Decentralized Greeting DApp

PROBLEM STATEMENT:

Objective: Expand the basic greeting smart contract into a functional DApp.
Key Features:
Create a smart contract to store and update greetings.
Build a frontend where users can read and update the greeting.
Connect the frontend with Ethereum using web3.js.
Objectives:
Basics of smart contracts, web3.js integration, and DApp structure.
Tools:
Solidity, Truffle, Ganache, MetaMask, and HTML/JavaScript.

—----------

Project Overview:

The goal of this project is to create a DApp that allows users to:

- Read the current greeting stored on the blockchain.
- Update the greeting to a new message.

The smart contract stores the greeting, and the frontend allows users to interact with it. When the user updates the greeting, the new value is stored on the blockchain.

Setup the Development Environment:

Before starting the project, ensure you have the following installed:

→Node.js
→Truffle
→Ganache
→MetaMask
→web3.js

Installation

Truffle :

npm install -g truffle

Node.js:

https://nodejs.org/dist/v22.12.0/node-v22.12.0-x64.msi

Ganache:

https://github.com/trufflesuite/ganache-ui/releases

MetaMask:

Browser extension and create an account.

Write the Smart Contract:

Create a folder for the project, and navigate to it using the terminal.

1.Initialize the Truffle project:

```
mkdir GreetingDApp
cd GreetingDApp
 truffle init
```

2.Inside the **contracts** folder, create a new file called **Greeting.sol:**

Greeting.sol:

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

contract Greeting {
   string public greeting;

   constructor(string memory _greeting) {
      greeting = _greeting;
   }

   function getGreeting() public view returns (string memory) {
      return greeting;
   }

   function setGreeting(string memory _greeting) public {
      greeting = _greeting;
   }
}
```

Test the Smart Contract:

Create a migration script in the **migrations** folder:

1.Create a new file **2_deploy_greeting.js**:

2_deploy_greeting.js:

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

module.exports = function(deployer) {
deployer.deploy(Greeting, "Hello, Ethereum!");
};
```

2.Compile and deploy the smart contract to the local Ganache blockchain:

```
truffle compile
truffle migrate
```

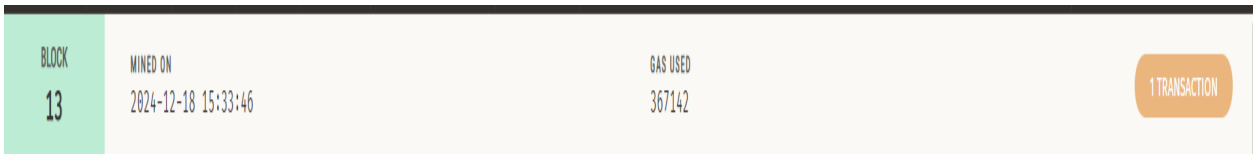
When we migrate we will get an build.js file

OUTPUT:

```
C:\Users\TEJASWINI.K\DecentralizedGreeting>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\Greeting.sol
> Artifacts written to C:\Users\TEJASWINI.K\DecentralizedGreeting\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang
```

```
C:\Users\TEJASWINI.K\DecentralizedGreeting>truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\Greeting.sol
> Artifacts written to C:\Users\TEJASWINI.K\DecentralizedGreeting\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_contracts.js
=====================

   Replacing 'Greeting'
   --------------------
   > transaction hash:    0x12e73507d7f807357381be665875d5374b23320b7aaa8179ed2fe4b5dca82468
   > Blocks: 0            Seconds: 0
   > contract address:    0x63ad331d189039C77bcaF2F455a8A072Bdb6689f
   > block number:        12
   > block timestamp:     1734516173
   > account:             0x28c203924599Bf027dA28dDe49915A593cA3d914
   > balance:             98.997813386798288018
   > gas used:            367142 (0x59a26)
   > gas price:           2.707044547 gwei
   > value sent:          0 ETH
   > total cost:          0.000993869749074674 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:     0.000993869749074674 ETH

Summary
```

The block has bee added:



Build the Frontend:

Create a frontend folder inside your project and create an **index.html** file

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Decentralized Greeting DApp</title>
</head>
<body>
  <h1>Decentralized Greeting DApp</h1>
  <div>
    <p>Current Greeting: <span id="currentGreeting">Loading...</span></p>
    <input type="text" id="newGreeting" placeholder="Enter new greeting">
    <button onclick="updateGreeting()">Update Greeting</button>
  </div>
  <script
    src="https://cdn.jsdelivr.net/npm/web3@1.8.1/dist/web3.min.js"></script>
  <script src="app.js"></script>
</body>
</html>
```

Connect the Frontend with Ethereum using web3.js:

In the same frontend folder, create a **app.js** file:

app.js:

```
const contractAddress = "0x1a1D3a32111B4a4873ed8DE589175b49ACB15C8c"; //
    Replace with your contract address
const contractABI = [{
  "inputs": [
    {
      "internalType": "string",
      "name": "initialGreeting",
      "type": "string"
    }
  ],
  "stateMutability": "nonpayable",
```

```
  "type": "constructor"
},
{
  "inputs": [],
  "name": "greeting",
  "outputs": [
    {
      "internalType": "string",
      "name": "",
      "type": "string"
    }
  ],
  "stateMutability": "view",
  "type": "function",
  "constant": true
},
{
  "inputs": [
    {
      "internalType": "string",
      "name": "newGreeting",
      "type": "string"
    }
  ],
  "name": "updateGreeting",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
},
{
  "inputs": [],
  "name": "getGreeting",
  "outputs": [
    {
      "internalType": "string",
      "name": "",
      "type": "string"
```

```
      }
    ],
    "stateMutability": "view",
    "type": "function",
    "constant": true
  }]; // Replace with your contract ABI

const web3 = new Web3("http://127.0.0.1:7545"); // Connect to Ganache
const greetingContract = new web3.eth.Contract(contractABI, contractAddress);

// Load the current greeting
async function loadGreeting() {
    const greeting = await greetingContract.methods.getGreeting().call();
    document.getElementById("greeting").innerText = greeting;
}

// Update the greeting
async function updateGreeting() {
    const newGreeting = document.getElementById("newGreeting").value;
    const accounts = await web3.eth.getAccounts(); // Fetch accounts from Ganache
    await greetingContract.methods.updateGreeting(newGreeting).send({ from:
      accounts[0] });
    loadGreeting(); // Reload greeting
}

// Initialize
window.onload = loadGreeting;
```

Deploy the Smart Contract to the Blockchain:

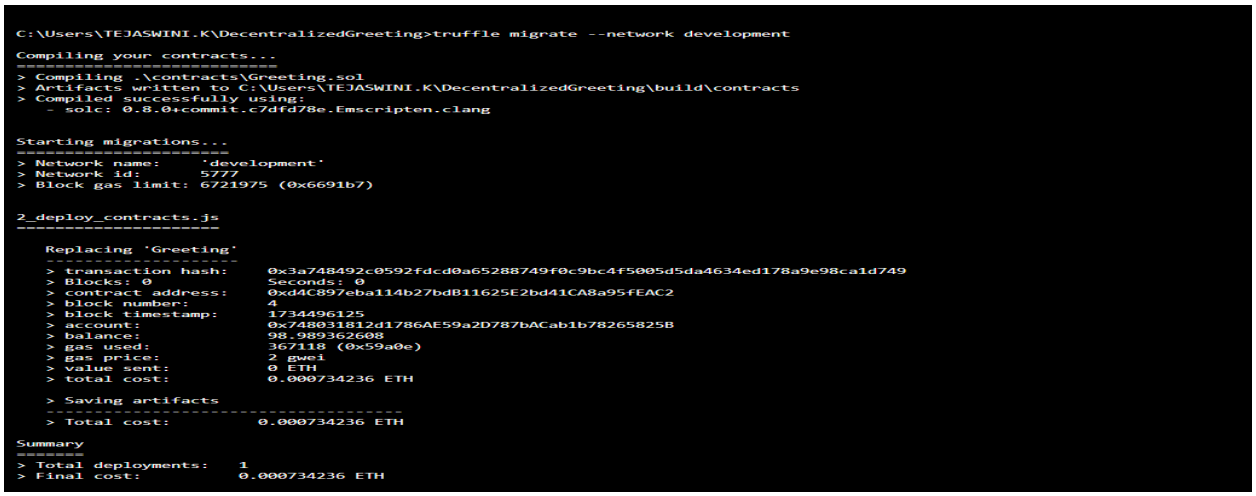
→First, ensure Ganache is running.

→Deploy the contract to Ganache using the following command:

truffle migrate --network development

→After deployment, copy the contract address from the terminal and update the contractAddress in app.js.

OUTPUT:



Run and Test the DApp:

Run the DApp;

→Copy the ABI JSON from the `build/contracts/Greeting.json` file into `app.js` in the `contractABI` section.

→Replace `<DEPLOYED_CONTRACT_ADDRESS>` with the contract address displayed after deployment.

→Start a simple HTTP server to serve the frontend:

npx http-server src

Test the DApp:

→Open the **index.html** file in your browser.

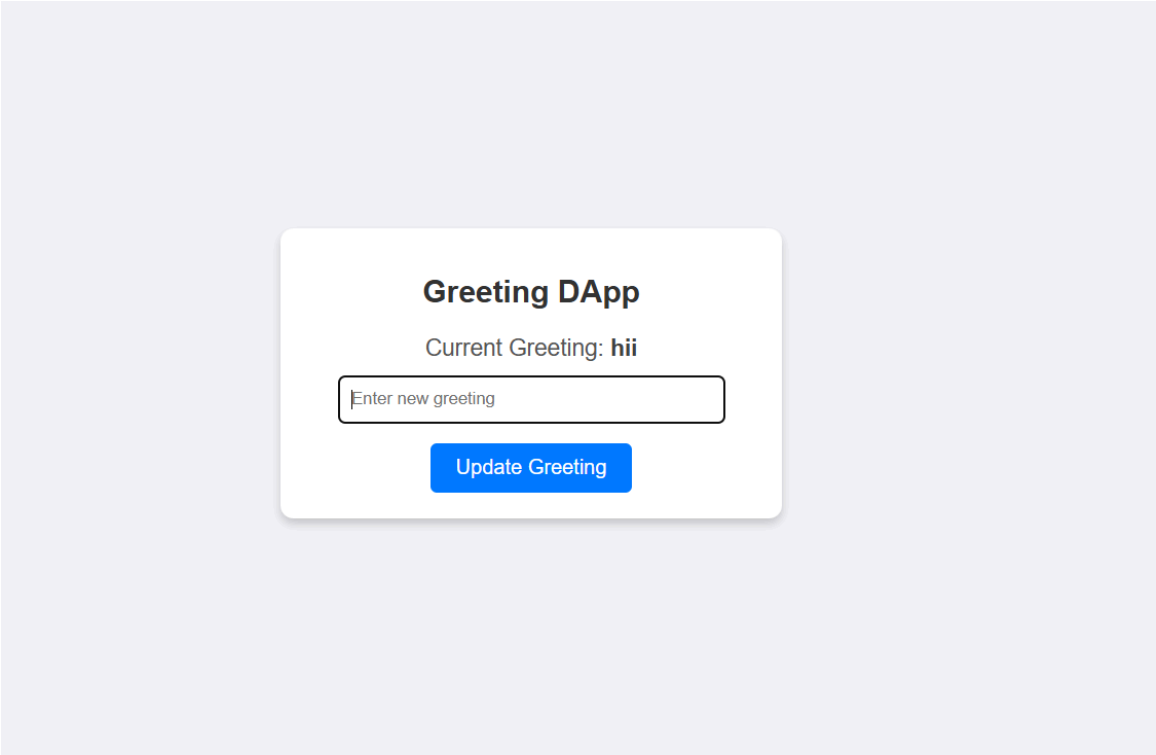
→Ensure MetaMask is connected to your Ganache account.

→You should see the current greeting displayed. Enter a new greeting and click the button to update it.

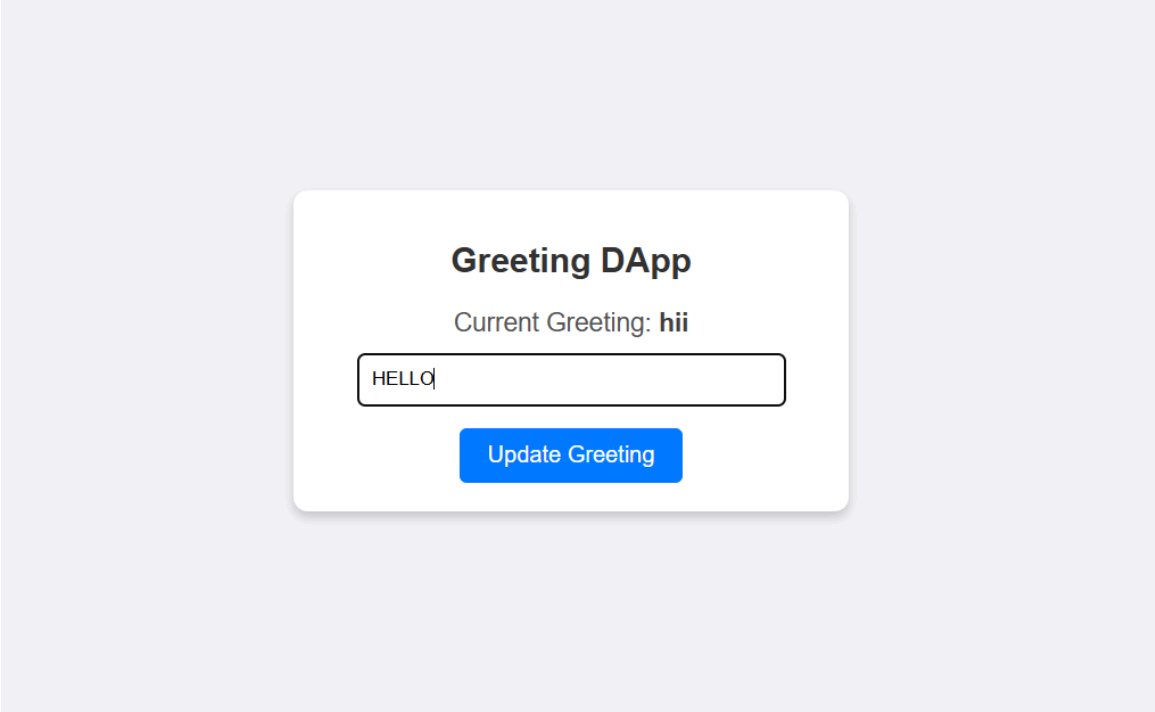
→The greeting will be updated on the blockchain, and the page will reload to reflect the change.

OUTPUTS:

Add the new Greeting in the entry box:



Click on Update Greeting the block will be added in the Ganche:

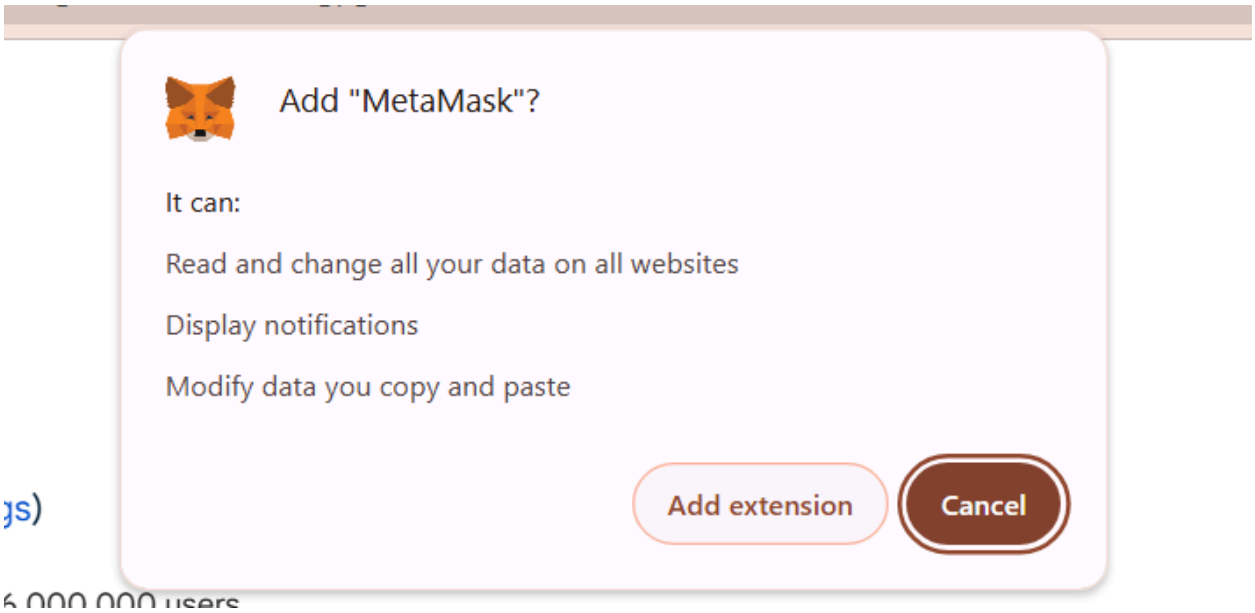


Click on update greeting.Then will be added in the Ganache.

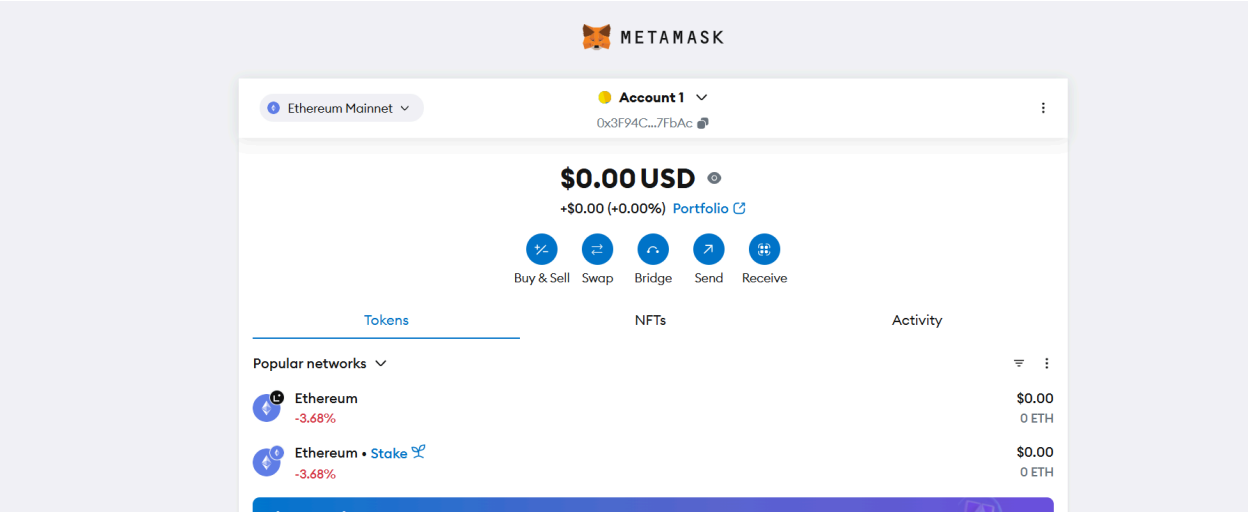
BLOCK	MINED ON	GAS USED	
14	2024-12-18 15:37:07	21508	1 TRANSACTION
13	2024-12-18 15:33:46	367142	1 TRANSACTION
12	2024-12-18 15:32:53	367142	1 TRANSACTION
11	2024-12-18 15:23:53	21532	1 TRANSACTION
10	2024-12-18 15:23:52	21532	1 TRANSACTION
9	2024-12-18 15:22:46	21532	1 TRANSACTION
8	2024-12-18 15:22:46	21532	1 TRANSACTION
7	2024-12-18 15:22:45	21532	1 TRANSACTION
BLOCK	MINED ON	GAS USED	

METAMASK:

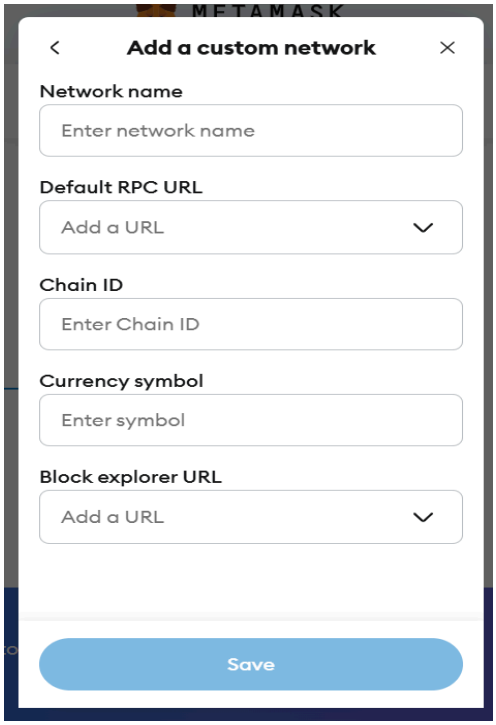
Add MetaMask in the chrome:



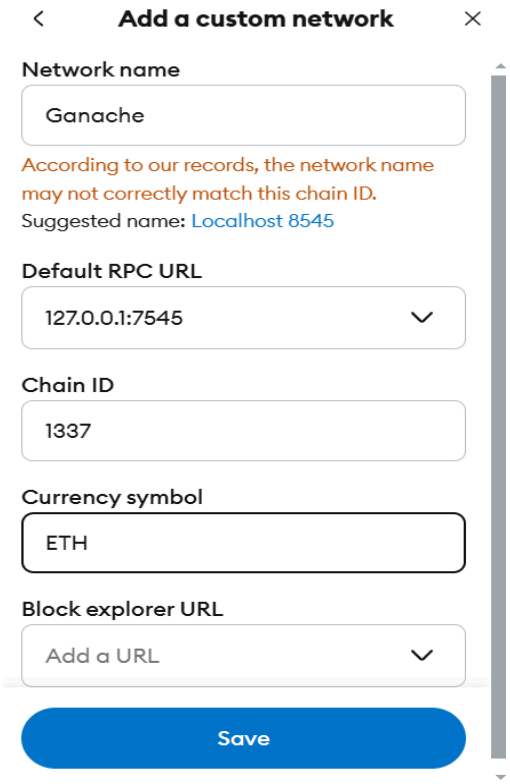
The MetaMask frame will be displayed like this:



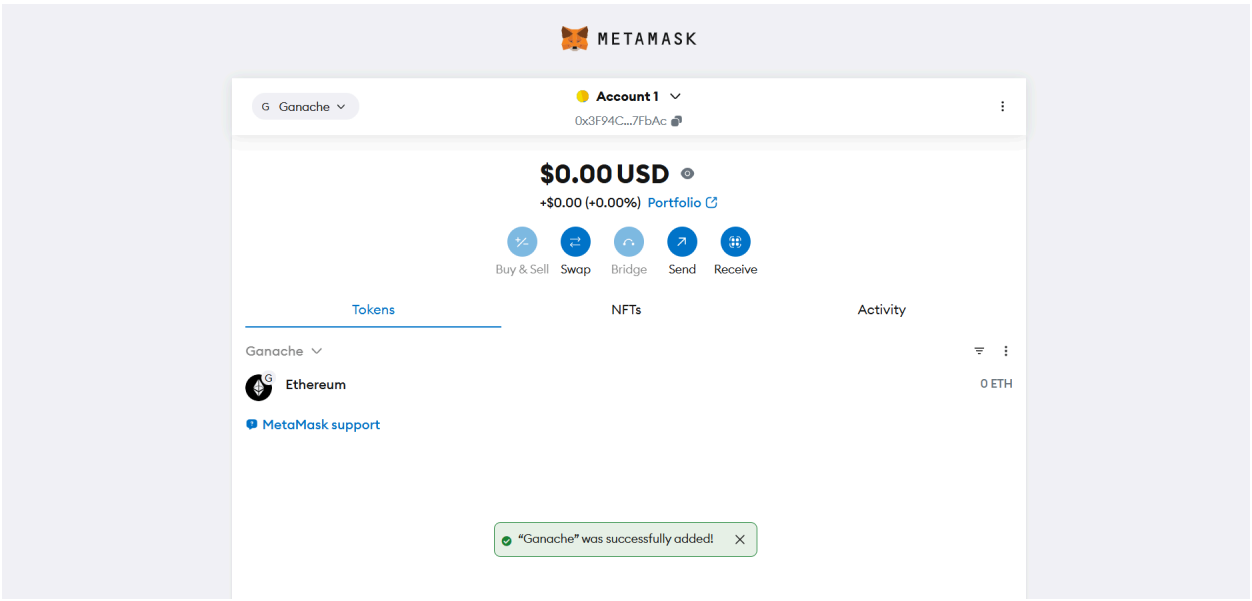
Add a custom network:



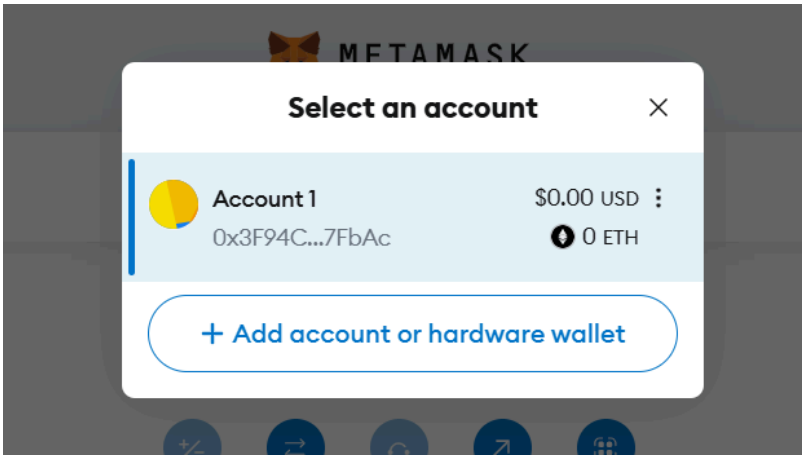
Add Ganache network and fill all the details,then click on save:



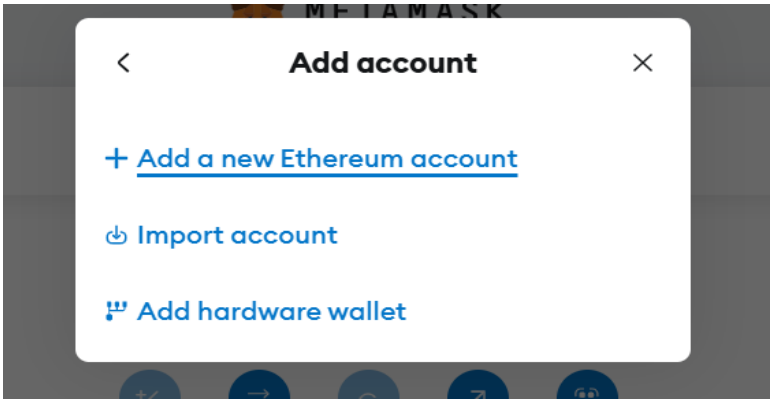
It open an Ganache platform:



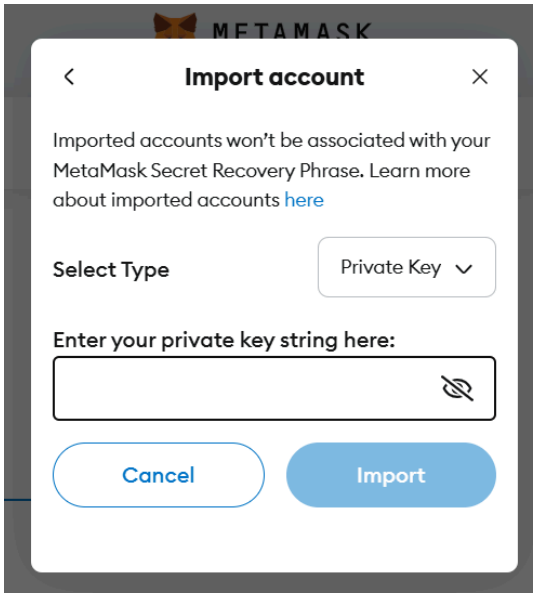
To connect with account in the Ganache select Account in Top of the page and click on Add account or hardware wallet.



Click on import account:

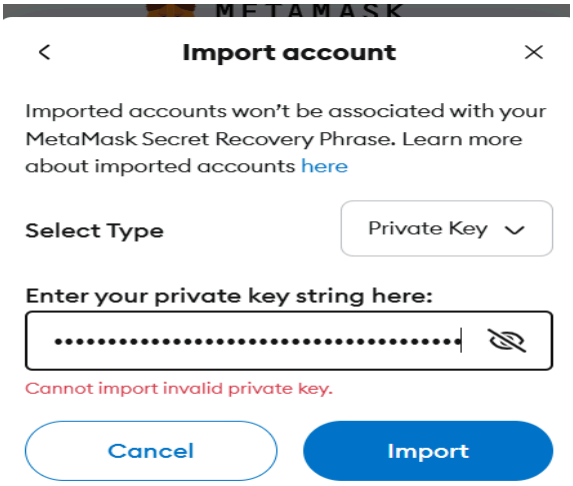


Add the account private key which is located in the Ganache:

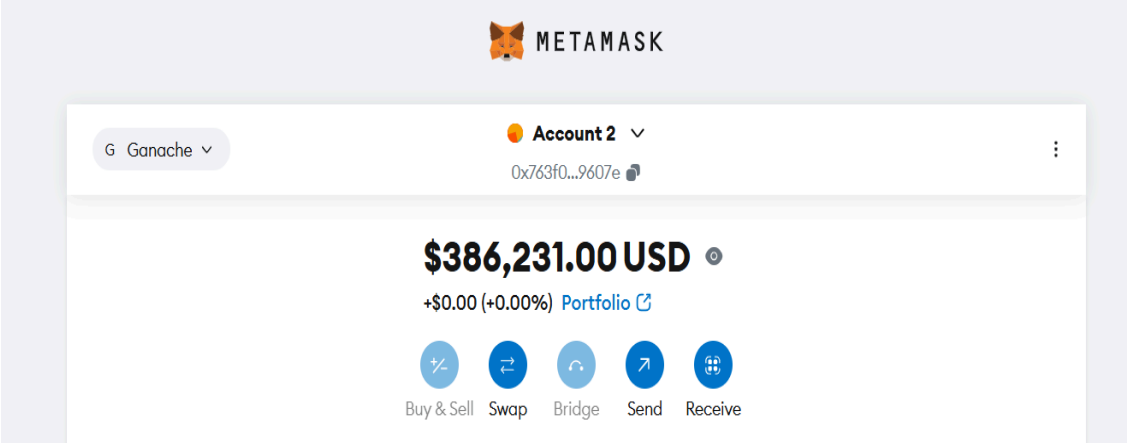


Copy the private key and paste in the entry filed:

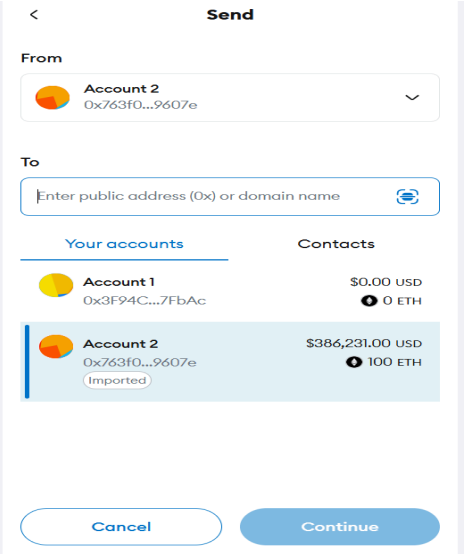
After entering the key click on import:



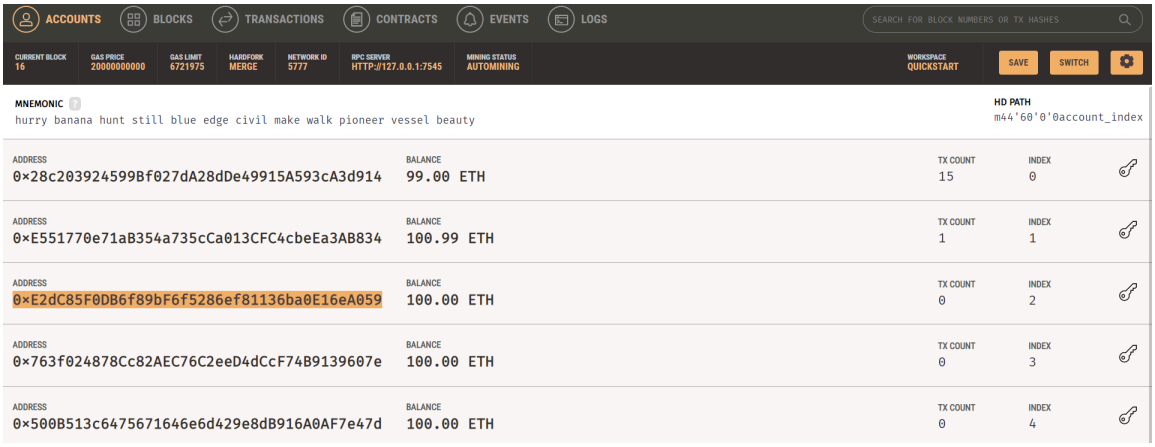
After the account has been added the money will be displayed:



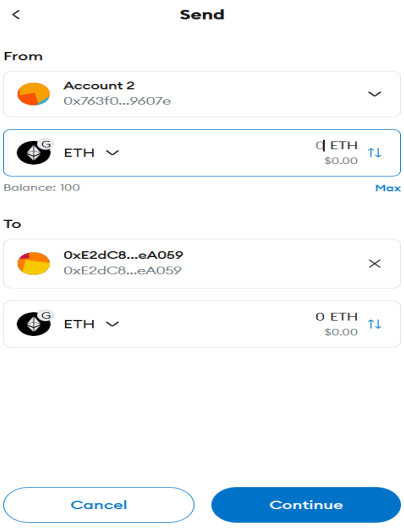
To send money to some one click on send and enter the receiver public key:



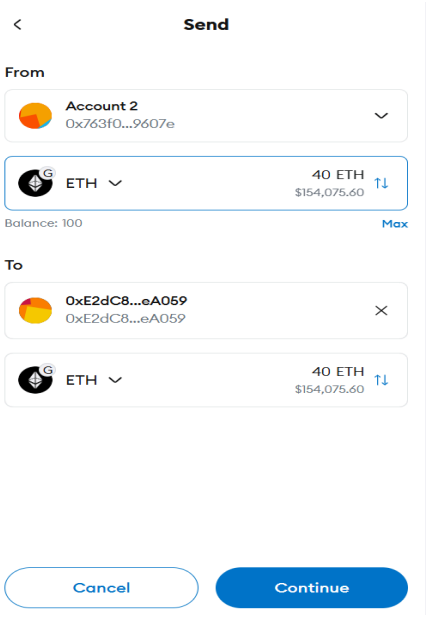
Copy the public key from the Ganache:



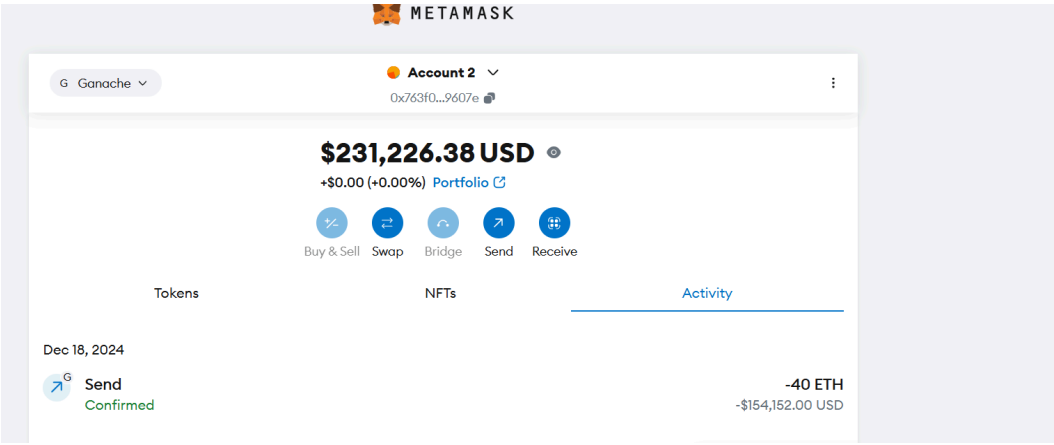
After entering public key the page will be in the way:



Now enter how much amount we need to send and click on continue and confirm:



The amount will be transferred and money will be credited to the receiver:



Now check the Ganache the money will be added: