**TASK : 1. Decentralized Greeting DApp  
in this there are certain tasks :  
Objective: Expand the basic greeting smart contract into a functional DApp.**

**Key Features:**

**1. Create a smart contract to store and update greetings.**

**2. Build a frontend where users can read and update the greeting.**

**3. Connect the frontend with Ethereum using web3.js.**

**Objectives:**

**1.Basics of smart contracts, web3.js integration, and DApp structure.**

**Tools:**

**Solidity, Truffle, Ganache, MetaMask, and HTML/JavaScript.**

**Name : G Srinivasa reddy**

**Section : CSE – A**

**ROOL: 21761A0518**

**Email :** [**g.svreddyy@gmail.com**](mailto:g.svreddyy@gmail.com)

**Steps :**

**1. Install Node.js and npm**

**2. Install Truffle**

**3. Install Ganache**

**4. Install MetaMask**

**5. Initialize the Project**

**6. Install web3.js**

**commands as follows :**

1. **Install Node.js and npm**
2. **npm install -g truffle**
3. **npm install -g ganache (or download the desktop version)**
4. **Install MetaMask as a browser extension.**
5. **npm install web3**
6. **Install a local server (pip install http-server or npm install -g http-server).**

**FROM HERE THE DETAIL EXECUTION:**

**1. Setup the Development Environment**

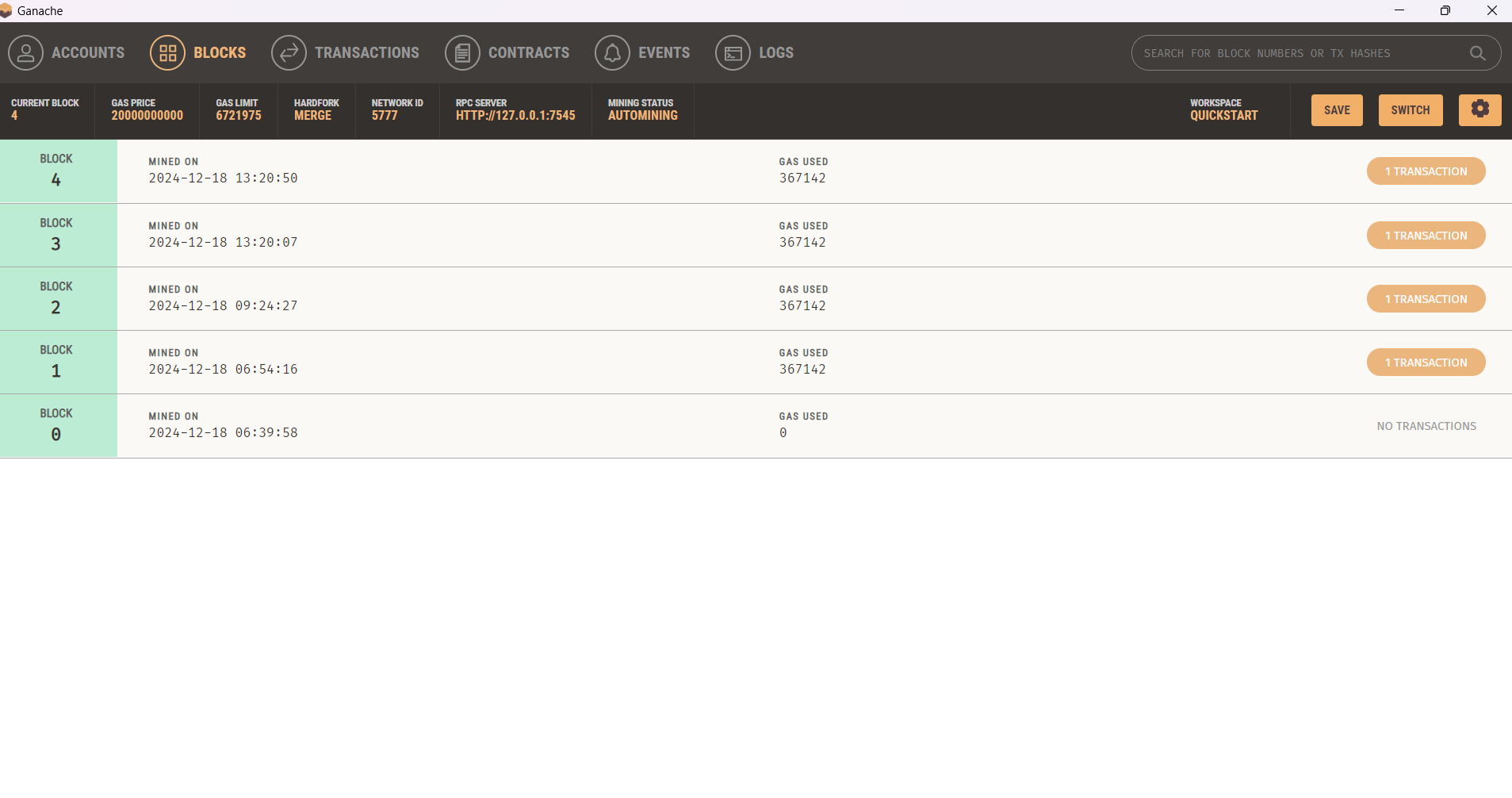
1. **Install Necessary Tools**:
   * Install **Node.js** and **npm** for managing dependencies.
   * Install **Truffle** (npm install -g truffle) to handle smart contract development and deployment.
   * Install **Ganache** to run a local Ethereum blockchain for testing.
   * Install a code editor like **VS Code**.
2. **Create a Truffle Project**:
3. mkdir GreetingDApp
4. cd GreetingDApp
5. truffle init
   * This initializes a basic Truffle project structure.
6. **Install Dependencies**:
   * Install **web3.js** for connecting the frontend with the Ethereum blockchain:
   * npm install web3

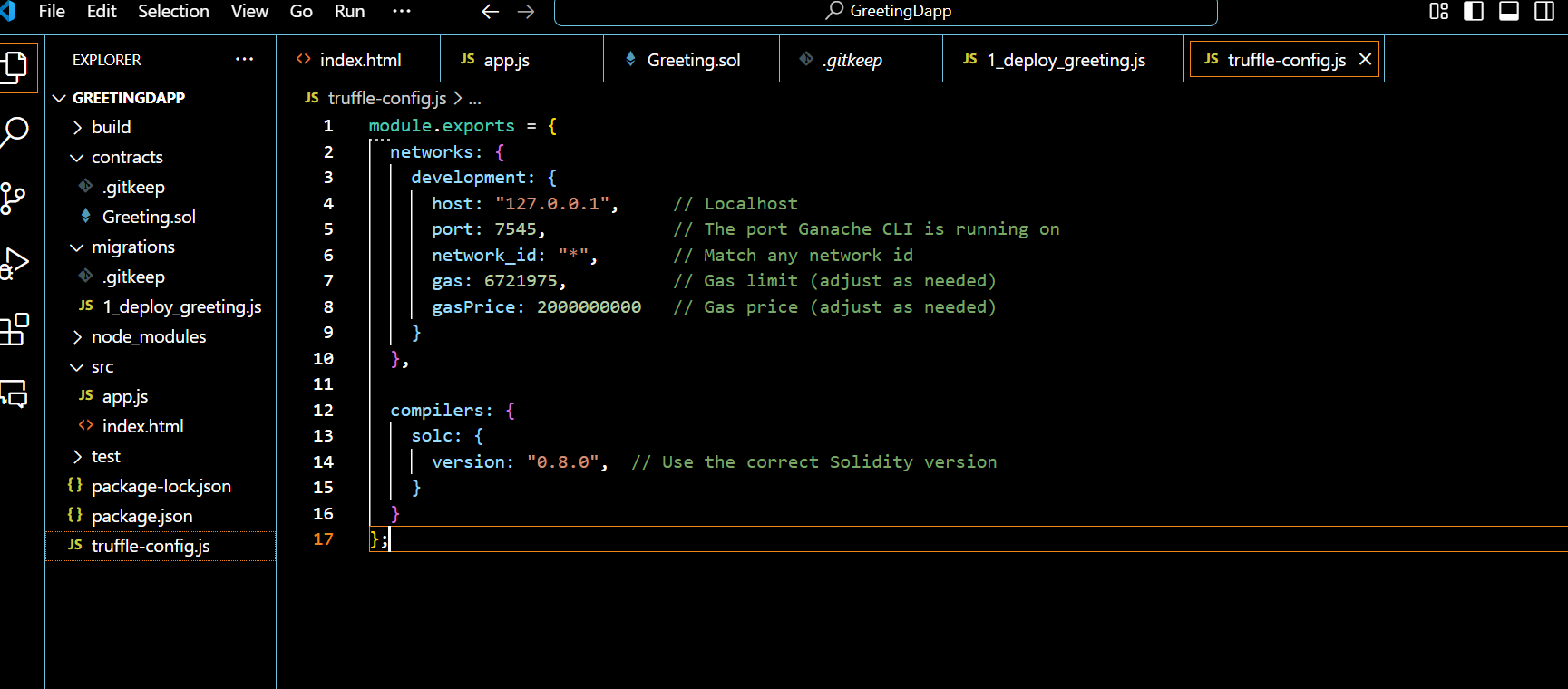


**2. Write the Smart Contract**

* **Navigate to the Contracts Folder**:
  + Go to contracts/ and create a new file, e.g., Greeting.sol.
* **Create the Greeting Smart Contract**:
* // SPDX-License-Identifier: MIT
* pragma solidity ^0.8.0;
* contract Greeting {
* string public greeting;
* constructor(string memory \_greeting) {
* greeting = \_greeting;
* }
* function setGreeting(string memory \_greeting) public {
* greeting = \_greeting;
* }
* function getGreeting() public view returns (string memory) {
* return greeting;
* }
* }
* **Compile the Contract**:
* truffle compile

**3. Deploy the Contract**

1. **Set up a Migration File**:
   * Create a new file 2\_deploy\_contracts.js in the migrations/ folder:
   * const Greeting = artifacts.require("Greeting");
   * module.exports = function (deployer) {
   * deployer.deploy(Greeting, "Hello, Blockchain!");
   * };
2. **Start Ganache**:
   * Launch Ganache and note the local blockchain network details (e.g., <http://127.0.0.1:7545>).
   * 
3. **Deploy the Contract**:
4. truffle migrate --network development



**4. Build the Frontend**

1. **Create a Frontend Folder**:
   * Inside the project, create a client/ folder for the frontend.
2. **Setup the Frontend Structure**:
   * Add the following folders/files:
   * client/
   * index.html
   * app.js
   * style.css
3. **Write the HTML** (index.html):

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Greeting DApp</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h1>Greeting DApp</h1>

<div>

<p>Current Greeting: <span id="currentGreeting"></span></p>

<input type="text" id="newGreeting" placeholder="Enter new greeting">

<button id="updateGreeting">Update Greeting</button>

</div>

<script src="https://cdn.jsdelivr.net/npm/web3/dist/web3.min.js"></script>

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

</body>

</html>

**Write the JavaScript** (app.js):

let web3;

let greetingContract;

const contractAddress = "YOUR\_DEPLOYED\_CONTRACT\_ADDRESS";

const abi = [

// Add your contract ABI here

];

async function init() {

// Connect to MetaMask

if (window.ethereum) {

web3 = new Web3(window.ethereum);

await window.ethereum.enable();

} else {

alert("MetaMask not detected!");

return;

}

// Connect to the contract

greetingContract = new web3.eth.Contract(abi, contractAddress);

// Fetch and display the current greeting

const currentGreeting = await greetingContract.methods.getGreeting().call();

document.getElementById("currentGreeting").textContent = currentGreeting;

// Set up the update greeting function

document.getElementById("updateGreeting").onclick = async () => {

const newGreeting = document.getElementById("newGreeting").value;

const accounts = await web3.eth.getAccounts();

await greetingContract.methods.setGreeting(newGreeting).send({ from: accounts[0] });

document.getElementById("currentGreeting").textContent = newGreeting;

};

}

window.onload = init;

1. **Style the Frontend** (style.css):

body {

1. font-family: Arial, sans-serif;

text-align: center;

padding: 20px;

}

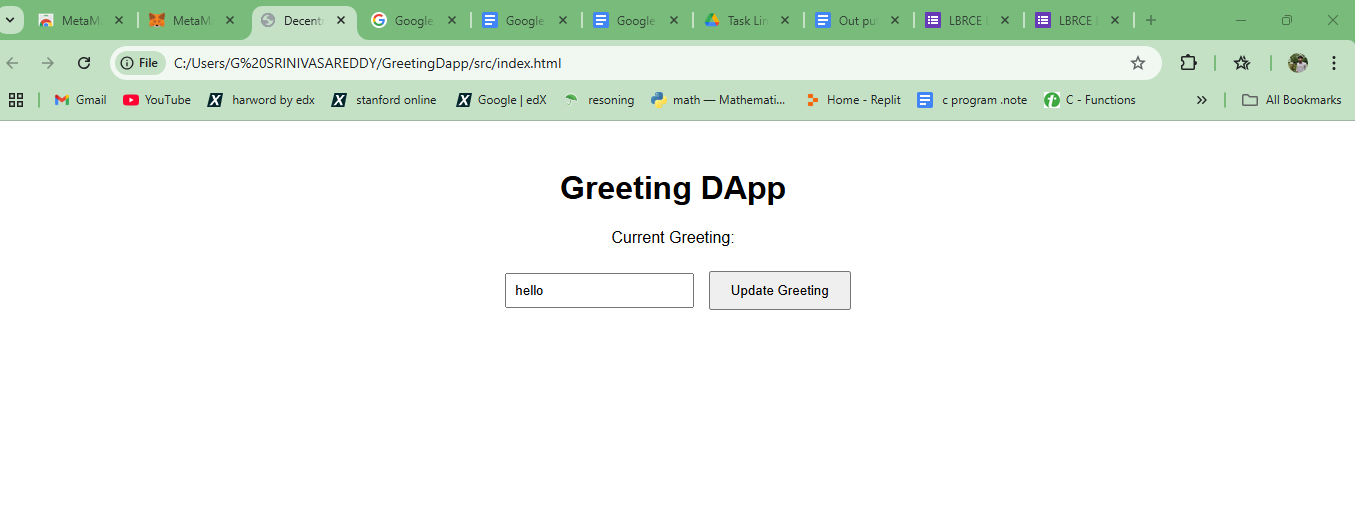
input, button {

margin: 5px;

padding: 10px;

font-size: 16px;

}



**5. Test the DApp**

1. **Run a Local Server**:
   * Use any local server (e.g., Python’s http.server or Node’s http-server):
   * cd client
   * python -m http.server 8000

OR

npx http-server

1. **Access the DApp**:
   * Open the browser and go to http://localhost:8000.
2. **Test Features**:
   * View the current greeting.
   * Update the greeting using MetaMask.

.

