Assignment no. 1

Aim :- Installation of Metamask and Study spending Ether per transaction:-

Theory :-

MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications. MetaMask is developed by ConsenSys Software In, a blockchain software company focusing on Ethereum-based tools and infrastructure.

MetaMask allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser.

Websites or other decentralized applications are able to connect, authenticate, and/or integrate other smart contract functionality with a user's MetaMask wallet (and any other similar blockchain wallet browser extensions) via JavaScript code that allows the website to send action prompts, signature requests, or transaction requests to the user through MetaMask as an intermediary.

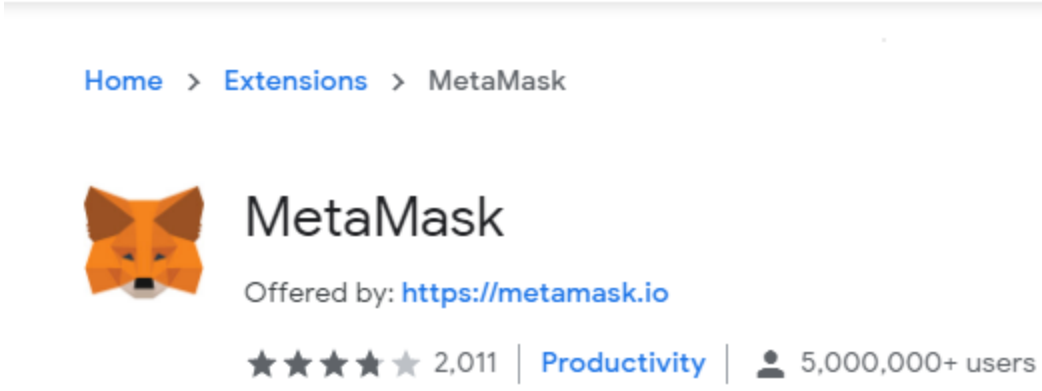
The application includes an integrated service for exchanging Ethereum tokens by aggregating several decentralized exchanges (DEXs) to find the best exchange rate. This feature, branded as MetaMask Swaps, charges a service fee of 0.875% of the transaction amount.

As of November 2021, MetaMask's browser extension had over 21 million monthly active users, according to Bloomberg.

Installation Process:-

MetaMask is a type of Ethereum wallet that bridges the gap between the user interfaces for Ethereum (For example, Mist browsers, DApps) and the regular web (For example, Google Chrome, Mozilla Firefox, websites). Its function is to inject a JavaScript library called web3.js into the namespace of each page your browser loads. Web3.js is written by the Ethereum core team. MetaMask is mainly used as a plugin in the web browser. Let’s walk through the steps to install it on Google Chrome.

Step 1: Go to Chrome Web Store Extensions Section.
Step 2: Search MetaMask.
Step 3: Check the number of downloads to make sure that the legitimate MetaMask is being installed, as hackers might try to make clones of it.



Step 4: Click the Add to Chrome button.
Step 5: Once installation is complete this page will be displayed. Click on the Get Started button.
Step 6: This is the first time creating a wallet, so click the Create a Wallet button. If there is already a wallet then import the already created using the Import Wallet button.
Step 7: Click I Agree button to allow data to be collected to help improve MetaMask or else click the No Thanks button. The wallet can still be created even if the user will click on the No Thanks button.
Step 8: Create a password for your wallet. This password is to be entered every time the browser is launched and wants to use MetaMask. A new password needs to be created if chrome is uninstalled or if there is a switching of browsers. In that case, go through the Import Wallet button. This is because MetaMask stores the keys in the browser. Agree to Terms of Use.

Results:-

Include your snapshots of Installation Process

Assignment no. 2

Aim :- Create Your Own Wallet using Metamask for Crypto Transactions:-

Theory:-

To access this new world we need the tool called “MetaMask”, the connection to the new web. It’s a key that connects the users to new types of applications, a wallet that keeps the data and valuables safe and sound. This also works as a shield that protects its users from hackers and data collectors. With metamask the users are easily ready to explore this new internet safely and securely. The possibilities in this new world are endless from using next generation applications that provide control back to the users and the community. They can move seamlessly between sites and send money across the world at a fraction of the cost.

The user controls its own identity on this world, once the MetaMask is installed you are ready to explore the best version of the internet.

Software cryptocurrency wallet used to mainly interact with Ethereum blockchain. metamask was created by ConsenSys(founded by Aaron Davis), which is a software company involved in the creation of Ethereum based tools. The easy accessibility of this wallet through browser extensions makes it the most popular and commonly used metaverse wallets.

The process seems to be a little complicated but it isn't difficult to use a metamask and it is an extremely convenient metaverse wallet option. Users can benefit from the recent update that enables them to slap tokens from any decentralised exchange. Users can also configure multiple addresses for holding their tokens.

Steps to create your own MetaMask Wallet

Step 1. Installing MetaMask on your browser

You will require to install the extension first to get started with MetaMask wallet. There are various marketplaces to find it, depending on your browser.

Most common browsers like Chrome, Firefox and Opera have metamask on their stores itself so finding it is not that tough.

1. Open google and click on Extensions.
2. Click on Install MetaMask.
3. Click Add to Chrome
4. Click Add Extension

Step 2. Creating an Account

1. Go on the upper right Corner of google home page and click on MM extension icon

2. You will be directed to create a new password.
3. Click Create and proceed by pressing Next and accept the terms of use.

The Metamask account was built successfully.

Step 3. Creating a Wallet

1. Click on the option to create a new Wallet.
2. Choose between “agree to” or “opt out” of MetaMask usage data gathering.
3. You will be directed to create a password. This password will make you login to your MM account through the browser extension or app on your smartphone.
4. This step includes adding your seed phrase or secret backup phrase. You will obtain a randomly developed series of words and will be requested to verify this word by entering it back into the browser or app. After confirming you will reach the main page of your MetaMask Wallet.

Step 4. Configure Settings

1. Click on the account photo in the top right of the main page. This will lead you to the general settings page. Account name, contact list and notification settings can be revised here.
2. Your wallet's unique Ethereum address is recorded straight below the Account 1 area. This address can be utilised to interact with the Ethereum blockchain and to send and receive the best nft tokens.

Depositing Funds

Click on View Account. You can see your public address here. You can receive or buy ETH and other tokens by this address.

MetaMask outperforms other wallets regarding ease of access with the Ethereum DApp world with its effortless setup and availability on both mobile and desktop platforms. The DeFi sector has seen a wide rise because of the MetaMask Browser extension.

Secure your wallet

MetaMask fees

It allocates gas fees for each transaction according to the situation of the ethereum network. Users have an option to adjust the gas fees and gas limits. It is available in the advanced tab provided that enables parameter customization. fees are the expense of processing transactions, while gas limits are the highest transaction expenses a user is ready to bet on a single transaction.

Results:-

Include your snapshots of Wallet Creation Process

Assignment no. 3

Aim:- Write a Smart Contract on a Test Network for Bank Account of a Customer for following Operations :-

- **Deposit Money**
- **Withdraw Money**
- **Show Balance**

Theory :-

A Smart Contract (or cryptocontract) is a computer program that directly and automatically controls the transfer of digital assets between the parties under certain conditions. A smart contract works in the same way as a traditional contract while also automatically enforcing the contract. Smart contracts are programs that execute exactly as they are set up(coded, programmed) by their creators. Just like a traditional contract is enforceable by law, smart contracts are enforceable by code.

How smart contracts work –

A smart contract is just a digital contract with the security coding of the blockchain. A smart contract has details and permissions written in code that require an exact sequence of events to take place to trigger the agreement of the terms mentioned in the smart contract. It can also include the time constraints that can introduce deadlines in the contract.

This contract is embedded in the blockchain making it transparent, immutable, inexpensive and decentralized. Every smart contract has its address in the blockchain. The contract can be interacted with by using its address presuming the contract has been broadcasted in the network.

The bitcoin network was the first to use some sort of smart contracts by using them to transfer value from one person to another. The smart contract involved employs basic conditions like checking if the amount of value to transfer is actually available in the sender account. Later, ethereum platform emerged which was considered more powerful, precisely because the developers/programmers could make custom contracts in a Turing-complete language. It is to be noted that the contracts written in the case of bitcoin network were written in a Turing-incomplete language, restricting the potential of smart contracts implementation in the bitcoin network.

The idea behind smart contracts is pretty simple. They are executed on a basis of simple logic, IF-THEN for example:

- **IF** you send me the object A, **THEN** the sum (of money, in cryptocurrency) will be transferred to you

- **IF** you transfer a certain amount of digital assets (cryptocurrency, for example, ether, bitcoin), **THEN** the A object will be transferred to you
- **IF** I finish the work, **THEN** the digital assets mentioned in the contract will be transferred to me

Features –

- **Trust:**
  The smart contract can't be lost as its embedded in the blockchain itself.
- **Accuracy:**
  Smart contracts are accurate to the limit a programmer has accurately coded them for execution.
- **Speed:**
  Smart contracts use software code to automate tasks, thereby reducing the time it takes to maneuver through all the human interaction related processes. Because everything is coded, the time taken to do all the work is the time taken for the code in the smart contract to execute.
- **Backup:**
  Every node in the blockchain maintains the shared ledger, providing probably the best backup facility.
- **Autonomy:**
  There is no third part involved. The contract is made by you and shared between the parties. No intermediaries involved which minimizes bullying and grants fully authority to the dealing parties. Also, the smart contract is maintained and executed by all the nodes on the network, thus removing all the controlling power from any one party's hand
- **Safety:**
  Cryptography can make sure that the assets are safe and sound. Even if someone breaks the encryption, the hacker will have to modify all the blocks that come after the block which has been modified. Please note that this is a highly difficult and computation intensive task and is practically impossible for a small or medium sized organisation to do.
- **Savings:**
  Smart contracts save money as they eliminate the presence of intermediaries in the process. Also, the money spent on the paperwork is minimal to zero.

Example Use Cases –

1. Real Estate. Reduce money paid to the middleman and distribute between the parties actually involved. For example, a smart contract to transfer ownership of an apartment once a certain amount of resources have been transferred to the seller's account(or wallet).
2. A smart contract can be deployed in a blockchain that keeps track of vehicles maintenance and ownerships. The smart contract can, for example, enforce vehicle maintenance service every six months; failure of which will lead to suspension of driving license.

3. The music industry could record the ownership of music in a blockchain. A smart contract can be embedded in the blockchain and royalties can be credited to the owner's account when the song is used for commercial purposes. It can also work in resolving ownership disputes.
4. Government elections. Once the votes are logged in the blockchain, it would be very hard to decrypt the voter address and modify the vote leading to more confidence against the ill practices.
5. Management. The blockchain application in management can streamline and automate many decisions that are taken late or deferred. Every decision is transparent and available to any party who has the authority(an application on private blockchain). For example, a smart contract can be deployed to trigger the supply of raw materials when 10 tonnes of plastic bags are produced.
6. Automating healthcare payment processes using smart contracts can prevent fraud. Every treatment is registered on the ledger and in the end, the sum of all the transactions can be calculated by the smart contract. The patient can't be discharged from the hospital until the bill has been paid can be coded in the smart contract.
7. In supply chain.

//CODE//

Assignment No 3 Smart Contract for Bank Account

```
pragma
solidity
^0.4.19;

        //declared the solidity version

        //defining a contract

        contract MyContract {

           address private owner;

           //define a construstor

           constructor() public {

              owner=msg.sender;

           }


           //function to get the address of the owner

           function getOwner()


              public view returns (address) {

                 return owner;

              }

              //function to get return the balancr of the owner

              function getbalance()

              public view returns(uint256){

                 return owner.balance;
```

```
        }
        //function to deposit money
        function deposit() external payable{
           require(msg.value == 2 ether, "Please send two ethers");
        }
        //Function to withdraw money
        function withdraw() external{
           require(msg.sender == owner, "No");
           msg.sender.transfer(address(this).balance);
        }
    }
```

Results :- Include Printout with Snapshot of Output

Assignment no. 4

Aim:- Write a Program in Solidity to create students data. Use the following Constructs

- **Structures**
- **Arrays**
- **Fallback**

Deploy this as Smart Contract on Etherum and Observe the transaction fees and Gas Value

Theory :-

Solidity is an object-oriented programming language for implementing smart contracts on various blockchain platforms, most notably, Ethereum. It was developed by Christian Reitwiessner, Alex Beregszaszi, and several former Ethereum core contributors.Programs in Solidity run on Ethereum Virtual Machine.

Smart Contract in Solidity

Solidity's code is encapsulated in contracts which means a contract in Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. A contract is a fundamental block of building an application on Ethereum.

```
// Solidity program to demonstrate how to write a smart contract

pragma solidity >= 0.4.16 < 0.7.0;

 // Defining a contract

contract Storage

{      // Declaring state variables

   uint public setData;

    // Defining public function that sets the value of the state variable

   function set(uint x) public

   {       setData = x;    }

       // Defining function to print the value of state variable

   function get(

   ) public view returns (uint) {

      return setData;

   }   }
```

Explanation:

1. Version Pragma:

pragma solidity >=0.4.16 <0.7.0;

Pragmas are instructions to the compiler on how to treat the code. All solidity source code should start with a “version pragma” which is a declaration of the version of the solidity compiler this code should use. This helps the code from being incompatible with the future versions of the compiler which may bring changes. The above-mentioned code states that it is compatible with compilers of version greater than and equal to 0.4.16 but less than version 0.7.0.

2. The contract keyword:

```
contract Storage{
//Functions and Data
}
```

The contract keyword declares a contract under which is the code encapsulated.

3. State variables:

uint public setData;

State variables are permanently stored in contract storage that is they are written in Ethereum Blockchain. The line uint setData declares a state variable called setData of type uint (unsigned integer of 256 bits). Think of it as adding a slot in a database.

4. A function declaration:

function set(uint x) public

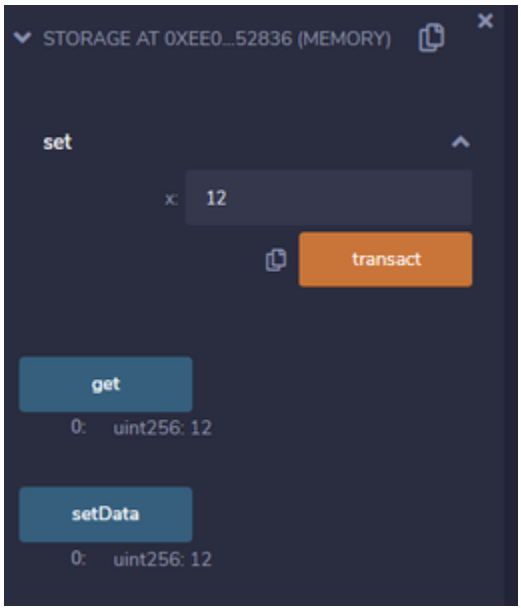
function get() public view returns (uint)

This is a function named *set* of access modifier type *public* which takes a variable *x* of datatype *uint* as a parameter.

This was an example of a simple smart contract which updates the value of setData. Anyone can call the function set and overwrite the value of setData which is stored in Ethereum blockchain and there is possibly no way for anyone to stop someone from using this function. This is an example of a decentralized application that is censorship proof and unaffected to the shutdown of any centralized server. As long as someone is running a single node of Ethereum blockchain, this smart contract will be accessible.

Function get will retrieve and print the value of the state variable.

Output:



Smart Contracts: Smart Contracts are the block of instructions that are not dependent on any third party or centralized database. They are executed in a decentralized environment.
The benefits of smart contracts are as follows:

- It removes centralized issues.
- It is safe and more secure.

Assignment no. 4 (CODE)

Aim:- Write a Program in Solidity to create students data. Use the following Constructs

- **Structures**
- **Arrays**
- **Fallback**

Deploy this as Smart Contract on Etherum and Observe the transaction fees and Gas Value

```
pragma
solidit
y
^0.6.6;


contract BankContract {

    struct client_account{
        int client_id;
        address client_address;
        uint client_balance_in_ether;
    }

    client_account[] clients;

    int clientCounter;
    address payable manager;
    mapping(address => uint) public interestDate;

    modifier onlyManager() {
        require(msg.sender == manager, "Only
manager can call this!");
        _;
    }
```

```
        modifier onlyManager() {
            require(msg.sender == manager, "Only
    manager can call this!");
            _;
        }

        modifier onlyClients() {
            bool isclient = false;
            for(uint i=0;i<clients.length;i++){
                if(clients[i].client_address ==
    msg.sender){
                    isclient = true;
                    break;
                }
            }
            require(isclient, "Only clients can call
    this!");
            _;

    }

    constructor() public{
        clientCounter = 0;
    }

    receive() external payable { }

    function setManager(address managerAddress)
public returns(string memory){
        manager = payable(managerAddress);
        return "";
    }

    function joinAsClient() public payable
returns(string memory){
        interestDate[msg.sender] = now;

clients.push(client_account(clientCounter++,
msg.sender, address(msg.sender).balance));
        return "";
    }

    function deposit() public payable onlyClients{

payable(address(this)).transfer(msg.value);
    }
```

```
    function deposit() public payable onlyClients{

payable(address(this)).transfer(msg.value);
    }

    function withdraw(uint amount) public payable
onlyClients{
        msg.sender.transfer(amount * 1 ether);
    }

    function sendInterest() public payable
onlyManager{
        for(uint i=0;i<clients.length;i++){
            address initialAddress =
clients[i].client_address;



                uint lastInterestDate =
    interestDate[initialAddress];
                if(now < lastInterestDate + 10
    seconds){
                    revert("It's just been less than
    10 seconds!");
                }
                payable(initialAddress).transfer(1
    ether);
                interestDate[initialAddress] = now;
            }
        }

        function getContractBalance() public view
    returns(uint){
            return address(this).balance;
        }
    }
```

Results :- Include Printout with Snapshot of Output

Code using Fallback

```
pragma solidity ^0.4.0;

// Creating a contract
contract fback
{
        // Declaring the state variable
        uint x;

        // Mapping of addresses to their balances
        mapping(address => uint) balance;

        // Creating a constructor
        constructor() public
        {       // Set x to default
                // value of 10
                x=10;
        }
        // Creating a function
        function SetX(uint _x) public returns(bool)
        {
                // Set x to the
                // value sent
                x=_x;
                return true;
        }

        // This fallback function will keep all the Ether
        function() public payable
        {
                balance[msg.sender] += msg.value;
        }
}
// Creating the sender contract
contract Sender
{function transfer() public payable

{       // Address of Fback contract

        address _receiver =

                        0xbcD310867F1b74142c2f5776404b6bd97165FA56;

                        // Transfers 100 Eth to above contract

        _receiver.transfer(100);

}

}
```

Assignment no. 5

Aim:- Write a Survey Report on types of Blockchains and its real time use cases

The basic application of the **blockchain** is to perform transactions in a secure network. That's why people use blockchain and ledger technology in different scenarios. One can set up multichain to prevent unauthorized access to sensitive data. It is not available to the public, and can only be available to authorized entities in the organization. It depends on the organization which type it requires to choose for their work.
By using blockchain we can track orders and payments from end to end.

Advantage using blockchain :

1. It provides greater trust among users.
2. It provides greater security among data.
3. Reduce the cost of production.
4. Improve Speed.
5. Invocation and tokenization.
6. It provides immutable records.
7. Smart contracts

Disadvantages using blockchain :

1. Data modification is not possible.
2. It requires large storage for a large database.
3. The owner cannot access the private key again if they forget or lose it.

Real life application of blockchain :

Here is a list of real world problem where we can use blockchain :

1. In a secure and full-proof voting management system.
2. To supply chain management.
3. In healthcare management.
4. Real estate project.
5. NFT marketplace.
6. Avoid copyright and original content creation.
7. In the personal identity system
8. To make an immutable data backup.
9. Internet of Things

Permissionless Blockchain

It is also known as trustless or public blockchains, are available to everyone to participate in the blockchains process that use to validate transactions and data. These are used in the network where high transparency is required.

Characteristics:

- Permissionless blockchain has no central authority.
- The platform is completely open-source.
- Full transparency of the transaction.
- Heavy use of tokens.
-

Advantages:

- Everyone can participate only requirement is good hardware and internet.
- Bring trust among users or entities.
- It has a high level of transparency as it's a larger network.
- Broader decentralization of access to more participants.

Disadvantages:

- Poor energy efficiency due to large network.
- Lower performance scalability.
- Less privacy as many of the things is visible.

Permissioned Blockchain

These are the closed network only a set of groups are allowed to validate transactions or data in a given blockchain network. These are used in the network where high privacy and security are required.

Characteristics:

- A major feature is a transparency based on the objective of the organization.
- Another feature is the lack of anatomy as only a limited number of users are allowed.
- It does not have a central authority.
- Developed by private authority.

Advantages:

- This blockchain tends to be faster as it has some nodes for validations.
- They can offer customizability.
- Strong Privacy as permission is needed for accessing transaction information.
- As few nodes are involved performance and scalability are increased.

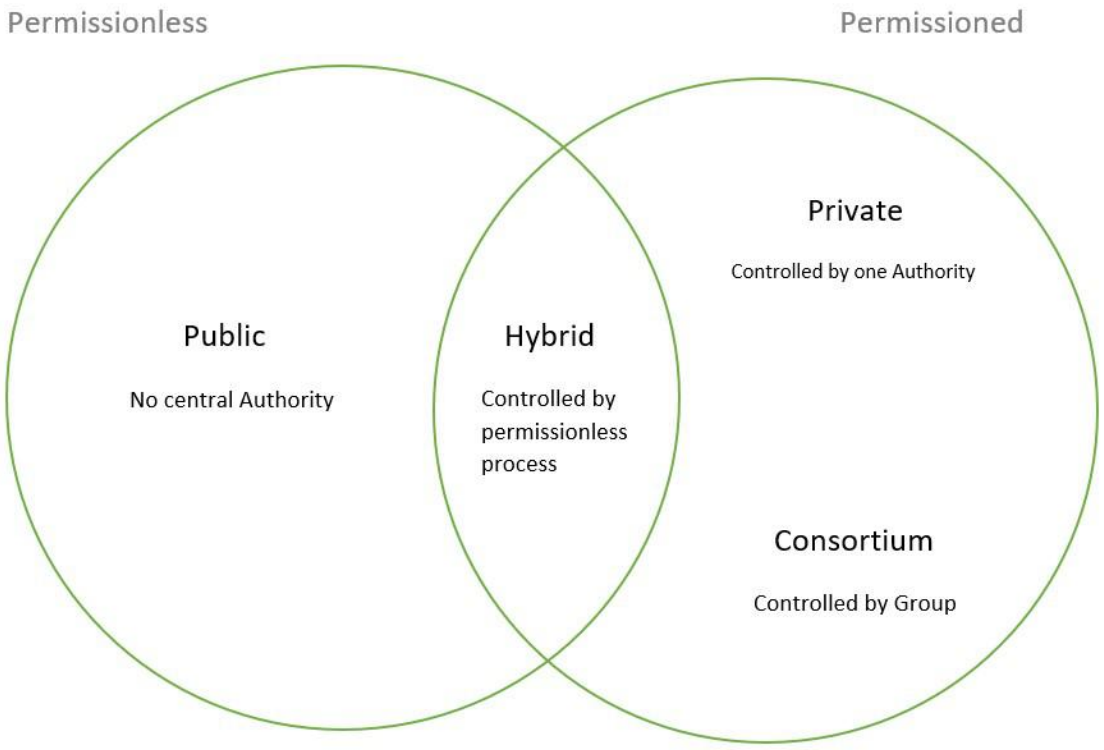
Disadvantages:

- Not truly decentralized as it requires permission
- Risk of corruption as only a few participants are involved.
- Anytime owner and operator can change the rules as per their need.

Types of Blockchain

There are 4 types of blockchain:

- **Public Blockchain.**
- **Private Blockchain.**
- **Hybrid Blockchain.**
- **Consortium Blockchain.**



Let's discuss each of these topics in detail.

1. Public Blockchain

These blockchains are completely open to following the idea of decentralization. They don't have any restrictions, anyone having a computer and internet can participate in the network.

- As the name is public this blockchain is open to the public, which means it is not owned by anyone.
- Anyone having internet and a computer with good hardware can participate in this public blockchain.

- All the computer in the network hold the copy of other nodes or block present in the network
- In this public blockchain, we can also perform verification of transactions or records

Advantages:

- **Trustable:** There are algorithms to detect no fraud. Participants need not worry about the other nodes in the network
- **Secure:** This blockchain is large in size as it is open to the public. In a large size, there is greater distribution of records
- **Anonymous Nature:** It is a secure platform to make your transaction properly at the same time, you are not required to reveal your name and identity in order to participate.
- **Decentralized:** There is no single platform that maintains the network, instead every user has a copy of the ledger.

Disadvantages:

- **Processing:** The rate of the transaction process is very slow, due to its large size. Verification of each node is a very time-consuming process.
- **Energy Consumption:** Proof of work is high energy-consuming. It requires good computer hardware to participate in the network
- **Acceptance:** No central authority is there so governments are facing the issue to implement the technology faster.

Use Cases: Public Blockchain is secured with proof of work or proof of stake they can be used to displace traditional financial systems. The more advanced side of this blockchain is the smart contract that enabled this blockchain to support decentralization. Examples of public blockchain are Bitcoin, Ethereum.

2. Private Blockchain

These blockchains are not as decentralized as the public blockchain only selected nodes can participate in the process, making it more secure than the others.

- These are not as open as a public blockchain.
- They are open to some authorized users only.
- These blockchains are operated in a closed network.
- In this few people are allowed to participate in a network within a company/organization.

Advantages:

- **Speed:** The rate of the transaction is high, due to its small size. Verification of each node is less time-consuming.
- **Scalability:** We can modify the scalability. The size of the network can be decided manually.
- **Privacy:** It has increased the level of privacy for confidentiality reasons as the businesses required.
- **Balanced:** It is more balanced as only some user has the access to the transaction which improves the performance of the network.

Disadvantages:

- Security- The number of nodes in this type is limited so chances of manipulation are there. These blockchains are more vulnerable.
- Centralized- Trust building is one of the main disadvantages due to its central nature. Organizations can use this for malpractices.
- Count- Since there are few nodes if nodes go offline the entire system of blockchain can be endangered.

Use Cases: With proper security and maintenance, this blockchain is a great asset to secure information without exposing it to the public eye. Therefore companies use them for internal auditing, voting, and asset management. An example of private blockchains is Hyperledger, Corda.

3. Hybrid Blockchain

It is the mixed content of the private and public blockchain, where some part is controlled by some organization and other makes are made visible as a public blockchain.

- It is a combination of both public and private blockchain.
- Permission-based and permissionless systems are used.
- User access information via smart contracts
- Even a primary entity owns a hybrid blockchain it cannot alter the transaction

Advantages:

- **Ecosystem:** Most advantageous thing about this blockchain is its hybrid nature. It cannot be hacked as 51% of users don't have access to the network
- **Cost:** Transactions are cheap as only a few nodes verify the transaction. All the nodes don't carry the verification hence less computational cost.
- **Architecture:** It is highly customizable and still maintains integrity, security, and transparency.
- **Operations:** It can choose the participants in the blockchain and decide which transaction can be made public.

Disadvantages:

- **Efficiency:** Not everyone is in the position to implement a hybrid Blockchain. The organization also faces some difficulty in terms of efficiency in maintenance.
- **Transparency:** There is a possibility that someone can hide information from the user. If someone wants to get access through a hybrid blockchain it depends on the organization whether they will give or not.
- **Ecosystem:** Due to its closed ecosystem this blockchain lacks the incentives for network participation.

Use Case: It provides a greater solution to the health care industry, government, real estate, and financial companies. It provides a remedy where data is to be accessed publicly but needs to be shielded privately. Examples of Hybrid Blockchain are Ripple network and XRP token.

4. Consortium Blockchain

It is a creative approach that solves the needs of the organization. This blockchain validates the transaction and also initiates or receives transactions.

- Also known as Federated Blockchain.
- This is an innovative method to solve the organization's needs.
- Some part is public and some part is private.
- In this type, more than one organization manages the blockchain.

Advantages:

- **Speed:** A limited number of users make verification fast. The high speed makes this more usable for organizations.
- **Authority:** Multiple organizations can take part and make it decentralized at every level. Decentralized authority, makes it more secure.
- **Privacy:** The information of the checked blocks is unknown to the public view. but any member belonging to the blockchain can access it.
- **Flexible:** There is much divergence in the flexibility of the blockchain. Since it is not a very large decision can be taken faster.

Disadvantages:

- **Approval:** All the members approve the protocol making it less flexible. Since one or more organizations are involved there can be differences in the vision of interest.
- **Transparency:** It can be hacked if the organization becomes corrupt. Organizations may hide information from the users.
- **Vulnerability:** If few nodes are getting compromised there is a greater chance of vulnerability in this blockchain

Use Cases: It has high potential in businesses, banks, and other payment processors. Food tracking of the organizations frequently collaborates with their sectors making it a federated solution ideal for their use. Examples of consortium Blockchain are Tendermint and Multichain.

Include the Following Case Studies in Survey :

- **Asset registry**
- **Cryptocurrencies**
- **Interbank reconciliation**
- **Smart contracts**
- **Supply chain traceability**

Assignment No. 6 : Create a d-App for e-Voting System

Code for Miniproject

```
// SPDX-
License-
Identifier:
MIT
        pragma solidity ^ 0.7;     // This defines the solidity version in which the code
        should be compiled
        contract Ballot {   // contract named ballot
          // VARIBLES
          struct vote {      // struct = dictionary in solidity
            address voterAddresss;     // used address datatype to store voter address
            bool choice;       // bool value to give the choice of voter
          }
          struct voter {
            string voterName;   // string dt used to store name of the voter
            bool voted;     // to check wether the voter voted or not
          }
          // some variables to store some values
          uint private countResult = 0;      // private var to count the result
          uint public finalResult = 0;       // public var to display the result
          uint public totalVoter = 0;        // public var to display the total no of voters
          uint public totalVote = 0;         // public var to display no of total votes
          address public ballotOfficialAddress;   // used address datatype to store the sys
        admin address
          string public ballotOfficalName;    // str used to store name of the ballot
        official(admin)
          string public proposal;     // proposal for which voting is done
          mapping(uint => vote) private votes;    // mapping integer to vote (like in a
        dictionary)
          mapping(address => voter) public voterRegister;     // mapping address to the
        voter
          enum State { Created, Voting, Ended }
          // enums keep track of state (data types that restrict the variable to have only
        one of the predefined values)
          State public state;
          // MODIFIER
          modifier condition(bool _condition) {      //modifiers = conditional statements in
        solidity
            require(_condition);
            _;
```

```
    }
    modifier onlyOfficial() {
        require(msg.sender == ballotOfficialAddress);
        // condition set to ensure only official/admin address can send message
        _;
    }
    modifier inState(State _state) {
        require(state == _state);
         // condition set to ensure if the global state is matching the passed state
        _;
    }
    // FUNCTION
    constructor(string memory _ballotofficalName,string memory _proposal)  {
        // created constructor that takes above arguments, underscores are used in
parameters to avoid ambiguity
        // this const enables the admin to register himself as the official using his
address and
        ballotOfficialAddress = msg.sender;
        ballotOfficalName = _ballotofficalName;
        proposal = _proposal;   // used to create proposal
        state = State.Created;  // used to change state
    }
    //function created to add voter takes voter address and name as arguments
    function addVoter(address _voterAdress, string memory _voterName )
        public
        inState(State.Created)      // checks whether if someone is registered as an
official
        onlyOfficial           // specifies that only official can register new voters
    {
        voter memory v;     // calling struct to create voter record
        v.voterName = _voterName;
        v.voted = false;    // to specift that new voter hasnt voted
        voterRegister[_voterAdress] = v;
        totalVoter++;   // to move index of the struct for new record
    }
    function startVote()       // function created to start voting
        public
        inState(State.Created)  // the state must be in created state to move to voting
state
        onlyOfficial    // only official can change state
    {
        state = State.Voting;   // changes state from creating to voting state
    }
```

```
    function doVote(bool _choice)      // function created to enable people to vote
      public
      inState(State.Voting)   // the current function can only be called in voting state
      returns (bool voted)    // returns if the voter has votesd or not
    {
      bool isFound = false;   // voter can only vote if the his voted state = false
      if(bytes(voterRegister[msg.sender].voterName).length != 0
        && voterRegister[msg.sender].voted == false )
        // to check if the voter has name and has not voted before
      {
        voterRegister[msg.sender].voted = true;
        vote memory v;  // to store the voters vote in the struct
        v.voterAddresss = msg.sender;
        v.choice = _choice;
        if(_choice) {
          countResult++;  // increment the result counter
        }
        votes[totalVote] = v;   //to calculate total number of people who voted
        totalVote++;    // increment vote if voted = true
        isFound = true;
      }
      return isFound;
    }
    function endVote()     // function created to end voting
      public
      inState(State.Voting)   // the current function can only be called in voting state
      onlyOfficial    // only official can change state
    {
      state = State.Ended;    // state changed to ended so voting ends
      finalResult = countResult;    // return the result after voting
    }
  }
```

Results :- Include Printout with Snapshot of Output