UNIVERSITY COLLEGE OF ENGINEERING KANCHEEPURAM

(A Constituent college of Anna University Chennai)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CCS339 - CRYPTOCURRENCY AND BLOCKCHAIN TECHNOLOGIES

Name: ______________________________

Register no: ______________________________

Year/Semester:____________**Branch:** __________

UNIVERSITY COLLEGE OF ENGINEERING KANCHEEPURAM

KANCHEEPURAM - 631 552

BONAFIDE CERTIFICATE

REGISTER NO												

Certified that this is the bonafide record of work done by Mr/Ms……………………………………….. of …… semester B.E. Computer Science and Engineering Branch / Batch during the academic year 20….to 20…. in the **CCS339 – CRYPTOCURRENCY AND BLOCKCHAIN TECHNOLOGIES.**

Staff In-Charge **Head of the Department**

Submitted for the University Practical examination held on ………………………

Internal Examiner **External Examiner**

INDEX

EX.NO: 1	Install and understand Docker container, Node.js, Java and Hyperledger Fabric, Ethereum and perform necessary software installation on Local machine.
DATE:	

AIM:

- Verify the installation of Docker, Node.js, and Java.
- Set up Hyperledger Fabric using Docker.
- Set up a local Ethereum environment using Docker and Ganache.

PREREQUISITES:

- A machine running Ubuntu.
- Basic understanding of command-line operations

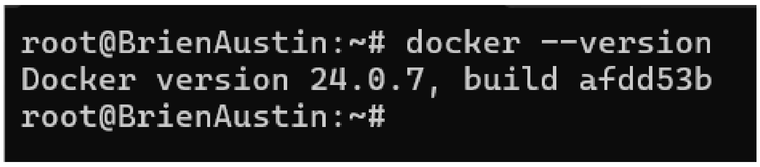
PROCEDURE:

VERIFYING DOCKER, NODEJS, JAVA INSTALLATION:

1. Open Linux terminal using **WSL** (**W**indows **S**ubsystem for **L**inux) or using VirtualBox.

Docker:

2. Verify the installation of Docker.



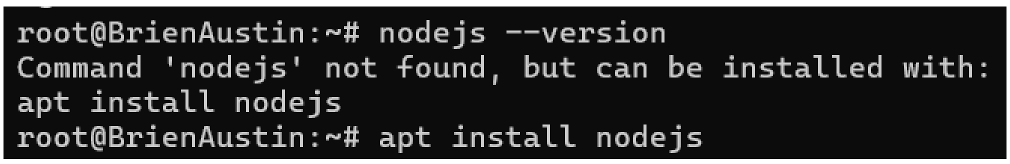
```
root@BrienAustin:~# docker --version
Docker version 24.0.7, build afdd53b
root@BrienAustin:~#
```

3. If not download using the below command:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

Nodejs:

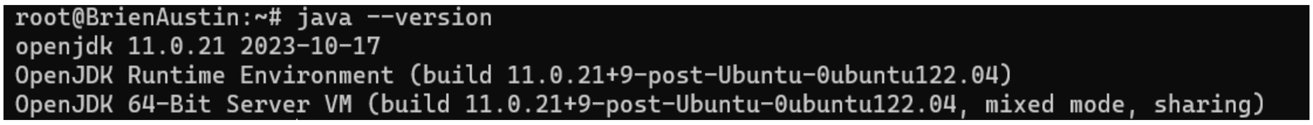
4. Verify the installation of Nodejs and install if not present.



```
root@BrienAustin:~# nodejs --version
Command 'nodejs' not found, but can be installed with:
apt install nodejs
root@BrienAustin:~# apt install nodejs
```

Java:

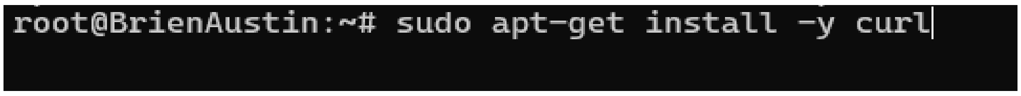
5. Verify the installation of Java and install if not present



```
root@BrienAustin:~# java --version
openjdk 11.0.21 2023-10-17
OpenJDK Runtime Environment (build 11.0.21+9-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.21+9-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
```

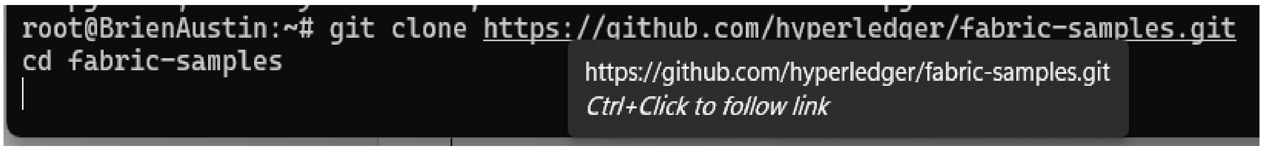
SETUP HYPERLEDGER FABRIC USING DOCKER:

1. Install cURL



```
root@BrienAustin:~# sudo apt-get install -y curl
```

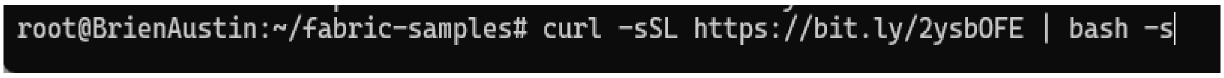
2. Cloning sample fabric repository



```
root@BrienAustin:~# git clone https://github.com/hyperledger/fabric-samples.git
cd fabric-samples
```

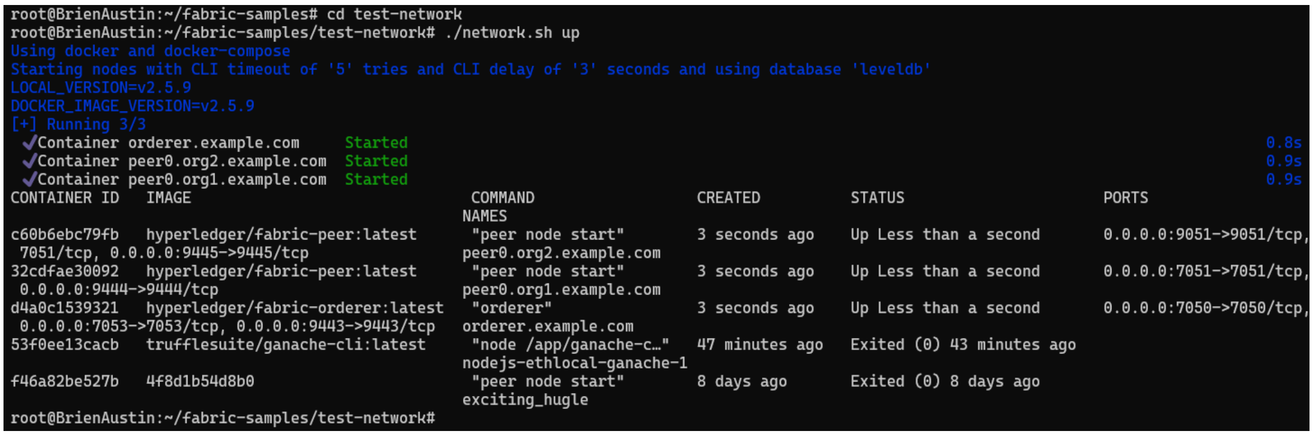
https://github.com/hyperledger/fabric-samples.git
Ctrl+Click to follow link

3. Install Fabric samples, Binaries and Docker images.



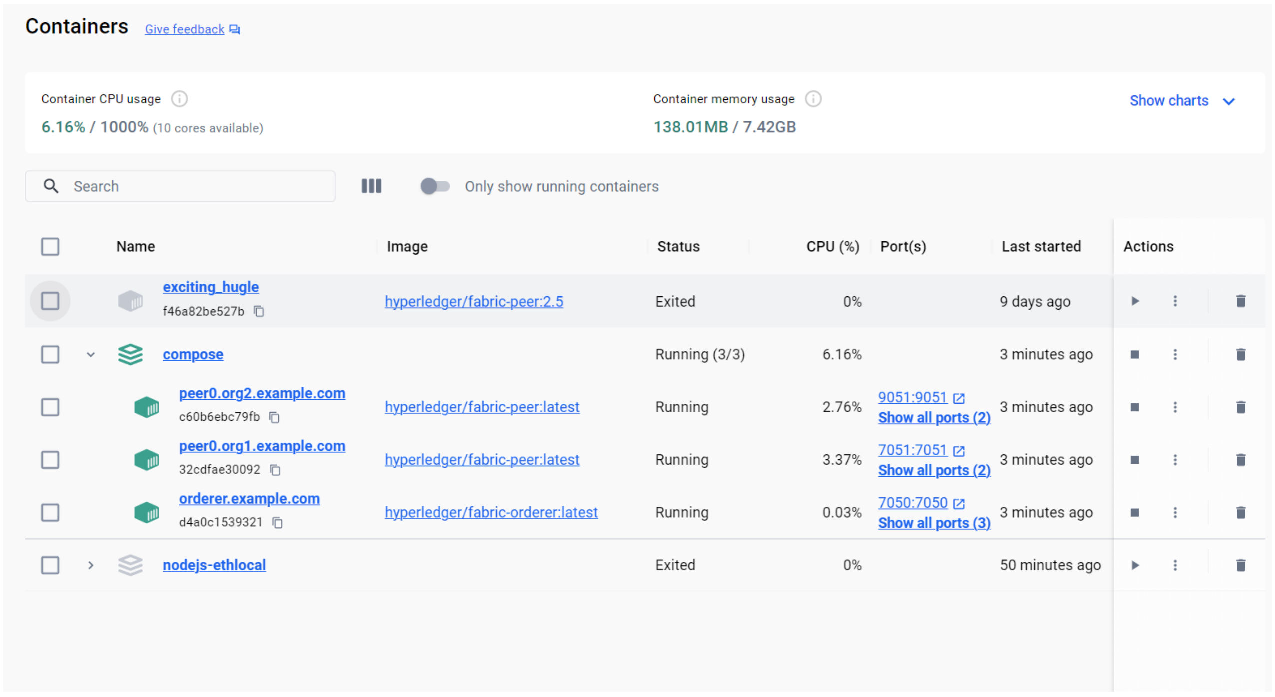
```
root@BrienAustin:~/fabric-samples# curl -sSL https://bit.ly/2ysbOFE | bash -s
```

4. Start the containers



```
root@BrienAustin:~/fabric-samples# cd test-network
root@BrienAustin:~/fabric-samples/test-network# ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'
LOCAL_VERSION=v2.5.9
DOCKER_IMAGE_VERSION=v2.5.9
[+] Running 3/3
 ✓Container orderer.example.com     Started                                                                     0.8s
 ✓Container peer0.org2.example.com  Started                                                                     0.9s
 ✓Container peer0.org1.example.com  Started                                                                     0.9s
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS                        PORTS
                                                  NAMES
c60b6ebc79fb   hyperledger/fabric-peer:latest      "peer node start"        3 seconds ago    Up Less than a second         0.0.0.0:9051->9051/tcp,
 7051/tcp, 0.0.0.0:9445->9445/tcp                 peer0.org2.example.com
32cdfae30092   hyperledger/fabric-peer:latest      "peer node start"        3 seconds ago    Up Less than a second         0.0.0.0:7051->7051/tcp,
 0.0.0.0:9444->9444/tcp                           peer0.org1.example.com
d4a0c1539321   hyperledger/fabric-orderer:latest   "orderer"                3 seconds ago    Up Less than a second         0.0.0.0:7050->7050/tcp,
 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp   orderer.example.com
53f0ee13cacb   trufflesuite/ganache-cli:latest     "node /app/ganache-c…"   47 minutes ago   Exited (0) 43 minutes ago
                                                  nodejs-ethlocal-ganache-1
f46a82be527b   4f8d1b54d8b0                        "peer node start"        8 days ago       Exited (0) 8 days ago
                                                  exciting_hugle
root@BrienAustin:~/fabric-samples/test-network#
```

5. Verify whether the containers are running using **Docker Desktop**



Containers Give feedback

Container CPU usage
6.16% / 1000% (10 cores available)

Container memory usage
138.01MB / 7.42GB

Show charts

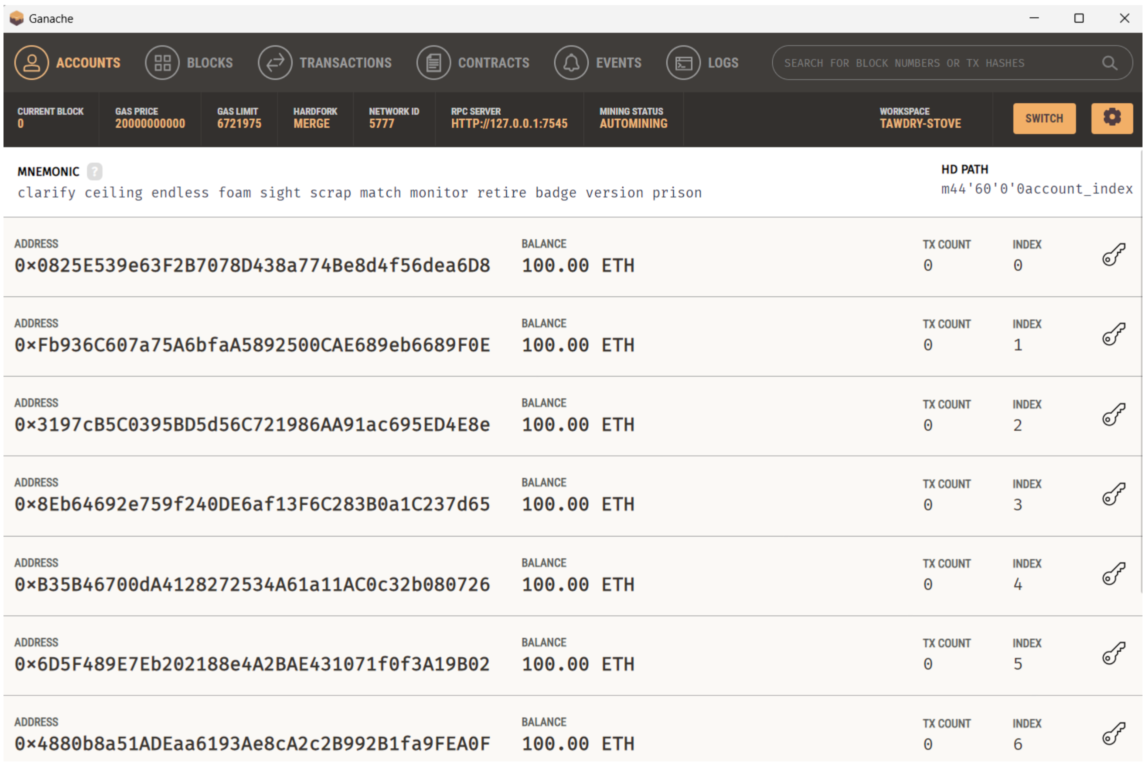
Search | Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started
exciting_hugle f46a82be527b	hyperledger/fabric-peer:2.5	Exited	0%		9 days ago
compose		Running (3/3)	6.16%		3 minutes ago
peer0.org2.example.com c60b6ebc79fb	hyperledger/fabric-peer:latest	Running	2.76%	9051:9051 Show all ports (2)	3 minutes ago
peer0.org1.example.com 32cdfae30092	hyperledger/fabric-peer:latest	Running	3.37%	7051:7051 Show all ports (2)	3 minutes ago
orderer.example.com d4a0c1539321	hyperledger/fabric-orderer:latest	Running	0.03%	7050:7050 Show all ports (3)	3 minutes ago
nodejs-ethlocal		Exited	0%		50 minutes ago

Thus, **Hyperledger Fabric** is setup on **Local machine** through **Docker.**

<u>SETUP ETHEREUM ENVIRONMENT USING DOCKER AND GANACHE:</u>

1. Install Ganache
2. The interface will be like this :



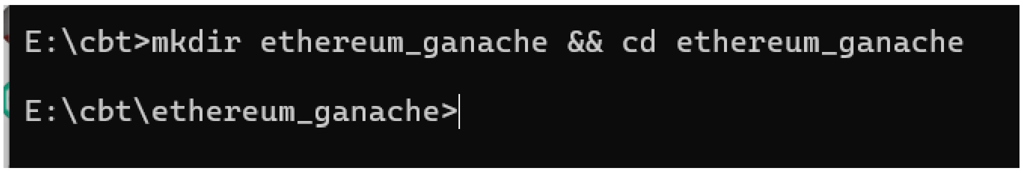
CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
0	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	TAWDRY-STOVE

MNEMONIC
clarify ceiling endless foam sight scrap match monitor retire badge version prison

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×0825E539e63F2B7078D438a774Be8d4f56dea6D8	100.00 ETH	0	0
0×Fb936C607a75A6bfaA5892500CAE689eb6689F0E	100.00 ETH	0	1
0×3197cB5C0395BD5d56C721986AA91ac695ED4E8e	100.00 ETH	0	2
0×8Eb64692e759f240DE6af13F6C283B0a1C237d65	100.00 ETH	0	3
0×B35B46700dA4128272534A61a11AC0c32b080726	100.00 ETH	0	4
0×6D5F489E7Eb202188e4A2BAE431071f0f3A19B02	100.00 ETH	0	5
0×4880b8a51ADEaa6193Ae8cA2c2B992B1fa9FEA0F	100.00 ETH	0	6

3. Create a folder **Ethereum_ganache**

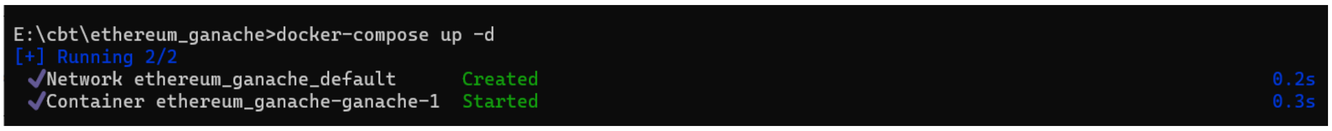


```
E:\cbt>mkdir ethereum_ganache && cd ethereum_ganache

E:\cbt\ethereum_ganache>
```

4. Inside it create a docker-compose.yml file.
5. Write the following code in it:

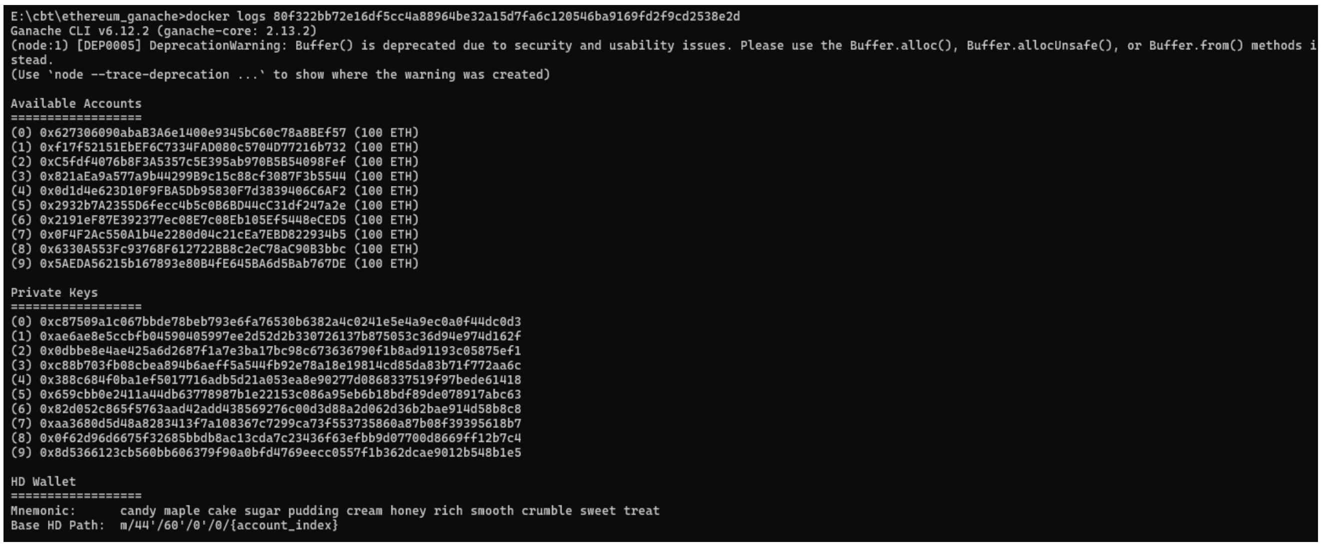
```
version: '3'
services:
  ganache:
    image: trufflesuite/ganache-cli
    ports:
      - "8545:8545"
    command: >
      ganache-cli
      --host 0.0.0.0
      --port 8545
      --networkId 5777
      --mnemonic "candy maple cake sugar pudding cream honey rich smooth crumble sweet treat"
```

6. Create and start the container



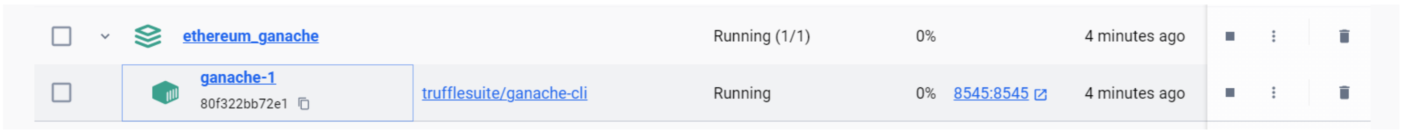
```
E:\cbt\ethereum_ganache>docker-compose up -d
[+] Running 2/2
 ✓Network ethereum_ganache_default      Created                                                        0.2s
 ✓Container ethereum_ganache-ganache-1  Started                                                        0.3s
```

7. View the Logs



```
E:\cbt\ethereum_ganache>docker logs 80f322bb72e16df5cc4a88964be32a15d7fa6c120546ba9169fd2f9cd2538e2d
Ganache CLI v6.12.2 (ganache-core: 2.13.2)
(node:1) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security and usability issues. Please use the Buffer.alloc(), Buffer.allocUnsafe(), or Buffer.from() methods i
stead.
(Use `node --trace-deprecation ...` to show where the warning was created)

Available Accounts
==================
(0) 0x627306090abaB3A6e1400e9345bC60c78a8BEf57 (100 ETH)
(1) 0xf17f52151EbEF6C7334FAD080c5704D77216b732 (100 ETH)
(2) 0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef (100 ETH)
(3) 0x821aEa9a577a9b44299B9c15c88cf3087F3b5544 (100 ETH)
(4) 0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2 (100 ETH)
(5) 0x2932b7A2355D6fecc4b5c0B6BD44cC31df247a2e (100 ETH)
(6) 0x2191eF87E392377ec08E7c08Eb105Ef5448eCED5 (100 ETH)
(7) 0x0F4F2Ac550A1b4e2280d04c21cEa7EBD822934b5 (100 ETH)
(8) 0x6330A553Fc93768F612722BB8c2eC78aC90B3bbc (100 ETH)
(9) 0x5AEDA56215b167893e80B4fE645BA6d5Bab767DE (100 ETH)

Private Keys
==================
(0) 0xc87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
(1) 0xae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f
(2) 0x0dbbe8e4ae425a6d2687f1a7e3ba17bc98c673636790f1b8ad91193c05875ef1
(3) 0xc88b703fb08cbea894b6aeff5a544fb92e78a18e19814cd85da83b71f772aa6c
(4) 0x388c684f0ba1ef5017716adb5d21a053ea8e90277d0868337519f97bede61418
(5) 0x659cbb0e2411a44db63778987b1e22153c086a95eb6b18bdf89de078917abc63
(6) 0x82d052c865f5763aad42add438569276c00d3d88a2d062d36b2bae914d58b8c8
(7) 0xaa3680d5d48a8283413f7a108367c7299ca73f553735860a87b08f39395618b7
(8) 0x0f62d96d6675f32685bbdb8ac13cda7c23436f63efbb9d07700d8669ff12b7c4
(9) 0x8d5366123cb560bb606379f90a0bfd4769eecc0557f1b362dcae9012b548b1e5

HD Wallet
==================
Mnemonic:      candy maple cake sugar pudding cream honey rich smooth crumble sweet treat
Base HD Path:  m/44'/60'/0'/0/{account_index}
```

8. Verify in **Docker Desktop**



	Name	Image	Status	CPU	Port	Last started
☐	ethereum_ganache		Running (1/1)	0%		4 minutes ago
☐	ganache-1 80f322bb72e1	trufflesuite/ganache-cli	Running	0%	8545:8545	4 minutes ago

RESULT:

Thus, **Docker, Java, NodeJS** have been successfully installed and the **Hyperledger Fabric** and **Ethereum Network** have been setup on the **Local machine** using **Docker**

.

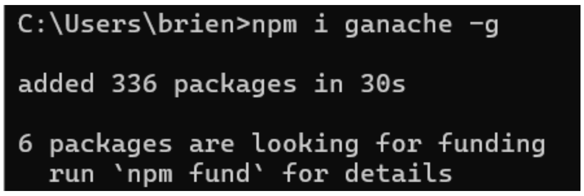
EX.NO: 2	Create and deploy a blockchain network and perform invoke and query on your blockchain network.
DATE:	

AIM:

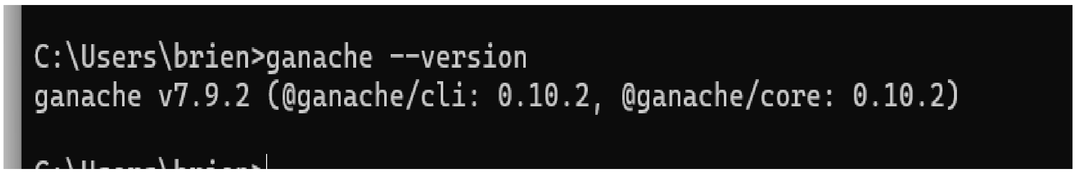
To create a blockchain network and perform invoke and query on the deployed blockchain network.

PROCEDURE:

1. Install Ganache and install **ganache** package globally.



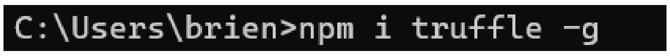
```
C:\Users\brien>npm i ganache -g

added 336 packages in 30s

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



```
C:\Users\brien>ganache --version
ganache v7.9.2 (@ganache/cli: 0.10.2, @ganache/core: 0.10.2)
```

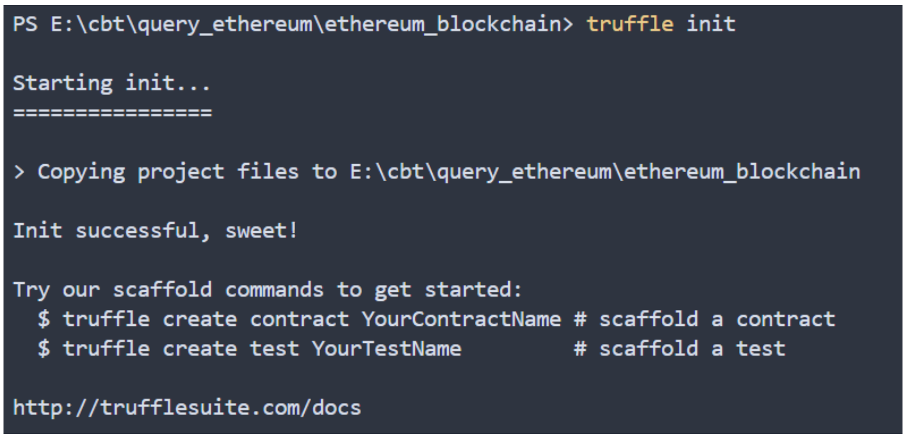
2. Ganache Desktop version can be downloaded from https://archive.trufflesuite.com/ganache/

3. Create a Folder and name it as *ethereum_blockchain*
4. Install **truffle** globally



```
C:\Users\brien>npm i truffle -g
```

5. Initialize **truffle**



```
PS E:\cbt\query_ethereum\ethereum_blockchain> truffle init

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

> Copying project files to E:\cbt\query_ethereum\ethereum_blockchain

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs
```

6. Modify the truffle with the code below

```
module.exports = {
  networks : {
    development : {
      host : '127.0.0.1',
      port : 5777,
      network_id : "*"
    }
  },
  compilers : {
    solc : {
      version : "0.8.19"
    }
  }
}
```

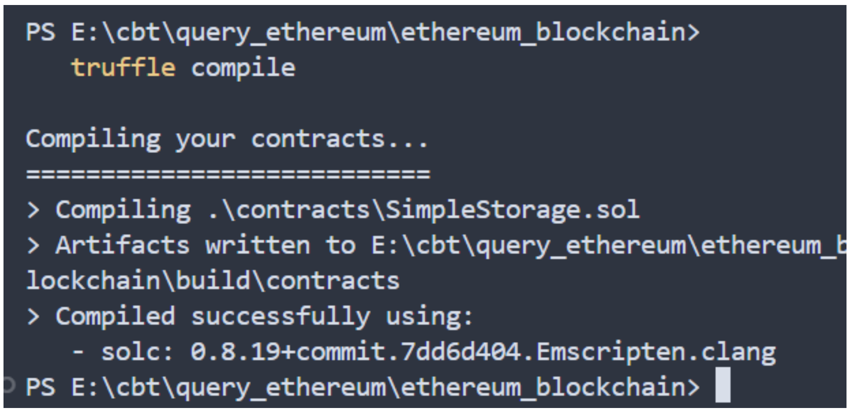
7. Now, in the contracts folder create a contract and name it as **SimpleStorage.sol**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract SimpleStorage {
    uint256 storedData;

    function set(uint256 x) public {
        storedData = x;
    }

    function get() public view returns (uint256) {
        return storedData;
    }
}
```

8. This contract allows us to **invoke** and **query** the locally deployed **Ethereum Blockchain.**
9. Now, in the migrations folder create a file and name it as **2_deploy_contracts.js** and write the code below

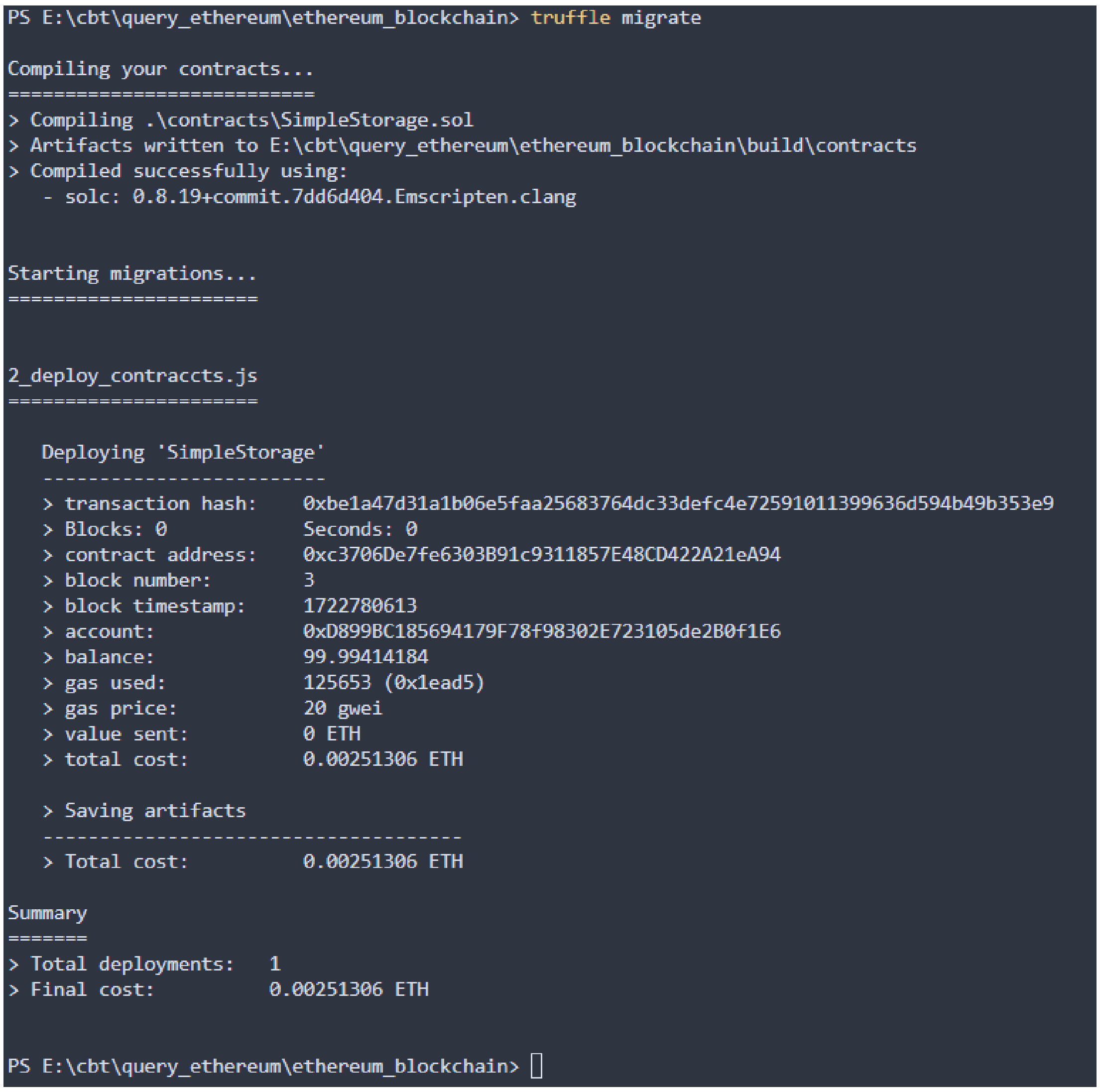
```
const SimpleStorage = artifacts.require("SimpleStorage");

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

10. Now compile the contract



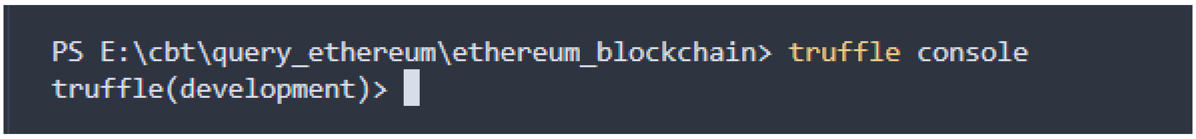
```
PS E:\cbt\query_ethereum\ethereum_blockchain>
   truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\SimpleStorage.sol
> Artifacts written to E:\cbt\query_ethereum\ethereum_b
lockchain\build\contracts
> Compiled successfully using:
   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang
PS E:\cbt\query_ethereum\ethereum_blockchain>
```

11. Now, migrate to the deployed local ethereum network.



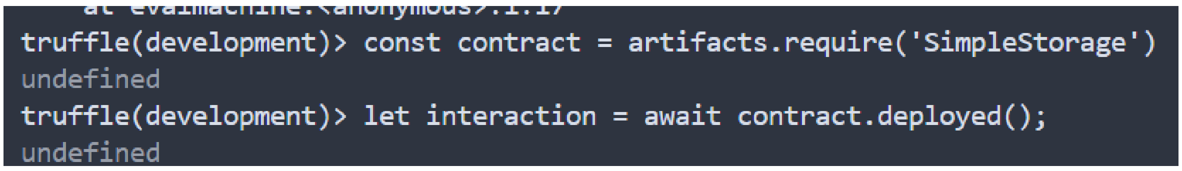
```
PS E:\cbt\query_ethereum\ethereum_blockchain> truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\SimpleStorage.sol
> Artifacts written to E:\cbt\query_ethereum\ethereum_blockchain\build\contracts
> Compiled successfully using:
   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang


Starting migrations...
======================


2_deploy_contraccts.js
======================

   Deploying 'SimpleStorage'
   -------------------------
   > transaction hash:    0xbe1a47d31a1b06e5faa25683764dc33defc4e72591011399636d594b49b353e9
   > Blocks: 0            Seconds: 0
   > contract address:    0xc3706De7fe6303B91c9311857E48CD422A21eA94
   > block number:        3
   > block timestamp:     1722780613
   > account:             0xD899BC185694179F78f98302E723105de2B0f1E6
   > balance:             99.99414184
   > gas used:            125653 (0x1ead5)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00251306 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00251306 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.00251306 ETH


PS E:\cbt\query_ethereum\ethereum_blockchain>
```

12. Now, we will be ale to interact with the blockchain, by using the command **truffle console.**



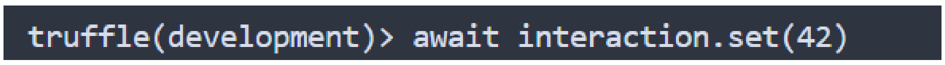
```
PS E:\cbt\query_ethereum\ethereum_blockchain> truffle console
truffle(development)>
```

13. Create an instance to invoke and query the blockchain

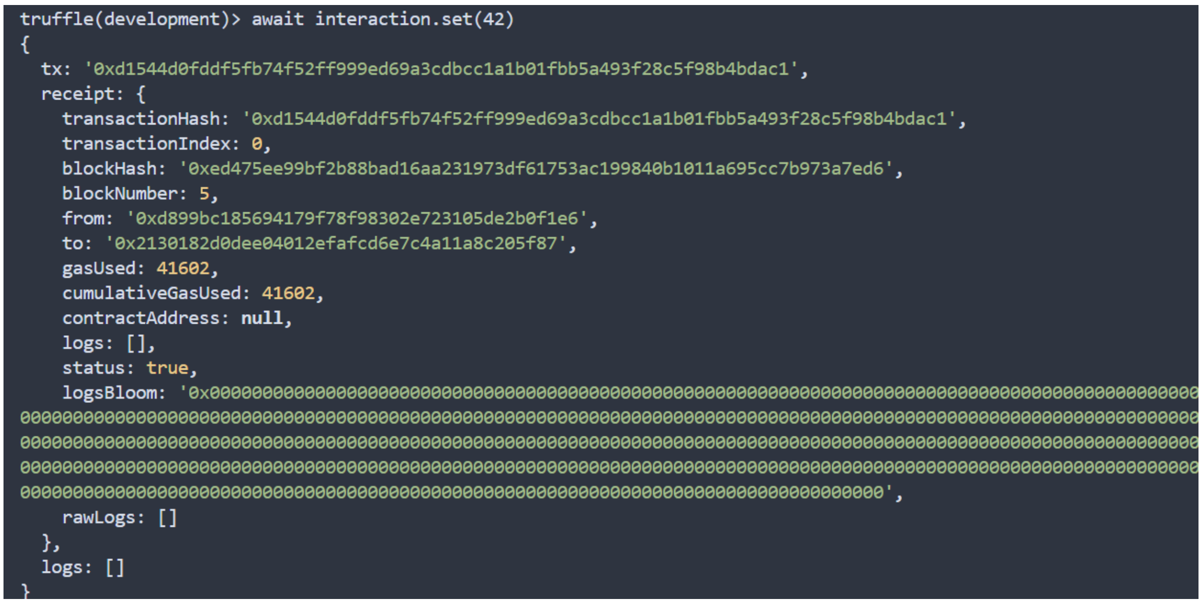


```
    at evalmachine.<anonymous>:1:17
truffle(development)> const contract = artifacts.require('SimpleStorage')
undefined
truffle(development)> let interaction = await contract.deployed();
undefined
```

14. Now, invoke a value to the SimpleStorage contract

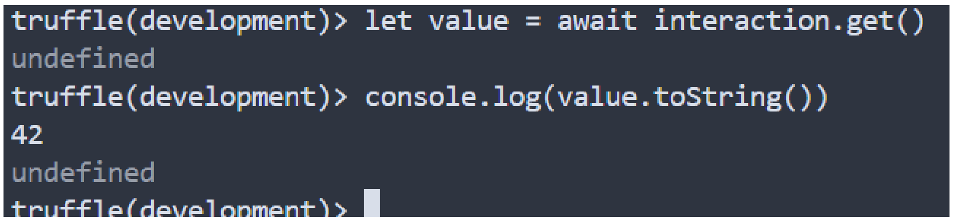


```
truffle(development)> await interaction.set(42)
```



```
truffle(development)> await interaction.set(42)
{
  tx: '0xd1544d0fddf5fb74f52ff999ed69a3cdbcc1a1b01fbb5a493f28c5f98b4bdac1',
  receipt: {
    transactionHash: '0xd1544d0fddf5fb74f52ff999ed69a3cdbcc1a1b01fbb5a493f28c5f98b4bdac1',
    transactionIndex: 0,
    blockHash: '0xed475ee99bf2b88bad16aa231973df61753ac199840b1011a695cc7b973a7ed6',
    blockNumber: 5,
    from: '0xd899bc185694179f78f98302e723105de2b0f1e6',
    to: '0x2130182d0dee04012efafcd6e7c4a11a8c205f87',
    gasUsed: 41602,
    cumulativeGasUsed: 41602,
    contractAddress: null,
    logs: [],
    status: true,
    logsBloom: '0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000',
    rawLogs: []
  },
  logs: []
}
```

15. Now, query for the value



```
truffle(development)> let value = await interaction.get()
undefined
truffle(development)> console.log(value.toString())
42
undefined
truffle(development)>
```

16. Thus, the contract works perfectly.

RESULT:

Thus, created a blockchain network and performed invoke and query on the deployed blockchain network.

EX.NO: 3	Interact with a blockchain network. Execute transactions and requests against a blockchain network by creating an app to test the network and its rules.
DATE:	

AIM:

To setup a Blockchain network and execute Transactions and Requests against a Blockchain network and test by developing an application.

PROCEDURE:

1. Install **Nodejs** and install the packages **ganache** and **truffle**.
2. Create a new folder and initialize **truffle** in the project by the below code

truffle init

3. Modify the **truffle-config.js** with the below code

```
module.exports = {
  networks: {
    development: {
      host: '127.0.0.1',
      port: 7545,
      network_id: '*'
    }
  },
  compilers: {
    solc: {
      version: "0.8.19"
    }
  }
}
```

4. Create a Smart Contract for a sample data storing app

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract SimpleStorage {

    uint256 private storedData;


    event DataStored(uint256 value);


    function set(uint256 x) public {
        storedData = x;
        emit DataStored(x);
    }


    function get() public view returns (uint256) {
        return storedData;
    }
}
```

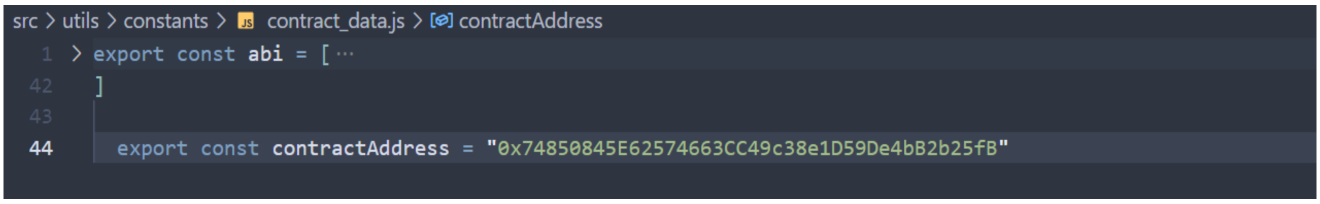
5. Compile and deploy the contract

```
truffle migrate -reset
truffle compile
truffle console
```

6. Develop a Frontend Application to test the contract
7. Create a React Vite App

npx create-vite@latest <project-name>

8. Now create a folder named **utils/constants** and create a file named **contract_data.js** and store the **abi** and **contract_address**



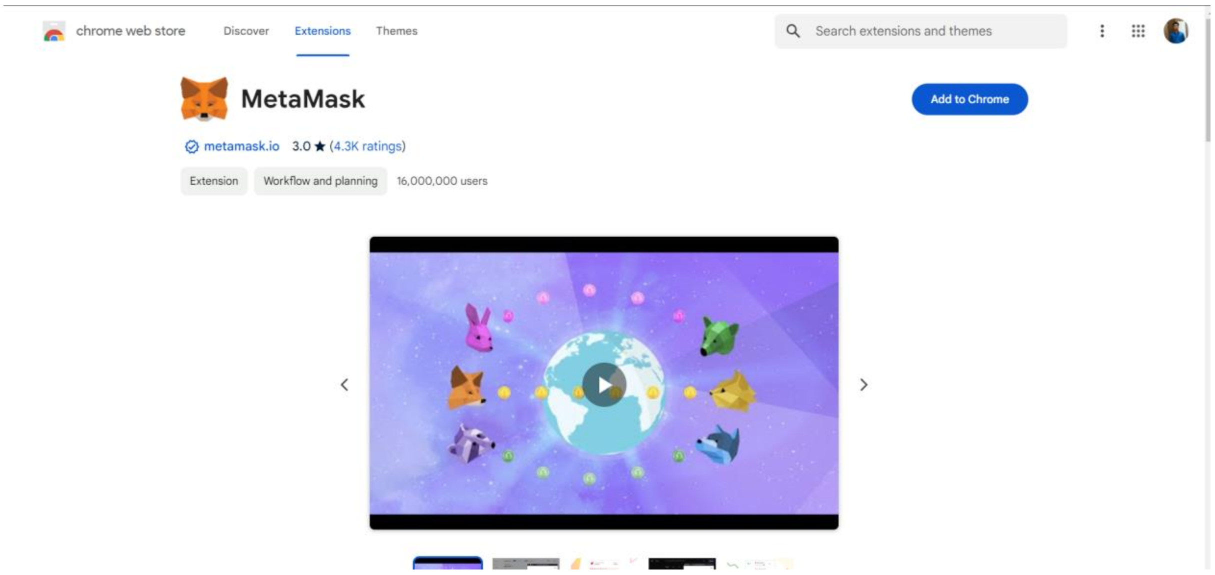
src > utils > constants > contract_data.js > contractAddress

```
> export const abi = [ ···
]

  export const contractAddress = "0x74850845E62574663CC49c38e1D59De4bB2b25fB"
```

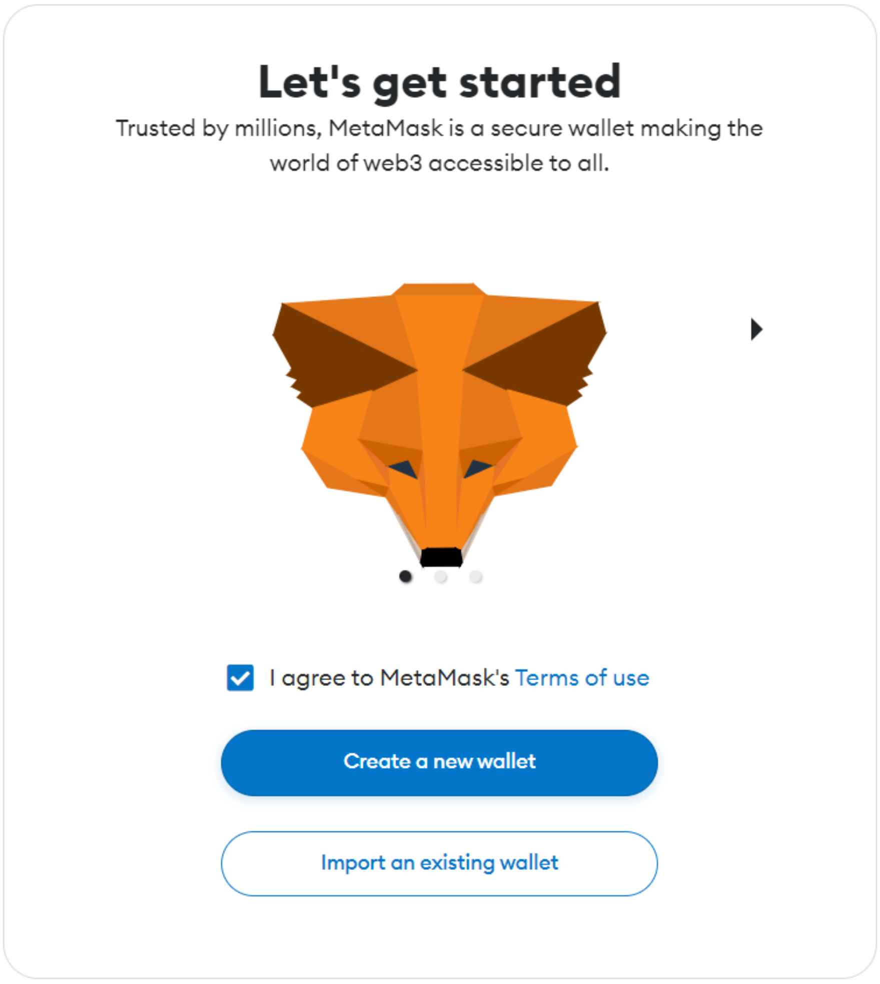
9. Now create a component for interacting with the contract

```
import { ethers } from "ethers";
import { useEffect, useState } from "react";
import toast from "react-hot-toast";
import { abi, contractAddress } from "./utils/constants/contract_data";

const SimpleStorageComponent = () => {
  const [storedValue, setStoredValue] = useState("");
  const [valueToSet, setValueToSet] = useState("");
  const [loading, setLoading] = useState(false);

  useEffect(() => {
    fetchStoredValue();
  }, []);

  const fetchStoredValue = async () => {
    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);
    try {
      const value = await contract.get();
      setStoredValue(value.toString());
    } catch (error) {
      console.error(error);
      toast.error('Failed to fetch stored value from contract');
    }
  };

  const setValue = async (e) => {
    e.preventDefault();
    setLoading(true);
    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);
    try {
      const signer = await provider.getSigner();
      const tx = await contract.connect(signer).set(valueToSet);
      await tx.wait();
      toast.success('Value set successfully!');
      fetchStoredValue();
    } catch (error) {
      console.error(error);
      toast.error('Error setting value');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="flex flex-col items-center mt-16">
      <h1 className="text-3xl text-neutral-700 font-bold">Simple Storage</h1>

      <form onSubmit={setValue} className="flex flex-col items-center mt-6">
        <input
          type="number"
          value={valueToSet}
          onChange={(e) => setValueToSet(e.target.value)}
          className="border rounded-md p-2 mb-2"
          placeholder="Set a value"
          required
        />
        <button
          type="submit"
          className="bg-blue-500 text-white px-3 py-2 rounded-md shadow-sm"
          disabled={loading}
        >
          {loading ? 'Setting Value...' : 'Set Value'}
        </button>
      </form>

      <div className="mt-6">
        <h2 className="text-xl">Stored Value: {storedValue}</h2>
      </div>
    </div>
  );
};

export default SimpleStorageComponent;
```

10. Now let’s create a **metamask wallet** and test out our **contract.**



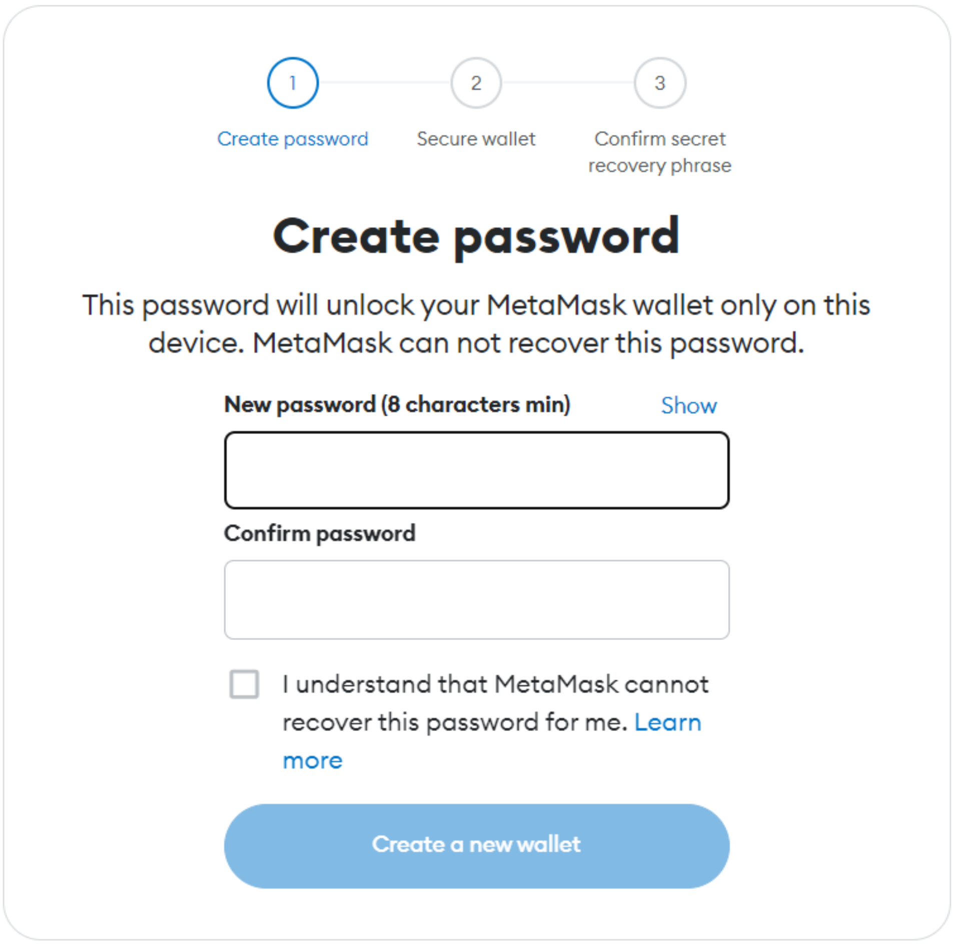
- Create a new wallet



Let's get started

Trusted by millions, MetaMask is a secure wallet making the world of web3 accessible to all.

- Add password and setup the account



1 Create password
2 Secure wallet
3 Confirm secret recovery phrase

Create password

This password will unlock your MetaMask wallet only on this device. MetaMask can not recover this password.

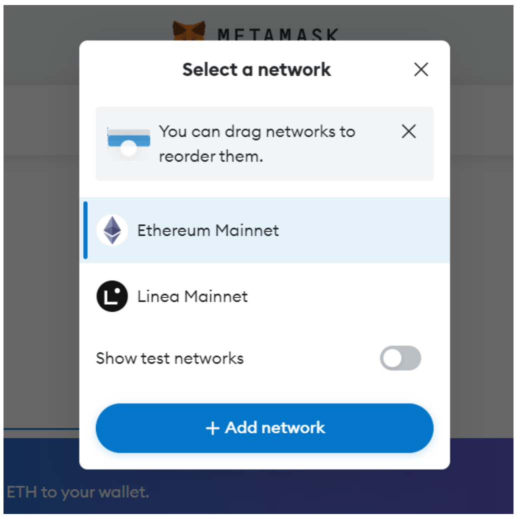
New password (8 characters min) Show

Confirm password

I understand that MetaMask cannot recover this password for me. Learn more

Create a new wallet

11. Add **ganache** to the metamask



Select a network

You can drag networks to reorder them.

Ethereum Mainnet

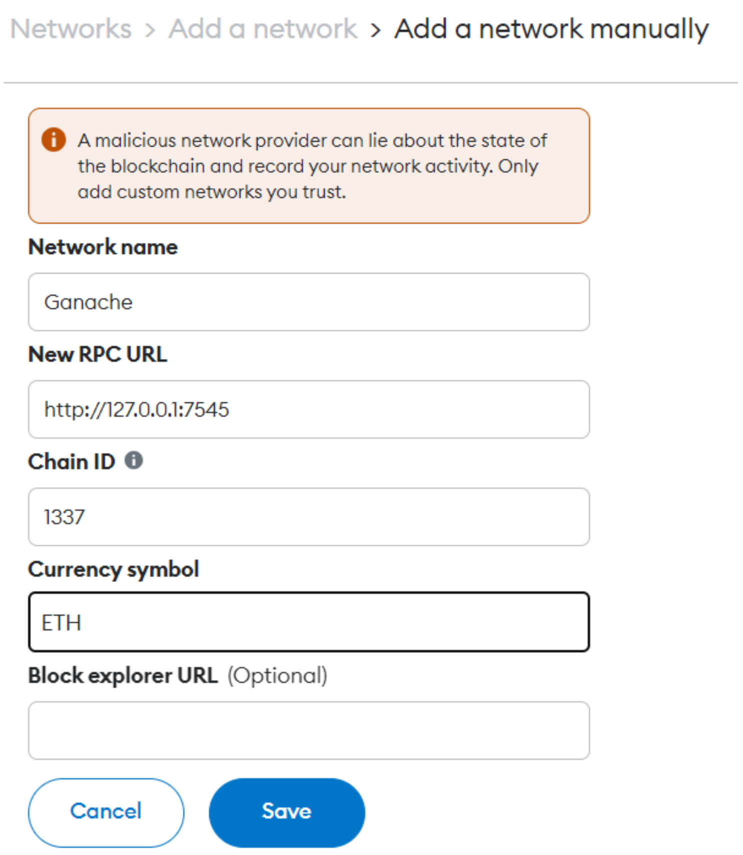
Linea Mainnet

Show test networks

+ Add network

ETH to your wallet.

Add a network manually



Networks > Add a network > Add a network manually

A malicious network provider can lie about the state of the blockchain and record your network activity. Only add custom networks you trust.

Network name: Ganache

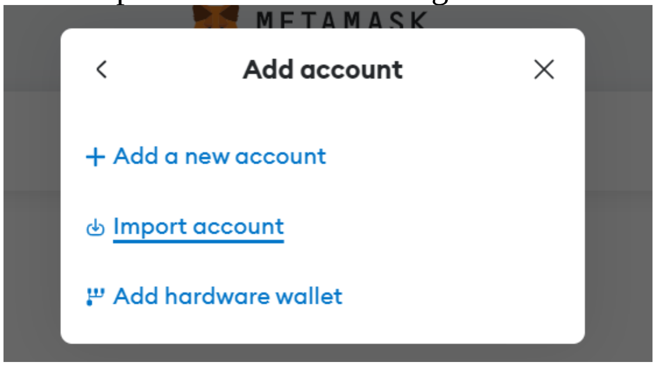
New RPC URL: http://127.0.0.1:7545

Chain ID: 1337

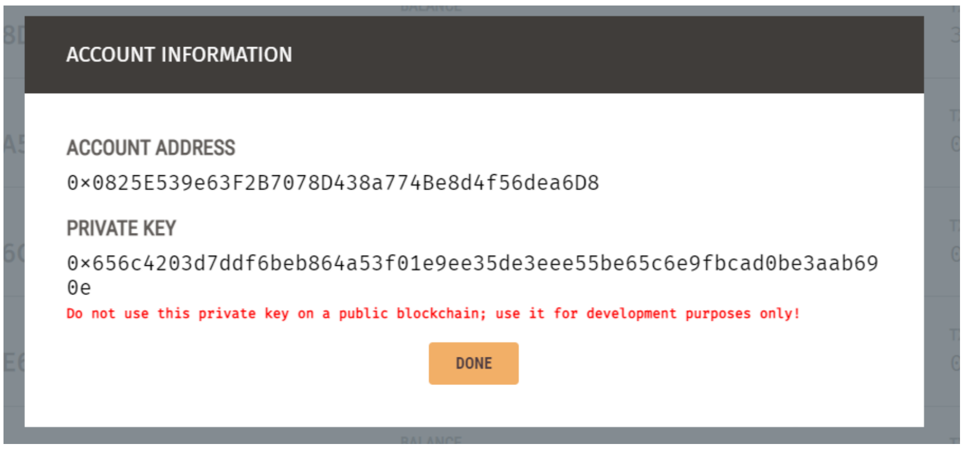
Currency symbol: ETH

Block explorer URL (Optional):

12. Now import a test wallet from ganache



13. Paste the private key of a wallet from Ganache

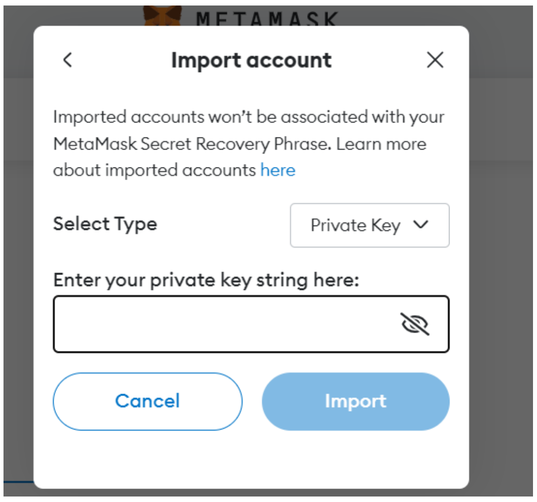


ACCOUNT INFORMATION

ACCOUNT ADDRESS
0x0825E539e63F2B7078D438a774Be8d4f56dea6D8

PRIVATE KEY
0x656c4203d7ddf6beb864a53f01e9ee35de3eee55be65c6e9fbcad0be3aab69
0e

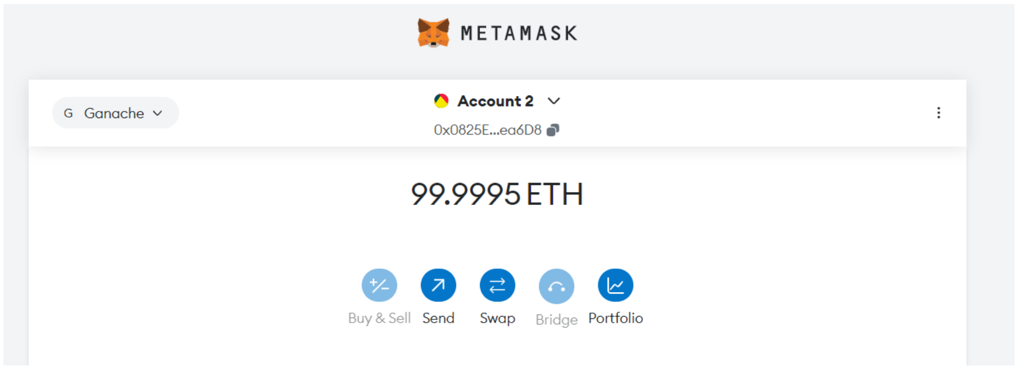
Do not use this private key on a public blockchain; use it for development purposes only!


Import account

Imported accounts won't be associated with your MetaMask Secret Recovery Phrase. Learn more about imported accounts here

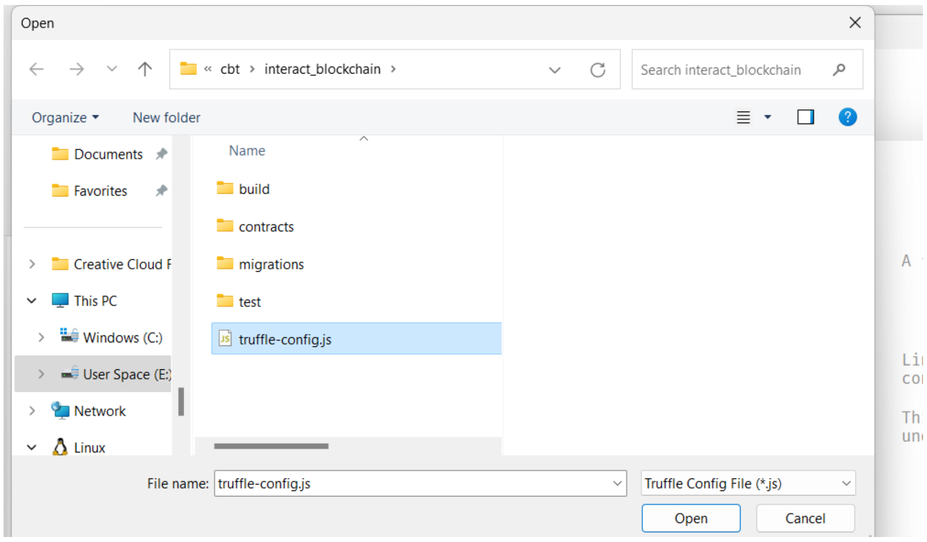
Select Type: Private Key

Enter your private key string here:



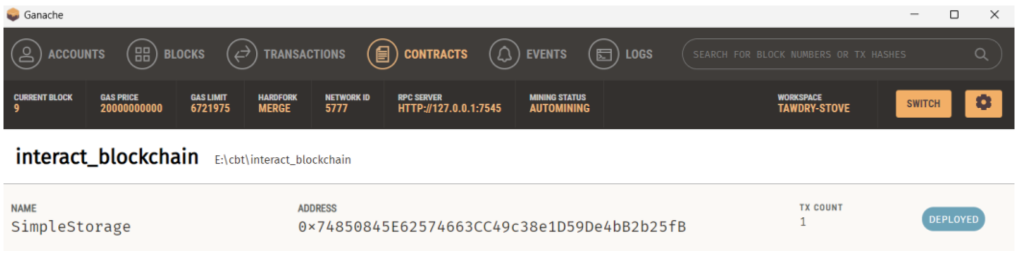
After pasting, we can test our app

14. Now link the contract to ganache , by opening ganache and adding **truffle-config.js**



File name: truffle-config.js

Truffle Config File (*.js)

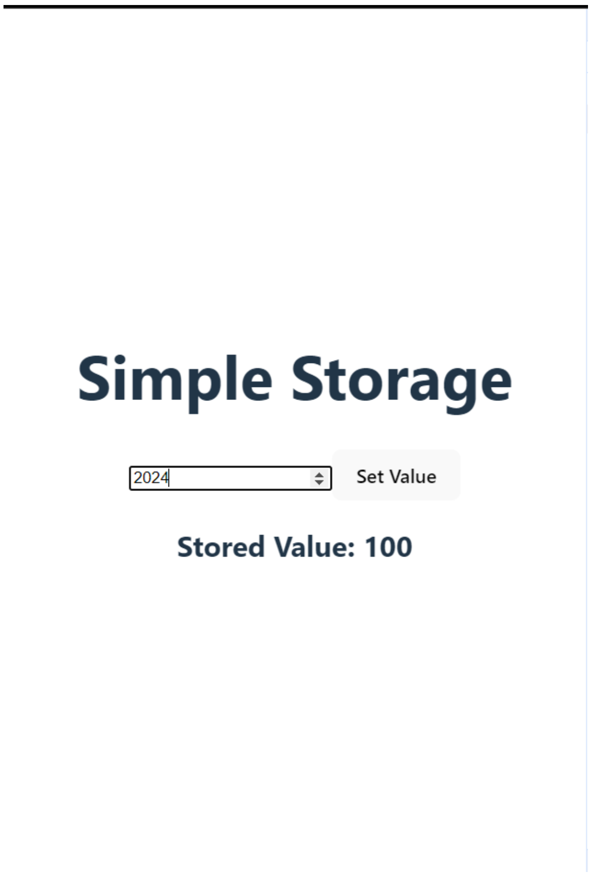


CURRENT BLOCK 9 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE TAWDRY-STOVE

interact_blockchain E:\cbt\interact_blockchain

NAME	ADDRESS	TX COUNT	
SimpleStorage	0×74850845E62574663CC49c38e1D59De4bB2b25fB	1	DEPLOYED

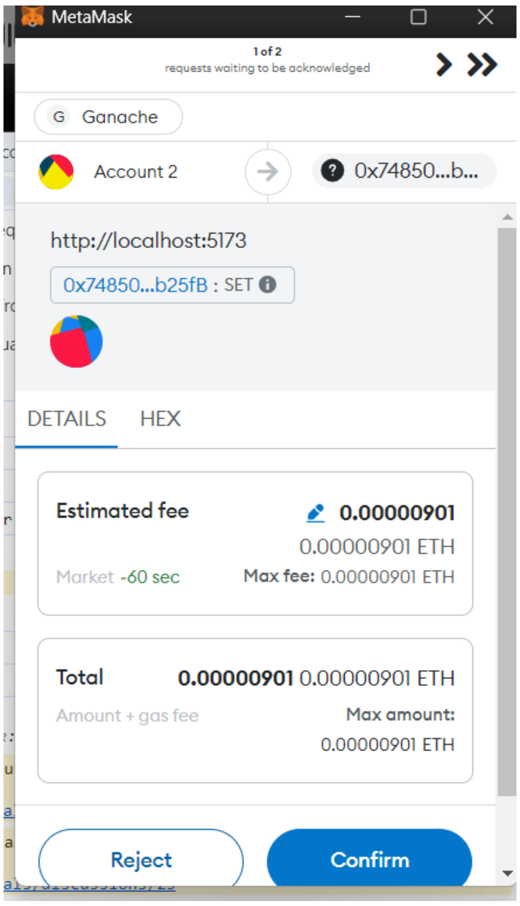
15. Now, testing our web app



Simple Storage

2024 Set Value

Stored Value: 100



MetaMask

1 of 2
requests waiting to be acknowledged

Ganache

Account 2 → 0x74850...b...

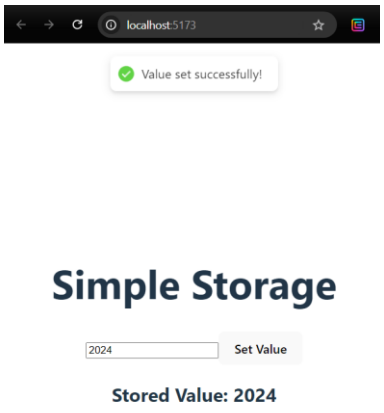
http://localhost:5173

0x74850...b25fB : SET

DETAILS HEX

Estimated fee	**0.00000901** 0.00000901 ETH
Market ~60 sec	**Max fee:** 0.00000901 ETH
Total	**0.00000901** 0.00000901 ETH
Amount + gas fee	**Max amount:** 0.00000901 ETH

Reject Confirm


localhost:5173

Value set successfully!

Simple Storage

2024

Stored Value: 2024

16. And now we can see the changed value as well as do transactions.

RESULT:

Thus, interacted with a blockchain network and executed transactions and requests against a blockchain network by creating an app to test the network and its rules.

EX.NO: 4	Deploy an asset-transfer app using blockchain.
DATE:	

AIM:

To create an asset transfer app using solidity smart contract and testing out using truffle

PROCEDURE:

1. Create a folder and initialize smart contract, using command truffle init
2. Inside the contracts folder create contract file and write the following code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleAssetTransfer {
    struct Asset {
        string name;
        address owner;
    }

    mapping(uint256 => Asset) public assets;
    mapping(address => uint256[]) private ownerAssets;

     event  AssetTransferred(uint256  indexed  assetId,  address  indexed  from,
address indexed to);

    function createAsset(uint256 _id, string memory _name) public {
        assets[_id] = Asset(_name, msg.sender);
        ownerAssets[msg.sender].push(_id);
    }

    function transferAsset(uint256 _assetId, address _to) public {
        require(assets[_assetId].owner == msg.sender, "Not the asset owner");
        require(_to != address(0), "Invalid recipient address");

        address previousOwner = assets[_assetId].owner;
        assets[_assetId].owner = _to;

        removeAssetFromOwner(previousOwner, _assetId);
        ownerAssets[_to].push(_assetId);

        emit AssetTransferred(_assetId, previousOwner, _to);
    }

    function getAssetOwner(uint256 _assetId) public view returns (address) {
        return assets[_assetId].owner;
    }

     function  getOwnerAssets(address  _owner)  public  view  returns  (uint256[]
memory) {
        return ownerAssets[_owner];
    }

    function removeAssetFromOwner(address _owner, uint256 _assetId) internal {
        uint256[] storage assetsOfOwner = ownerAssets[_owner];
        for (uint256 i = 0; i < assetsOfOwner.length; i++) {
            if (assetsOfOwner[i] == _assetId) {
                assetsOfOwner[i] = assetsOfOwner[assetsOfOwner.length - 1];
                assetsOfOwner.pop();
                break;
                }
            }
        }
    }
```

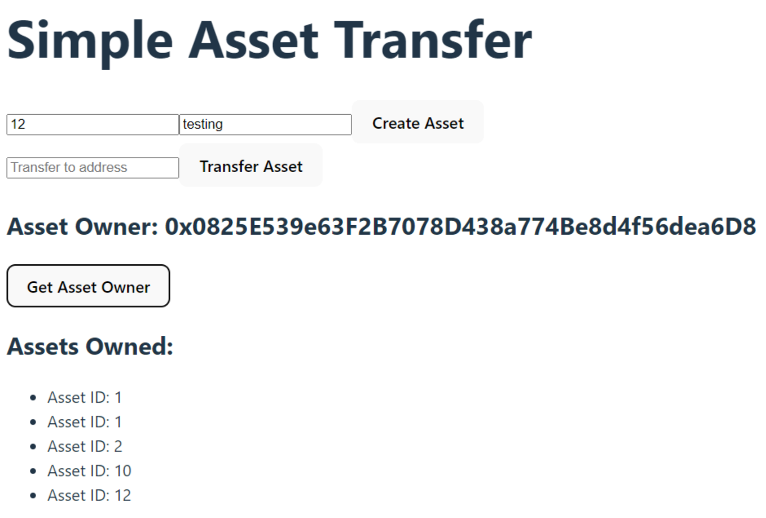
3. Inside migrations folder create a file 2_deploy_contracts.js and write the following code

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

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

4. Now deploy by using command truffle migrate –reset
5. After deploying, test the contract using command truffle console
6. Now develop a Frontend using React, using the code below

```
import { ethers } from "ethers";
import { useEffect, useState } from "react";
import toast from "react-hot-toast";
import { abi, contractAddress } from "./utils/constants/contract_data";

const SimpleAssetTransferComponent = () => {
  const [assetId, setAssetId] = useState("");
  const [assetName, setAssetName] = useState("");
  const [storedOwner, setStoredOwner] = useState("");
  console.log(storedOwner)
  const [valueToTransfer, setValueToTransfer] = useState("");
  const [loading, setLoading] = useState(false);
  const [ownerAssets, setOwnerAssets] = useState([]);

  useEffect(() => {
    if (storedOwner) {
      fetchOwnerAssets(storedOwner);
    }
  }, [storedOwner]);

  const createAsset = async (e) => {
    e.preventDefault();
    setLoading(true);
    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);
    try {
      const signer = await provider.getSigner();
      const tx = await contract.connect(signer).createAsset(assetId, assetName);
      await tx.wait();
      toast.success('Asset created successfully!');
    } catch (error) {
      console.error(error);
      toast.error('Error creating asset');
    } finally {
      setLoading(false);
    }
  };

  const transferAsset = async (e) => {
    e.preventDefault();
    setLoading(true);
    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);
    try {
      const signer = await provider.getSigner();
           const   tx   =   await   contract.connect(signer).transferAsset(assetId,
valueToTransfer);
      await tx.wait();
      toast.success('Asset transferred successfully!');
      fetchOwnerAssets(valueToTransfer);
    } catch (error) {
      console.error(error);
      toast.error('Error transferring asset');
    } finally {
      setLoading(false);
    }
  };

  const fetchOwner = async (id) => {
    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);
    try {
      const owner = await contract.getAssetOwner(id);
      setStoredOwner(owner);
      toast.success('Fetched asset owner successfully!');
    } catch (error) {
      console.error(error);
      toast.error('Error fetching asset owner');
    }
  };

  const fetchOwnerAssets = async (owner) => {
    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);
    try {
      const assets = await contract.getOwnerAssets(owner);
      setOwnerAssets(assets);
      console.log(assets)
    } catch (error) {
      console.error(error);
      toast.error('Error fetching owner assets');
    }
  };

  return (
    <div className="flex flex-col items-center mt-16">
```

```
          <h1  className="text-3xl  text-neutral-700  font-bold">Simple  Asset
    Transfer</h1>

          <form onSubmit={createAsset} className="flex flex-col items-center mt-6">
            <input
              type="number"
              value={assetId}
              onChange={(e) => setAssetId(e.target.value)}
              className="border rounded-md p-2 mb-2"
              placeholder="Asset ID"
              required
            />
            <input
              type="text"
              value={assetName}
              onChange={(e) => setAssetName(e.target.value)}
              className="border rounded-md p-2 mb-2"
              placeholder="Asset Name"
              required
            />
            <button
              type="submit"
              className="bg-blue-500 text-white px-3 py-2 rounded-md shadow-sm"
              disabled={loading}
            >
              {loading ? 'Creating Asset...' : 'Create Asset'}
            </button>
          </form>

          <form onSubmit={transferAsset} className="flex flex-col items-center mt-6">
            <input
              type="text"
              value={valueToTransfer}
              onChange={(e) => setValueToTransfer(e.target.value)}
              className="border rounded-md p-2 mb-2"
              placeholder="Transfer to address"
              required
            />
            <button
              type="submit"
              className="bg-blue-500 text-white px-3 py-2 rounded-md shadow-sm"
              disabled={loading}
            >
              {loading ? 'Transferring Asset...' : 'Transfer Asset'}
            </button>
          </form>

          <div className="mt-6">
            <h2 className="text-xl">Asset Owner: {storedOwner}</h2>
            <button
              onClick={() => fetchOwner(assetId)}
              className="mt-2 bg-blue-500 text-white px-3 py-2 rounded-md"
            >
              Get Asset Owner
            </button>
          </div>

          <div className="mt-6">
            <h2 className="text-xl">Assets Owned:</h2>
            <ul>
              {ownerAssets.map((id) => (
                <li key={id.toString()}>Asset ID: {id.toString()}</li>
              ))}
            </ul>
          </div>
        </div>
      );
    };

    export default SimpleAssetTransferComponent;
```


Simple Asset Transfer

12 | testing | Create Asset

Transfer to address | Transfer Asset

Asset Owner: 0x0825E539e63F2B7078D438a774Be8d4f56dea6D8

Get Asset Owner

Assets Owned:

- Asset ID: 1
- Asset ID: 1
- Asset ID: 2
- Asset ID: 10
- Asset ID: 12

RESULT:

Thus, an asset transfer app created using Ethereum Blockchain.

EX.NO: 5	Use blockchain to track fitness club rewards. Build a web app to track and trace member rewards
DATE:	

AIM:

To develop a web app to track the fitness club rewards using Blockchain

PROCEDURE:

1. Develop the smart contract for tracking fitness club rewards.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Fitness {
    struct Member {
        address memberAddress;
        uint256 points;
        bool isMember;
    }

    mapping(address => Member) public members;
    address public owner;

    event MemberAdded(address indexed memberAddress);
    event PointsUpdated(address indexed memberAddress, uint256 points);

    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner can execute this function");
        _;
    }

    modifier onlyMember() {
        require(members[msg.sender].isMember,  "Only  members  can  execute  this
function");
        _;
    }

    constructor() {
        owner = msg.sender;
    }

    function addMember(address _memberAddress) public onlyOwner {
        require(!members[_memberAddress].isMember, "Address is already a member");
        members[_memberAddress] = Member({
            memberAddress: _memberAddress,
            points: 0,
            isMember: true
        });
        emit MemberAdded(_memberAddress);
    }

    function updatePoints(address _memberAddress, uint256 _points) public onlyOwner {
        require(members[_memberAddress].isMember, "Address is not a member");
        members[_memberAddress].points += _points;
        emit PointsUpdated(_memberAddress, members[_memberAddress].points);
    }

    function getPoints(address _memberAddress) public view onlyMember returns (uint256) {
        require(members[_memberAddress].isMember, "Address is not a member");
        return members[_memberAddress].points;
    }

    function redeemPoints(address _memberAddress, uint256 _points) public onlyOwner {
        require(members[_memberAddress].isMember, "Address is not a member");
        require(members[_memberAddress].points >= _points, "Insufficient points");
        members[_memberAddress].points -= _points;
        emit PointsUpdated(_memberAddress, members[_memberAddress].points);
    }
}
```

2. Initialize a truffle project, write the above smart contract in the contracts folder and then in migrations folder create a js file and name it as **2_deploy_contracts.js**

   the 2_deploy_contracts.js should be in the below format

   ```
   const contract = artifacts.require(<contract-name>)
   module.exports = function (deployer) {
       deployer.deploy(contract);
   };
   ```

3. Then, modify the **truffle-config.js**

```
module.exports = {
  networks: {
    development: {
      host: '127.0.0.1',
      port: 7545,
      network_id: '*'
    }
  },
  compilers: {
    solc: {
      version: "0.8.19"
    }
  }
}
```

4. After configuring, deploy the contract by **truffle migrate –reset**

5. From build/contracts/<contract-name>.json , copy the **abi** and the **contract address** from **networks key** in the json file.

6. Now, integrate with Frontend Framework like React
7.
8. Install the following

```
npm i react-hot-toast ethers @metamask/detect-provider @metamask/sdk-react
```

9. In the src, create a folder utils/constants/contract.js and then paste the **abi** and **contractAddress.**

10. Wrap the Providers of react-hot-toast and metamask

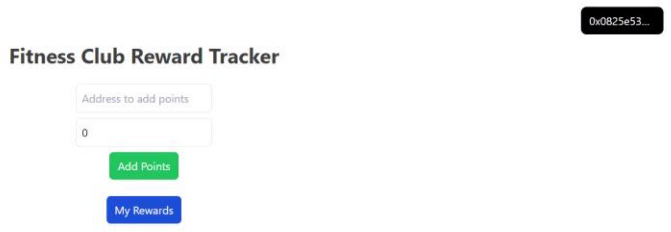
```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import { Toaster } from 'react-hot-toast'
import {MetaMaskProvider} from '@metamask/sdk-react'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <MetaMaskProvider>
    <Toaster/>
    <App />
    </MetaMaskProvider>
  </React.StrictMode>,
)
```

11. Create a component for the Fitness Club Reward Tracker

```
import detectEthereumProvider from "@metamask/detect-provider";
import { ethers } from "ethers";
import { useEffect, useState } from "react";

import { Trophy } from "lucide-react";
import toast from "react-hot-toast";
import { abi, contractAddress } from "./utils/constants/contract";

const FitnessComponent = () => {
  const [account, setAccount] = useState(null);
  const [dataloading,setDataLoading] = useState(false);
  const [addPointsLoading,setAddPointsLoading] = useState(false);
  const [isConnected, setIsConnected] = useState(false);
  const [network, setNetwork] = useState(null);
  console.log(network)
  const [data, setData] = useState(null);
  const [pointsToAdd, setPointsToAdd] = useState(0); // State for points to add
  const [addressToAdd, setAddressToAdd] = useState(""); // State for the address
to which points will be added

  useEffect(() => {
    const connectMetaMask = async () => {
      const provider = await detectEthereumProvider();
      if (provider) {
        const accounts = provider.selectedAddress ? [provider.selectedAddress] :
[];
        handleAccountsChanged(accounts);
        handleNetworkChanged(provider.networkVersion);
        provider.on('accountsChanged', handleAccountsChanged);
        provider.on('chainChanged', handleNetworkChanged);
      } else {
        toast.error('Please install MetaMask!');
      }
    };

    connectMetaMask();
  }, []);

  const handleAccountsChanged = (accounts) => {
    if (accounts.length === 0) {
      toast.error('Please connect to MetaMask.');
      setAccount(null);
      setIsConnected(false);
      setData(null);
    } else {
      setAccount(accounts[0]);
      setIsConnected(true);
    }
  };

  const fetchData = async () => {
    setDataLoading(true)
    if (!account) {
      toast.error('No account connected. Please connect your wallet.');
      return;
    }

    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);
    try {
      const data = await contract.checkRewardPoints(account);
      setData(data.toString() === '0' ? '0' : data.toString());
      setDataLoading(false)
    } catch (error) {
      console.error(error);
      toast.error('Failed to fetch data from contract');
    }
  };

  const handleNetworkChanged = (networkId) => {
    setNetwork(networkId);
  };

  const connectWallet = async () => {
    const provider = await detectEthereumProvider();
    if (provider) {
      try {
        const accounts = await provider.request({ method: 'eth_requestAccounts'
});
        handleAccountsChanged(accounts);
      } catch (err) {
        if (err.code === 4001) {
          toast.error('Please connect to MetaMask.');
        } else {
          console.error(err);
          toast.error('Something went wrong');
        }
      }
    } else {
      toast.error('Please install MetaMask!');
    }
  };

  const addRewardPoints = async () => {
    setAddPointsLoading(true)
    if (!account || !isConnected) {
      toast.error('Please connect to MetaMask and select an account.');
      return;
    }

    const provider = new ethers.BrowserProvider(window.ethereum);
    const contract = new ethers.Contract(contractAddress, abi, provider);

    try {
      const signer = await provider.getSigner();
      const tx = await contract.connect(signer).addRewardPoints(addressToAdd,
pointsToAdd);
      await tx.wait();
      fetchData();
      setPointsToAdd(0);
      setAddressToAdd("");
      setAddPointsLoading(false) // Reset address input after adding points
      toast.success('Reward points added successfully!'); // Success notification
    } catch (error) {
      console.error(error);
      toast.error('Error adding points');
    }
  };

  return (
    <div className="">
      <div className="fixed right-3 top-3">
        <button onClick={connectWallet} className="flex bg-black justify-center
text-white px-3 py-2 rounded-lg">
          <h1 className="w-24 truncate">{isConnected ? `${account}` : `Connect
Wallet`}</h1>
        </button>
      </div>
      <div className="mt-16 flex items-center flex-col ">
        <h1 className="text-3xl text-neutral-700 font-bold">Fitness Club Reward
Tracker</h1>

        <div className="flex flex-col items-center mt-6">
          <input
            type="text"
            value={addressToAdd}
            onChange={(e) => setAddressToAdd(e.target.value)}
            className="border rounded-md p-2 mb-2"
            placeholder="Address to add points"
          />
          <input
            type="number"
            value={pointsToAdd}
            onChange={(e) => setPointsToAdd(e.target.value)}
            className="border rounded-md p-2"
            placeholder="Points to add"
          />
          <button
            onClick={addRewardPoints}
            className="bg-green-500 mt-2 text-white px-3 py-2 rounded-md shadow-
sm"
            disabled={addPointsLoading || !isConnected || pointsToAdd <= 0 ||
!addressToAdd }
          >
           {
            addPointsLoading ? <div className="flex items-center gap-2">
                  <div className="h-3 w-3 border border-t-transparent rounded-full
animate-spin"></div>
                  Adding ...
            </div> : <div> Add Points</div>
           }
          </button>
        </div>


        <button
          onClick={fetchData}
          className="bg-blue-700 mt-6 text-white px-3 py-2 rounded-md shadow-sm"
          disabled={!isConnected}
        >
          {
            dataloading ? <div className="flex items-center gap-2">
                  <div className="h-3 w-3 border border-t-transparent rounded-full
animate-spin"></div>
                  Fetching...
            </div> : <div> My Rewards</div>
           }
        </button>
        {data !== null && <p className="flex items-center gap-2 mt-5">{data}
<Trophy className="text-yellow-400" size={18} /></p>}
```

```
            </div>
          </div>
        );
      };

      export default FitnessComponent;
```

12. The created web app will be like this



RESULT:

Thus, a web application for Tracking Fitness Club Rewards was doen with the help of Ethereum Blockchain.

EX.NO: 6	Use blockchain to create a Car Auction Network
DATE:	

AIM:

To create a car Auction network using Blockchain technology using Ethereum Blockchain

PROCEDURE:

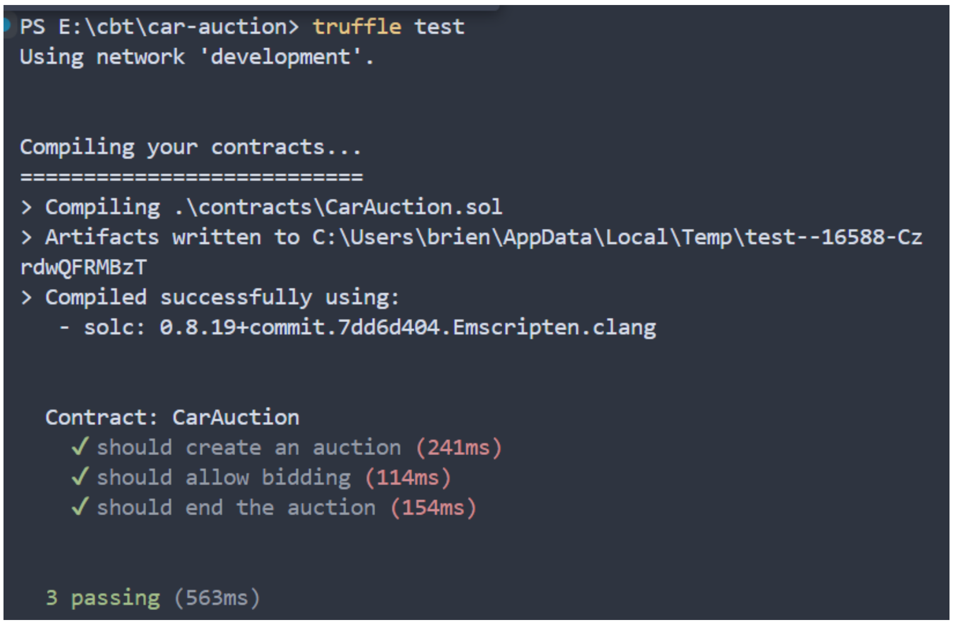
1. Develop the Smart contract by applying all logics required for creating a Car Auction Network.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CarAuction {
    struct Car {
        string make;
        string model;
        uint256 year;
    }

    struct Auction {
        Car car;
        address payable seller;
        uint256 startingBid;
        uint256 highestBid;
        address payable highestBidder;
        bool active;
    }

    uint256 public auctionCount;
    mapping(uint256 => Auction) public auctions;

    event AuctionCreated(uint256 indexed auctionId, address indexed seller,
string make, string model, uint256 year, uint256 startingBid);
    event HighestBidIncreased(uint256 indexed auctionId, address indexed bidder,
uint256 amount);
    event AuctionEnded(uint256 indexed auctionId, address indexed winner,
uint256 amount);

    function createAuction(string memory _make, string memory _model, uint256
_year, uint256 _startingBid) public {
        auctions[auctionCount] = Auction({
            car: Car({
                make: _make,
                model: _model,
                year: _year
            }),
            seller: payable(msg.sender),
            startingBid: _startingBid,
            highestBid: 0,
            highestBidder: payable(address(0)),
            active: true
        });
        emit AuctionCreated(auctionCount, msg.sender, _make, _model, _year,
_startingBid);
        auctionCount++;
    }

    function bid(uint256 _auctionId) public payable {
        Auction storage auction = auctions[_auctionId];
        require(auction.active, "Auction is not active");
         require(msg.value > auction.highestBid, "There already is a higher
bid");
       require(msg.value >= auction.startingBid, "Bid is lower than the starting
bid");

        if (auction.highestBidder != address(0)) {
            auction.highestBidder.transfer(auction.highestBid);
        }

        auction.highestBid = msg.value;
        auction.highestBidder = payable(msg.sender);
        emit HighestBidIncreased(_auctionId, msg.sender, msg.value);
    }

    function endAuction(uint256 _auctionId) public {
        Auction storage auction = auctions[_auctionId];
         require(msg.sender == auction.seller, "Only the seller can end the
auction");
        require(auction.active, "Auction is not active");

        auction.active = false;
        auction.seller.transfer(auction.highestBid);
                    emit AuctionEnded(_auctionId, auction.highestBidder,
auction.highestBid);
    }

    function getAuctionDetails(uint256 _auctionId) public view returns (string
memory make, string memory model, uint256 year, uint256 startingBid, uint256
highestBid, address highestBidder, bool active) {
        Auction storage auction = auctions[_auctionId];
             return (auction.car.make, auction.car.model, auction.car.year,
auction.startingBid, auction.highestBid, auction.highestBidder,
auction.active);
    }
}
```

2. Deploy the contract to truffle after configuring the truffle project
3. Write the test cases for the contract

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


contract("CarAuction", accounts => {
  it("should create an auction", async () => {
    const instance = await CarAuction.deployed();
    await instance.createAuction("Toyota", "Camry", 2020, web3.utils.toWei("1",
"ether"), { from: accounts[0] });
    const auction = await instance.auctions(0);
    assert.equal(auction.car.make, "Toyota");
    assert.equal(auction.car.model, "Camry");
    assert.equal(auction.car.year, 2020);
  });


  it("should allow bidding", async () => {
    const instance = await CarAuction.deployed();
    await instance.bid(0, { from: accounts[1], value: web3.utils.toWei("2",
"ether") });
    const auction = await instance.auctions(0);
    assert.equal(auction.highestBid, web3.utils.toWei("2", "ether"));
    assert.equal(auction.highestBidder, accounts[1]);
  });
```

```
    it("should end the auction", async () => {
      const instance = await CarAuction.deployed();
      await instance.endAuction(0, { from: accounts[0] });
      const auction = await instance.auctions(0);
      assert.equal(auction.active, false);
    });
  });
```



```
PS E:\cbt\car-auction> truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Compiling .\contracts\CarAuction.sol
> Artifacts written to C:\Users\brien\AppData\Local\Temp\test--16588-Cz
rdwQFRMBzT
> Compiled successfully using:
   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang


  Contract: CarAuction
    ✓ should create an auction (241ms)
    ✓ should allow bidding (114ms)
    ✓ should end the auction (154ms)


  3 passing (563ms)
```

RESULT:

Thus, a Car Auction network have been created by Ethereum Blockchain and tested