when the state meets, the smart contract automatically gets terminated.
2. The smart contract is created using some programming language in which solidity is the leading choice and allows different people to collaborate, coordinate, and cooperate to make transactions that do not trust each other.
3. Solidity is the same as another programming language where creating a class is the same as creating a contract and differs in some exceptions like string should always be given with memory in function.
4. The syntax of solidity programming is 90 percent the same as javascript. Still, javascript directly runs on the browser while solidity runs on the Ethereum blockchain platform, which needs a certain amount of Ether (ETH) to deploy the contract and gas to create a transaction.

5. Blockchain technology today is booming, and many businesses across the globe are trying to acquire blockchain technology in many domains of finance, insurance, security, etc

## Assignment No.5

Aim :- Write a survey report on types of Blockchains and its real time use cases

Objective: Student will be able to learn

1. Concept of types of Blockchains
2. survey report on types of Blockchains and its real time use cases

Theory:-

### Blockchain & Types

Blockchain technology is being used to carry and transfer the transactions or exchange of information through a secure network. Blockchain technology and distributed ledger technology is used parallel to the digital cryptocurrency to the people. Blockchain is being used for the purpose of private networking and uses too where only the restricted network users can get the authorization and access. Here network administrators are authorized to administrate the activities and any new nodes or users who wish to get permission, need to contact with the system or network administrators. Primarily there are two types of Blockchain technology viz. private Blockchain and public Blockchain. Though based on some other criteria and analysis Blockchain technology can also be noted and called as consortium blockchain technology, and hybrid blockchain technology. It is important to note that every kind of Blockchain basically consists of a cluster of nodes, and this is working on the peer-to-peer (P2P) network system.

Every node in the network has a copy of the shared ledger and further that is timely updated and also being verified the transactions, initiate and receive transactions. Keeping in mind the broad nature, experts

classified Blockchain Technology into following three

^ ^ Public Blockchain,

^ ^ Private Blockchain, and

^ ^ Hybrid Blockchain.

1. **Public Blockchain** is a major type of Blockchain, and that is not only open but also decentralized in nature. And in this type of Blockchain technology computer networks are basically accessible to anyone interested in transactions. Here based on validation the validated person basically receives the transaction rewards and furthermore, two kinds of Proof-of-work and Proof-of-stake models are being used. The Public Blockchain is furthermore a non-restrictive and distributed ledger system which is doesn't seek any kind of permission, and anyone having access can be authorized one to get the data or part pf the Blockchain. This kind of Blockchain also gives authorization regarding the

current and past records verification. Additionally, this is being used for mining and exchanging cryptocurrencies

In this segment most common is Bitcoin and Litecoin blockchains. It is mostly secure upon following strict security rules as well as methods. However, upon non-following the security protocols it may be risky. Some of the examples of this type of Blockchain are—

^ ^ Bitcoin,

^ ^ Ethereum,

^ ^ Litecoin.

Here Fig. 3 depicted some of the features and advantages regarding Public Blockchain Systems and Technology.



Fig. 3: Salient features and functions of the public blockchains

There are two common examples of public blockchains and these are Bitcoin and Ethereum as per the experts This type of Blockchain is concerned with the following type of features viz.

High Security and Privacy,

- ^ ^ Open and Flexible Environment,

- Anonymous Nature,
- No regulations and strict Policies,
- Full Transparency and Systems.
- Distributed, etc.

However, according to the experts, the following are being considered as important advantages and benefits of the Public Blockchain.

#### Trustable and Faith

Public Blockchain is trusted and here unlike private blockchains, the participants don't need to think of authenticity. In this type of Public Blockchain, they don't need to know other nodes, and therefore there is no fraud in the transactions. In this category nodes can contact blindly without feeling the need to trust individual nodes.

#### Secure and Safe

In the Public Blockchain, there are opportunities in connecting with the other participants and nodes in the same public network, and this results in secure, large, and greater communication and participation. Owing to this feature, it is difficult for the attackers to enter the systems and here every node will do the verifications and transactions as per norms. Here thoughtful cryptogenic encrypting methods are being used and therefore it is much safer than the private blockchain according to some experts.

#### Open and Transparent

Public Blockchain is also having the features of openness and here data is basically transparent to all the nodes and in this mechanism, one blockchain record is normally available to all the authorized nodes. Therefore, here all the nodes become open and transparent and there is an absence of fake transactions or hiding any information. Though there are plenty of advantages and benefits but it is also having a different kind of disadvantages and weaknesses, and some of them are as under.

#### Lower Transaction per Second

In Public Blockchain System the rate of transaction per second is also very low, and this is due to having a large number of nodes and huge network. Here each node has to verify the transaction and also do

proof-of-work is time consuming. Here in public systems seven (07) transactions happen per second and additionally, here Ethereum network has about a 15 TPS rate.

## **Scalability Matters**

Similar to the previously mentioned issue on a lower transaction per second in public blockchain another issue is scalability according to the experts. The huge size basically creates the scalability in this regard and here bitcoins lightning networks are considered as important to overcome the problem according to the experts.

## **High Energy Consumption**

The public blockchain also suffers from higher energy consumption due to the proof-of-work energy consumption. As it needs special algorithms therefore high energy consumption is considered as important in energy, environment, and financial context.

## **2. Private blockchains**

Private Block Chain are restricted and not open, such kind of blockchain also has features of access. This blockchain allows permission for the transaction from the support of the system administrator

Private blockchain solutions develop these platforms having the features of the following—

- ^ ^ Full of privacy,
- ^ ^ High efficiency,
- ^ ^ Faster transaction,
- ^ ^ Better scalability,
- ^ ^ Faster and speediness.

This type of blockchain is works on closed systems and networks only and these are usually useful in the organizations, enterprises from which only selected members can be joined. This type of blockchain contains proper security, authorizations, permissions as well as accessibility. According to the experts, private blockchains are deployed for voting, regarding supply chain management, for finding and managing digital identity, regarding asset ownership, and so on. There are certain popular private blockchains like Multichain, Hyperledger projects, Corda, etc. Participants are





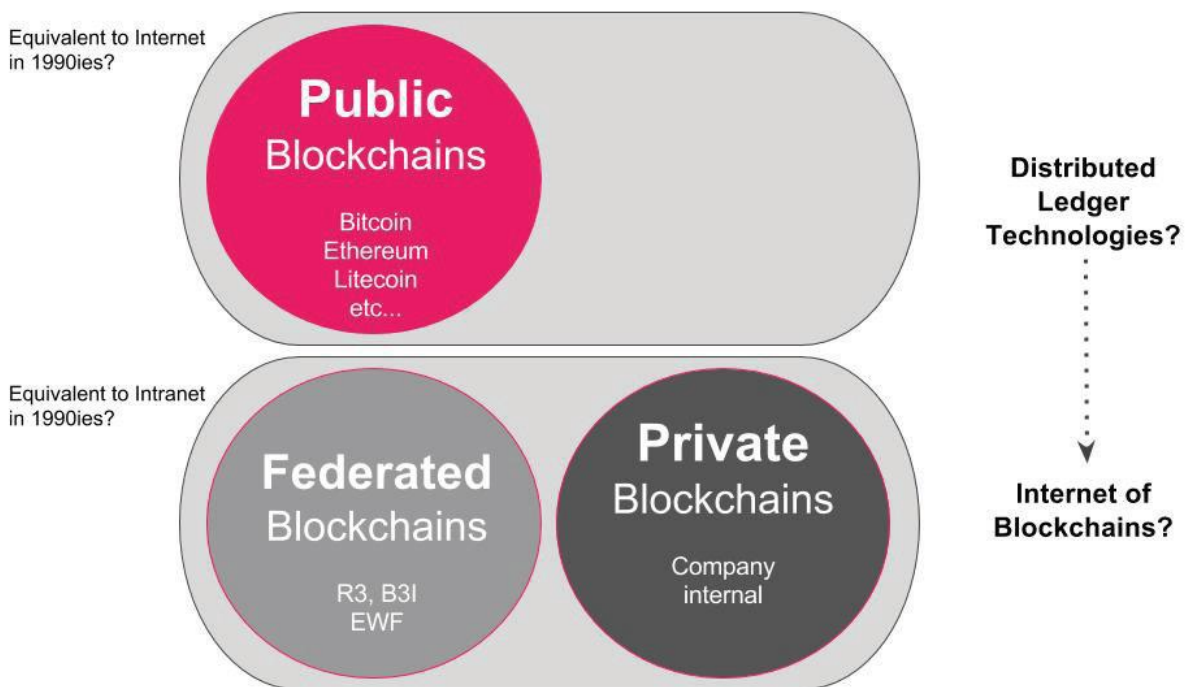
Fig. 4: Salient features and functions of the private blockchains

Private blockchains as running with the authorized nodes; therefore no one from the outside of the private network is able in accessing information and transaction related data exchanged between two nodes.

3. **Hybrid Blockchains** is a merger of public blockchain as well as private blockchain and it is required in better control for achieving higher goals. Hybrid Blockchain deals with centralized and decentralized systems and it is not open; however, it has the features of integrity, transparency, as well as security. It has several advantages over traditional blockchains as depicted in Fig. 5. In Hybrid Blockchains maximum customization is being considered as main benefits with private permission-based system as well as a public permission-less system. In this type of blockchain systems users are able in getting access and selected sections and rest can be recorded or keeps safe due to the benefits of the records from the ledger. Hybrid Blockchains is flexible enough so that users can join easily as private blockchain. This type of blockchain is able in enhancing the security and transparency of the blockchain network’

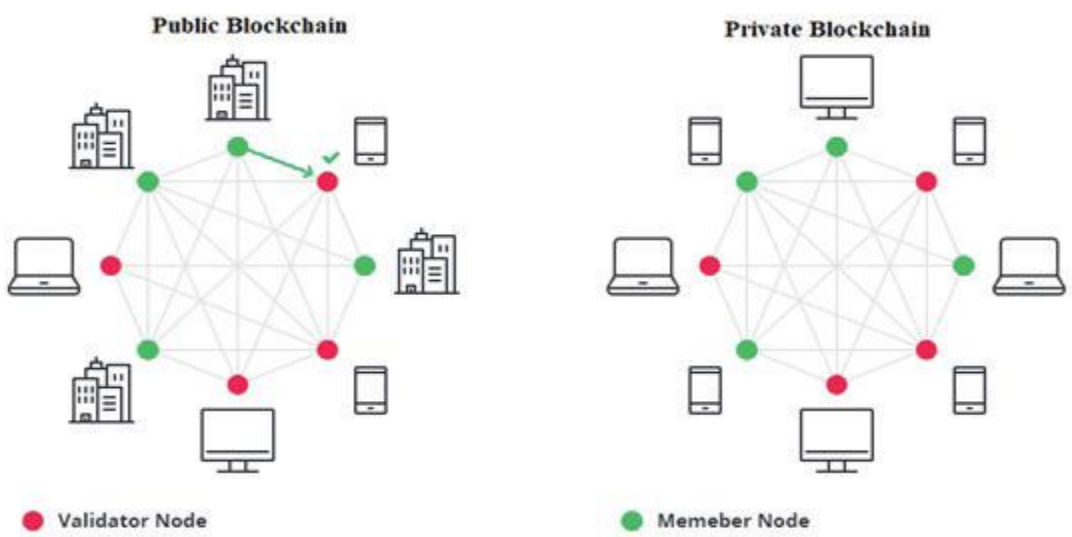
4. **Consortium Blockchain** is another type of semi-decentralized type of blockchain, and this type of blockchain is able in the organization of managing the blockchain network. This type of blockchain is able in doing activities even from a single organization. Here blockchain is able in exchange information or do the mining and are being used in the areas such as banks,

government organizations, etc. Some of the examples of this type of consortium are Energy Web Foundation, R3, etc



**Fig. 5:** Types of Blockchains and roadmaps

Therefore, in a nutshell, all the Blockchains are having their own benefits and advantages and as a whole public and private is considered as major or worthy in terms of operations (as depicted in Fig. According to the experts security, scalability, and transparency are considered as worthy and main points in the Blockchains of public and private types. It is important to note that private blockchains are not trustworthy; while the public network is important in proof-of-work based.



**Fig. 6:** Architecture wise differences in public and private blockchains





## Top blockchain use cases


Blockchain is "a general-purpose technology, which means it is applicable across sectors," said Christos Makridis, a research professor at Arizona State University, senior adviser at Gallup, digital fellow at Stanford University's Digital Economy Lab and CTO at arts and education technology startup Living Opera. "For example, financial services can use it to write [smart contracts](#) between consumers and their banking institution. Similarly, healthcare can use it to write smart contracts between insurers and hospitals, as well as between patients and hospitals. The possibilities are endless."

[Blockchain use cases continue to expand](#). Here are some common commercial applications:

1. **Smart contracts.** The primary function of computer programs called "smart contracts" is to automate the execution of contract terms when conditions warrant them. The computer code follows a relatively simple command of "when/if \_then\_\_\_" to ensure that all parties receive the benefits or penalties as the contract stipulates and actions require. Smart contracts are useful to, and used by, most industries today for a variety of uses traditionally governed by paper contracts. The blockchain also makes a permanent record of every action and reaction in the transaction.
2. **Cybersecurity.** Blockchains are highly secure because of their permanency, transparency and distributed nature. With [blockchain storage](#), there's no central entity to attack and no centralized database to breach. Because blockchains are decentralized, including those privately owned, and the data stored in each block is unchangeable, criminals can't access the information. "Essentially, the intruder needs keys to many different locations versus just one," Makridis noted. "The computing requirements for the intruder grow exponentially."

## No weak links

 <p><b>CHRISTOS MAKRIDIS</b> Research professor, Arizona State University</p> <p>"Essentially, the intruder needs keys to many different locations versus just one. The computing requirements for the intruder grow exponentially."</p>	 <p><b>NIR KSHETRI</b> Professor, University of North Carolina-Greensboro</p> <p>Companies can "benefit from transparency, commercial confidentiality of data, and an immutable record of transactions."</p>	 <p><b>RAJAT KAPUR</b> Senior manager, blockchain practice, EY</p> <p>"Specifically, the need to preserve data integrity and the ability to build new revenue or business models are the top drivers for more than half of all respondents."</p>	 <p><b>CHEN ZUR</b> EY U.S. blockchain practice leader</p> <p>Respondents said "they prioritize use cases that would improve efficiencies ... like supply chain track and trace, payment support processes and digitization of document flows."</p>
---	---	---	--

SOURCE: TESTAMENTS. ALL RIGHTS RESERVED. 

3. **IoT.** Two primary IoT uses of blockchains are in the supply chain sector and for asset tracking and inventory management. A third use is in recording measurements made by machines whether those sensors are in the Arctic, the Amazon jungle, a manufacturing plant or on a NASA drone surveying Mars. "Whether it be reports of chemical data regarding oil grades or tracking shipments of electronics across the world through various ports of entry, the blockchain can be utilized anywhere there is data interacting with the real world," explained Aaron Rafferty, CEO of cryptocurrency investment firm R.F. Capital.
4. **Cryptocurrencies.** The blockchain concept was originally developed to manage digital currencies such as bitcoin. While the two technologies still compete against each other in alternative transactions, they've also been separated so blockchains could serve other purposes. Given the anonymity of crypto coins, blockchain is the only way to document transactions with accuracy and privacy for the parties involved.
5. **NFTs.** Nonfungible tokens are units of data certified to be unique and not interchangeable. In short, they are digital assets. According to Rafferty, NFTs are revolutionizing the digital art and collectibles world. "We are using decentralization and the ethereum blockchain to create a music live stream network where artists and streamers can connect with fans directly, sell their NFTs, receive contributions from fans and trade in their rewards and contributions for

crypto tokens," said Shantal Anderson, founder and CEO of music and pop culture streaming network Reel Mood.

**Conclusion :- We Have Study survey report on types of Blockchains and its real time use cases.**

## Assignment No.6

Aim :- Write a program to create a Business Network using Hyperledger

Objective: Student will be able to learn

1. Concept of types of Hyperledger
2. create a Business Network using Hyperledger

### Theory :-

#### Self-paced training

We offer blockchain introduction, Hyperledger for system admin, Ethereum, Solidity, Corda R3, Hyperledger for developers, blockchain cybersecurity and more classes in self-paced video format starting at \$60. Click [here](#) to learn more and register. For complete self-paced blockchain training, visit our Complete Blockchain Development Training page.

#### Recipe Overview

**Hyperledger Composer** is a set of collaboration tools for business owners and developers that make it easy to write chaincode for Hyperledger Fabric and **decentralized applications (DApps)**. With Composer, you can quickly build POC and deploy chaincode to the blockchain in a short amount of time. Hyperledger Composer consists of the following toolsets:

- **A modeling language called CTO:** A domain modeling language that defines a business model, concept, and function for a business network definition
- **Playground:** Rapid configuration, deployment, and testing of a business network
- **Command-line interface (CLI) tools:** The client command-line tool is used to integrate business network with Hyperledger Fabric

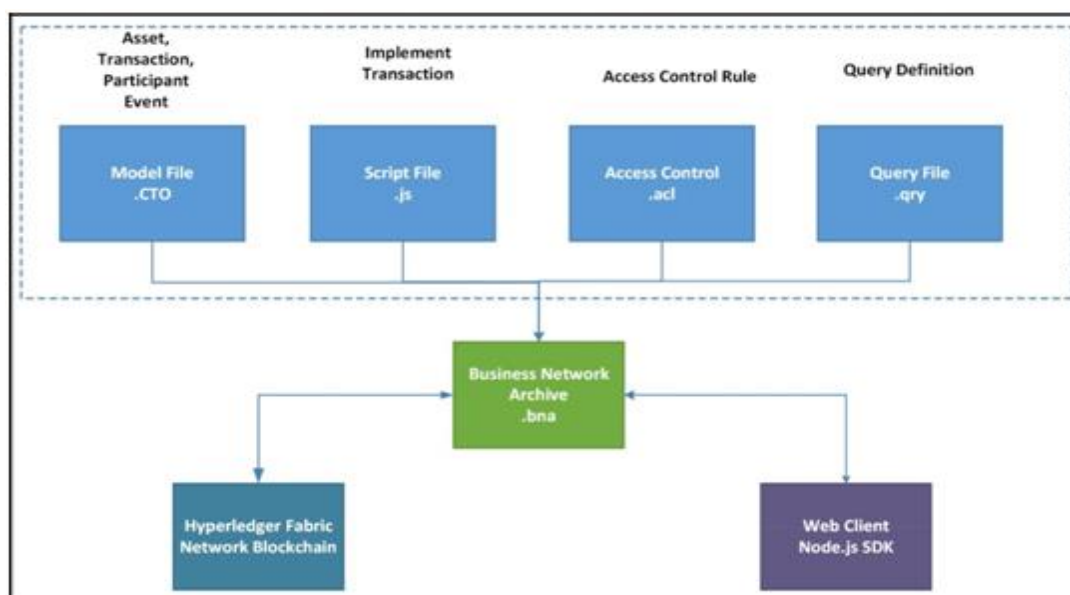
In this recipe, we will explore the Composer business network and development components, including implementing models, transaction logic, access control, and query definitions. We will set up a development environment, and cover the use of the Hyperledger Playground for testing. For those who are not familiar with Hyperledger project, reading Blockchain Overview, Intro to Hyperledger Family and Hyperledger Blockchain Ecosystem, Hyperledger Design Philosophy and Framework Architecture, Overview of Building Blockchain Smart Contracts in

Hyperledger and The Survey of Hyperledger Fabric Architecture and Components for Blockchain Developers articles are strongly recommended beforehand.

In this recipe, we will cover the Hyperledger Composer business network and development components followed by Setting up the Hyperledger Composer Prerequisites & Development environment and Configuring a Hyperledger Composer business network recipes. After completing 3 mentioned recipes, it's recommended to learn the following topics: i- Implementing models, transaction logic, access control, and query definitions, ii- Deploying, testing, and exporting business network archives using the Composer command-line interface and iii- Interacting with Composer through the RESTful API.

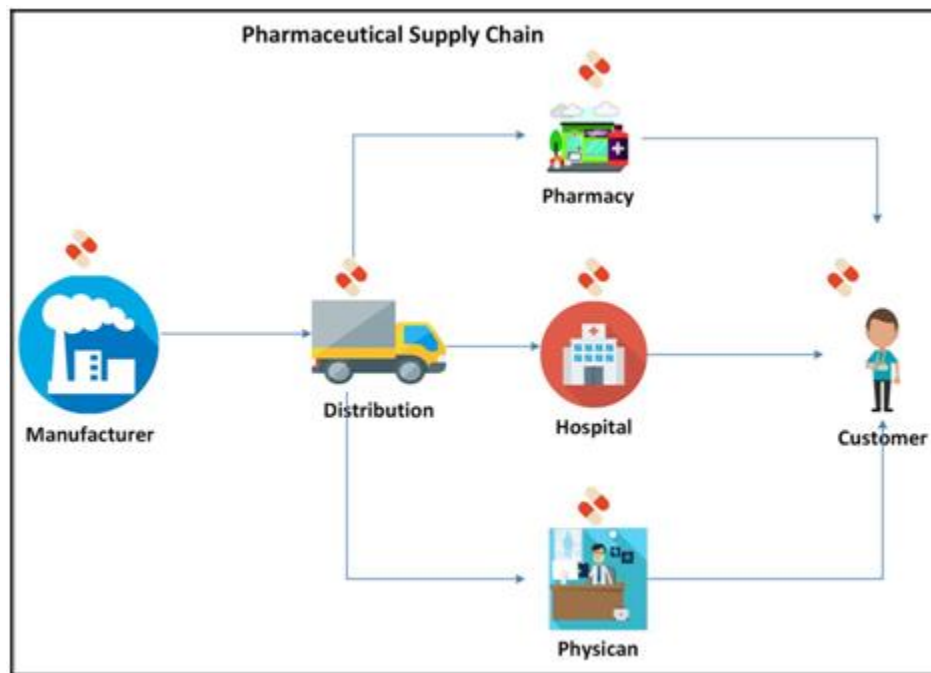
### **The Hyperledger Composer Business Network and Development Components**

You can use Hyperledger Composer to quickly build a business network. Here, you can define assets, participants, transactions, access-control rules, and optional events and queries in the business network. A model (.cto) file contains all of the preceding definitions in the business network. Asset is an associate with a real-world object, and participants have their own unique identity. The transactions will interact with assets. Participants can do this via transactions. The structure of a business network also includes an access control (permissions.acl), which specifies the access-control rules, a script (logic.js) file that implements transaction logic, and a package.json file that contains project metadata:



## Pharmaceutical Supply Chain Business Network

In this recipe, we will develop an application called **FarmaTrace Enterprise (FTE)**, which will use the Hyperledger Composer tool. FTE will enable end-to-end tracking of medicinal products throughout the pharmaceutical supply chain. There are a variety of challenges that the pharmaceutical supply chain and its stakeholders face daily. The process is time-consuming and information about it is not efficient and transparent to all parties. Creating a flexible and holistic supply chain strategy can allow companies to have better control over processes, enhance communication, and reduce costs:



We aim to provide a general design and implementation approach for Composer applications. This commonly-used procedure will help you to understand how to quickly start. In the FarmaTrace application, we will build a business network and components. Here, we show how a typical pharmaceutical supply chain works. However, it could be much more complex in a real-world use case. The following is how the FarmaTrace process flow works.

### Pharmaceutical Supply Chain Process flow

First, we need to understand the entire process flow:

The **manufacturer** produces drugs, then packages and labels them

The **distribution** company gathers packaged products and begins delivery

The **distribution** company sends drugs to **pharmacies**

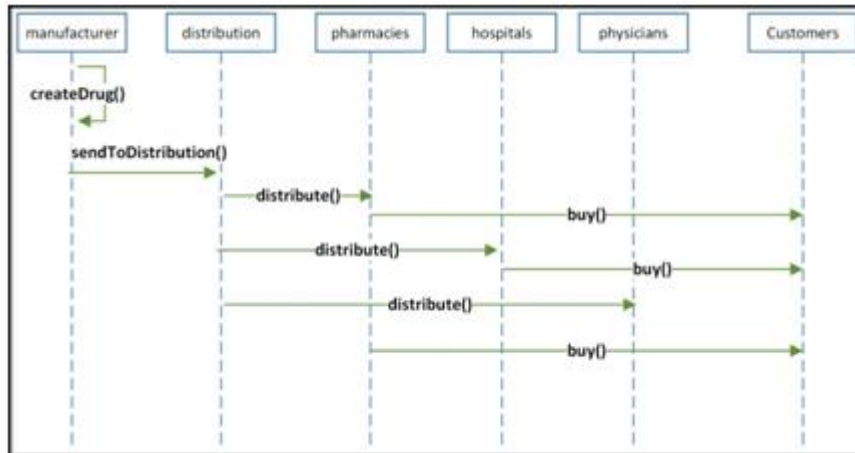


The **distribution** company sends drugs to **hospitals**

The **distribution** company sends drugs to **physicians**

**Customers** buy drugs from a pharmacy, hospital, or physician

Process flow can be illustrated as below:



### Pharmaceutical Supply Chain Entities

Next, we need to identify entities for a FarmaTrace application; participants are categorized into six entities:

- Manufacturer
- Distribution
- Pharmacy
- Hospital
- Physician
- Customer

### Pharmaceutical Supply Chain Assets

We will record every transaction in the blockchain; information for these records is stored in a receipt asset object. All participants in the network can trace this receipt. The drug receipt will have all process evidence information for each transaction. This can be used to provide proof for certain steps, which need to display required documents. The asset also defines certain rules that

only allow authorized parties to perform permitted transaction actions. The receipt will contain other asset information, such as the drug status, to trace the transaction proof in each step.

### **Pharmaceutical Supply Chain Query**

Now we will check which parameters are needed to allow us to search for drug information:

- Drug ID
- Drug name
- FDA action date
- Marketing status (prescription, over-the-counter, or discontinued)
- Approval type (type of supplement or other regulatory action)
- Search participants by ID

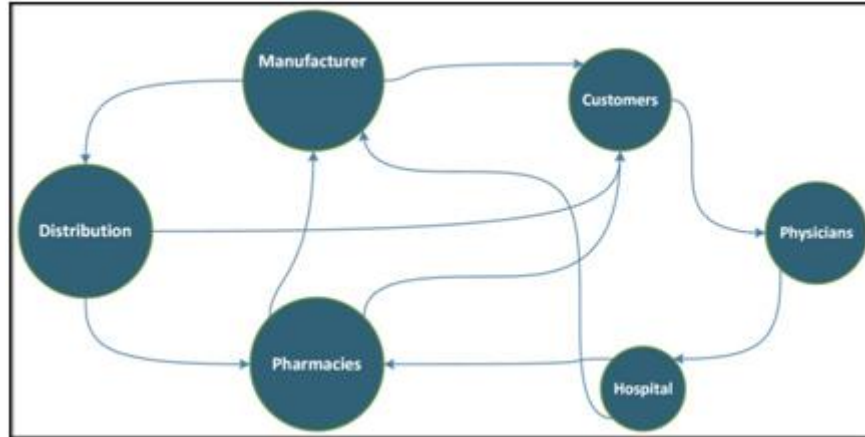
We have analyzed the FarmaTrace application business use case and development components. To build a business network, we will create all of the following critical files:

- **Model file:** org.packt.farmatrace.cto
- **Script file:** logic.js
- **ACL file:** permissions.acl
- **Query file:** queries.qry

We will create these files after we set up a development environment in the Configuring a Composer business network recipe.

### **Putting Things Together**

Designing network topology is a critical step for an enterprise blockchain. For our FarmaTrace application, the deployed Composer business network will look like the following:



All six participants are attending the network and can communicate with each other. The methods written in farmatrace-logic.js will be converted into Fabric chaincode. By invoking chaincode, the Fabric order service will provide a shared communication channel and broadcast transactions to peers. Peers will verify the transactions and commit and apply the same sequence of transactions. Eventually peers will update their state in the same way.

**Conclusion: We Have Study a Business Network using Hyperledger**