**VIVEKANAND EDUCATION SOCIETY’S**

**INSTITUTE OF TECHNOLOGY**

**Hashu Advani Memorial Complex, Collector’s Colony, R. C. Marg, Chembur, Mumbai – 400074.**



**A PROJECT REPORT**

**Project Title**

**BY**

**02-Sharven Angne**

**06-Sachin Bairi**

**07-Drishti Bhatia**

**Year 2022-23**

**Under the Guidance of**

**Mrs. Vaishali Gatty**

CERTIFICATE

This is to certify that the project entitled **“Eth- To do list”** is a bonafide work of “**Sharven Angne,** **Sachin Bairi, Drishti Bhatia” submitted** to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Master of Computer Applications”**.

(Name and sign) (Name and sign)

Supervisor/Guide Co-Supervisor/Guide

(Name and sign) (Name and sign)

Head of Department Principal

**ACKNOWLEDGEMENT**

With immense please we are presenting “**Eth- To do list**” Project report as part of the curriculum of ‘**Master of Computer Application**’.

We wish to thank all the people who gave us unending support. We, **Sharven Angne, Sachin Bairi, Drishti Bhatia** express our profound thanks to **Mrs. Vaishali Gatty** and all those who have indirectly guided and helped us in preparation of this project.

**Aim: To make a To-Do list Application using Ethereum.**

* **Abstract:**

Aim is to create a client-side application for the todo list that will talk directly to the blockchain. We've used the Ethereum blockchain in this project, which we can access by connecting our client-side application to a single Ethereum node. We'll write a smart contract in Solidity that powers the to-do list, and we'll deploy it to the Ethereum blockchain. We'll also connect to the blockchain network with our personal account using an Ethereum wallet in order to interact with the to-do list application. This server is a centralized entity that has full control over every aspect of the application. Anyone with full access to the server can change any part of the code or the data at any time. A blockchain application works quite differently. All the code and the data to the to-do list does not lie on a centralized server. Instead, it is distributed across the blockchain. All of the code and the data is shared and unchangeable on the blockchain.

* **Introduction:**

A blockchain is a peer-to-peer network of computers, or nodes, that talk to one another. It's a distributed network where all of the participants share the responsibility of running the network. Each network participant maintains a copy of the code and the data on the blockchain. All of this data is contained in bundles of records called "blocks" which are "chained together" to make up the blockchain.

All of the nodes on the network ensure that this data is secure and unchangeable, unlike a centralized application where the code and data can be changed at any time. Because the blockchain is responsible for storing data, it fundamentally is a database. And because it's a network of computers that talk to one another, it's a network. To be able to communicate with blockchain we’ll need to write a smart contract.

A smart contract can also be understood as a backend script that contacts the blockchain. The smart contract will allow us to store our to-do list tasks in the blockchain. By creating a to-do list using Ethereum we will learn about blockchain, how to use smart contracts with the Solidity programming language and deploy it to blockchain. In this way we will create a client side application for the todo list.

* **Working:**

1. We have created a client-side application for the to-do list that talks directly to the blockchain.
2. Using the Ethereum blockchain which we can access by connecting our client-side application to a single Ethereum node.
3. Programmed a smart contract in Solidity that powers the to-do list, and deployed it to the Ethereum blockchain.
4. Connected to the blockchain network with our personal account using an Ethereum wallet in order to interact with the to-do list application.

* **Requirements:**

To access the to-do list, users would use a web browser that would communicate with a web server over the Internet. The server contains all of the code and data for the to-do list.

**Ganache Personal Blockchain** - The dependency is a personal blockchain, which is a local development blockchain that can be used to mimic the behavior of a public blockchain. It will allow user to deploy smart contracts, develop applications, and run tests.

**Node.JS -** Used to configure users’ environment for developing smart contracts. The first dependency needed is Node Package Manager, or NPM, which comes with Node.js.

**Truffle Framework-** It provides a suite of tools for developing Ethereum smart contacts with the Solidity programming language.

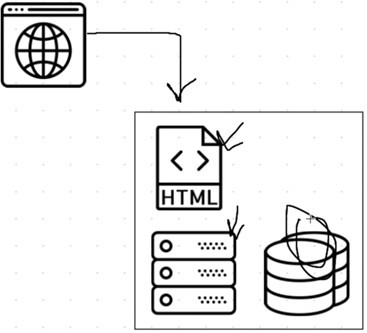
**MetaMask-** MetaMask allows people to manage their personal account when they connect to the blockchain, as well as manage their Ether funds that they'll need to pay for transactions.

**Solidity**- For compilation smart contracts.

Here is a list of what user would find on the server:

* Client-side files in HTML, CSS, and JavaScript
* Back-end code responsible for the application's business logic
* Database that stores the tasks in the to-do list

To illustrate this, let's examine how our blockchain-based todo list will work.



**Blockchain Application Diagram**

To access the blockchain todo list, we'll use a web browser to talk to the client side application, which will be written in HTML, CSS, and JavaScript. Instead of talking to a back end web server, the client side application will talk directly to the blockchain.

* **Source Code:**

**TodoList.sol**

pragma solidity ^0.5.0;

contract TodoList {

uint public taskCount = 0;

struct Task {

uint id;

string content;

bool completed;

}

mapping(uint => Task) public tasks;

event TaskCreated(

uint id,

string content,

bool completed

);

event TaskCompleted(

uint id,

bool completed

);

constructor() public {

createTask("SYMCA Block chain project");

createTask("Sachin Bairi");

createTask("Drishti Bhatia");

createTask("Sharven Angne");

}

function createTask(string memory \_content) public {

taskCount ++;

tasks[taskCount] = Task(taskCount, \_content, false);

emit TaskCreated(taskCount, \_content, false);

}

function toggleCompleted(uint \_id) public {

Task memory \_task = tasks[\_id];

\_task.completed = !\_task.completed;

tasks[\_id] = \_task;

emit TaskCompleted(\_id, \_task.completed);

}

}

**Migration.sol:**

pragma solidity >=0.4.21 <0.6.0; contract Migrations {

address public owner;

uint public last\_completed\_migration;

constructor() public {

owner = msg.sender;

}

modifier restricted() {

if (msg.sender == owner) \_;

}

function setCompleted(uint completed) public restricted { last\_completed\_migration = completed;

}

function upgrade(address new\_address) public restricted { Migrations upgraded = Migrations(new\_address);

upgraded.setCompleted(last\_completed\_migration);

}

}

**2\_deployed\_contracts.js:**

var TodoList = artifacts.require("./TodoList.sol");

module.exports = function(deployer) {

deployer.deploy(TodoList);

};

**Migrations.js:**

var Migrations = artifacts.require("./Migrations.sol"); module.exports = function(deployer) { deployer.deploy(Migrations);

};

**Package.json:**

{

"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": "[gregory@dappuniversity.com](mailto:gregory@dappuniversity.com)", "license": "ISC",

"devDependencies": { "bootstrap": "4.1.3",

"chai": "^4.1.2",

"chai-as-promised": "^7.1.1",

"chai-bignumber": "^2.0.2",

"lite-server": "^2.3.0",

"nodemon": "^1.17.3",

"truffle": "5.0.2",

"truffle-contract": "3.0.6",

"web3": "^0.20.6"

}

}

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Todo List</title>

<style>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;

}

.body {

height: 100%;

}

footer {

position: relative;

bottom: 0;

height: 100px;

background-color: skyblue;

display: flex;

align-items: center;

justify-content: center;

font-weight: 500;

font-size: large;

}

nav {

height: 85px;

background-color: skyblue;

padding: 10px 40px;

display: flex;

justify-content: space-between;

align-items: center;

}

.nav-title {

color: black;

font-size: 40px;

font-weight: 700;

}

.nav-address {

font-size: 20px;

font-weight: 700;

}

main {

margin-top: 20px;

width: 100vw;

display: flex;

justify-content: center;

align-items: center;

height: 74vh;

}

.form-control {

height: 50px;

width: 350px;

border: none;

outline: none;

border-radius: 8px;

padding-left: 10px;

font-size: x-large;

font-weight: 700;

/\* color: white; \*/

background-color: rgb(6, 181, 229);

}

::placeholder {

color: black;

}

.taskList {

margin-top: 40px;

}

.taskTemplate {

height: 50px;

font-weight: 600;

font-size: large;

background-color: rgb(181, 232, 237);

border-radius: 10px;

padding: 10px;

margin: 5px;

transition: all 250ms ease-out;

}

.taskTemplate:hover {

background-color: rgb(128, 193, 198);

cursor: pointer;

}

.checkbox {

height: 18px;

width: 18px;

padding: 10px;

}

.checkbox:hover {

cursor: pointer;

}

#content {

display: none;

}

form {

width: 350px;

margin-bottom: 10px;

}

ul {

margin-bottom: 0px;

}

#completedTaskList .content {

color: black;

text-decoration: line-through;

}

</style>

</head>

<body>

<nav class="navbar navbar-dark fixed-top bg-dark flex-md-nowrap p-0 shadow">

<p class="nav-title" href="" target="\_blank">Todo List</p>

<small>

<p class="nav-address" href="#"><span id="account"></span></p>

</small>

</nav>

<div class="body">

<div class="main">

<main role="main" class="card-todo">

<div id="loader" class="text-center">

<p class="text-center">Loading...</p>

</div>

<div id="content">

<form onSubmit="App.createTask(); return false;">

<input id="newTask" type="text" class="form-control" placeholder="Add task..." required />

<input type="submit" hidden="" />

</form>

<ul id="taskList" class="list-unstyled">

<div class="taskTemplate" class="checkbox" style="display: none">

<label>

<input type="checkbox" class="checkbox" />

<span class="content">Task content goes here...</span>

</label>

</div>

</ul>

<ul id="completedTaskList" class="list-unstyled"></ul>

</div>

</main>

</div>

</div>

<footer>

<p>Developed by Sharven Angne, Sachin Bairi and Drishti Bhatia </p>

</footer>

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<!-- Include all compiled plugins (below), or include individual files as needed -->

<script src="vendor/web3/dist/web3.js"></script>

<script src="vendor/truffle-contract/dist/truffle-contract.js"></script>

<script src="app.js"></script>

</body>

</html>

**App.js:**

App = {

loading: false,

contracts: {},

load: async () => {

await App.loadWeb3();

await App.loadAccount();

await App.loadContract();

await App.render();

web3.eth.defaultAccount = web3.eth.accounts[0];

},

loadWeb3: async () => {

if (typeof web3 !== "undefined") {

App.web3Provider = web3.currentProvider;

web3 = new Web3(web3.currentProvider);

} else {

window.alert("Please connect to Metamask.");

}

// Modern dapp browsers...

if (window.ethereum) {

window.web3 = new Web3(ethereum);

try {

// Request account access if needed

await ethereum.enable();

// Acccounts now exposed

web3.eth.sendTransaction({

/\* ... \*/

});

} catch (error) {

// User denied account access...

}

}

// Legacy dapp browsers...

else if (window.web3) {

App.web3Provider = web3.currentProvider;

window.web3 = new Web3(web3.currentProvider);

// Acccounts always exposed

web3.eth.sendTransaction({

/\* ... \*/

});

}

// Non-dapp browsers...

else {

console.log(

"Non-Ethereum browser detected. You should consider trying MetaMask!"

);

}

},

loadAccount: async () => {

// Set the current blockchain account

App.account = web3.eth.accounts[0];

console.log(App.account);

},

loadContract: async () => {

// Create a JavaScript version of the smart contract

const todoList = await $.getJSON("TodoList.json");

App.contracts.TodoList = TruffleContract(todoList);

App.contracts.TodoList.setProvider(App.web3Provider);

// Hydrate the smart contract with values from the blockchain

App.todoList = await App.contracts.TodoList.deployed();

},

render: async () => {

// Prevent double render

if (App.loading) {

return;

}

// Update app loading state

App.setLoading(true);

// Render Account

$("#account").html(App.account);

// Render Tasks

await App.renderTasks();

// Update loading state

App.setLoading(false);

},

renderTasks: async () => {

// Load the total task count from the blockchain

const taskCount = await App.todoList.taskCount();

for (var i = 1; i <= taskCount; i++) {

console.log("Task: " + i);

}

const $taskTemplate = $(".taskTemplate");

// Render out each task with a new task template

for (var i = 1; i <= taskCount; i++) {

// Fetch the task data from the blockchain

const task = await App.todoList.tasks(i);

const taskId = task[0].toNumber();

const taskContent = task[1];

const taskCompleted = task[2];

// Create the html for the task

const $newTaskTemplate = $taskTemplate.clone();

$newTaskTemplate.find(".content").html(taskContent);

$newTaskTemplate

.find("input")

.prop("name", taskId)

.prop("checked", taskCompleted)

.on("click", App.toggleCompleted);

// Put the task in the correct list

if (taskCompleted) {

$("#completedTaskList").append($newTaskTemplate);

} else {

$("#taskList").append($newTaskTemplate);

}

// Show the task

$newTaskTemplate.show();

}

},

createTask: async () => {

App.setLoading(true);

const content = $("#newTask").val();

await App.todoList.createTask(content);

window.location.reload();

},

toggleCompleted: async (e) => {

App.setLoading(true);

const taskId = e.target.name;

await App.todoList.toggleCompleted(taskId);

window.location.reload();

},

setLoading: (boolean) => {

App.loading = boolean;

const loader = $("#loader");

const content = $("#content");

if (boolean) {

loader.show();

content.hide();

} else {

loader.hide();

content.show();

}

},

};

$(() => {

$(window).load(() => {

App.load();

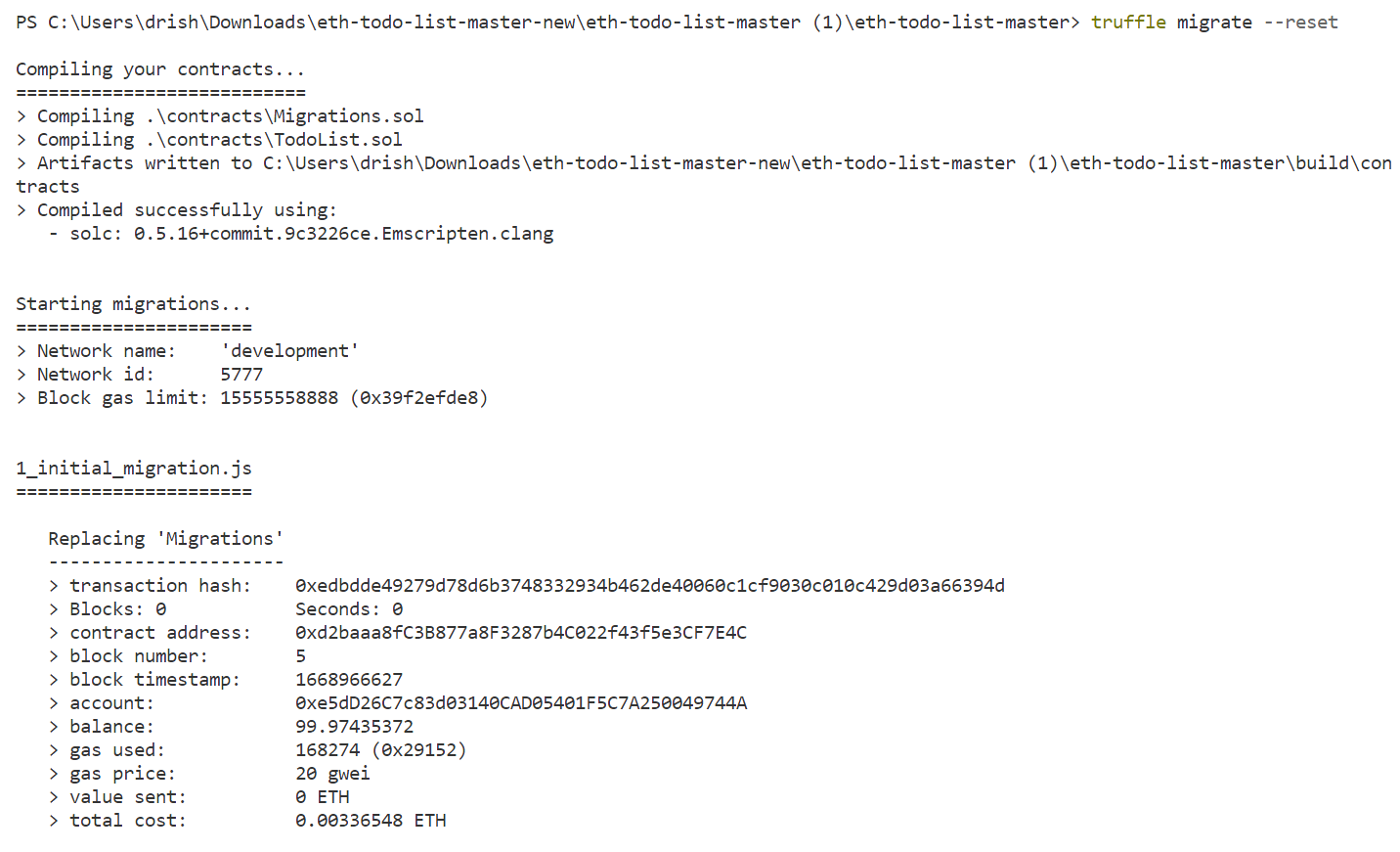
});

});

**Output:**

**Console**

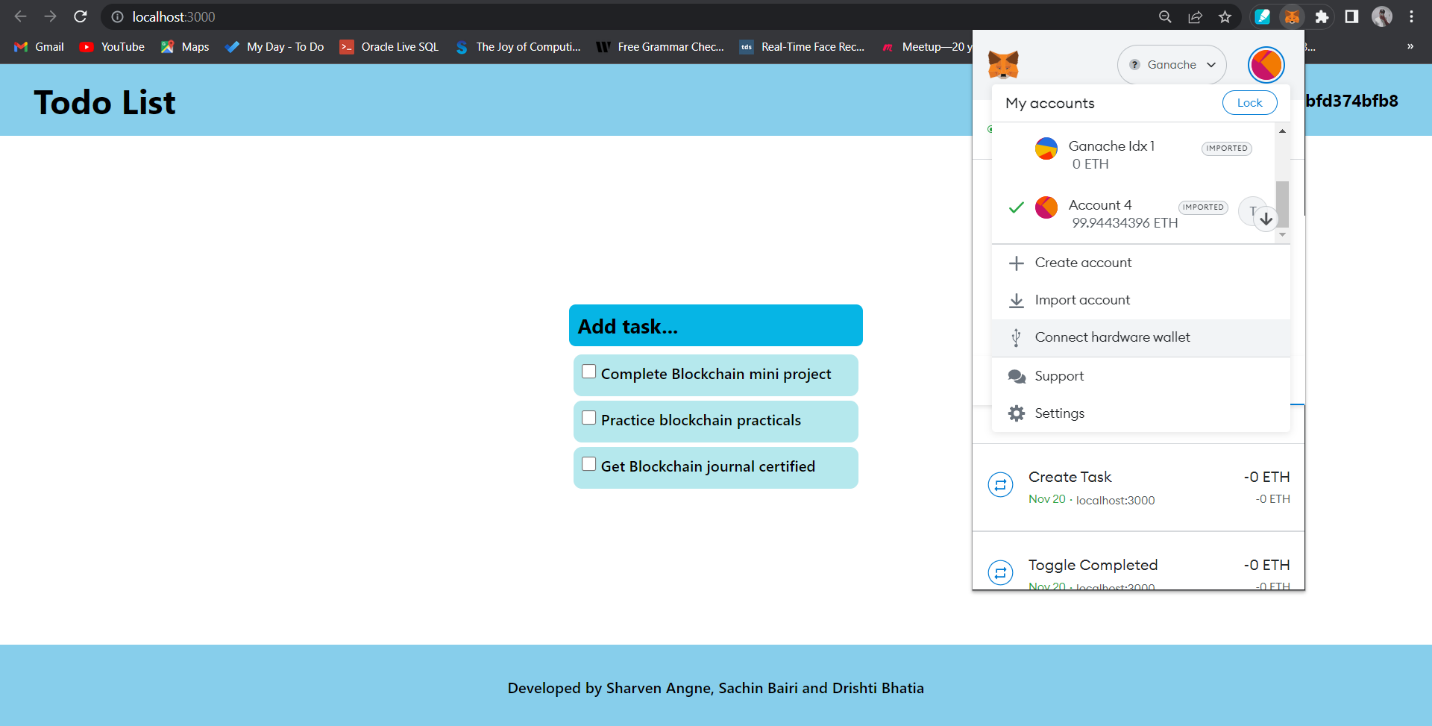
**Command: truffle migrate –reset**

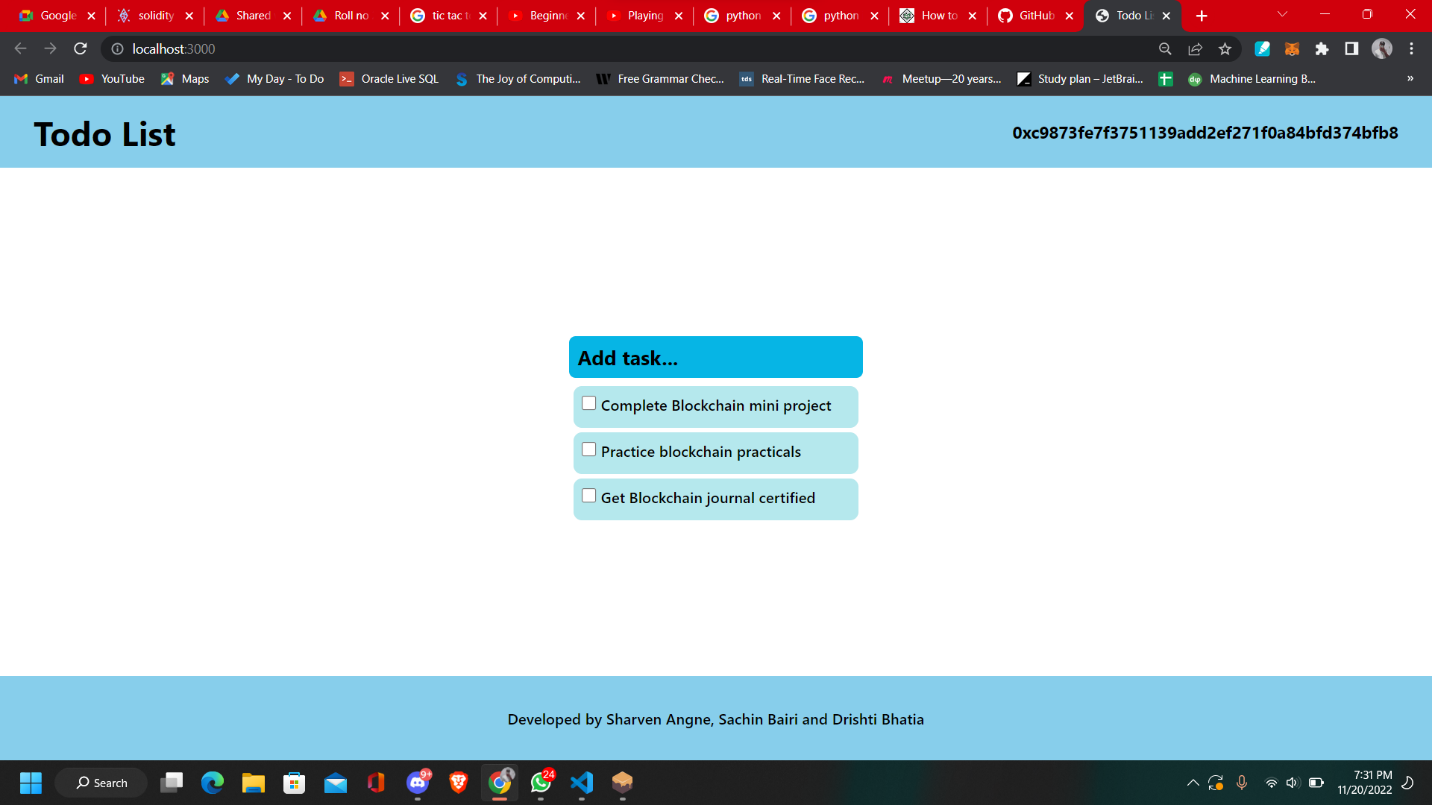
****

**Command: npm run dev**

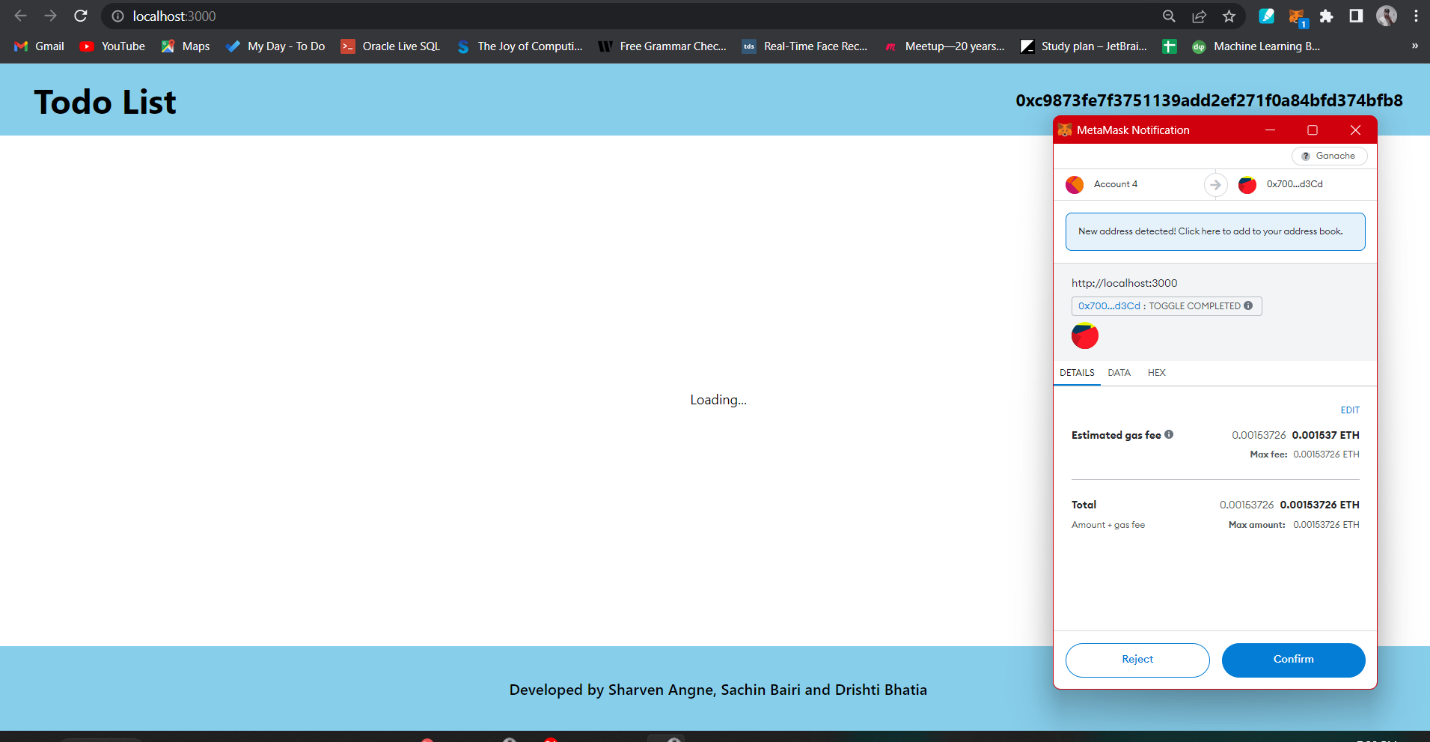
****

**Connected to metamask as we deploy the project**

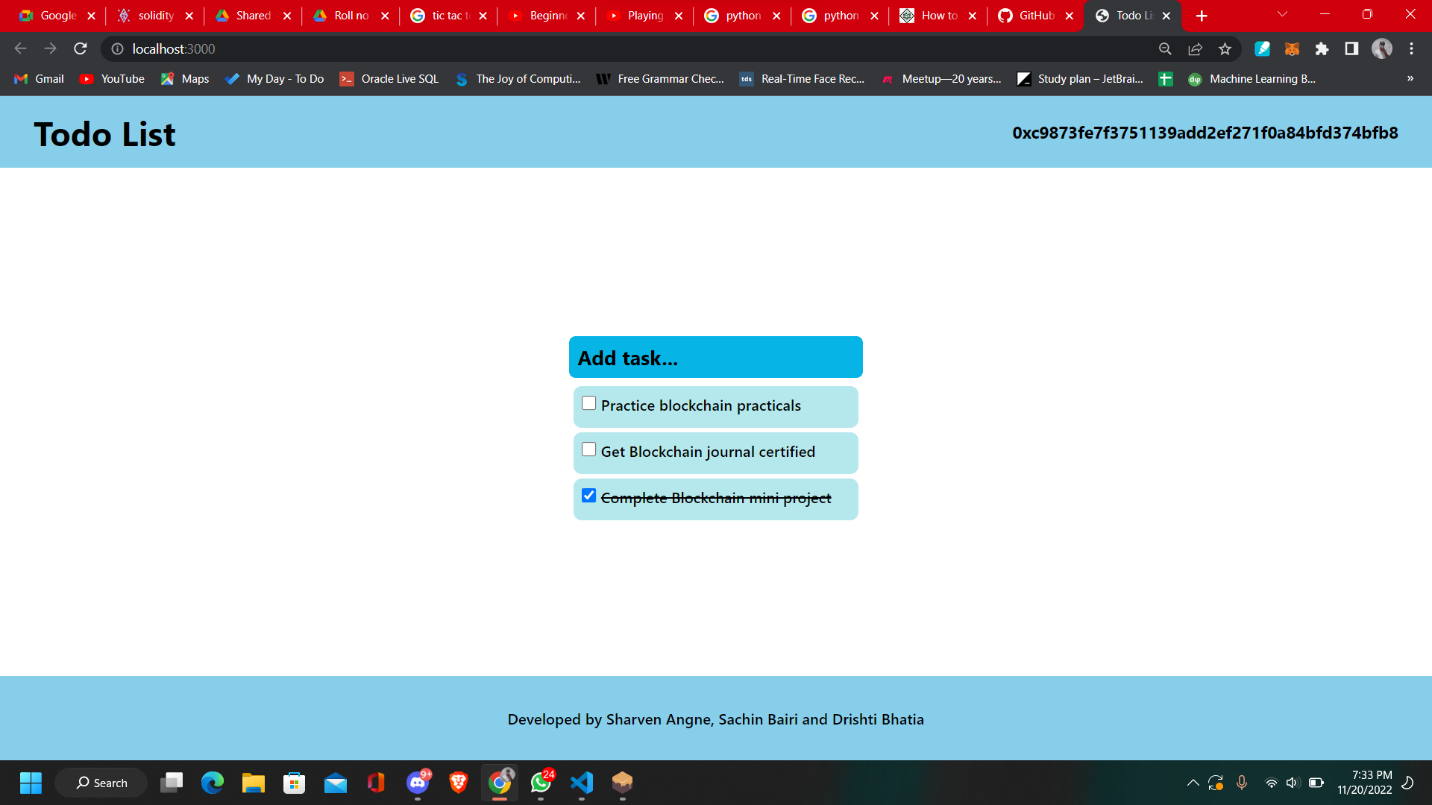
**To-do list appears**



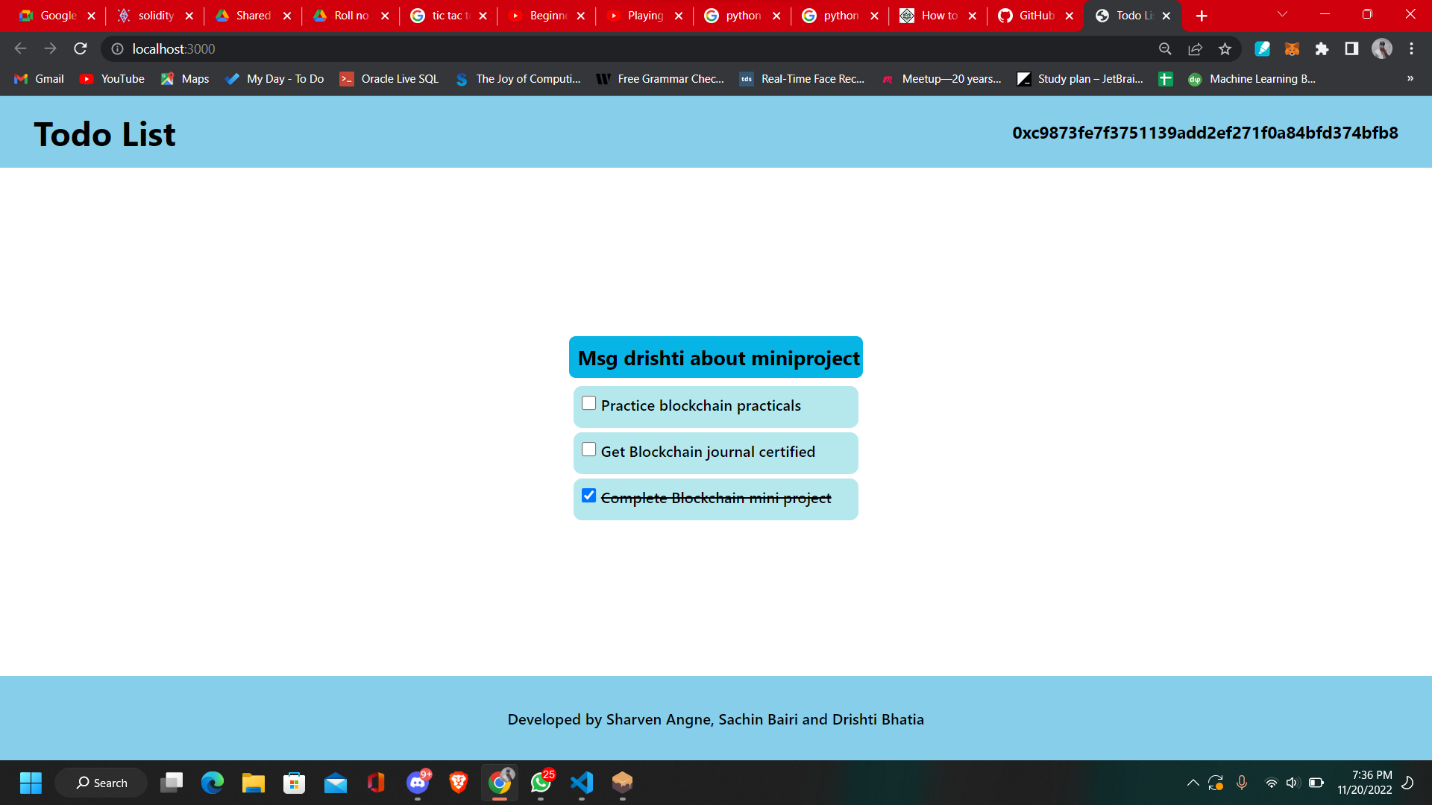
**Checking the task and confirm the transactions**



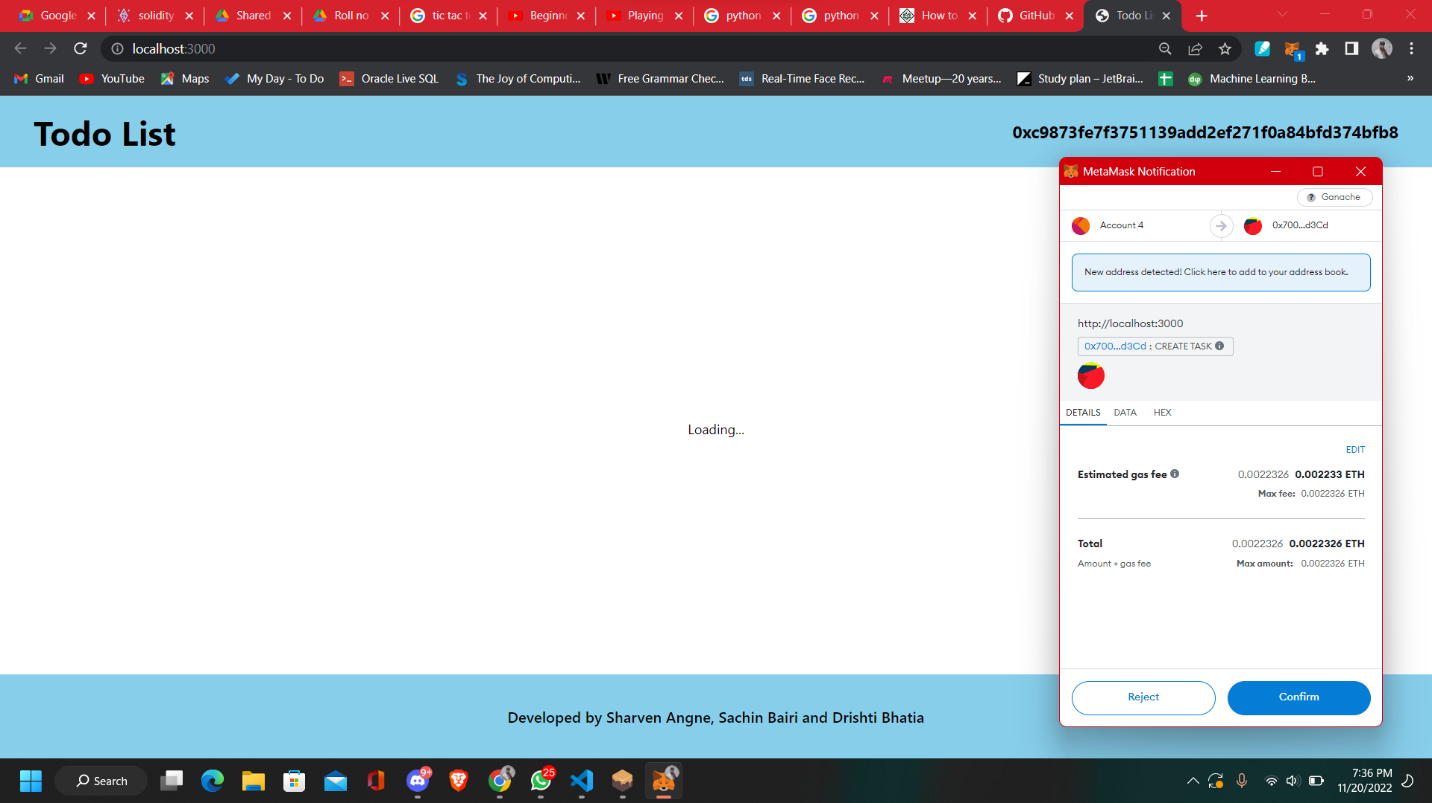
**After confirming:**



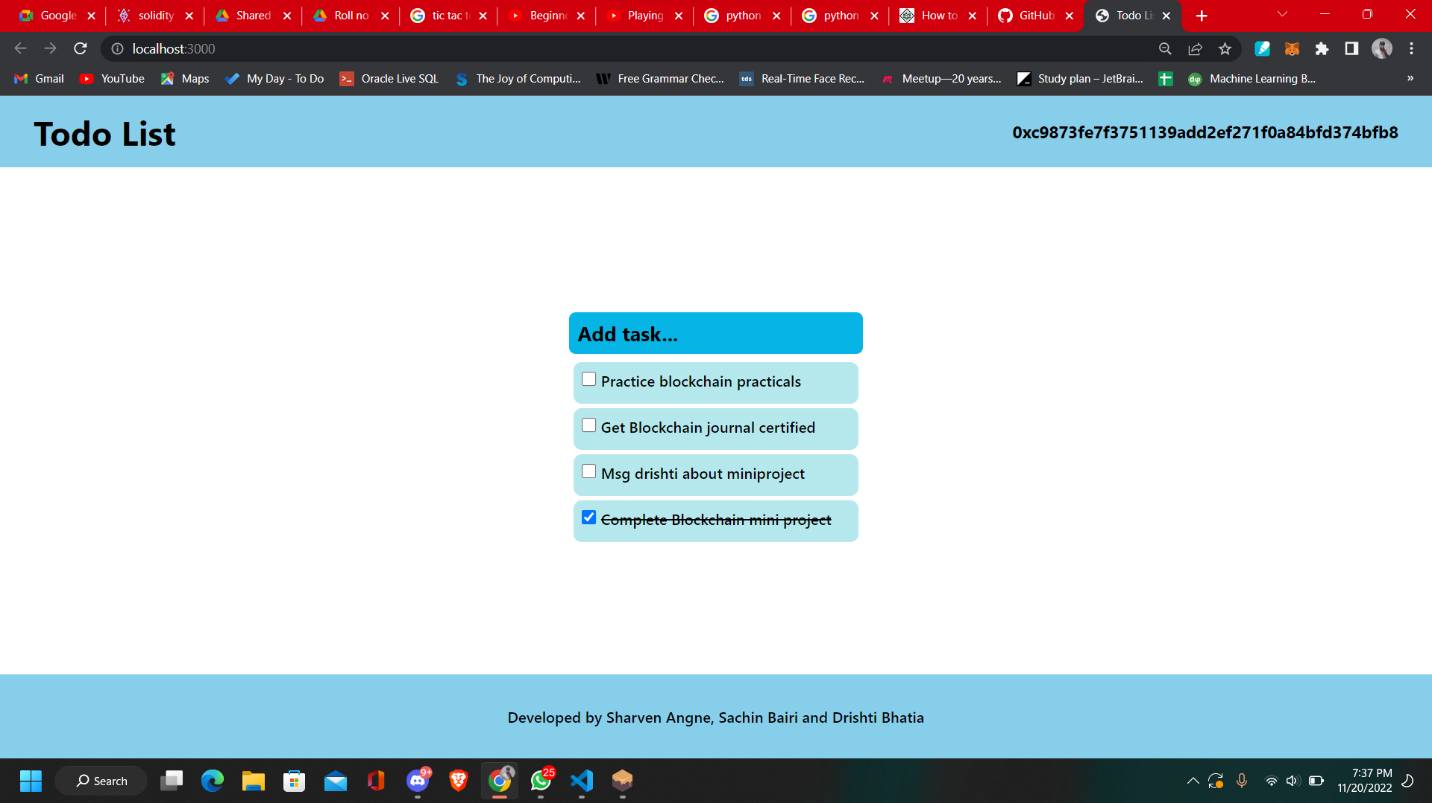
**Adding task in the list:**



**Confirming transaction**

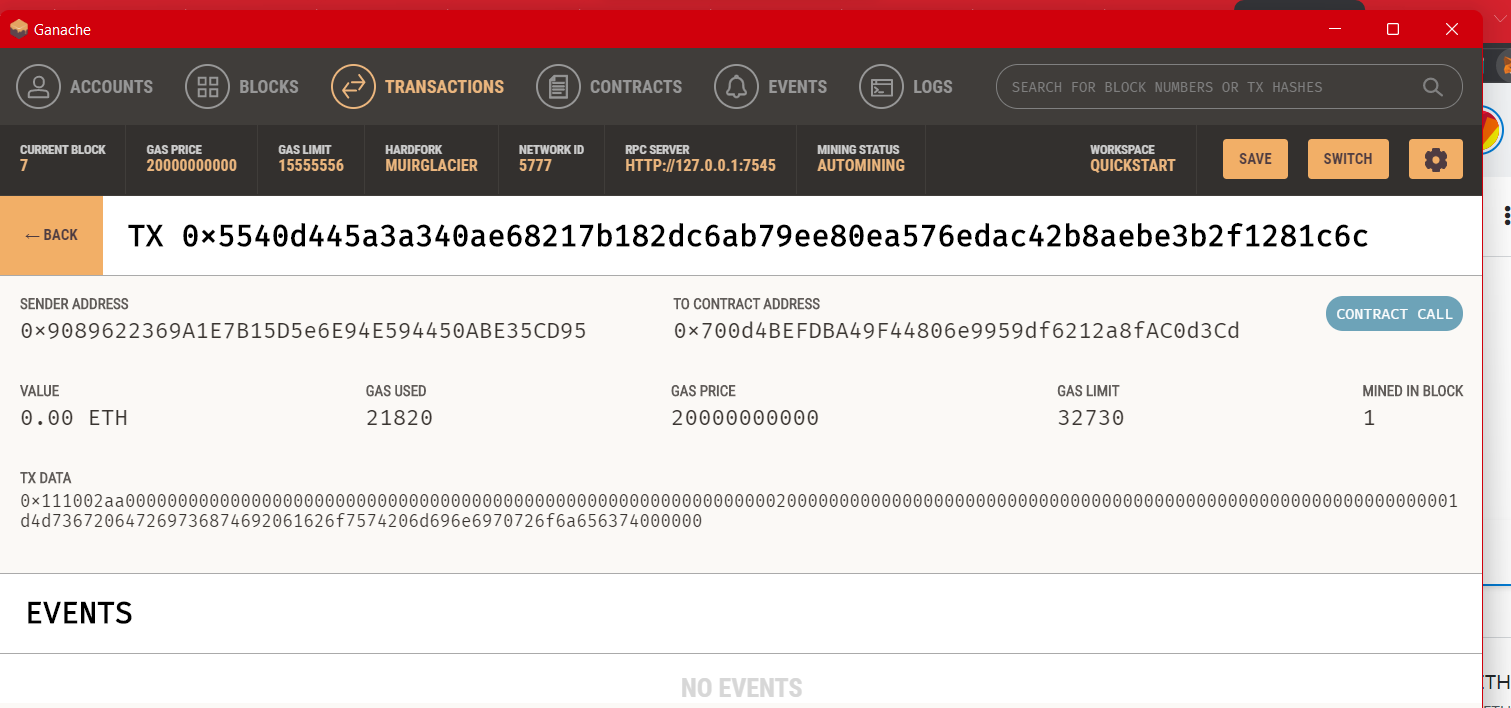


**New Task Added**



**The new item will be added to the list. Similarly, many items can be added or deleted from the list.**

**Transaction Loss**



**Conclusion:**

Successfully built a full stack blockchain application powered by Ethereum smart contracts (A To-do list application in ethereum) and learnt some of the concepts used for making block chain applications.

**Reference:**

Adhami S, Giudici G, Martinazzi S (2018) Why do businesses go crypto? An empirical analysis of initial coin offerings. J Econ Bus 100:64–75

Dabbagh, Mohammad, Mehdi Sookhak, and Nader Sohrabi Safa. "The evolution of blockchain: A bibliometric study." Ieee Access 7 (2019): 19212-19221.

Dabbagh, M., Sookhak, M. and Safa, N.S., 2019. The evolution of blockchain: A bibliometric study. Ieee Access, 7, pp.19212-19221.

Lapointe, Cara, and Lara Fishbane. "The blockchain ethical design framework."

Innovations: Technology, Governance, Globalization 12.3-4 (2019): 50-71. Dapp University “Blockchain Application” reference from Youtube

Herian, Robert. "Taking blockchain seriously." Law and Critique 29.2 (2018): 163-171. Link: https://en.wikipedia.org/wiki/Blockchain