Name : Pratik Tiruwa
Seat No : 31031523035

B. Create a DApp to implement transactions between two accounts.

1. index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>DApp-3</title>
    </head>
    <body>
        <h1>Blockchain Transactions DApp</h1>
        <h2>Send Ether:</h2>
        <input type="text" id="toAddr" placeholder="To Address" />
        <input type="number" id="amount" placeholder="Amount" />
        <button onclick="send()">Send</button>
        <h2>Check Balance:</h2>
        <button onclick="checkBalance()">Check Balance</button>
        <p>Your Balance is: <span id="bal"></span></p>
        <script
src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
        <script src="app.js"></script>
    </body>
</html>
```

2. app.js

```
const contractAddress = ""; // Replace with your deployed contract address
const contractABI = []; // Use ABI from compiled contract

let web3;
let contract;

window.addEventListener("load", async () => {
  if (window.ethereum) {
    web3 = new Web3(window.ethereum);
```

```
    await window.ethereum.enable();
  } else {
    console.log("MetaMask not detected. Please install MetaMask.");
  }

  contract = new web3.eth.Contract(contractABI, contractAddress);
});


async function send() {
  const accounts = await web3.eth.getAccounts();
  const amount = web3.utils.toWei(document.getElementById('amount').value, 'ether');
  const toAddress = document.getElementById('toAddr').value;
  const sender = accounts[0];

  console.log("Sender: ", accounts[0]);
  console.log("Receiver: ", toAddress);
  console.log("Amount: ", amount);

  if (amount <= 0) {
    alert("Amount must be greater than 0");
    return;
  }
  else if (toAddress == "") {
    alert("Please enter receiver address");
    return;
  }
  else {
    contract.methods.transfers(toAddress).send({
      from: sender,
      value: amount
    }).on('transactionHash', (hash) => {
      console.log('Transaction Hash:', hash);
    }).on('receipt', (receipt) => {
      console.log('Transaction Receipt:', receipt);
    }).on('error', (error) => {
```

```
      console.error('Error:', error);
    });
  }
};

async function checkBalance() {
  const accounts = await web3.eth.getAccounts();
  const balance = await web3.eth.getBalance(accounts[0]);
  const balanceInEther = web3.utils.fromWei(balance, 'ether');
  document.getElementById("bal").innerText = `${balanceInEther}`;
}
```

3. transactions.sol

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.19;

contract transactions {
    event Transfer(address indexed from, address indexed to, uint256 value);

    function transfers(address payable _to) public payable {
        require(msg.value > 0, "Send some ether");
        _to.transfer(msg.value);
        emit Transfer(msg.sender, _to, msg.value);
    }

    receive() external payable {
        emit Transfer(msg.sender, address(this), msg.value);
    }
}
```

1. 1_deploy.js

Name : Pratik Tiruwa
Seat No : 31031523035

```
const transaction = artifacts.require("transactions");

module.exports = async function (deployer) {
```

```
  await deployer.deploy(transaction);
  const instance = await transaction.deployed();
  console.log("Contract deployed at:", instance.address);
};
```

1. bs-config.json

```
{
    "server":{
        "baseDir": ["./frontend"]
    }
}
```

2. Output: