**Annexure 1**

**Ethereum based Todo-List Implementation Documentation**

### Download ganache

|  |
| --- |
| https://archive.trufflesuite.com/ganache/ |

### Install Truffle

|  |
| --- |
| npm install -g [truffle@5.0.2](mailto:truffle@5.0.2) |

### Install MetaMask google extension

### Verify the installation

|  |
| --- |
| Node -v - v20.12.2  Truffle version - v5.0.2 |

NOTE: For error like the file cannot be loaded because running scripts is disabled on this system, run the following command on the windows prompt

|  |
| --- |
| Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestricted |

### Initialize truffle

|  |
| --- |
| truffle init |

### Make a new file named ‘package.json’ in the directory of the project

(This file contains all the dependencies to run this project)

|  |
| --- |
| {  "name": "eth-todo-list",  "version": "1.0.0",  "description": "Blockchain Todo List Powered By Ethereum",  "main": "truffle-config.js",  "directories": {  "test": "test"  },  "scripts": {  "dev": "lite-server",  "test": "echo \"Error: no test specified\" && sexit 1"  },  "author": "sarthakaggarwal120@gmail.com",  "license": "ISC",  "devDependencies": {  "bootstrap": "4.1.3", //used for building client side application  //Dependencies to test the smart contracts  "chai": "^4.1.2",  "chai-as-promised": "^7.1.1",  "chai-bignumber": "^2.0.2",  "lite-server": "^2.3.0", //Server for running client side appn  "nodemon": "^1.17.3",  "truffle": "5.0.2",  "truffle-contract": "3.0.6"  }  } |

### Run the following command to install these dependencies

|  |
| --- |
| Npm install |

### Go to the contracts directory and make a new solidity file named ‘TodoList.sol’

# Smart Contract Implementation with explanation

|  |
| --- |
| // SPDX-License-Identifier: MIT  pragma solidity ^0.5.0;  contract TodoList{  uint public taskCount = 0;  } |

## **taskCount** – It is a state variable in solidity.  Variables declared in a contract that are not within any function are called state variables. Whenever this variable changes, the entire state of the contract/ blockchain changes. The scope of this variable belongs to the entire smart contract

## **public** - Public variables are accessible from within the contract and can be accessed from external contracts as well. Solidity automatically generates a getter function for public state variables.

### Compile the given code using this command

|  |
| --- |
| Truffle compile |

### You will see a new ‘TodoList.json’ file created in builds/contracts directory.

* This file contains ‘abi’ - Application Binary Interface that is used for the communication between the smart contract and the data script and other necessary details such as Bytecode that it is running on the Ethereum machine, etc

### Open truffle-config.js file and put this code into the file. Here we create a network that is connected to ganache which is defined by the host and port. This code helps us talk to the local blockchain.

|  |
| --- |
| module.exports = {  networks: {  development: {  host: "127.0.0.1", // Local host  port: 7545, // Port on which Ganache is running  network\_id: "\*" // Match any network id  }  },  solc: {  optimizer: {  enabled: true,  runs: 200  }  }  } |

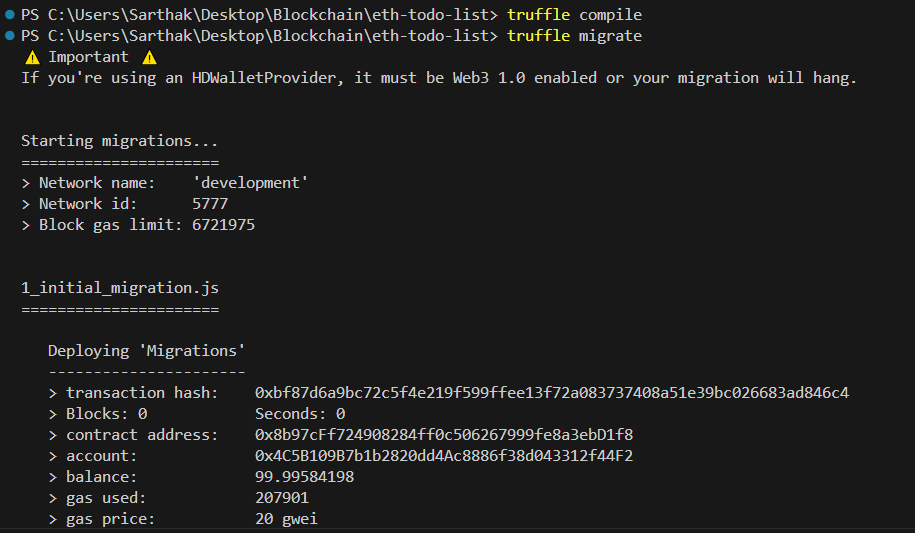
### Go to the migrant’s directory and create a file named ‘2\_deploy\_contracts.js’ and copy the contents of ‘1\_initial\_migration.js’ into this file. We are creating the migration file to get the smart contract on the blockchain

* The migration file is changing the state of the blockchain as we put our smart contract onto the blockchain. And hence we need migration.js file to do that.
* These files are numbered so that truffle knows in which order it has to execute them.

|  |
| --- |
| const TodoList = artifacts.require("TodoList");  module.exports = function(deployer) {  deployer.deploy(TodoList);  }; |

### To deploy the smart contract onto the blockchain. Use the following command

|  |
| --- |
| Truffle migrate |



A computer screen with white text

Description automatically generated

A computer screen with white text

Description automatically generated

A black background with white numbers

Description automatically generated

### Now to check the smart contract, we will open the truffle console using:

|  |
| --- |
| Truffle console |

### To retrieve the smart contract from the blockchain use:

|  |
| --- |
| todoList = await TodoList.deployed() |

* ‘TodoList” – is the name of the smart contract that we created in the migration
* We have retrieved the deployed copy of the smart contract into the variable ‘todoList’.
* Await – We must interact with the blockchain in an asynchronous manner.
* This is just saying that wait for the finished result of TodoList.deployed() and then assign it to the variable ‘todoList’

### To see the result of the smart contract, run the command in the truffle console

|  |
| --- |
| todoList |

### To get the address of the smart contract, use

|  |
| --- |
| todoList.address |

truffle(development)> todoList

TruffleContract {

  constructor: [Function: TruffleContract] {

    \_constructorMethods: {

      setProvider: [Function: setProvider],

      new: [Function: new],

      at: [Function: at],

      deployed: [Function: deployed],

      defaults: [Function: defaults],

      hasNetwork: [Function: hasNetwork],

      isDeployed: [Function: isDeployed],

      detectNetwork: [Function: detectNetwork],

      setNetwork: [Function: setNetwork],

      setWallet: [Function: setWallet],

      resetAddress: [Function: resetAddress],

      link: [Function: link],

      clone: [Function: clone],

      addProp: [Function: addProp],

      toJSON: [Function: toJSON],

      decodeLogs: [Function: decodeLogs]

    },

    \_properties: {

      contract\_name: [Object],

      contractName: [Object],

      gasMultiplier: [Object],

      timeoutBlocks: [Object],

      autoGas: [Object],

      numberFormat: [Object],

      abi: [Object],

      network: [Function: network],

      networks: [Function: networks],

      address: [Object],

      transactionHash: [Object],

      links: [Function: links],

      events: [Function: events],

      binary: [Function: binary],

      deployedBinary: [Function: deployedBinary],

      unlinked\_binary: [Object],

      bytecode: [Object],

      deployedBytecode: [Object],

      sourceMap: [Object],

      deployedSourceMap: [Object],

      source: [Object],

      sourcePath: [Object],

      legacyAST: [Object],

      ast: [Object],

      compiler: [Object],

      schema\_version: [Function: schema\_version],

      schemaVersion: [Function: schemaVersion],

      updated\_at: [Function: updated\_at],

      updatedAt: [Function: updatedAt],

      userdoc: [Function: userdoc],

      devdoc: [Function: devdoc]

    },

    \_property\_values: {},

    \_json: {

      contractName: 'TodoList',

      abi: [Array],

      bytecode: '0x608060405260008055348015601357600080fd5b50609e806100226000396000f3fe608060405260043610603f576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff168063b6cb58a5146044575b600080fd5b348015604f57600080fd5b506056606c565b6040518082815260200191505060405180910390f35b6000548156fea165627a7a72305820cf1f2e7364efb6f1359eba6973b3cb67b0ee09f06958537523a2629cfb4a37750029',

      deployedBytecode: '0x608060405260043610603f576000357c0100000000000000000000000000000000000000000000000000000000900463ffffffff168063b6cb58a5146044575b600080fd5b348015604f57600080fd5b506056606c565b6040518082815260200191505060405180910390f35b6000548156fea165627a7a72305820cf1f2e7364efb6f1359eba6973b3cb67b0ee09f06958537523a2629cfb4a37750029',

      sourceMap: '62:294:1:-;;;343:1;319:25;;62:294;8:9:-1;5:2;;;30:1;27;20:12;5:2;62:294:1;;;;;;;',

      deployedSourceMap: '62:294:1:-;;;;;;;;;;;;;;;;;;;;;;;;319:25;;8:9:-1;5:2;;;30:1;27;20:12;5:2;319:25:1;;;;;;;;;;;;;;;;;;;;;;;;;;;:::o',      source: '// SPDX-License-Identifier: MIT\r\n' +

        '\r\n' +

        'pragma solidity ^0.5.0;\r\n' +

        '\r\n' +

        'contract TodoList{\r\n' +

        '    // It is the state variable, Whenever this changes, the state of the contract/ Blockchain changes\r\n' +

        '    // The scope of this variable belongs to the entire smart contract\r\n' +

        '    // public - read the value from the smart contract\r\n' +

        '\r\n' +

        '    uint public taskCount = 0;\r\n' +

        '\r\n' +

        '\r\n' +

        '\r\n' +

        '\r\n' +

        '}\r\n',

      sourcePath: 'C:/Users/Sarthak/Desktop/Blockchain/eth-todo-list/contracts/TodoList.sol',

      ast: [Object],

      legacyAST: [Object],

      compiler: [Object],

      networks: [Object],

      schemaVersion: '3.0.1',

      updatedAt: '2024-08-09T07:19:57.720Z',

      devdoc: [Object],

      userdoc: [Object]

    },

    setProvider: [Function: bound setProvider],

    new: [Function: bound new] {

      estimateGas: [Function: bound estimateDeployment]

    },

    at: [Function: bound at],

    deployed: [Function: bound deployed],

    defaults: [Function: bound defaults],

    hasNetwork: [Function: bound hasNetwork],

    isDeployed: [Function: bound isDeployed],

    detectNetwork: [Function: bound detectNetwork],

    setNetwork: [Function: bound setNetwork],

    setWallet: [Function: bound setWallet],

    resetAddress: [Function: bound resetAddress],

    link: [Function: bound link],

    clone: [Function: bound clone],

    addProp: [Function: bound addProp],

    toJSON: [Function: bound toJSON],

    decodeLogs: [Function: bound decodeLogs],

    web3: Web3 {

      currentProvider: [Getter/Setter],

      \_requestManager: [RequestManager],

      givenProvider: null,

      providers: [Object],

      \_provider: [HttpProvider],

      setProvider: [Function (anonymous)],

      BatchRequest: [Function: bound Batch],

      extend: [Function],

      version: '1.0.0-beta.37',

      utils: [Object],

      eth: [Eth],

      shh: [Shh],

      bzz: [Bzz]

    },

    class\_defaults: {

      from: '0x4C5B109B7b1b2820dd4Ac8886f38d043312f44F2',

      gas: 6721975,

      gasPrice: 20000000000

    },

    currentProvider: HttpProvider {

      host: 'http://127.0.0.1:7545',

      httpAgent: [Agent],

      timeout: 0,

      headers: undefined,

      connected: true,

      send: [Function (anonymous)],

      \_alreadyWrapped: true

    },

    network\_id: '5777'

  },

  methods: {

    'taskCount()': [Function (anonymous)] {

      call: [Function (anonymous)],

      sendTransaction: [Function (anonymous)],

      estimateGas: [Function (anonymous)],

      request: [Function (anonymous)]

    }

  },

  abi: [

    {

      constant: true,

      inputs: [],

      name: 'taskCount',

      outputs: [Array],

      payable: false,

      stateMutability: 'view',

      type: 'function',

      signature: '0xb6cb58a5'

    }

  ],

  address: '0xbEdc7dC791ba30e72Bde6a20Cf872671eda2c923',

  transactionHash: undefined,

  contract: Contract {

    currentProvider: [Getter/Setter],

    \_requestManager: RequestManager {

      provider: [HttpProvider],

      providers: [Object],

      subscriptions: {}

    },

    givenProvider: null,

    providers: {

      WebsocketProvider: [Function: WebsocketProvider],

      HttpProvider: [Function: HttpProvider],

      IpcProvider: [Function: IpcProvider]

    },

    \_provider: HttpProvider {

      host: 'http://127.0.0.1:7545',

      httpAgent: [Agent],

      timeout: 0,

      headers: undefined,

      connected: true,

      send: [Function (anonymous)],

      \_alreadyWrapped: true

    },

    setProvider: [Function (anonymous)],

    BatchRequest: [Function: bound Batch],

    extend: [Function: ex] {

      formatters: [Object],

      utils: [Object],

      Method: [Function: Method]

    },

    clearSubscriptions: [Function (anonymous)],

    options: { address: [Getter/Setter], jsonInterface: [Getter/Setter] },

    defaultAccount: [Getter/Setter],

    defaultBlock: [Getter/Setter],

    methods: {

      taskCount: [Function: bound \_createTxObject],

      '0xb6cb58a5': [Function: bound \_createTxObject],

      'taskCount()': [Function: bound \_createTxObject]

    },

    events: { allEvents: [Function: bound ] },

    \_address: '0xbEdc7dC791ba30e72Bde6a20Cf872671eda2c923',

    \_jsonInterface: [ [Object] ]

  },

  taskCount: [Function (anonymous)] {

    call: [Function (anonymous)],

    sendTransaction: [Function (anonymous)],

    estimateGas: [Function (anonymous)],

    request: [Function (anonymous)]

  },

  sendTransaction: [Function (anonymous)],

  send: [Function (anonymous)],

  allEvents: [Function (anonymous)],

  getPastEvents: [Function (anonymous)]

}

truffle(development)> todoList.address

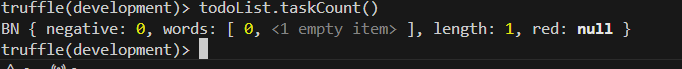
'0xbEdc7dC791ba30e72Bde6a20Cf872671eda2c923'

truffle(development)>

### To get the number of tasks that are created in the smart contract use:

Truffle saves the task as a big number BN

|  |
| --- |
| todoList.taskCount() |



### To convert the big number into a number use:

|  |
| --- |
| taskCount = await todoList.taskCount() taskCount.toNumber() |

A screenshot of a computer program

Description automatically generated

### Now we will initialize the git

|  |
| --- |
| Git init |

### Make a .gitignore file and write node\_modules/ into it so that it does not import all the node\_modules/

To commit the files

|  |
| --- |
| Git add . git commit -am “Project Setup” |