Term Project Report

Subject: Database Management System Design

Subject code:CS631003

FALL- 2021

Professor: Vincent Oria

Project Members:

- Kavya Umashanker(UCID:kbu2)
- Prajwal Mani(UCID:pbm6)

INDEX

Introduction

The Project lets us explore Bank databases and practical implementation of the operations carried out by the bank. After doing the requirement analysis and listing out the CS631-Bank operations and requirements.This website follows all the requirements and even some more features are added to make this as a complete experience using proper tools and technologies. There are two portals created based on the operations, one is the customer portal where customers can perform transactions and view all the transactions from various accounts. The second portal is for the employees to add, modify, delete customers and add employees.

System Requirements

Operating System :	Windows
Environment :	Python flask with mysql server
RAM Specifications :	1GB RAM
Storage Specifications :	50 GB Hard disk

ER Diagram

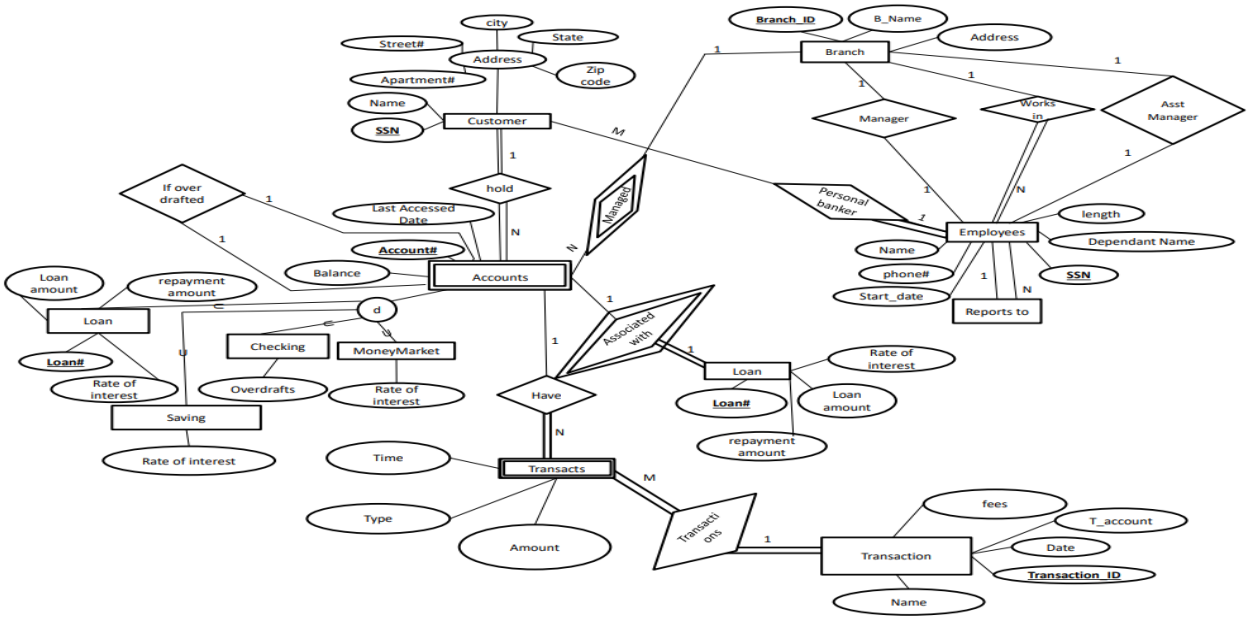

Figure 1: Entity Relationship Diagram

Relational Schema

Step -1:

//Listing all the entities [i.e, Customer and Employee]

Customer(CSSN, Name, Apartment#, street#, City, State, Zipcode)

Employee(ESSN, Name, Phone#, Start_date, Length_of_employment, Dependent_name)

Branch(BranchID, Bname,Address)

Transaction(Transaction_ID, Tname, T_account, date, fees)

Step -2:

//listing the weak attributes

Account(Account#, Balance, last_accessed_date, Account_type, Interest/Overdraft,CSSN)

Transacts(Transaction_ID,type,amount,time)

Step-3(1:1):

// Relationship between Branch and Employees — Manager and Assistant Manager

Branch(BranchID, Bname,Address,Manager, Asst_manager)

//Relationship between Account AND LOAN – each loan account has to be linked to the account.

Loan(Loan#, Amount, Repayment_Amount, Interest_Rate, Account#)

//Relationship between Checking account, if overdrafted, from which account should the money be withdrawn.

Account(Account#, Balance, last_accessed_date, Account_type, Interest/Overdraft, overdrafted_account#, CSSN)

Step -4(1:N):

//Relationship between branch and its employees – One branch can have many employees and Each

employees can work in one branch only.

Employee(ESSN, Name, Phone#, Start_date, Length_of_employment, Dependent_name, BranchID)

//Relationship between employees and itself – employees report to the branch manager.

Employee(ESSN, Name, Phone#, Start_date, Length_of_employment, Dependent_name, BranchID,

Manager_ESSN)

//Relationship between Employee and Customer- Each customer can have a personal banker and one

employee can handle many customer

Customer(CSSN, Name, Apartment#, street#, City, State, Zipcode,PersonalBanker_ESSN)

//Relationship between Customer and Account – one customer can have many accounts: it is already

handled

Account(Account#, Balance, last_accessed_date, Account_type, CSSN)

//Account can be either savings or money market or checking or loan.

Savings(SavingsAccount#, last_accessed_date, savings_interest_rate)

Checking(CheckingAccount#,Overdrafted_amount,overdrafted_account, date)

MoneyMarket(MarketAccount#, updated_date, market_interest_rate)

//Relationship between Account and branch – Each customer account resides in its homebranch

Account(Account#, Balance, last_accessed_date, Account_type, Interest/Overdraft, overdrafted_account#,

CSSN,BranchID)

//Relationship between Account and Transacts – Each account can perform many transactions

Transacts(TransactionID, Account#, type,amount,time)

Step-5(M:N):

//Relationship between Transacts and Transaction. This data can be retrieved from the table Transacts,

Hence we are not considering this table.

Transactions(Type,TransactionID)

• The Final set of Tables are:

1. Transaction(Transaction_ID, Tname, T_account, date, fees)

2. Branch(BranchID, Bname, Address, Manager, Asst_manager)

3. Loan(Loan#, Amount, Repayment_Amount, Interest_Rate, Account#)

4. Savings(SavingsAccount#, last_accessed_date, savings_interest_rate)

5. Checking(CheckingAccount#,Overdrafted_amount,overdrafted_account, date)

6. MoneyMarket(MarketAccount#, updated_date, market_interest_rate)

7. Employee(ESSN, Name, Phone#, Start_date, Length_of_employment, Dependent_name, BranchID,

Manager_ESSN)

8. Customer(CSSN, Name, Apartment#, street#, City, State, Zipcode,PersonalBanker_ESSN)

9. Account(Account#, Balance, last_accessed_date, Account_type, Interest/Overdraft,

overdrafted_account#, CSSN,BranchID)

10. Transacts(TransactionID, Account#, ,type,amount,time)

Application Design

Tools used:

Front end : Html,CSS,JavaScript

Backend : Python Flask, PHP

Database : MySQL

Server : XAMPP

User Manual

Home Page

Since the website and the database is accessed by both the customers and the employees so we have a common home page which directs to the login page of the desired user request.

Figure 2: Home page

Customer login page

Using this page, the customers are authenticated based on the email and password if the customer is not registered then customer is redirected to the register page

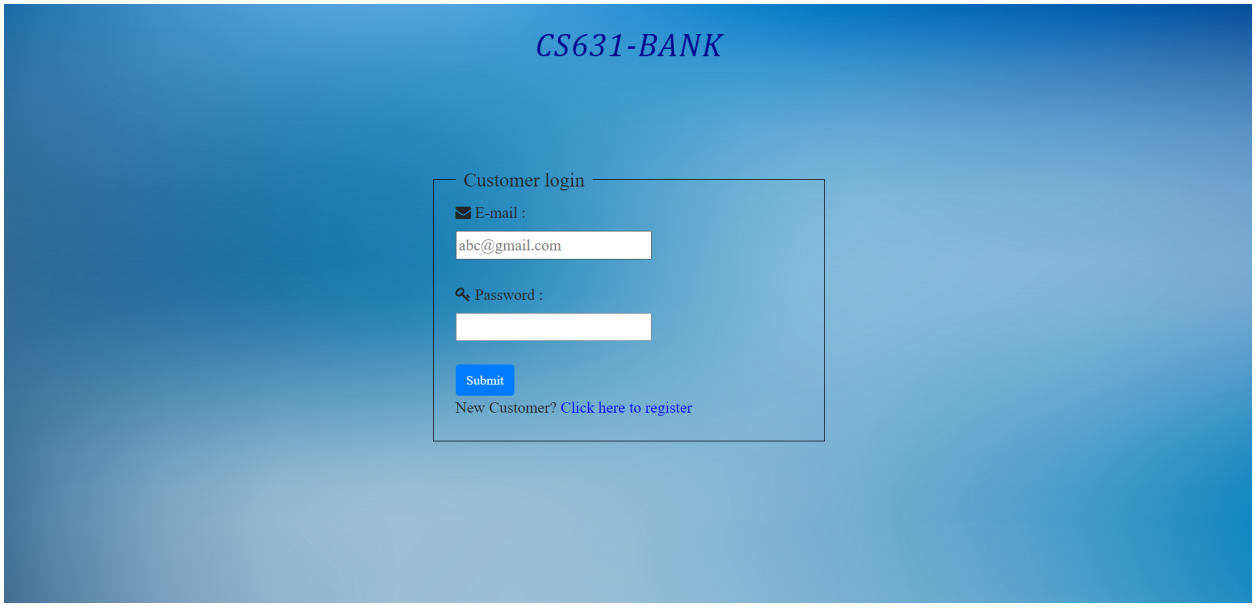

Figure 3: Customer Login Page

Customer Registration Page

Our website requires the customer to register before logging into the website.

Figure 4: Customer registration page

Customer accounts display page

This page shows all the accounts that the customer owns in the bank irrespective of the branches

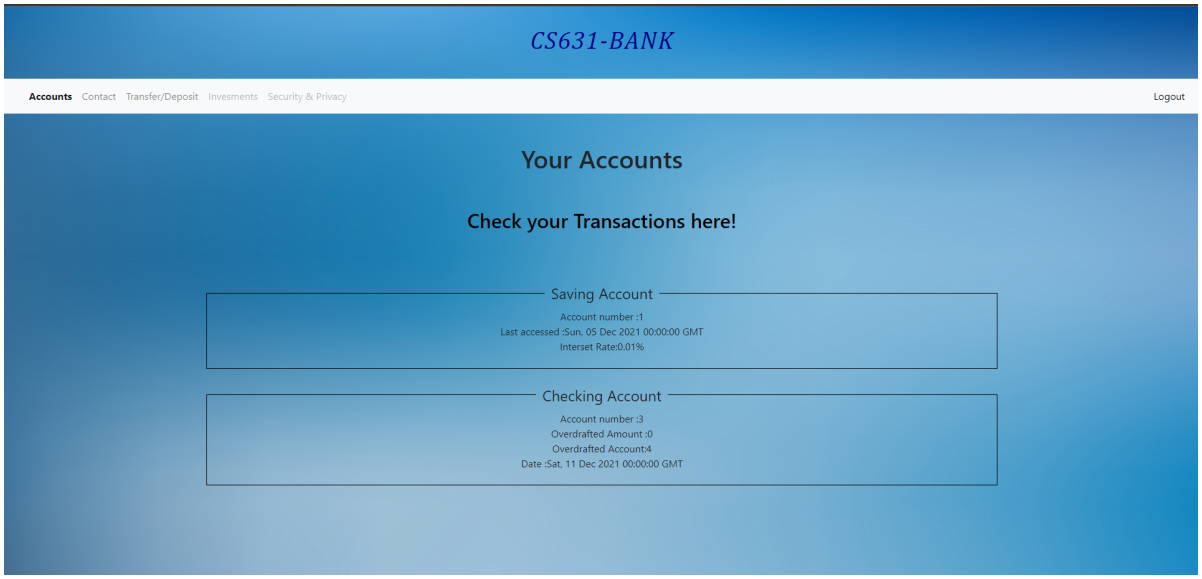

Figure 5: Accounts Display Page

Customer Transactions Display Page:

In this page the customer transactions are listed as passbook differentiating credit and debit to the account with there balance, transaction id ,transaction name, type and date.

Figure 6: Transactions Display page

Transfer Page

The accounts have three main operations to be performed that are cheque deposit,money withdrawal and cash deposit .All of the above operations are done on a single page and in background the database gets updated and values are inserted on different tables too.After a successful transaction the system generates a unique transaction id.

In case if the customer withdraws more than the balance the account is overdrafted and overdrafted interest and amount is updated to the account.

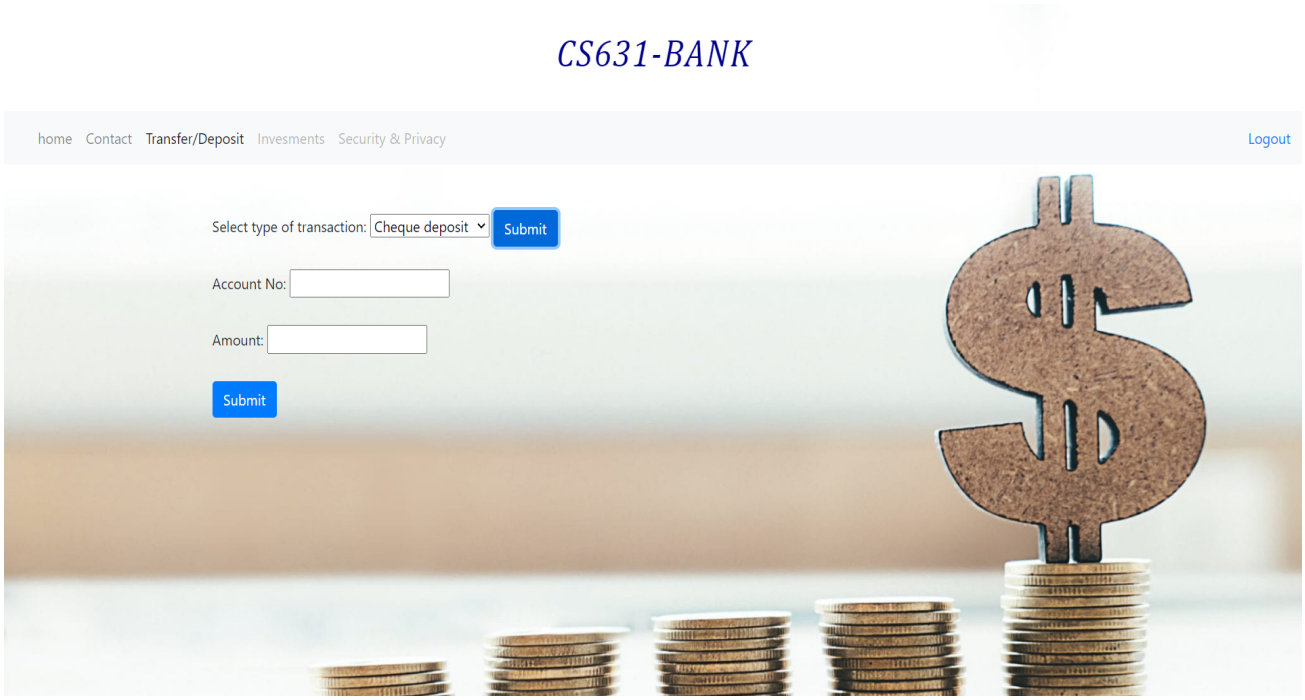

Figure 7: Options for Cheque deposit

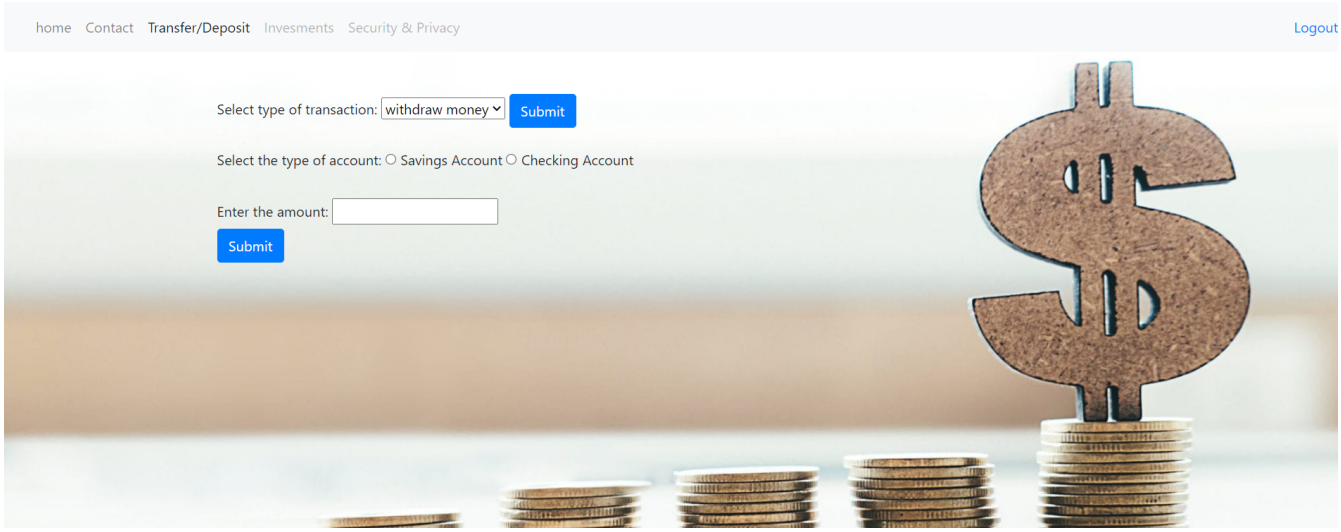

Figure 8: Option for money withdrawal

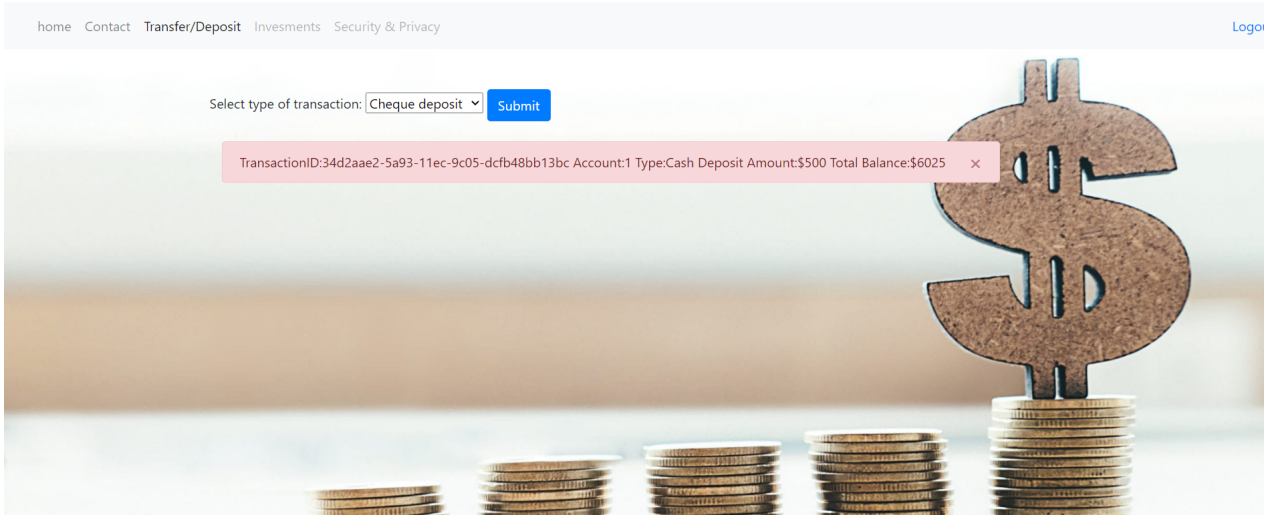

Figure 9: Successful Transaction details

Contact Us:

If there is any issue with the account the customer can contact the bank employee through the contact us page .

Figure 10: Contact us page

Employee login and register Page :

Same as the customer, the bank employee needs to register before logging in to the website.

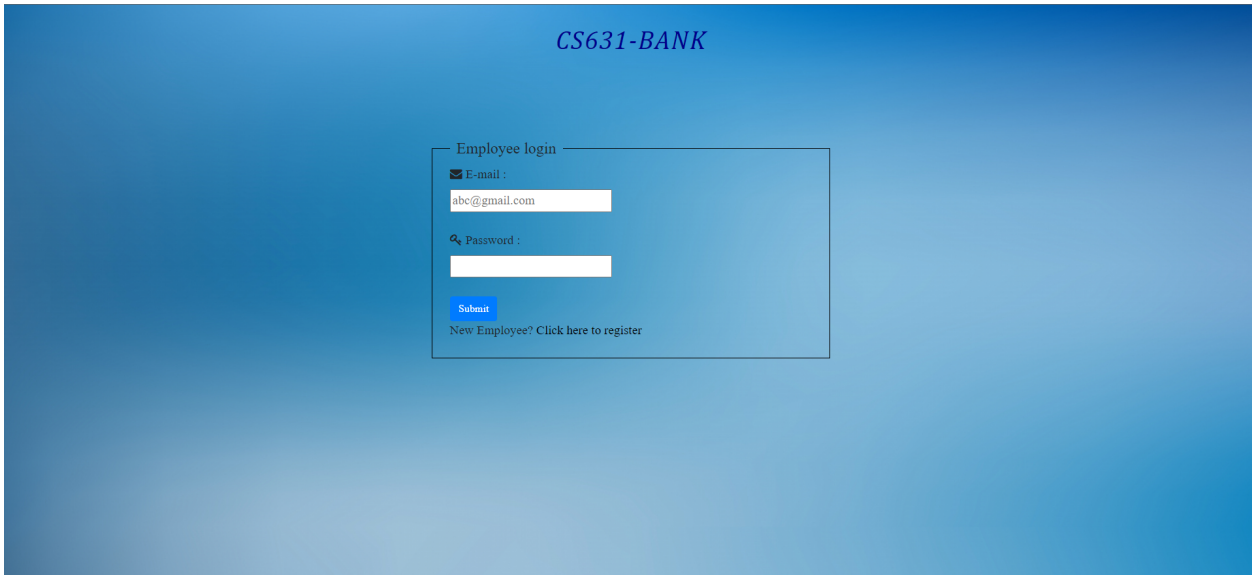

Figure 11: Employee login page

Figure 12: Employee Registration page

Employee operations Page

The bank employees can mainly perform three operations add, delete and modify the customer for all this operations the common page is employee operations page

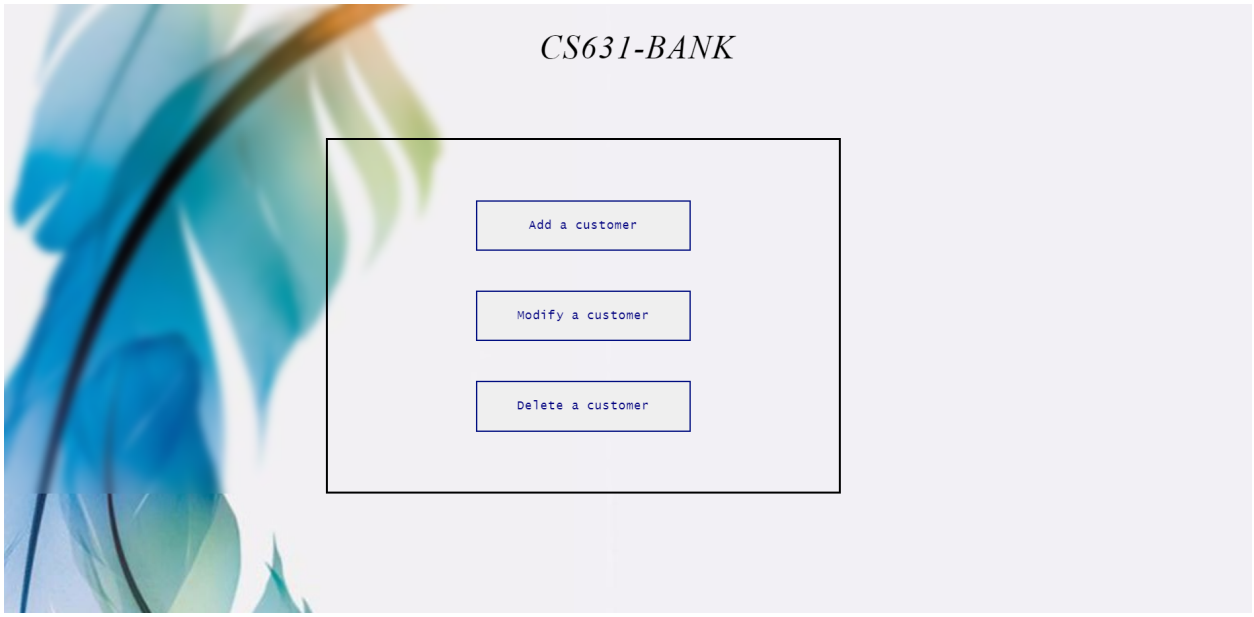

Figure 13: Employee operation page

Customer Account creation

This page allows the bank employees to create an account after getting all the basic customer details like name,address ssn etc. and assign personal banker

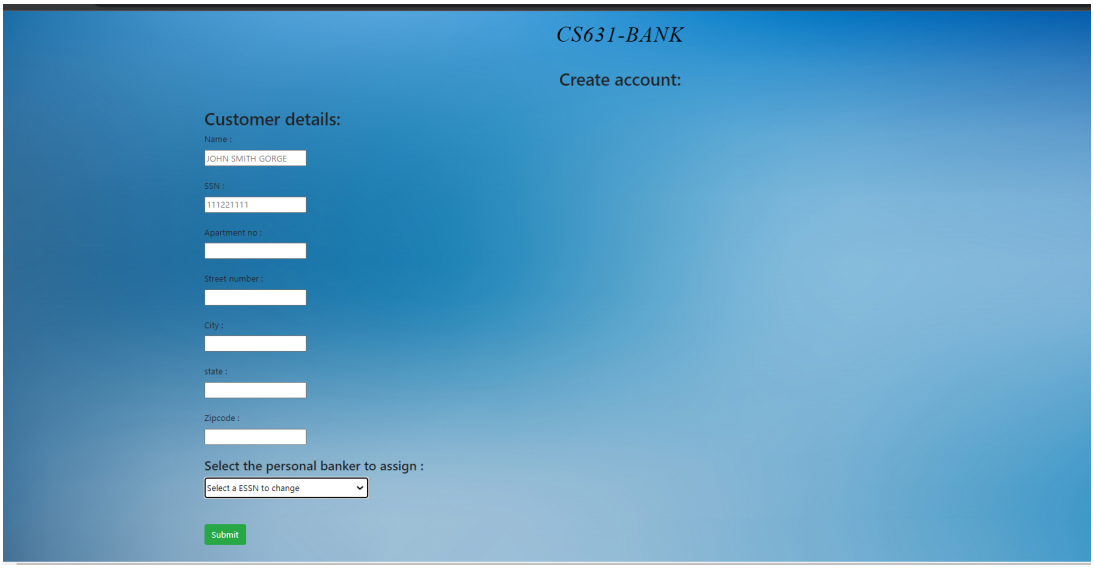

Figure 14: Create Customer Account.

Create different accounts

This page shows different fields based on the type of the account that is selected. The 4 types of accounts are savings,checking,loan and money market. Different accounts have different parameters such as saving account needs deposit and last accessed date, whereas checking account needs an overdraft account.

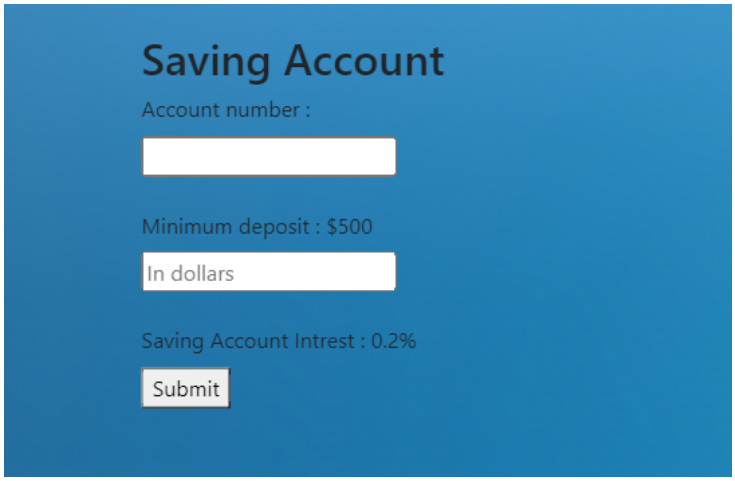

Figure 15: create Saving Account

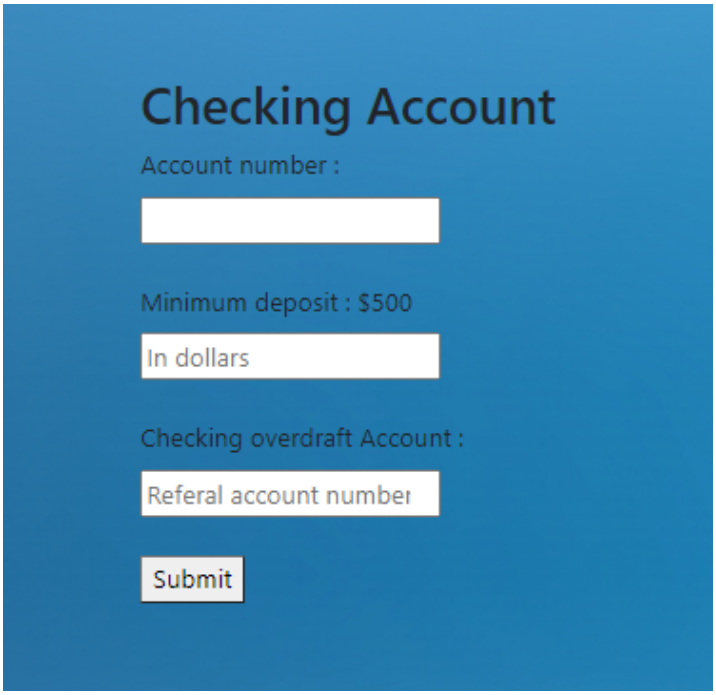

Figure 16: Create Checking Account

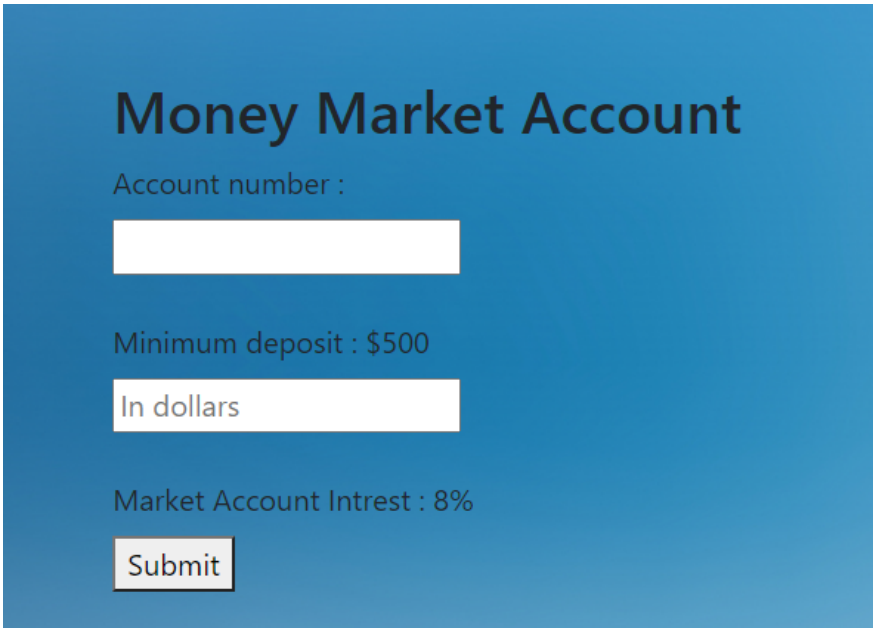

Figure 17: Create Money Market Account

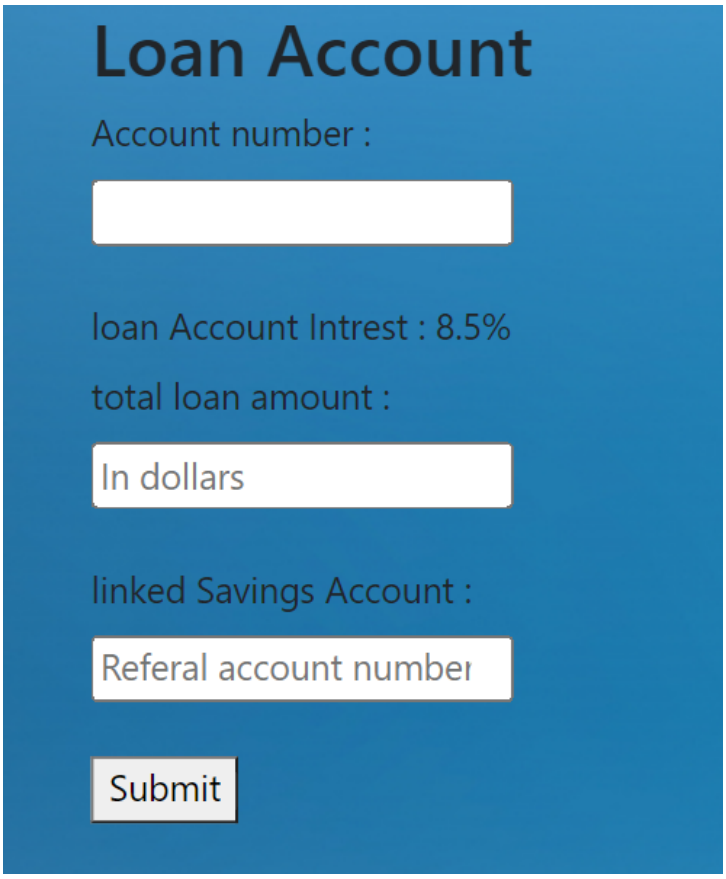

Figure 18: Creating a loan account

Modify Customer Page

This page is used to modify customer basic details.

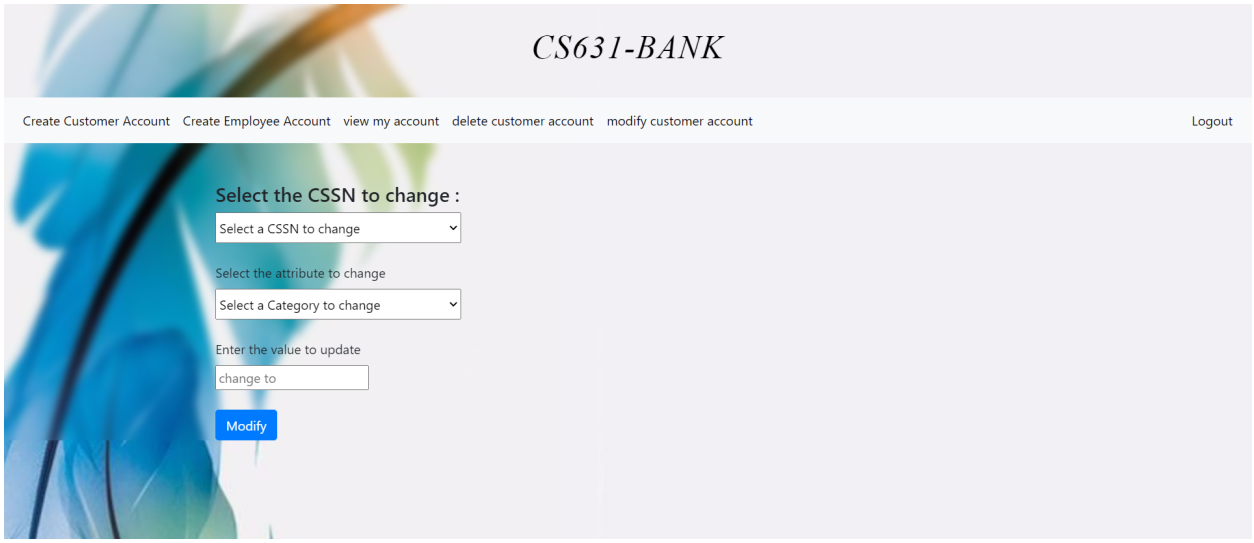

Figure 19: Modifying Details of a Customer

Deleting Customer Account Page

Based on the selected customer ssn the employee can delete the account this action could lead to deletion all the records for that particular customer is deleted on all the tables.

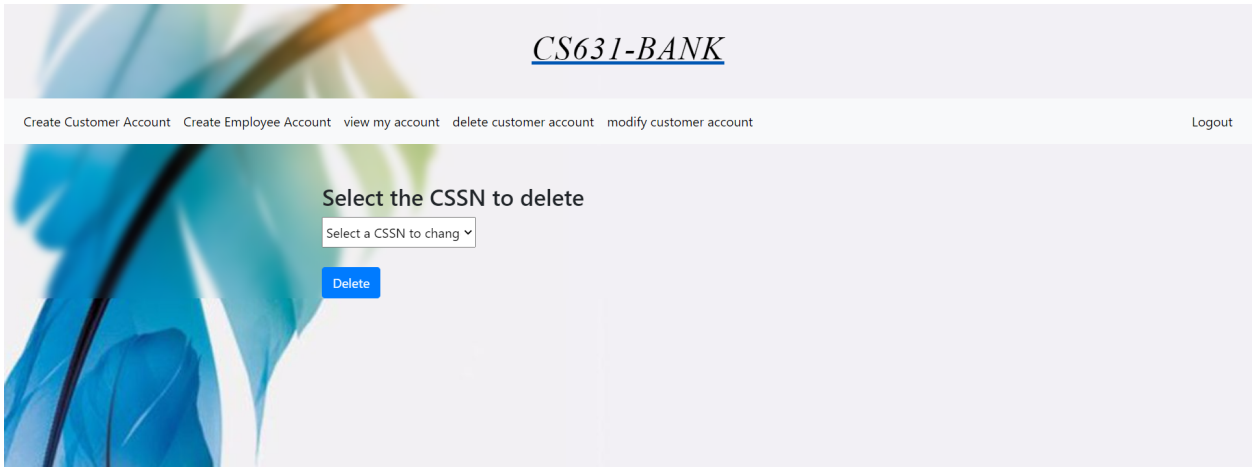

Figure 20: Deleting the customer account.

// Comment : To keep the report simple we added just a part of the code of different technologies if you want to check out the entire source code then please visit this link:
https://github.com/prajwalmani/Bank_Database_System_Design

SQL

Database name: 'Bank'

```
CREATE TABLE `account` (
  `account` int(50) NOT NULL,
  `balance` int(15) NOT NULL,
  `last_accessed_date` date NOT NULL,
  `account_type` varchar(5) NOT NULL,
  `intereset_overdraft` int(15) DEFAULT NULL,
  `cssn` varchar(10) NOT NULL,
  `Branch_id` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `auth` (
  `email` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `SSN` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

```
CREATE TABLE `branch` (
  `branchid` int(10) NOT NULL,
  `bname` varchar(50) NOT NULL,
  `address` text NOT NULL,
  `manager` varchar(50) NOT NULL,
  `asstmanager` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `checking` (
  `checkingaccount` varchar(50) NOT NULL,
  `overdrafted_amount` varchar(50) NOT NULL,
  `overdrafted_account` varchar(50) NOT NULL,
  `date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `customer` (
  `CSSN` varchar(50) NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Apartment` int(5) NOT NULL,
  `Street` int(5) NOT NULL,
```

```
  `City` varchar(50) NOT NULL,
  `State` text NOT NULL,
  `Zipcode` int(11) NOT NULL,
  `PersonalBanker_ESSN` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `employee` (
  `ESSN` varchar(50) NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Phone` int(50) NOT NULL,
  `Start_date` date NOT NULL,
  `Length_of_employment` int(5) NOT NULL,
  `Dependent_Name` varchar(50) DEFAULT NULL,
  `Branch_ID` int(10) NOT NULL,
  `Manager_SSN` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `loan` (
  `loan` varchar(50) NOT NULL,
  `Amount` int(50) NOT NULL,
  `Repayment_amount` int(50) NOT NULL,
```

```
  `interset_rate` int(5) NOT NULL DEFAULT 0,
  `account` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `moneymarket` (
  `marketaccount` varchar(50) NOT NULL,
  `updated_date` date NOT NULL,
  `market_interset_rate` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `savings` (
  `savingsaccount` varchar(50) NOT NULL,
  `last_accessed_date` date NOT NULL,
  `savings_interset_rate` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
CREATE TABLE `transcation` (
  `transactionid` varchar(50) NOT NULL,
  `tname` varchar(50) NOT NULL,
  `taccount` varchar(50) NOT NULL,
  `date` date NOT NULL,
  `fees` int(11) NOT NULL DEFAULT 0
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `transct` (
  `transactionid` varchar(50) NOT NULL,
  `account` varchar(50) NOT NULL,
  `type` varchar(10) NOT NULL,
  `tname` varchar(50) NOT NULL,
  `amount` int(50) NOT NULL,
  `balance` int(11) NOT NULL,
  `time` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


ALTER TABLE `account`
  ADD PRIMARY KEY (`account`);


ALTER TABLE `auth`
  ADD PRIMARY KEY (`email`);


ALTER TABLE `branch`
  ADD PRIMARY KEY (`branchid`);
```

```
ALTER TABLE `checking`
  ADD PRIMARY KEY (`checkingaccount`);

ALTER TABLE `customer`
  ADD PRIMARY KEY (`CSSN`);

ALTER TABLE `employee`
  ADD PRIMARY KEY (`ESSN`),
  ADD UNIQUE KEY `Phone` (`Phone`);

ALTER TABLE `loan`
  ADD PRIMARY KEY (`loan`);

ALTER TABLE `moneymarket`
  ADD PRIMARY KEY (`marketaccount`);

ALTER TABLE `transcation`
  ADD PRIMARY KEY (`transactionid`);

ALTER TABLE `transct`
```

```
  ADD PRIMARY KEY (`transactionid`);

COMMIT;


ALTER TABLE `account`

  ADD CONSTRAINT `account_ibfk_1` FOREIGN KEY (`cssn`) REFERENCES `customer` (`CSSN`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `account_ibfk_2` FOREIGN KEY (`Branch_id`) REFERENCES `branch` (`branchid`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `checking`

  ADD CONSTRAINT `checking_ibfk_1` FOREIGN KEY (`checkingaccount`) REFERENCES `account` (`account`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `checking_ibfk_2` FOREIGN KEY (`overdrafted_account`) REFERENCES `account` (`account`) ON DELETE CASCADE ON UPDATE CASCADE;

-- Constraints for table `customer`

ALTER TABLE `customer`

  ADD CONSTRAINT `customer_ibfk_1` FOREIGN KEY (`PersonalBanker_ESSN`) REFERENCES `employee` (`ESSN`) ON DELETE CASCADE ON UPDATE CASCADE;

-- Constraints for table `employee`

ALTER TABLE `employee`

  ADD CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`Manager_SSN`) REFERENCES `employee` (`ESSN`),
```

```
  ADD CONSTRAINT `employee_ibfk_2` FOREIGN KEY (`Branch_ID`) REFERENCES
`branch` (`branchid`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `loan`

  ADD CONSTRAINT `loan_ibfk_1` FOREIGN KEY (`loan`) REFERENCES `account`
(`account`) ON DELETE CASCADE ON UPDATE CASCADE,

  ADD CONSTRAINT `loan_ibfk_2` FOREIGN KEY (`account#`) REFERENCES `account`
(`account`) ON DELETE CASCADE ON UPDATE CASCADE;

`

ALTER TABLE `moneymarket`

  ADD CONSTRAINT `moneymarket_ibfk_1` FOREIGN KEY (`marketaccount`)
REFERENCES `account` (`account`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `savings`

  ADD CONSTRAINT `savings_ibfk_1` FOREIGN KEY (`savingsaccount`) REFERENCES
`account` (`account`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `transcation`

  ADD CONSTRAINT `transcation_ibfk_1` FOREIGN KEY (`taccount`) REFERENCES
`account` (`account`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `transct`
```

ADD CONSTRAINT `transct_ibfk_1` FOREIGN KEY (`transactionid`) REFERENCES `transcation` (`transactionid`) ON DELETE CASCADE ON UPDATE CASCADE,

ADD CONSTRAINT `transct_ibfk_2` FOREIGN KEY (`account#`) REFERENCES `account` (`account`) ON DELETE CASCADE ON UPDATE CASCADE;

Python FLASK

```
from types import MethodDescriptorType

from flask import Flask,render_template,request,flash,url_for,redirect

from flask.templating import render_template_string

from flask.wrappers import Request

import pymysql

from pymysql import DATE, ROWID, cursors

import json

import os

import time

import uuid

import datetime

from datetime import date

import calendar

import datetime

import time
```

```
app = Flask(__name__)

app.secret_key = "super secret key"



connection =pymysql.connect(host="localhost",user="root",passwd="",database="bank")
cursor=connection.cursor()

global_ssn=[]

@app.route('/', methods=['GET', 'POST'])
def home():
    servicecharge()
    if request.method=='POST':
        customerbutton=request.form.get('customerbutton')
        employeebutton=request.form.get('employeebutton')
        if customerbutton=='cb':
            return redirect(url_for("customerauth"))
        elif employeebutton=='eb':
            return redirect(url_for("employeeauth"))
```

```
        else:
            pass
    return render_template('home.html')


@app.route('/customerlogin.html', methods=['GET', 'POST'])
def customerauth():
    if request.method == 'POST':
        email=request.form['email']
        password=request.form['password']


        sql="SELECT * FROM `auth`;"
        cursor.execute(sql)
        rows=cursor.fetchall()
        for row in rows:
            if email==row[0] and password==row[1]:
                    global_ssn.append(row[2])
                    return redirect(url_for('customeraccountsdisp'))
        else:
            flash("Wrong credentials! Try again... ")
            return  render_template("customerlogin.html")
    return render_template("customerlogin.html")
```

```
@app.route("/employeelogin.html",methods=['GET',"POST"])
def employeeauth():
    if request.method == 'POST':
        email=request.form['email']
        password=request.form['password']

        sql="SELECT * FROM `auth`;"
        cursor.execute(sql)
        rows=cursor.fetchall()
        for row in rows:
            if email==row[0] and password==row[1]:
                return render_template("emphomedisplay.html")
        else:
            flash("Wrong credentials! Try again... ")
            return  render_template("employeelogin.html")
    return  render_template("employeelogin.html")

@app.route('/employeeregister.html',methods=['GET', 'POST'])
def employeeregister():
    if request.method == 'POST':
```

```
        ssn=request.form['SSN']
        email=request.form['email']
        branchid=request.form['Branch']
        password=request.form['Password']
        repassword=request.form['CPassword']

        if password!=repassword:
            flash("Passwords dont match!.")
            return render_template("employeeregister.html")

        sql="select ESSN,Branch_ID from employee where ESSN={0} and
Branch_ID={1}".format(ssn,branchid)
        cursor.execute(sql)
        rows=cursor.fetchall()
        for row in rows:
            sql=("""INSERT INTO `auth`(`email`, `password`, `SSN`) VALUES (%s,%s,%s)""")
            values=(email,password,ssn)
            cursor.execute(sql,values)
            connection.commit()
            return redirect(url_for('employee'))
        else:
```

```
            flash("Employee not found!")
    return render_template("employeeregister.html")


@app.route('/customerregister.html',methods=['GET', 'POST'])
def customerregister():
    if request.method == 'POST':
        ssn=request.form['SSN']
        email=request.form['email']
        password=request.form['Password']
        repassword=request.form['CPassword']
        if password!=repassword:
                flash("Passwords dont match!.")
                return render_template("customerregister.html")
        sql="select CSSN from customer where CSSN={0}".format(ssn)
        cursor.execute(sql)
        rows=cursor.fetchall()
        for row in rows:
            sql=("""INSERT INTO `auth`(`email`, `password`, `SSN`) VALUES (%s,%s,%s)""")
            values=(email,password,ssn)
            cursor.execute(sql,values)
            connection.commit()
```

```
            return redirect(url_for('customerlogin'))
        else:
            flash("Customer not found!")
    return render_template("customerregister.html")



@app.route('/customeraccountsdisp.html',methods=['POST','GET'])
def customeraccountsdisp():
    ssn=global_ssn[-1]
    f = open("phpdata.txt", "w")
    f.write(ssn)
    f.close()
    sql='select account,balance,last_accessed_date,account_type from `account`where
cssn={0};'.format(ssn)
    cursor.execute(sql)
    rows=cursor.fetchall()
    jsondata={}
    for row in rows:
        account_type=row[3]
        if account_type=='s':
            savings_account=row[0]
```

```
        savings_sql="select * from `savings` where
savingsaccount={0};".format(savings_account)
        cursor.execute(savings_sql)
        rows=cursor.fetchall()
        for row in rows:
            buff_dict={
                "s":{
                    "savingsaccount":row[0],
                    "last_accessed_data":row[1],
                    "savings_interset_rate":row[2]
                }
            }
            jsondata.update(buff_dict)

    if account_type=='l':
        loan_account=row[0]
        loan_sql="SELECT * FROM `loan` where loan#={0};".format(loan_account)
        cursor.execute(loan_sql)
        rows=cursor.fetchall()
        for row in rows:
            buff_dict={
```

```
                "l":{
                    "loanaccount":row[0],
                    "amount":row[1],
                    "repaymentamount":row[2],
                    "interesetrate":row[3],
                    "account":row[4]
                }
            }
            jsondata.update(buff_dict)

        if account_type=='c':
            checking_account=row[0]
            checkingsql="SELECT * FROM `checking` where
checkingaccount={0}".format(checking_account)
            cursor.execute(checkingsql)
            rows=cursor.fetchall()
            for row in rows:
                buff_dict={
                    "c":{
                        "checkingaccount":row[0],
                        "overdraftedamount":row[1],
```

```
                "overdraftedaccount":row[2],
                "date":row[3]
            }
        }
        jsondata.update(buff_dict)

    if account_type=='m':
        money_market_account=row[0]
        money_market_sql="SELECT * FROM `moneymarket` where
marketaccount={0}".format(money_market_account)
        cursor.execute(money_market_sql)
        rows=cursor.fetchall()
        for row in rows:
            buff_dict={
                "m":{
                    "marketaccount":row[0],
                    "updateddate":row[1],
                    "marketinterestrate":row[2]
                }
            }
            jsondata.update(buff_dict)
```

```
    return render_template("customeraccountsdisp.html",data=jsondata)


@app.route('/contactus.html')
def contactus():
    return render_template("contactus.html")



@app.route('/transfer.html', methods=['GET', 'POST'])
def transfer():
    flag=0
    ssn=global_ssn[-1]
    # cheque deposit
    if request.method == 'POST' and "toaccountno" in request.form:
        toaccountno=request.form['toaccountno']
        toamount=request.form['toamount']
        try:
            sql="select `account`,`balance` from `account` where `cssn`={0}".format(ssn)
            cursor.execute(sql)
            rows=cursor.fetchall()
            for row in rows:
```

```
            balance=int(row[1])-int(toamount)
            account=row[0]
            update_sql="""UPDATE `account`
            SET `balance`={0}
            where account={1}""".format(balance,account)
            cursor.execute(update_sql)
            connection.commit()
            insert_quer=("""
            INSERT INTO `transct` (`transactionid`, `account`, `type`, `tname`, `amount`,`balance`
,`time`) VALUES (%s,%s,%s,%s,%s,%s,%s)
            """)
            values=(uuid.uuid1(),account,"CDD","Cheque Deposit
Debit",toamount,balance,datetime.datetime.utcnow())
            cursor.execute(insert_quer,values)
            connection.commit()
        sql="select `balance` from `account` where `account`={0}".format(toaccountno)
        cursor.execute(sql)
        rows=cursor.fetchall()
        for row in rows:
            balance=int(row[0])+int(toamount)
            update_sql="""UPDATE `account`
            SET `balance`={0}
```

```
            where account={1}""".format(balance,toaccountno)
            cursor.execute(update_sql)
            connection.commit()
            insert_quer=("""
            INSERT INTO `transct` (`transactionid`, `account`, `type`, `tname`, `amount`,`balance`
,`time`) VALUES (%s,%s,%s,%s,%s,%s,%s)
            """)
            values=(uuid.uuid1(),toaccountno,"CDC","Cheque Depost Credit
",toamount,balance,datetime.datetime.utcnow())
            cursor.execute(insert_quer,values)
            connection.commit()
            flash("Cheque deposited to Account No:{0} worth of
${1}".format(toaccountno,toamount))

        except:
            flash("Failure Cheque not deposited!")

    # withdraw money
    if request.method=="POST" and "waccount" in request.form:
        radio_value=request.form['waccount']
        amount=request.form['amount']
        try:
```

```
        if radio_value=='s':
            sql="select `account`,`balance` from `account` where `account_type`= 's' and `cssn`={0}".format(ssn)
        if radio_value=='c':
            sql="select `account`,`balance` from `account` where `account_type`='c' and `cssn`={1}".format(ssn)
        cursor.execute(sql)
        rows=cursor.fetchall()
        for row in rows:
            balance=int(row[1])-int(amount)
            if balance<0 and radio_value=='s':
                flash("Insufficent Funds")
            if balance<0 and radio_value=='c':
                flash("Overdrafted But transaction was successfull")
                overdraftedsql=("""
                INSERT INTO `checking`(`checkingaccount`, `overdrafted_amount`, `overdrafted_account`, `date`) VALUES (%s,%s,%s,%s)
                """)
                today=date.today()
                values=(row[0],balance,"1234",today)
                cursor.execute(overdraftedsql,values)
                connection.commit()
```

```
            account=row[0]
            update_sql="""UPDATE `account`
            SET `balance`={0}
            where account={1}""".format(balance,account)
            cursor.execute(update_sql)
            connection.commit()
            insert_quer=("""
            INSERT INTO `transct` (`transactionid`, `account`, `type`, `tname`, `amount`,`balance`
,`time`) VALUES (%s,%s,%s,%s,%s,%s,%s)
            """)
            values=(uuid.uuid1(),account,"MW","Money
Withdraw",amount,balance,datetime.datetime.utcnow())
            cursor.execute(insert_quer,values)
            connection.commit()
            flash("Transcation was successfull")
        except:
            pass


    # cash deposit
    if request.method == 'POST' and "raccount" in request.form:
        ssn=global_ssn[-1]
        radio_value=request.form['raccount']
```

```
    eamount=request.form['eamount']
    try:
        if radio_value=='s':
            sql="select `account`,`balance` from `account` where `account_type`= 's' and `cssn`={0}".format(ssn)
        if radio_value=='c':
            sql="select `account`,`balance` from `account` where `account_type`='c' and `cssn`={0}".format(ssn)
        cursor.execute(sql)
        rows=cursor.fetchall()
        for row in rows:
            balance=int(row[1])+int(eamount)
            account=row[0]
            update_sql="""UPDATE `account`
            SET `balance`={0}
            where account={1}""".format(balance,account)
            cursor.execute(update_sql)
            connection.commit()
            insert_quer=("""
            INSERT INTO `transct` (`transactionid`, `account`, `type`, `tname`, `amount`,`balance`,`time`) VALUES (%s,%s,%s,%s,%s,%s,%s)
            """)
```

```
            values=(uuid.uuid1(),account,"CSD","Cash
Deposit",eamount,balance,datetime.datetime.utcnow())

            cursor.execute(insert_quer,values)

            connection.commit()

            flash("TransactionID:{0}\nAccount:{1}\nType:Cash Deposit\nAmount:${2}\nTotal
Balance:${3}\n".format(uuid.uuid1(),account,eamount,balance))

        except:

             flash("Couldn't find any savings/checking account link to this Email id Please contact
the branch.")

    return render_template("transfer.html")


# @app.route("acctransactiondisplay.html",methods=['GET', 'POST'])

def acctransdisp():

    return render_template("acctransactiondisplay.html")


def servicecharge():

        today = datetime.date.today()

        if today.day == calendar.monthrange(today.year, today.month):

            sql="SELECT account,balance FROM `account`;"

            cursor.execute(sql)

            rows=cursor.fetchall()

            for row in rows:
```

```
            balance=int(row[1])-10
            account=row[0]
            update_sql="""UPDATE `account`
               SET `balance`={0}
               where account={1}""".format(balance,account)
            cursor.execute(update_sql)
            connection.commit()
            insert_quer=("""
               INSERT INTO `transct` (`transactionid`, `account`, `type`, `tname`,
`amount`,`balance` ,`time`) VALUES (%s,%s,%s,%s,%s,%s,%s)
               """)
            values=(uuid.uuid1(),account,"SC","Service
Charge",int(10),balance,datetime.datetime.utcnow())
            cursor.execute(insert_quer,values)
            connection.commit()

if __name__ == "__main__":
   app.run(debug=True)
```

PHP Codes

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Transaction Details</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"
integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0s
ibVcOQVnN" crossorigin="anonymous">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2
xXr2" crossorigin="anonymous">
</head>

<style>
body{
  background-image: url('https://img.wallpapersafari.com/desktop/1680/1050/42/12/w3xlib.jpg');
  background-size: cover;
  background-repeat: no-repeat;
   background-color: rgb(220, 227, 230);

}
#heading{
```

```
    background-color: transparent;
    padding:30px 20px 30px 20px;
    color: darkblue;
    font-family:Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
    font-style: oblique;
    letter-spacing: 1.5px;
    word-spacing: 4px;

    text-align: center;
}
.center {
  margin-left: auto;
  margin-right: auto;
  border:"1";
}
th, td {
  padding: 15px;
  text-align:center
}

#head{
```

```
  color: white;
}

a:visited,link{
color: black;
}



#Entertainment{
  font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}
#right{
  margin-left:0.5cm;
}
#right2{
  margin-left:23cm;
}
fieldset.scheduler-border {
    border: 1px double black;
    padding: 0 1.4em 1.4em 1.4em;
    margin: 0 0 1.5em 0;
```

```
}

legend.scheduler-border {
    width:auto; /* Or auto */
    padding:0 10px; /* To give a bit of padding on the left and right */
    border-bottom:none;
}
#loginbox{
  font-family: 'Times New Roman', Times, serif;
  font-size: larger;
}

</style>
<title>transaction display</title>
<body id="main">
<a href="home.html" ><h1 id="heading">CS631-BANK</h1></a>

<nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#"></a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
```

```
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#"><b>Accounts</b><span
class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="http://127.0.0.1:5000/contactus.html">Contact</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="http://127.0.0.1:5000/transfer.html">Transfer/Deposit</a>
      </li>
      <li class="nav-item">
        <a class="nav-link " href="#">Invesments</a>
      </li>
      <li class="nav-item">
        <a class="nav-link " href="#">Security & Privacy</a>
      </li>
```

```
      </ul>
        <ul class="navbar-nav ml-auto">
          <li class="nav-item">
        <a  class="nav-link1" style="float: right;"
href="http://127.0.0.1:5000/customerlogin.html">Logout</a>
      </li>
      </ul>
    </div>
  </nav>
  <br>
<br>
<div class="row">
    <div class="offset-2 col-8">

          <?php
$servername = "localhost";
$username = "root";
$password = NULL;
$db = "bank";
// Create connection
$conn = mysqli_connect($servername, $username, $password,$db);
```

```
// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
//echo "Connected successfully<br>";

$myfile = fopen("C:\Users\prajw\OneDrive\Desktop\dbms project\phpdata.txt", "r") or die("Unable to open file!");
$file = fgets($myfile);

fclose($myfile);

$sql = "SELECT name FROM customer where cssn = '$file'; ";

$result = mysqli_query($conn,$sql);
$row = mysqli_fetch_array($result);
        $user = $row['name'];
echo "<h2>Account user name: $user<h2>";

$sql = "SELECT account,account_type FROM account where cssn = '$file'; ";
$result = $conn->query($sql);
```

```
$i=0;
while($rs = $result->fetch_assoc()){
    $data[$i] = array('account'=>$rs['account'],'type'=>$rs['account_type']);
    $i++;
}



$d= count($data);



for ($x = 0; $x < $d; $x++) {
$acc= $data[$x]['account'];
$type = $data[$x]['type'];
if ($type == 's'){
  $type = "SAVINGS ACCOUNT";
}
if ($type == 'c'){
  $type = "CHECKING ACCOUNT";
}
if ($type == 'm'){
  $type = "MONEY MARKET ACCOUNT";
```

```
}
if ($type == 'l'){
  $type = "LOAN ACCOUNT";
}
$sql = "SELECT * FROM transct where account = '$acc'; ";
$result = $conn->query($sql);


echo "
<h4>Transaction of Account number : $acc  </h4>
<br>
<h4> Account type : $type</h4>
        <br>
        <br>";
if ($result->num_rows > 0) {
        echo"
        <table class='center' border='2px black'>
          <tr>
            <th>Transaction ID</th>
            <th>Transaction name</th>
            <th>Date</th>
            <th>Type</th>
```

```
            <th>credit</th>
            <th>debit</th>
            <th>Balance</th>

          </tr>";

while($rs = $result->fetch_assoc()){
    $id = $rs['transactionid'];
    $tname = $rs['tname'];
    $type=$rs['type'];
    $amount = $rs['amount'];
    $balance=$rs['balance'];
    $time=$rs['time'];
    if (($type == 'CSD') or ($type == 'CD')or ($type == 'CDC')){
      $credit = $amount;
      $debit = '-';
    }
    if (($type == 'CDD') or ($type == 'SC')or ($type == 'MW')){
      $debit=$amount;
      $credit = '-';
```

```
    }
    echo"<tr>
    <td>$id</td>
    <td>$tname</td>
    <td>$time</td>

    <td>$type</td>
    <td>$credit</td>
    <td>$debit</td>
    <td>$balance</td>
  </tr>";
}
echo " </table><br><br><br>";
}
else{
  echo " <h3 style=text-align:center;>NO TRANSACTIONS TO DISPLAY!</h3> ";
}
}

?>
    </body>
```

```
    </html>
```

Accountdips.html

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Customer Login</title>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"
integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0s
ibVcOQVnN" crossorigin="anonymous">

    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2
xXr2" crossorigin="anonymous">

</head>

<style>

body{

  background-image: url('https://img.wallpapersafari.com/desktop/1680/1050/42/12/w3xlib.jpg');

  background-size: cover;

  background-repeat: no-repeat;

   background-color: rgb(220, 227, 230);
```

```
}
#heading{
    background-color: transparent;
    padding:30px 20px 30px 20px;
    color: darkblue;
    font-family:Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
    font-style: oblique;
    letter-spacing: 1.5px;
    word-spacing: 4px;
    text-align: center;
}



#head{
  color: white;
}


a
{color:black;
```

```
}
a:visited,link{
color: black;
}



#Entertainment{
  font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
}
#right{
  margin-left:0.5cm;
}
#right2{
  margin-left:23cm;
}
fieldset.scheduler-border {
    border: 1px double black;
    padding: 0 1.4em 1.4em 1.4em;
    margin: 0 0 1.5em 0;
}
```

```
legend.scheduler-border {
    width:auto; /* Or auto */
    padding:0 10px; /* To give a bit of padding on the left and right */
    border-bottom:none;
}
#accountbox{
  font-family: 'Times New Roman', Times, serif;
  font-size: larger;
}



.open-button {
  background-color: #555;
  color: white;
  padding: 16px 20px;
  border: none;
  cursor: pointer;
  opacity: 0.8;
  position: fixed;
  bottom: 23px;
  right: 28px;
```

```
  width: 280px;
}

/* The popup form - hidden by default */
.form-popup {
  display: none;
  position: fixed;
  bottom: 0;
  right: 15px;
  border: 3px solid #f1f1f1;
  z-index: 9;
}



.overlay {
  position: fixed;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
```

```
  background: rgba(0, 0, 0, 0.7);
  transition: opacity 500ms;
  visibility: hidden;
  opacity: 0;
}
.overlay:target {
  visibility: visible;
  opacity: 1;
}

.popup {
  margin: 70px auto;
  padding: 20px;
  background: #fff;
  border-radius: 5px;
  width: 30%;
  position: relative;
  transition: all 5s ease-in-out;
}

.popup h2 {
```

```
  margin-top: 0;
  color: #333;
  font-family: Tahoma, Arial, sans-serif;
}
.popup .close {
  position: absolute;
  top: 20px;
  right: 30px;
  transition: all 200ms;
  font-size: 30px;
  font-weight: bold;
  text-decoration: none;
  color: #333;
}
.popup .close:hover {
  color: #06D85F;
}
.popup .content {
  max-height: 30%;
  overflow: auto;
}
```

```
@media screen and (max-width: 700px){
  .box{
    width: 70%;
  }
  .popup{
    width: 70%;
  }
}
.nav-link1 {
      color:white;
      padding-right: .5rem !important;
      padding-left: .5rem !important;
    }

    /* Fixes dropdown menus placed on the right side */
    .ml-auto .dropdown-menu {
      left: auto !important;
      right: 0px;
    }
</style>
```

```
<body id="main">


  <a href="home.html" ><h1 id="heading">CS631-BANK</h1></a>

  <nav class="navbar navbar-expand-lg navbar-light bg-light">

    <a class="navbar-brand" href="#"></a>

    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>


    <div class="collapse navbar-collapse" id="navbarSupportedContent">

      <ul class="navbar-nav mr-auto">

        <li class="nav-item active">

          <a class="nav-link" href="#"><b>Accounts</b><span
class="sr-only">(current)</span></a>

        </li>

        <li class="nav-item">

          <a class="nav-link" href="{{ url_for('contactus')}}">Contact</a>

        </li>
```

```
      <li class="nav-item">
        <a class="nav-link" href="{{ url_for('transfer')}}">Transfer/Deposit</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Invesments</a>
      </li>
      <li class="nav-item">
        <a class="nav-link disabled" href="#">Security & Privacy</a>
      </li>
    </ul>
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
      <a  class="nav-link1" style="float: right;" href="{{ url_for('customerauth')}}">Logout</a>
    </li>
    </ul>
  </div>
</nav>

  <div id="blocks"style='text-align: center;'>
    <br>
    <br>
```

```
    <h1 >Your Accounts</h1><br>

    <br>

    <h2><a  href="http://localhost/acctransactiondisplay.php">Check your Transactions here!</a></h2>

    <br>

    <br><br>

   </div>


   <script>

    var flag=0;

    var jsondata = JSON.parse('{{ data | tojson | safe}}')

    console.log(jsondata.c)

    var blocks=document.getElementById('blocks').innerHTML;

    if (("s" in jsondata)==true){

     flag=1;

     var savingaccountnumber=String(jsondata.s.savingsaccount);

     var last_accessed_data=String(jsondata.s.last_accessed_data);

     var savings_interset_rate=String(jsondata.s.savings_interset_rate);

     var savingshtml=`

     <div class='row'>

      <div class='offset-2 col-8 '>
```

```
 <fieldset class="scheduler-border">


    <legend class="scheduler-border">Saving Account</legend>
    <div class="control-group">
    Account number :`+savingaccountnumber+`<br>
    Last accessed :`+last_accessed_data+`<br>
    Interset Rate:`+'0.01%'+`<br>
    </div>


 </fieldset>
 </div>
 </div>
  `;
  document.getElementById('blocks').innerHTML+=savingshtml;
}
if(("l" in jsondata)==true){
  flag=1;
  var loanaccount=String(jsondata.l.loanaccount);
  var amount=String(jsondata.l.amount);
  var repaymentamount=String(jsondata.l.repaymentamount);
  var interesetrate=String(jsondata.l.interesetrate);
```

```
 var account=String(jsondata.l.account);
 var loanhtml=`
 <div class='row'>
  <div class='offset-2 col-8'>
 <fieldset class="scheduler-border">

   <legend class="scheduler-border">Loan Account</legend>
   <div class="control-group">
   Loan Account number :`+loanaccount+`<br>
   Amount :`+amount+`<br>
   Repayment Amount :`+repaymentamount+`<br>
   Interset Rate:`+'9%'+`<br>
   Account :`+account+`<br>
   </div>

 </fieldset>
</div>
 </div>`;
 document.getElementById('blocks').innerHTML+=loanhtml;
}
if(("c" in jsondata)==true){
```

```
flag=1;
var checkingaccount=String(jsondata.c.checkingaccount);
var overdraftedamount=String(jsondata.c.overdraftedamount);
var overdraftedaccount=String(jsondata.c.overdraftedaccount);
var date=String(jsondata.c.date);
var checkinghtml=`
<div class='row'>
  <div class='offset-2 col-8'>
<fieldset class="scheduler-border">

   <legend class="scheduler-border">Checking Account</legend>
   <div class="control-group">
   Account number :`+checkingaccount+`<br>
   Overdrafted Amount :`+overdraftedamount+`<br>
   Overdrafted Account:`+overdraftedaccount+`<br>
   Date :`+date+`<br>
   </div>
 </fieldset>
</div>
 </div>`;
 document.getElementById('blocks').innerHTML+=checkinghtml;
```

```
}
if (("m" in jsondata)==true){
  flag=1;
    var marketaccount=String(jsondata.m.marketaccount);
    var updateddate=String(jsondata.m.updateddate);
    var marketinterestrate=String(jsondata.m.marketinterestrate);;
    var moneyhtml=`
    <div class='row'>
    <div class='offset-2 col-8'>
  <fieldset class="scheduler-border">

     <legend class="scheduler-border">Money Market Account</legend>
     <div class="control-group">
     Account number :`+marketaccount+`<br>
    Updated Date:`+updateddate+`<br>
     Interset Rate:`+'0.5%'+`<br>
     </div>
 </fieldset>
</div>
 </div>`;
 document.getElementById('blocks').innerHTML+=moneyhtml;
```

```
    }
    if(flag==0){
      document.getElementById('blocks').innerHTML="<h1>Oops!.. We couldn't find any accounts linked to your email id<h1>";
    }
    if (typeof browser === "undefined") {
   var browser = chrome;
}
   </script>
   </body>
   </html>
```