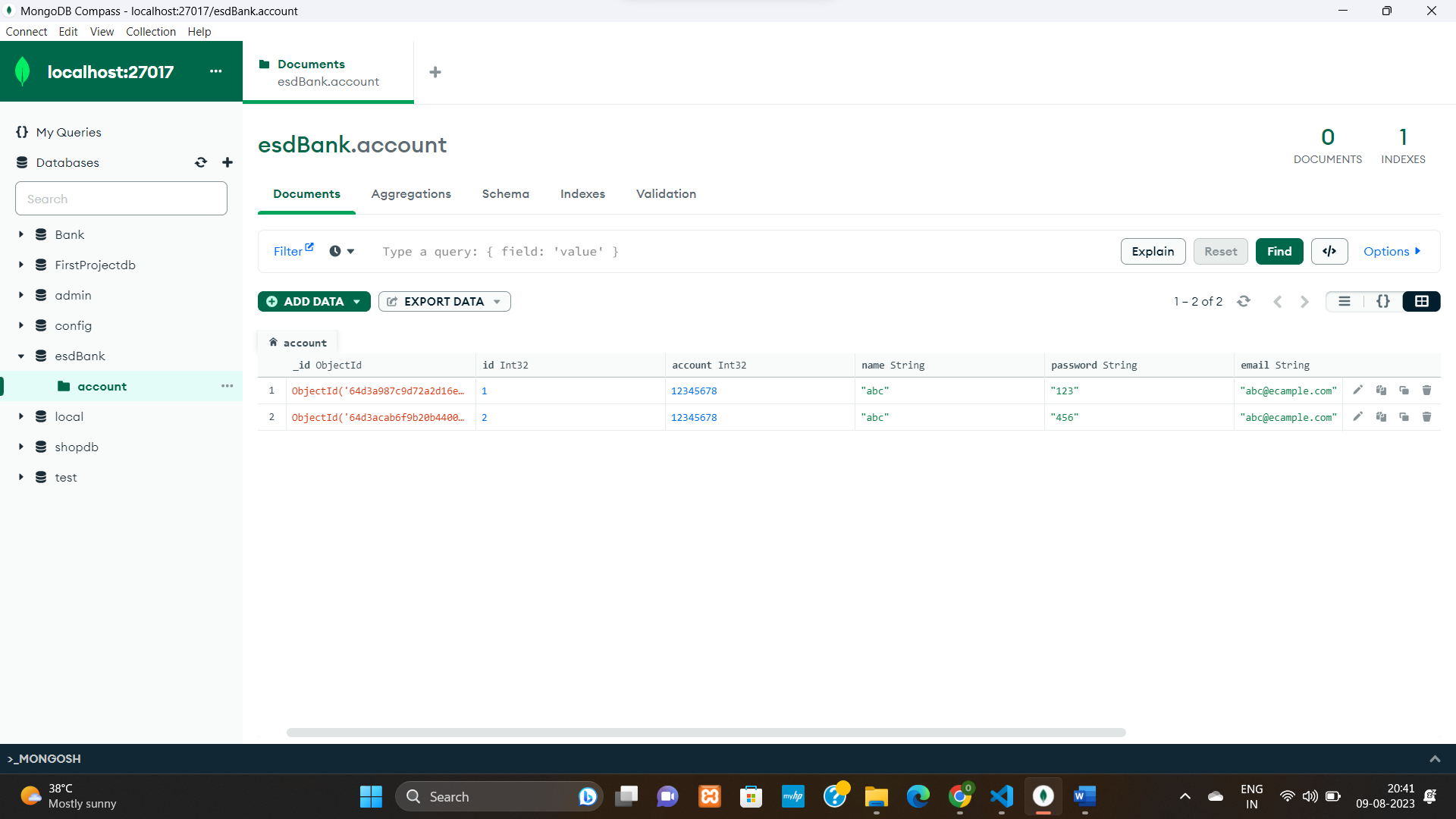
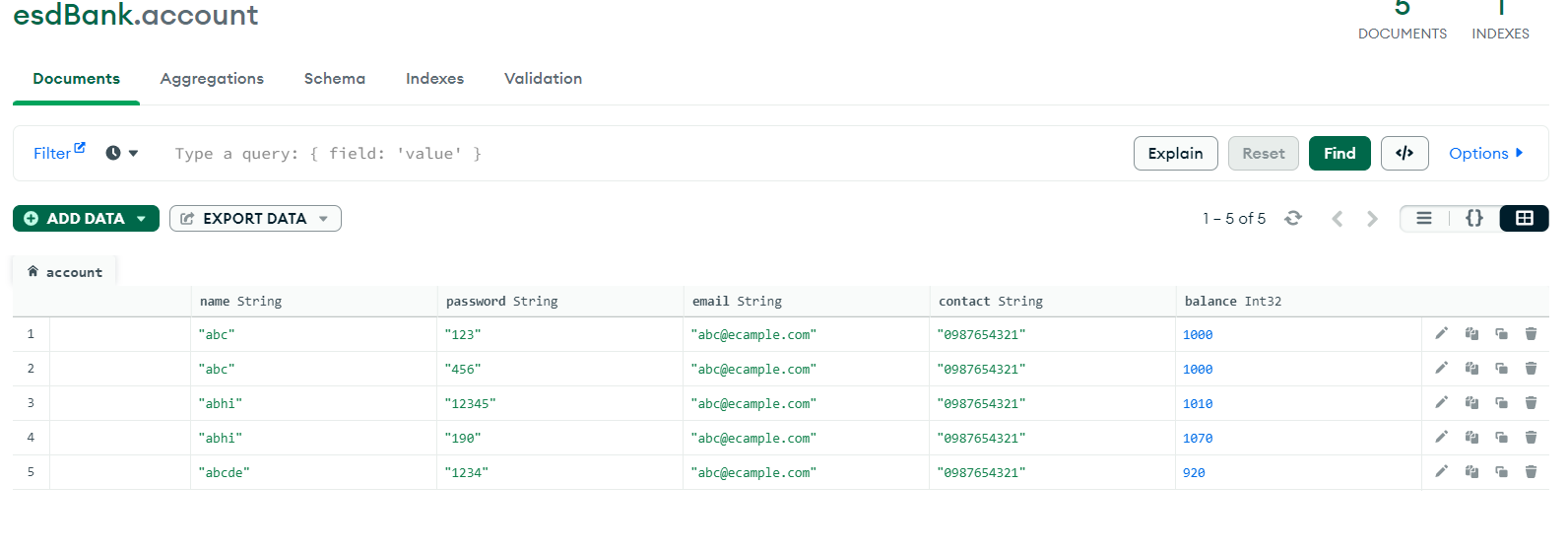
Roll No.: 47,48,49,50,51

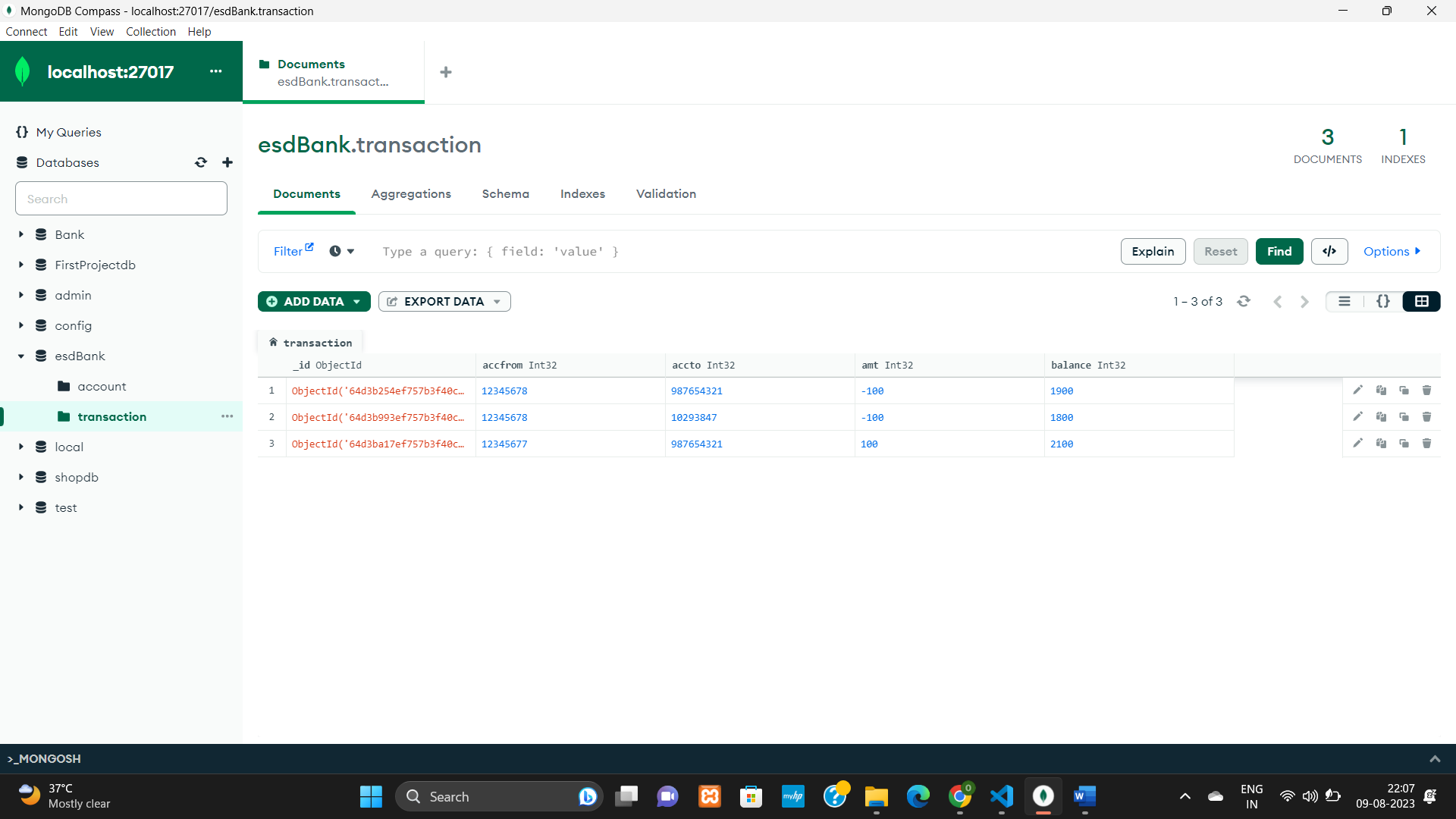
Changes done in database of account collection:



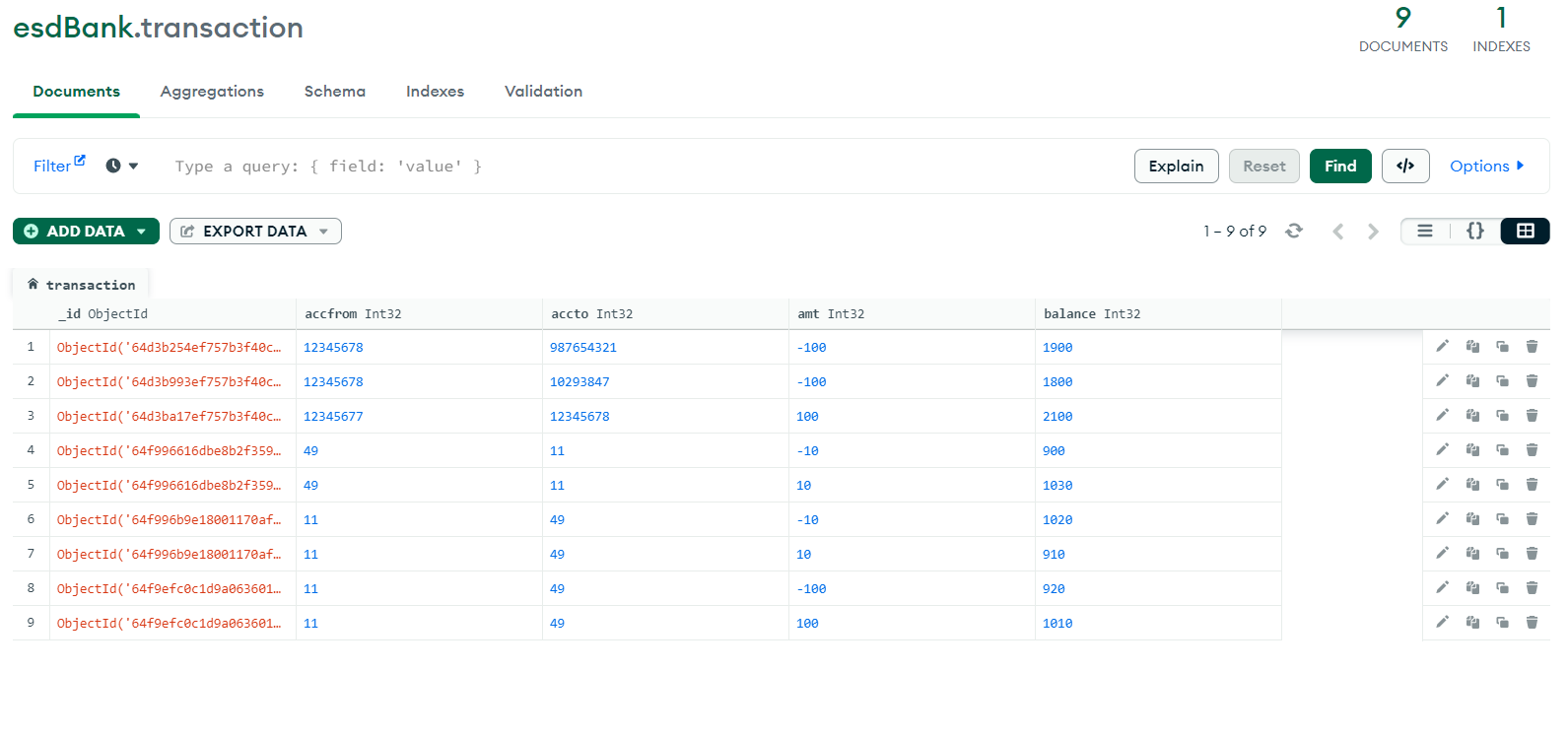




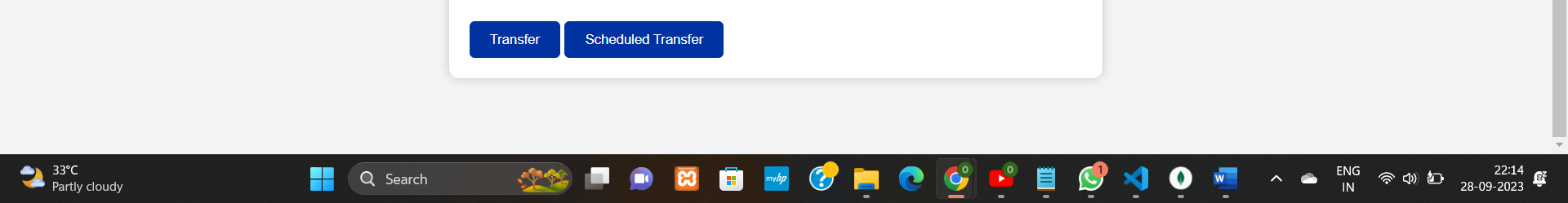
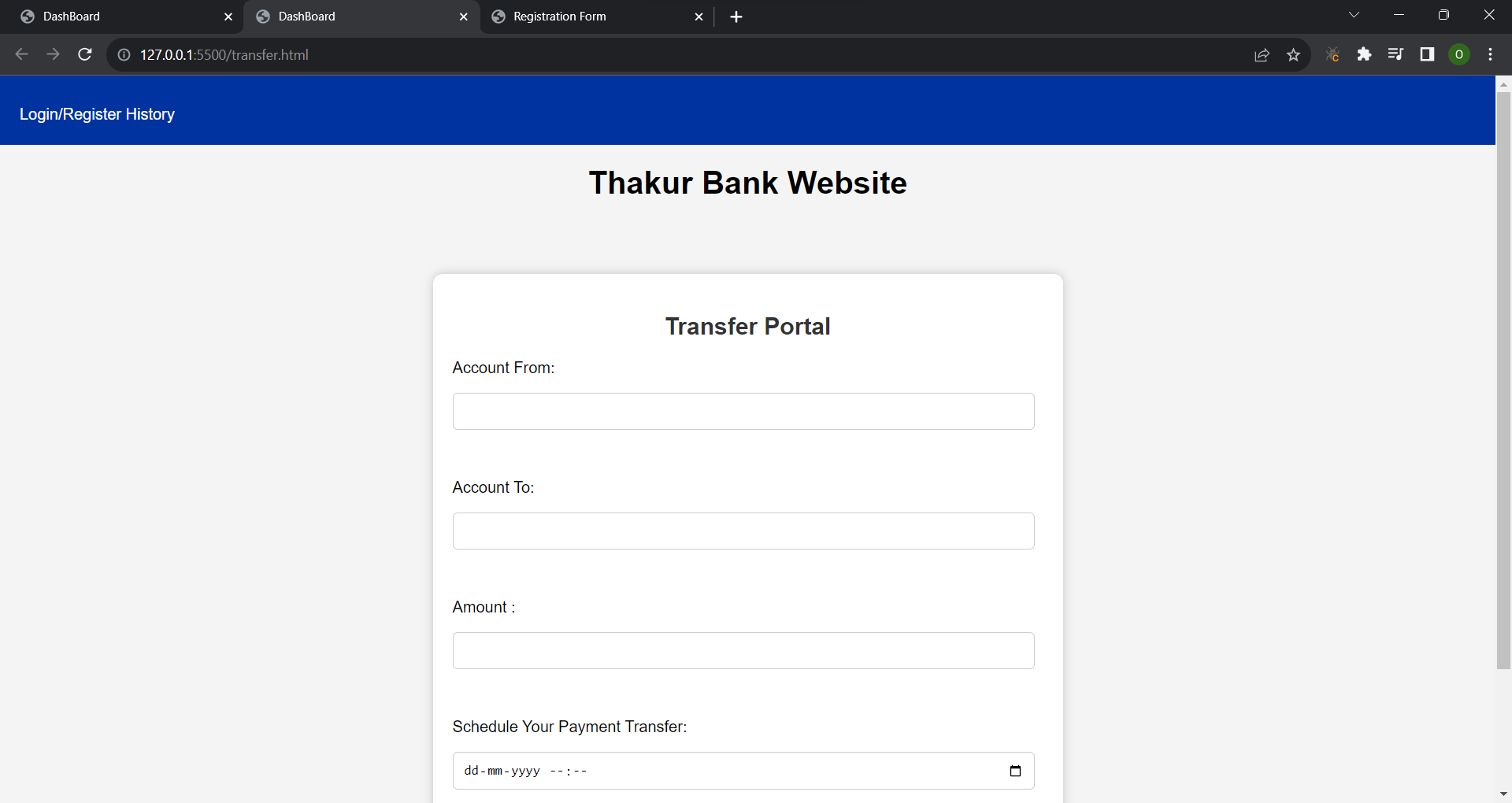
Before transaction collection of transaction:



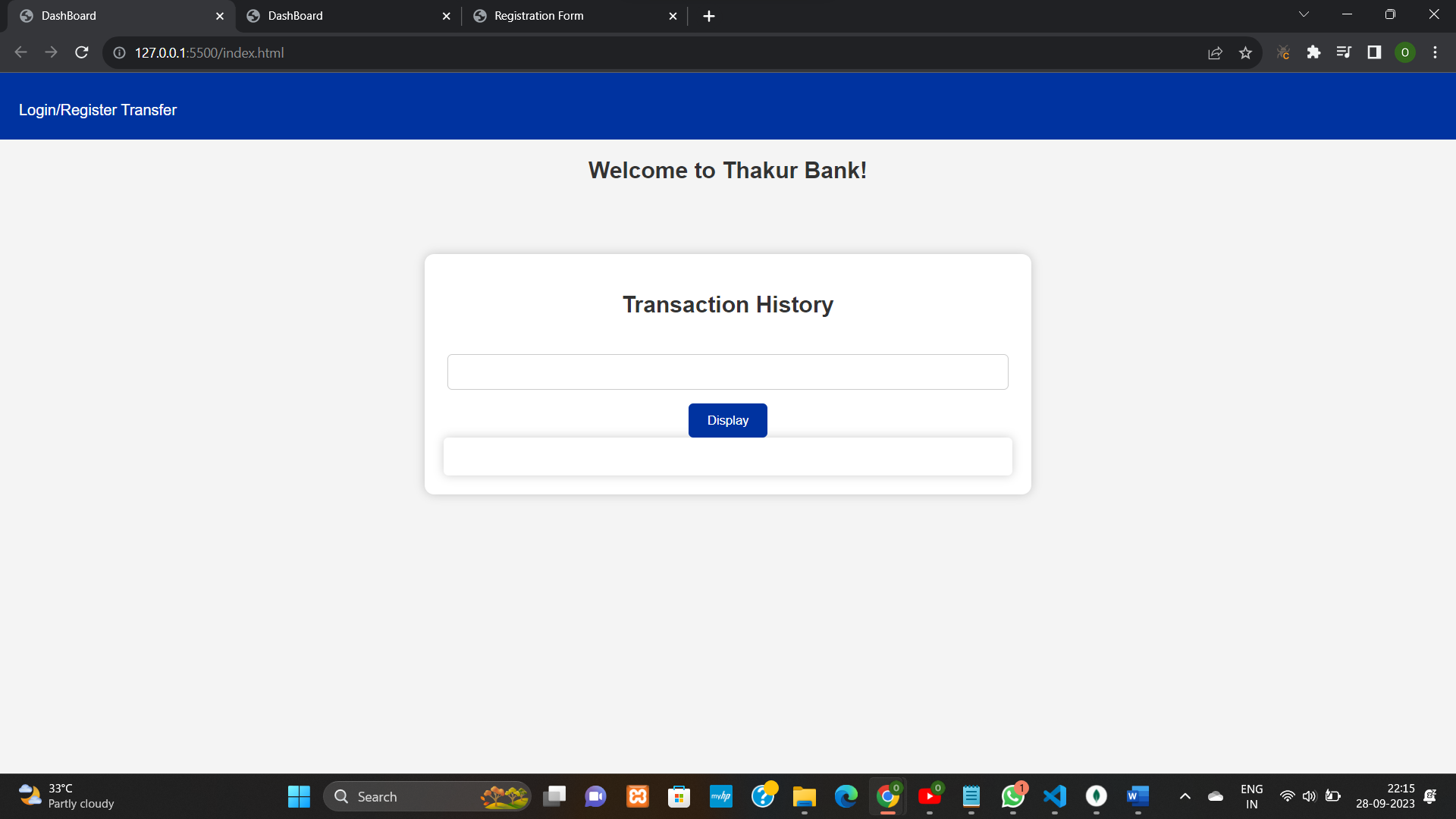
After transaction collection of transaction:



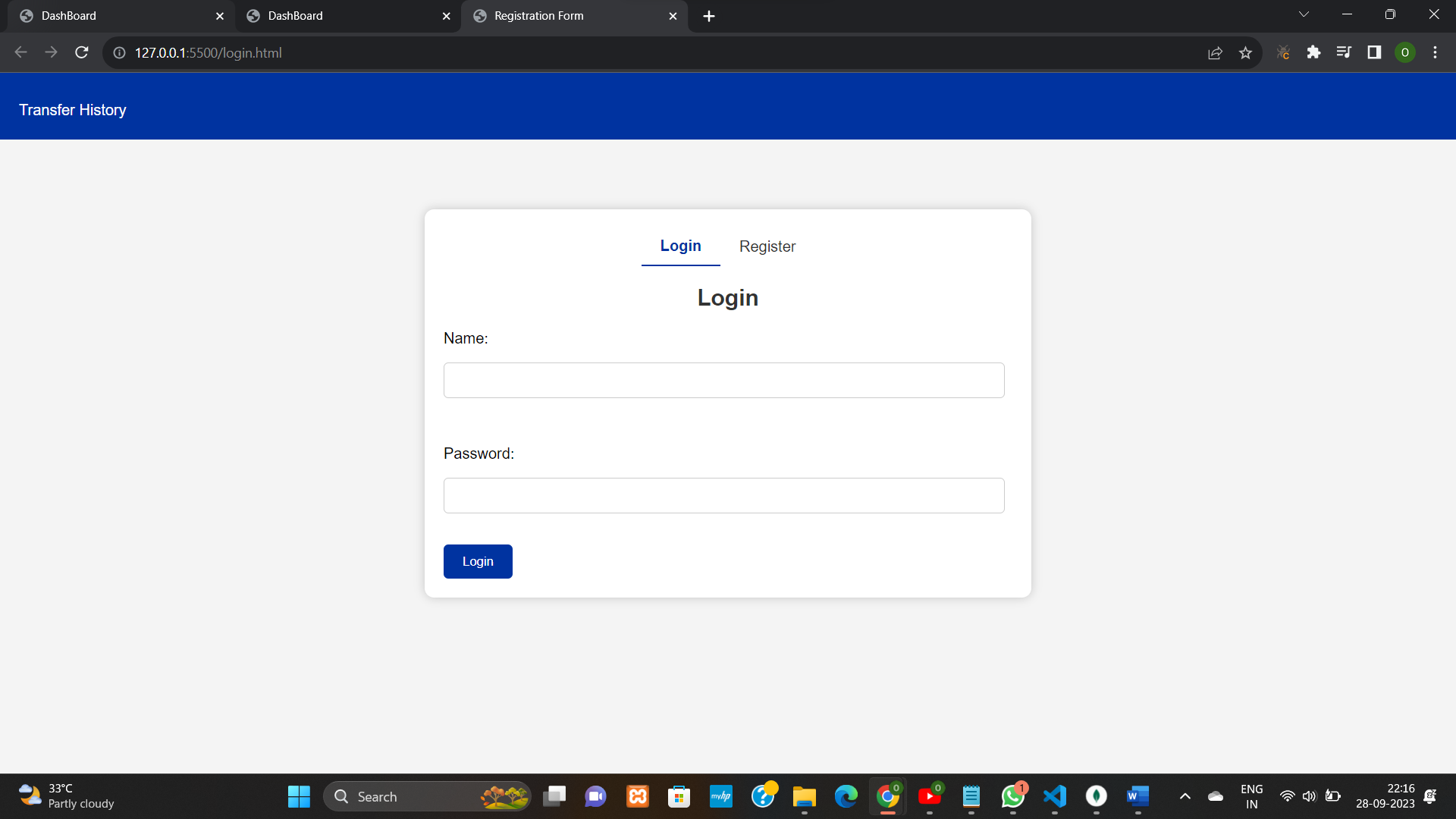
Transfer.html page:

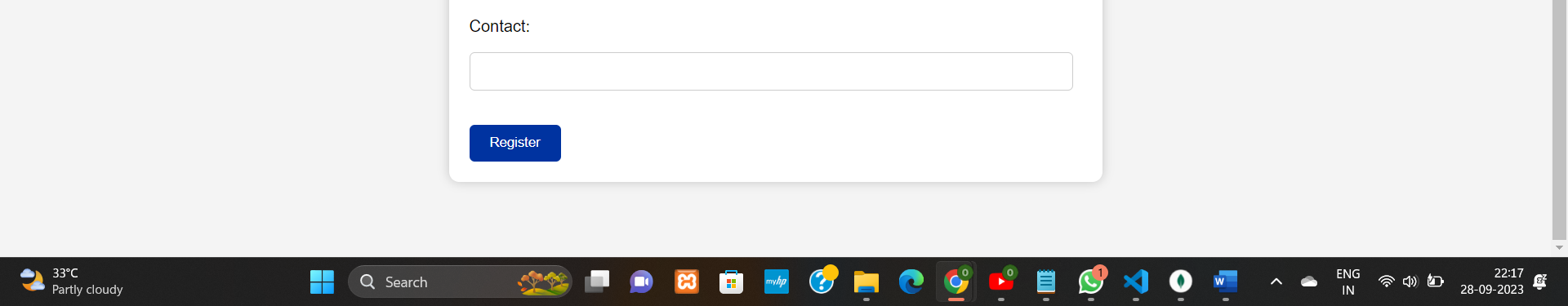
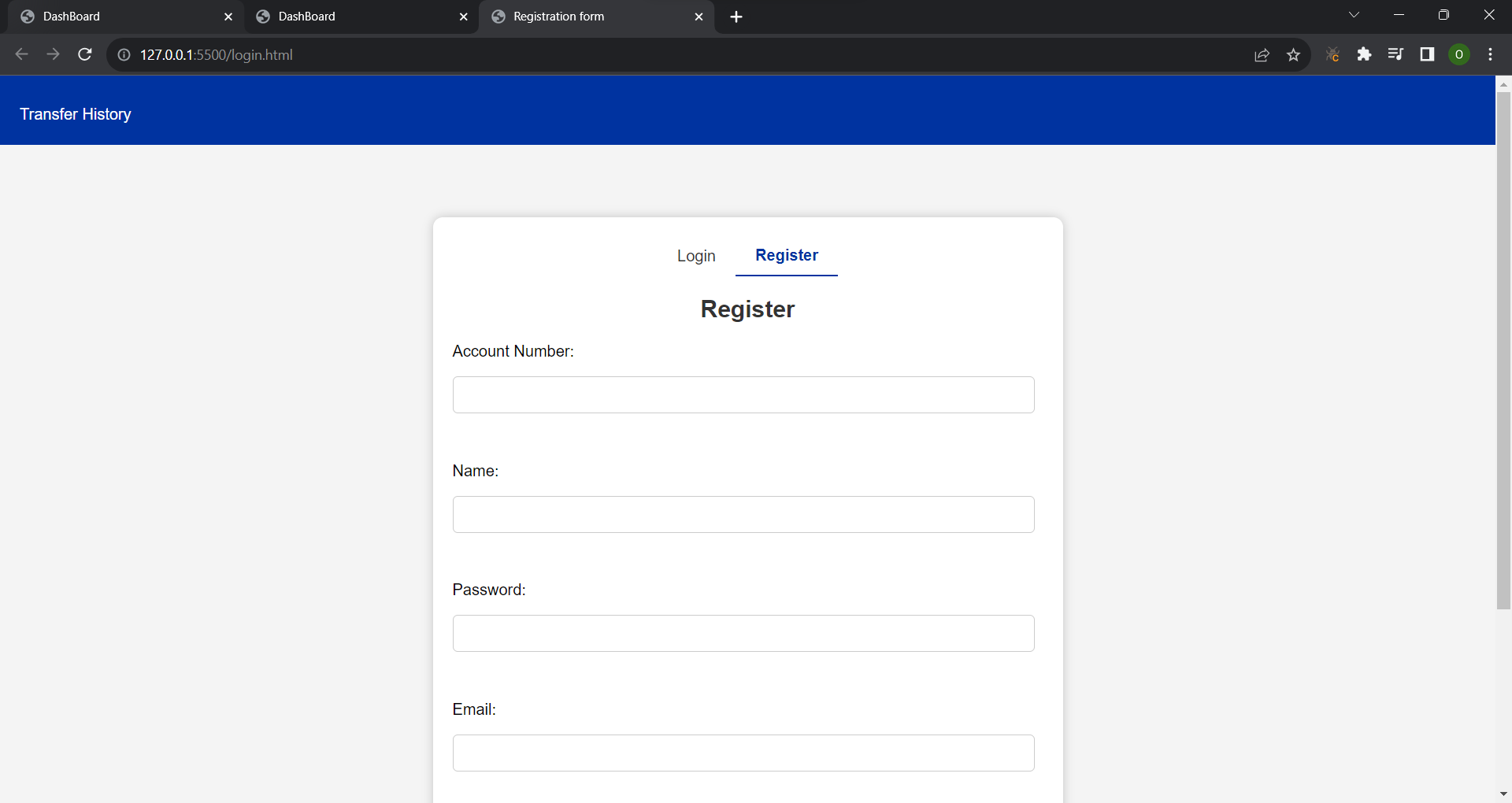


index.html:



Login.html:





Code:

Server.js:

const express = require("express");

const schedule = require('node-schedule');

const app = express();

const port = 3000;

const { MongoClient } = require("mongodb");

const url = "mongodb://127.0.0.1:27017";

const dbName = "esdBank";

app.use(express.json());

app.use(express.urlencoded());

async function createDocument(req, res) {

    const client = new MongoClient(url);

    try {

        await client.connect();

        console.log("Connected to the database");

        const db = client.db(dbName);

        const collection = db.collection('account');

        //Create user logic here

        const last\_id = await collection.countDocuments();

        const { account, name, password, email, contact } = req.body;

        const newUser = {

            id: (last\_id + 1),

            account: parseInt(account),

            name: name,

            password: password,

            email: email,

            contact: contact,

            balance: 1000

        }

        const result = await collection.insertOne(newUser);

        console.log("Created document:", result.insertedId);

        res.send(result.insertedId);

    } catch (err) {

        console.error("Error : ", err);

    } finally {

        await client.close();

        console.log("Disconnected from the database");

    }

}

async function readDocument(req, res) {

    const client = new MongoClient(url);

    try {

        await client.connect();

        console.log("Connected to the database");

        const db = client.db(dbName);

        const collection = db.collection("account");

        // Insert the document into the collection

        const { name, password } = req.body;

        filter = {

            name: name,

            password: password

        }

        const result = await collection.findOne(filter);

        console.log(`${name} and ${password}`)

        res.json(result);

    } catch (err) {

        console.error("Error:", err);

    } finally {

        await client.close();

        console.log("Disconnected from the database");

    }

}

async function updateAmount(senderAccount, receiverAccount, amount) {

    const client = new MongoClient(url);

    try {

        await client.connect();

        console.log("Connected to the database");

        const db = client.db(dbName);

        const collection = db.collection("account");

        // Check if sender and receiver accounts exist

        const senderAccountNumber = parseInt(senderAccount);

        const receiverAccountNumber = parseInt(receiverAccount);

        const senderFilter = { account: senderAccountNumber };

        const receiverFilter = { account: receiverAccountNumber };

        const senderDoc = await collection.findOne(senderFilter);

        const receiverDoc = await collection.findOne(receiverFilter);

        if (!senderDoc || !receiverDoc) {

            throw new Error("Sender or receiver account not found");

        }

        // Check if sender has enough balance

        if (parseInt(senderDoc.balance) >= 900) {

            // Calculate updated balances

            const senderBalanceAfterTransfer = senderDoc.balance - amount;

            const receiverBalanceAfterTransfer = parseInt(receiverDoc.balance) + parseInt(amount);

            // Update sender's balance

            const senderUpdate = {

                $set: { balance: senderBalanceAfterTransfer }

            };

            await collection.updateOne(senderFilter, senderUpdate);

            // Update receiver's balance

            const receiverUpdate = {

                $set: { balance: receiverBalanceAfterTransfer }

            };

            await collection.updateOne(receiverFilter, receiverUpdate);

            // Insert transaction records into the 'transaction' collection

            const transactionCollection = db.collection("transaction");

            // Create transaction records for sender and receiver

            const senderTransaction = {

                accfrom: senderAccountNumber,

                accto: receiverAccountNumber,

                amt: -amount, // Negative amount for sender

                balance: senderBalanceAfterTransfer

            };

            const receiverTransaction = {

                accfrom: senderAccountNumber,

                accto: receiverAccountNumber,

                amt: parseInt(amount), // Positive amount for receiver

                balance: receiverBalanceAfterTransfer

            };

            // Insert sender and receiver transactions

            await transactionCollection.insertOne(senderTransaction);

            await transactionCollection.insertOne(receiverTransaction);

            console.log(`Updated balances for sender (${senderAccount}) and receiver (${receiverAccount})`);

        } else {

            throw new Error("Sender does not have enough balance");

        }

    } catch (err) {

        console.error("Error:", err);

    } finally {

        await client.close();

        console.log("Disconnected from the database");

    }

}

async function displayRecord(req, res) {

    const client = new MongoClient(url);

    try {

        await client.connect();

        console.log("Connected to the database");

        const db = client.db(dbName);

        const collection = db.collection("transaction");

        const { account } = req.body;

        const filter = {

            $or: [{ accfrom: parseInt(account) }, { accto: parseInt(account) }]

        }

        const result = await collection.find(filter).toArray();

        console.log("Displayed document:", result);

        res.json(result);

    } catch (err) {

        console.error("Error:", err);

    } finally {

        await client.close();

        console.log("Disconnected from the database");

    }

}

app.get("/", (req, res) => {

    res.send("Hello");

});

app.post("/register", (req, res) => {

    createDocument(req, res);

    // res.send("Registration Successful");

});

app.post("/login", (req, res) => {

    readDocument(req, res);

});

app.post("/history", (req, res) => {

    displayRecord(req, res);

});

app.post("/transfer", (req, res) => {

    const { senderAccount, receiverAccount, amount } = req.body; // Extract data from request body

    if (!senderAccount || !receiverAccount || !amount) {

        return res.status(400).json({ error: "Invalid request data" });

    }

    updateAmount(senderAccount, receiverAccount, amount)

        .then(() => {

            res.json({ message: "Transfer successful" });

        })

        .catch((err) => {

            res.status(500).json({ error: "Internal Server Error" });

        });

});

// Route to schedule a transaction

app.post("/schedule-transaction", (req, res) => {

    const { senderAccount, receiverAccount, amount, scheduledTime } = req.body;

    if (!senderAccount || !receiverAccount || !amount || !scheduledTime) {

        return res.status(400).json({ error: "Invalid request data" });

    }

    // Parse the scheduledTime as a JavaScript Date object

    const scheduledDate = new Date(scheduledTime);

    // Schedule the transaction

    const job = schedule.scheduleJob(scheduledDate, () => {

        // Perform the transaction when the scheduled time is reached

        updateAmount(senderAccount, receiverAccount, amount);

        // Respond to the client

        res.json({ message: "Transaction scheduled successfully" });

    });

});

app.listen(port, () => {

    console.log(`App listening on port ${port}`);

});

Script.js:

function insertData() {

    var myHeaders = new Headers();

    myHeaders.append('Content-Type', 'application/json');

    let name = document.getElementById('txtName').value;

    let password = document.getElementById('txtPassword').value;

    let email = document.getElementById('txtEmail').value;

    let contact = document.getElementById('txtContact').value;

    let account = document.getElementById('txtAccount').value;

    var raw = JSON.stringify({

        'account': account,

        "name": name,

        "password": password,

        'email': email,

        'contact': contact

    });

    var requestOptions = {

        method: "POST",

        headers: myHeaders,

        body: raw,

        redirect: "follow"

    };

    fetch("http://localhost:3000/register", requestOptions)

        .then(response => response.text())

        .then(result => {

            console.log(result);

            alert(result);

        })

        .catch(error => console.log("error", error));

    window.location.href= "index.html";

}

function readData() {

    var myHeaders = new Headers();

    myHeaders.append('Content-Type', 'application/json');

    let name = document.getElementById('txtUserName').value;

    let password = document.getElementById('txtUserPassword').value;

    var raw = JSON.stringify({

        "name": name,

        "password": password,

    });

    var requestOptions = {

        method: "POST",

        headers: myHeaders,

        body: raw,

        redirect: "follow"

    };

    fetch("http://localhost:3000/login", requestOptions)

        .then(response => response.json())

        .then(result => {

            if(result) {

                alert("Login Successful");

                window.location.href="./index.html";

            }

            else {

                alert("Invalid username or password")

            }

        })

        .catch(error => console.log('error', error));

}

function transfer() {

    var myHeaders = new Headers();

    myHeaders.append('Content-Type', 'application/json');

    // Get values from input fields

    let senderAccount = parseInt(document.getElementById('txtAccFrom').value);

    let receiverAccount = parseInt(document.getElementById('txtAccTo').value);

    let amount = parseFloat(document.getElementById('txtAmount').value);

    // Create the request body

    var raw = JSON.stringify({

        "senderAccount": senderAccount,

        "receiverAccount": receiverAccount,

        "amount": amount

    });

    var requestOptions = {

        method: "POST",

        headers: myHeaders,

        body: raw,

        redirect: "follow"

    };

    fetch("http://localhost:3000/transfer", requestOptions)

        .then(response => response.json())

        .then(result => {

            if (result) {

                alert("Transfer Successful");

            } else {

                alert("Invalid Account From");

            }

        })

        .catch(error => console.log('error', error));

}

function scheduledtransfer() {

    const scheduledTime = document.getElementById('scheduledTime').value;

    let senderAccount = parseInt(document.getElementById('txtAccFrom').value);

    let receiverAccount = parseInt(document.getElementById('txtAccTo').value);

    let amount = parseFloat(document.getElementById('txtAmount').value);

    // Send a POST request to your backend with the scheduled time and other data

    fetch('http://localhost:3000/schedule-transaction', { // Update the URL to match your backend endpoint

        method: 'POST',

        headers: {

            'Content-Type': 'application/json'

        },

        body: JSON.stringify({ senderAccount, receiverAccount, amount, scheduledTime })

    })

    .then(response => response.json())

    .then(data => {

        console.log(data.message);

        // Handle the response from the server

    })

    .catch(error => {

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

    });

}

function displayData() {

    var myHeaders = new Headers();

    myHeaders.append('Content-Type', 'application/json');

    let account = document.getElementById('accfrom').value;

    var raw = JSON.stringify({

        'account': account

    });

    var requestOptions = {

        method: "POST",

        headers: myHeaders,

        body: raw,

        redirect: "follow"

    };

    fetch("http://localhost:3000/history", requestOptions)

        .then(response => response.json())

        .then(result => {

            // Clear the existing data before appending new data

            document.getElementById("TransactionData").innerHTML = '';

            result.forEach(element => {

                // Construct the HTML for each transaction entry

                var transactionHTML = `

                    <br>Account From: ${element.accfrom}

                    <br>Account To: ${element.accto}

                    <br>Amount debited/credited: ${element.amt}

                    <br>Balance: ${element.balance}

                    <hr>

                `;

                // Append the transaction entry to the container

                document.getElementById("TransactionData").innerHTML += transactionHTML;

            });

        })

        .catch(error => console.log('error', error));

}

function switchTab(tabName) {

    var loginTab = document.getElementById('login-tab');

    var registerTab = document.getElementById('register-tab');

    var loginForm = document.getElementById('login-form');

    var registerForm = document.getElementById('register-form');

    if (tabName === 'login') {

        loginTab.classList.add('active');

        registerTab.classList.remove('active');

        loginForm.style.display = 'block';

        registerForm.style.display = 'none';

        changePageTitle('Login form');

    } else if (tabName === 'register') {

        loginTab.classList.remove('active');

        registerTab.classList.add('active');

        loginForm.style.display = 'none';

        registerForm.style.display = 'block';

        changePageTitle('Registration form');

    }

}

function changePageTitle(newPageTitle) {

    document.title = newPageTitle;

}

Transfer.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>DashBoard</title>

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

</head>

<body>

    <nav>

        <div class="navbar">

               <a href="" style="color: white;text-decoration: none;">Login/Register</a>

               <a href="" style="color: white;text-decoration: none">History</a>

        </div>

    </nav>

    <center>

        <h1>Thakur Bank Website</h1>

    </center>

    <h2 style="text-align: left;"> <span id="accName"></span></h2>

    <div class="container">

        <div id="login-form">

            <h2>Transfer Portal</h2>

            <p>Account From:</p>

            <input type="text" name="txtAccFrom" id="txtAccFrom" required>

            <br>

            <br>

            <p>Account To:</p>

            <input type="text" name="txtAccTo" id="txtAccTo" required>

            <br>

            <br>

            <p>Amount :</p>

            <input type="number" name="txtAmount" id="txtAmount" required>

            <br>

            <br>

            <p>Schedule Your Payment Transfer: </p>

            <input type="datetime-local" name="scheduledTime" id="scheduledTime" placeholder="Scheduled Time">

            <br>

            <br>

            <div class="btn-back">

                <button class="btn" onclick="transfer()">Transfer</button>

                <button class="btn" onclick="scheduledtransfer()">Scheduled Transfer</button>

            </div>

        </div>

    </div>

    <script src="script.js"></script>

</body>

</html>

Style.css:

#register-form {

    display: none;

}

body {

    background-color: #f4f4f4;

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

}

.container {

    background-color: #ffffff;

    margin: 10vh auto;

    max-width: 600px;

    padding: 20px;

    border-radius: 10px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

h2 {

    text-align: center;

    color: #333;

}

nav {

    background-color: #0033a0;

    padding: 10px 0;

}

.btn {

    padding: 10px 20px;

    border: none;

    background-color: #0033a0;

    color: #fff;

    border-radius: 5px;

    cursor: pointer;

}

.btn:hover {

    background-color: #001f66;

}

input[type="text"],

input[type="password"],

input[type="number"],

input[type="email"],

input[type="datetime-local"] {

    width: 95%;

    padding: 10px;

    border: 1px solid #ccc;

    border-radius: 5px;

    margin-bottom: 15px;

}

.form-tab {

    display: flex;

    justify-content: center;

    margin-bottom: 20px;

}

.form-tab button {

    padding: 10px 20px;

    border: none;

    background-color: transparent;

    color: #333;

    cursor: pointer;

    outline: none;

    font-size: 16px;

}

.form-tab button.active {

    color: #0033a0;

    font-weight: bold;

    border-bottom: 2px solid #0033a0;

}

#accName {

    color: #0033a0;

    font-weight: bold;

}

#TransactionData {

    background-color:white;

    padding: 20px;

    border-radius: 5px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

/\* Navbar Styles \*/

.navbar {

    background-color: #0033a0;

    overflow: hidden;

    padding: 20px;

    text-decoration: none;

    color: #fff;

    height: 10px;

}