Payment DApp for Microtransactions

I selected this project:

2. Payment DApp for Microtransactions

Objective: Develop a payment DApp that allows small payments between accounts.

Key Features:

Write a smart contract to send Ether to a recipient.

Allow users to input recipient address and amount through the DApp UI.

Display transaction hash and confirmation status.

Objectives:

Ether transfers using Solidity, transaction handling, and web3.js integration.

Tools:

Solidity, Truffle, Ganache, MetaMask, and React/HTML.

What I understand about this project is…..

1.Payment of ethereums can be done by giving the recipient address by the sender from a GUI which was created by using React js

2.Smart contracts between user and receiver can be written in solidity language.

3.For GUI we have to use react.js.

Step by step explanation of what I have done:

1.Fist I have created a folder named paymentdapp on my desktop.

2.Then I opened it in the command prompt and type “truffle init” so that i can connect it with ganache.

3.Then I opened visual studio code and created a file called PaymentDapp.sol in the contracts folder present in paymentdapp folder which has my smart contract code.

4.Then I opened the migrations folder in paymentdapp and created a file named 2_deploy_payment_dapp.js which has the deployer code in it.

5.Next I have changed the code in truffle-config.js file which is already created in the folder paymentdapp.

6.I have updated the code by only leaving the host address of ganache and port of ganache and also used the solidity version.

7.For creating GUI using react I opened the command prompt and went to my folder by using the command “cd paymentdapp”.

8.Then I created a react project by using this command:

```
npx create-react-app payment-dapp
```

9.Then I have installed web3 by

```
npm install web3
```

10.After completing this process I have entered into the app.js file and updated my code there.

11.After compiling smart contract by :

```
truffle compile
truffle migrate
truffle migrate –network development
```

12.After compiling the smart contract it generates paymentdapp.json file.

13.Add a folder named contract in the src folder of payment-dapp.
14.copy the paymentdapp.json file into the contract folder.
15.Import this into the app.js by using :

```
import PaymentDapp from "./contracts/PaymentDapp.json";
```

16.compile the whole project by going to the command prompt and type the command:
npm start
17.Then it will be opened in the browser and connected to the metamask account which is extended already in the browser.
18.Then send the transaction from the metamask account to the ganache account.

Lets see the codes implementation:

Paymentdapp.sol

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

contract PaymentDapp {
    address public owner;

    constructor() {
        owner = msg.sender;
    }

    function sendPayment(address payable recipient) public payable {
        require(msg.value > 0, "Must send some Ether");
        recipient.transfer(msg.value);
    }
}
```

2_deploy_payment_dapp.js:

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

module.exports = async function (deployer) {
  console.log("Deploying the PaymentDapp smart contract...");
  await deployer.deploy(PaymentDapp);
  console.log("PaymentDapp smart contract deployed successfully!");
};
```

truffle-config.js:

```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",  // Ganache CLI or GUI
      port: 7545,         // Default Ganache port
      network_id: "*",    // Match any network id
    },
  },

  compilers: {
    solc: {
      version: "0.8.0", // Use the correct Solidity version
      settings: {
        optimizer: {
          enabled: true,
          runs: 200,
        },
      },
    },
  },
};
```

App.js:

```
import React, { useState, useEffect } from "react";
import Web3 from "web3";
import PaymentDapp from "./contracts/PaymentDapp.json"; // Import ABI

function App() {
  const [account, setAccount] = useState(""); // User's MetaMask account
  const [contract, setContract] = useState(null); // Smart contract
instance
  const [recipient, setRecipient] = useState(""); // Recipient address
  const [amount, setAmount] = useState(""); // Amount in Ether
  const [message, setMessage] = useState(""); // Optional message
  const [txHash, setTxHash] = useState(""); // Transaction hash
```

```
// Load web3, accounts, and the smart contract
useEffect(() => {
  const loadBlockchainData = async () => {
    if (window.ethereum) {
      const web3 = new Web3(window.ethereum);
      await window.ethereum.request({ method: "eth_requestAccounts" });

      const accounts = await web3.eth.getAccounts();
      setAccount(accounts[0]);

      const networkId = await web3.eth.net.getId();
      const deployedNetwork = PaymentDapp.networks[networkId];

      const instance = new web3.eth.Contract(
        PaymentDapp.abi,
        deployedNetwork && deployedNetwork.address
      );

      setContract(instance);
    } else {
      alert("Please install MetaMask to use this DApp!");
    }
  };

  loadBlockchainData();
}, []);

const sendPayment = async () => {
  if (contract && recipient && amount) {
    try {
      const weiAmount = Web3.utils.toWei(amount, "ether");
      const tx = await contract.methods
        .sendPayment(recipient, message)
        .send({ from: account, value: weiAmount });

      setTxHash(tx.transactionHash);
      alert("Transaction Successful!");
    } catch (error) {
      console.error("Error:", error);
```

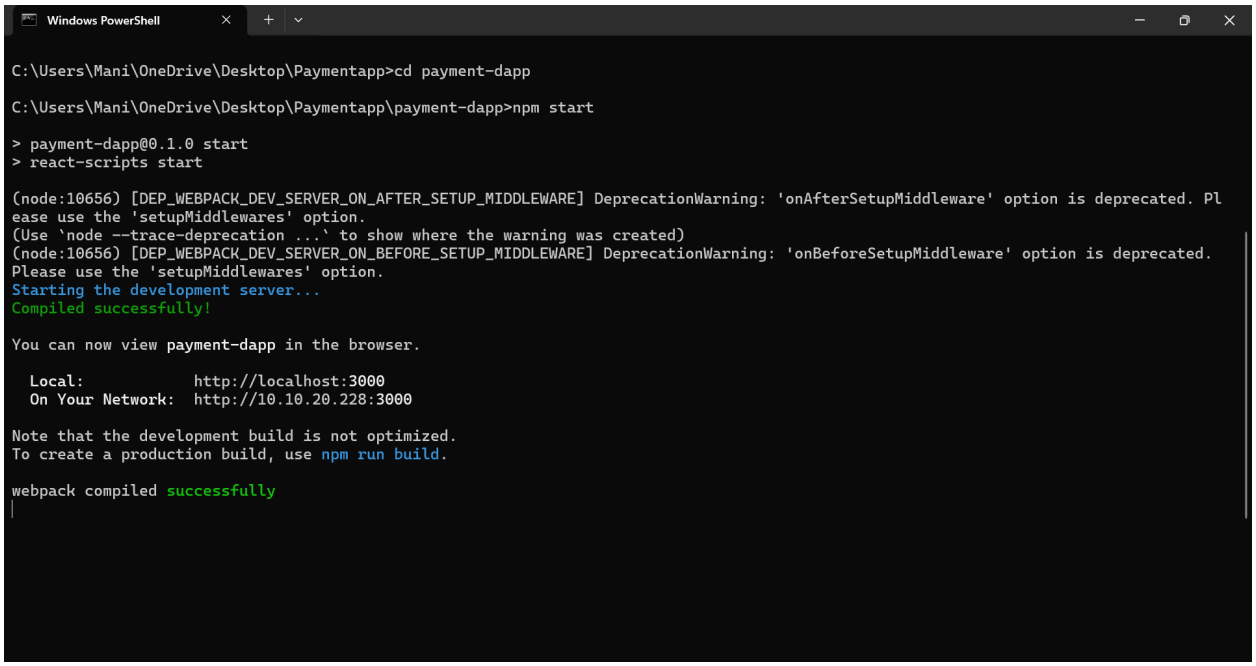
```
    }
  } else {
    alert("Please fill all fields correctly!");
  }
};

return (
  <div style={{ padding: "20px" }}>
    <h1>Payment DApp for Microtransactions</h1>
    <p>Connected Account: {account}</p>
    <div>
      <label>Recipient Address:</label>
      <input
        type="text"
        placeholder="0xRecipientAddress"
        value={recipient}
        onChange={(e) => setRecipient(e.target.value)}
      />
    </div>
    <div>
      <label>Amount (ETH):</label>
      <input
        type="number"
        placeholder="Amount in ETH"
        value={amount}
        onChange={(e) => setAmount(e.target.value)}
      />
    </div>
    <div>
      <label>Message:</label>
      <input
        type="text"
        placeholder="Optional message"
        value={message}
        onChange={(e) => setMessage(e.target.value)}
      />
    </div>
    <button onClick={sendPayment} style={{ marginTop: "10px" }}>
      Send Payment
    </button>
```

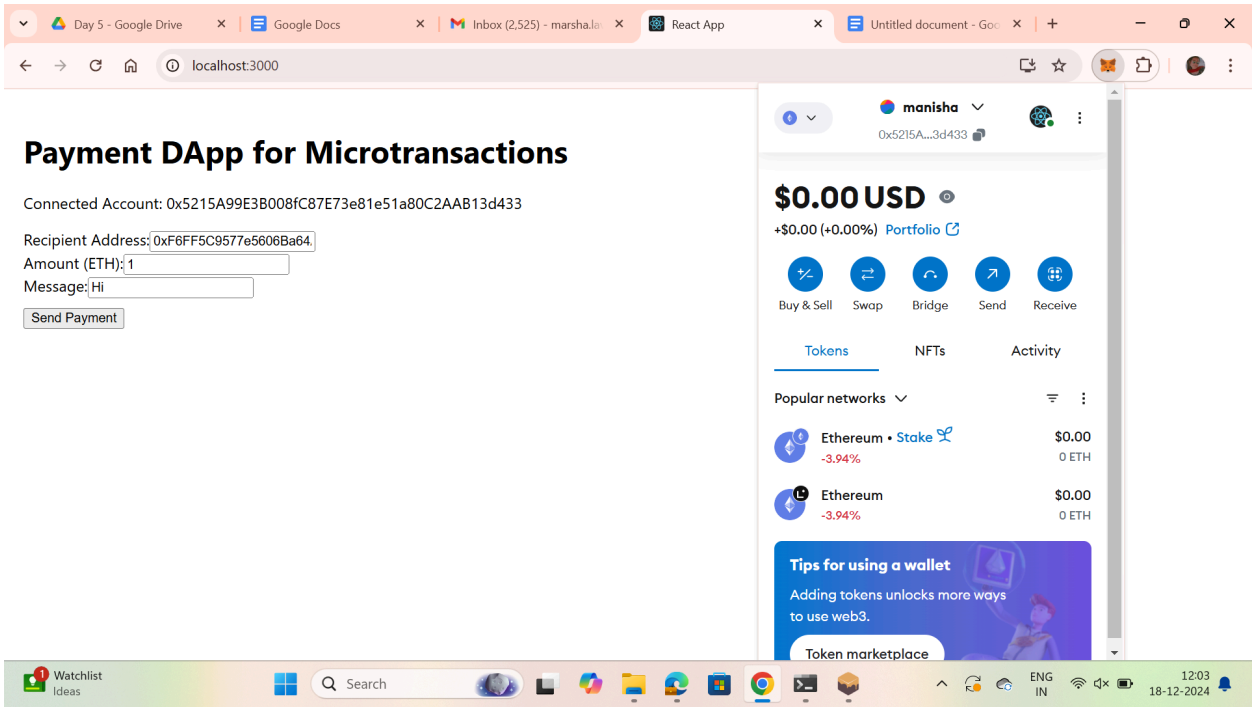
```
      {txHash && (
        <p>
          Transaction Hash:{" "}
          <a
            href={`https://etherscan.io/tx/${txHash}`}
            target="_blank"
            rel="noopener noreferrer"
          >
            {txHash}
          </a>
        </p>
      )}
    </div>
  );
}

export default App;
```

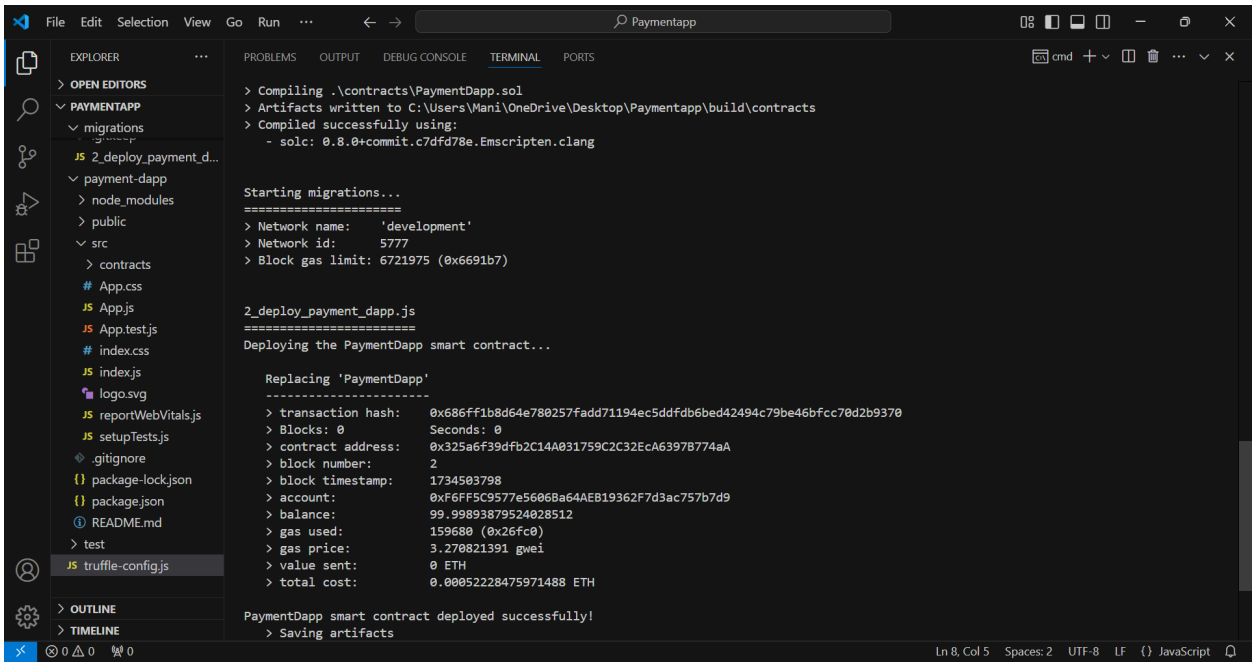
Command prompt commands:

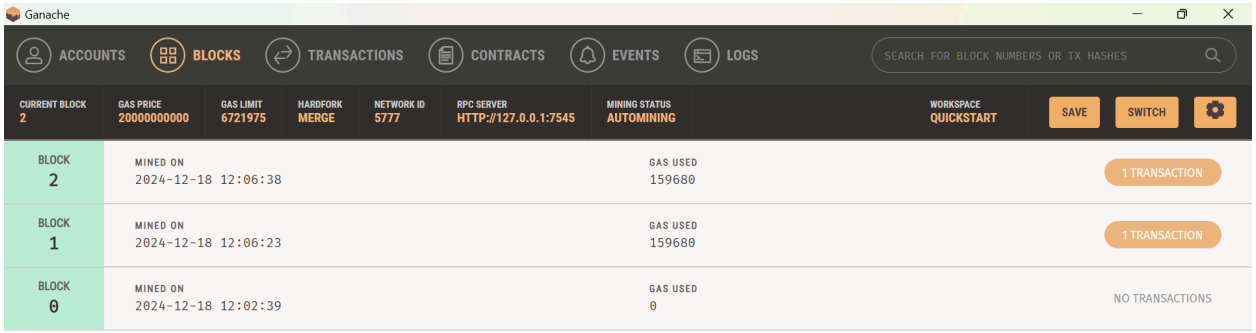
1. node -v
2. npm -v
3. npm install web3
4. npm install web-vitals
5. npx create-react-app payment-dapp
6. npm start

Screenshots:









Laveti Manisha
21761A05A3
CSE-B,8th sem