CIS344 Final Project Report: Bank’s Portal

Mst Tania

Due Date: 05-23-23

The MySQL Part:

Account Table: The accounts table takes in the accountsID, owner Name, Ownerssn, the balance and the account status of accounts. From the accounts table accountID is a primary key refer to the Transactions table of accountID is a foreign key.

Transactions Table: The Transactions table takes in the transactionID, accountID, transactionType, and transactionAmount. The transaction table transactionID is a primary key and accountId if foreign key refers to the account table of accountID.

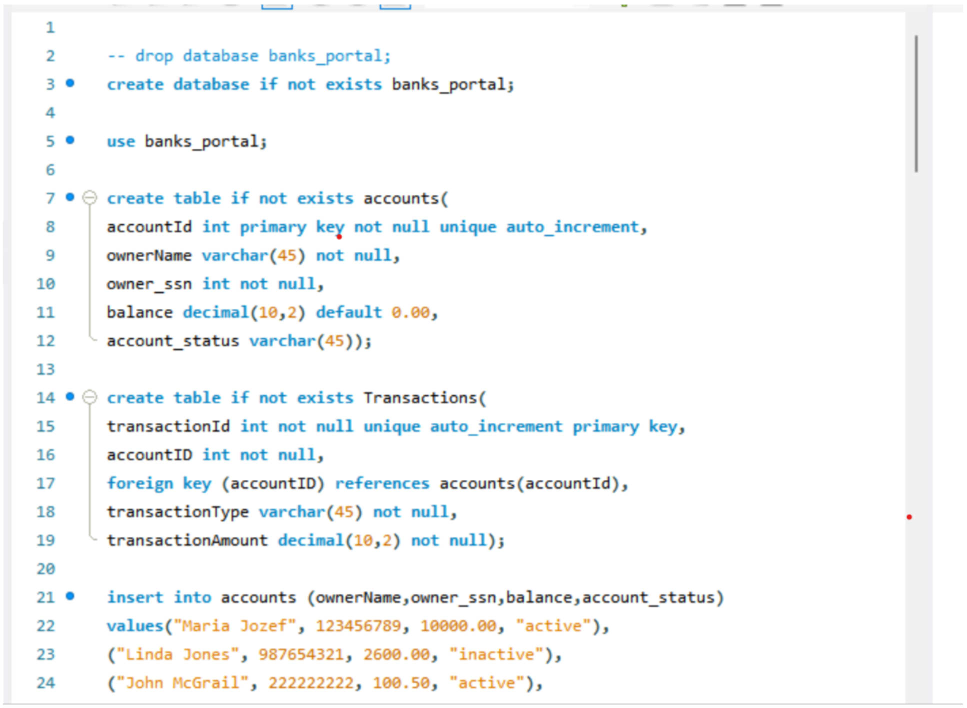
Values: Then the next account and transaction table, Inserting the values to it that was provided by Professor.

Deposit is a procedure that takes in the accountID, the amount of money to be deposited and returns a boolean value (returns 1 if the deposit went through or 0 if it failed). Inside the procedure, we first check that the account we are depositing money into is active. If it is active, then we insert a new transaction to the transactions table. After adding the transaction, we have to update the current balance for the account.

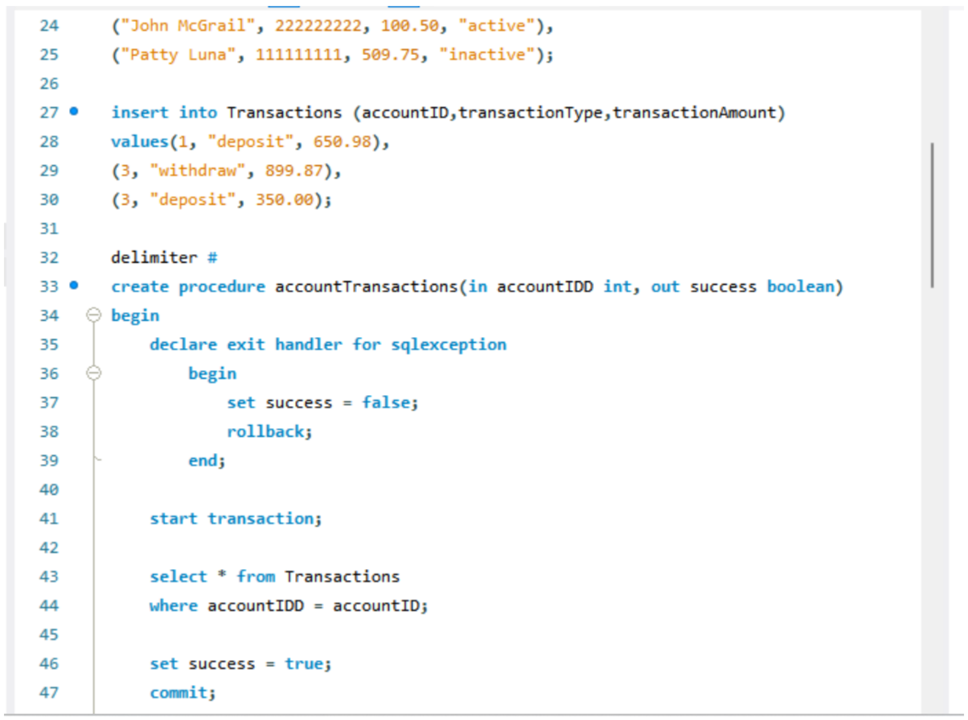
Withdraw does the same thing as the deposit procedure. Instead of adding money to the balance, it subtracts from it.

DeleteAccount is a procedure that takes in the accountID and returns a boolean value. It changes the account_status from "active" to "inactive" for the account that has the same accountId.

There are few Screenshots of the work Below:

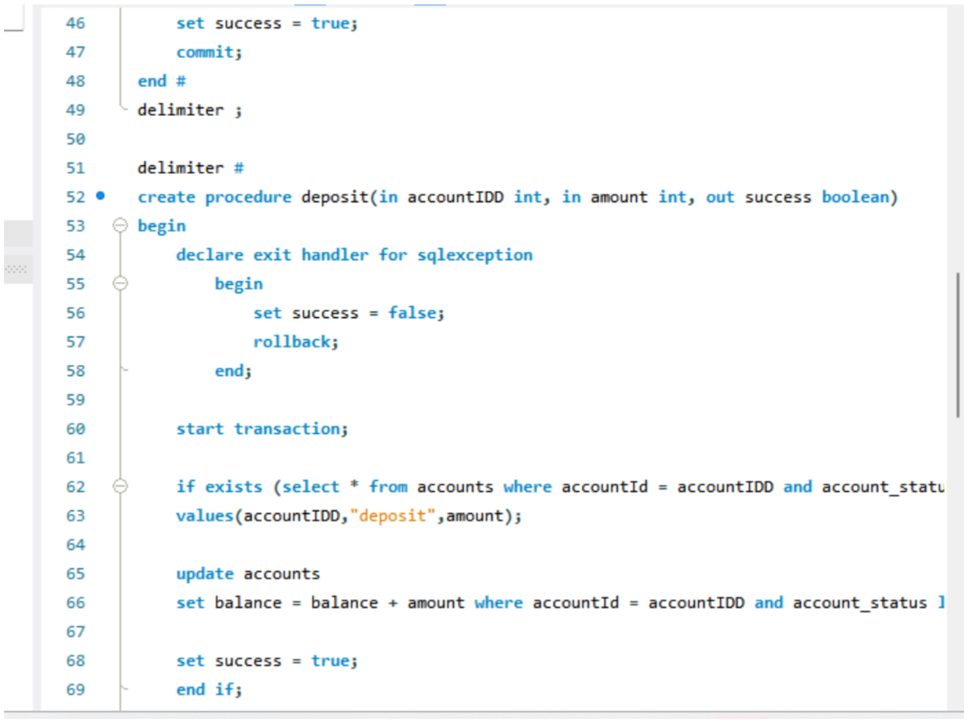


```sql

-- drop database banks_portal;
create database if not exists banks_portal;

use banks_portal;

create table if not exists accounts(
accountId int primary key not null unique auto_increment,
ownerName varchar(45) not null,
owner_ssn int not null,
balance decimal(10,2) default 0.00,
account_status varchar(45));

create table if not exists Transactions(
transactionId int not null unique auto_increment primary key,
accountID int not null,
foreign key (accountID) references accounts(accountId),
transactionType varchar(45) not null,
transactionAmount decimal(10,2) not null);

insert into accounts (ownerName,owner_ssn,balance,account_status)
values("Maria Jozef", 123456789, 10000.00, "active"),
("Linda Jones", 987654321, 2600.00, "inactive"),
("John McGrail", 222222222, 100.50, "active"),
```

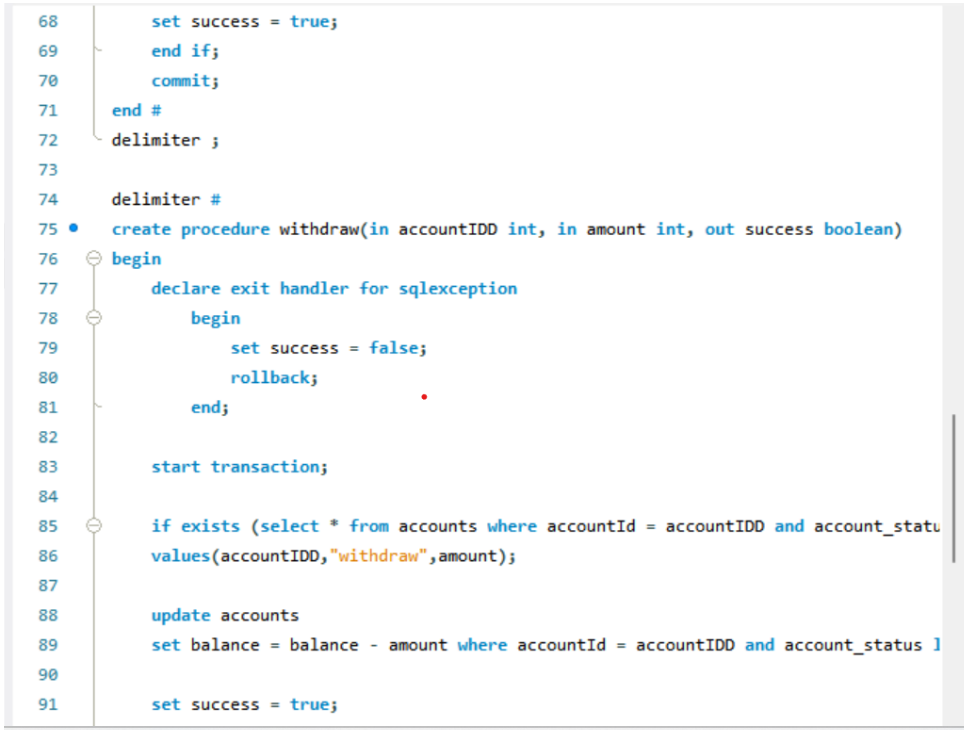


("John McGrail", 222222222, 100.50, "active"),

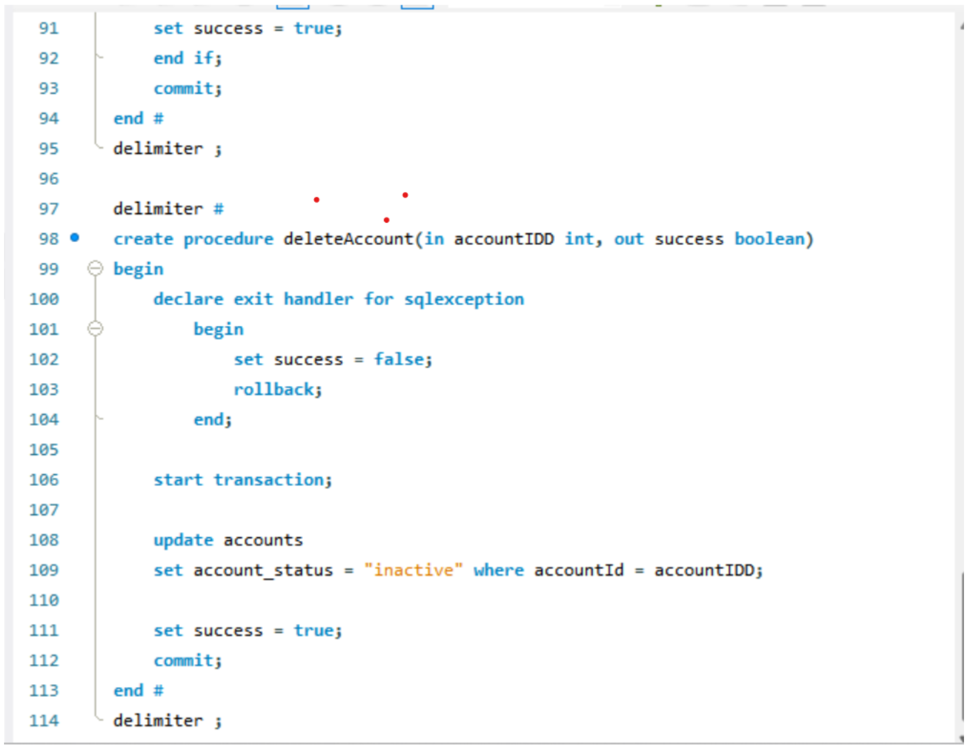
```sql
("Patty Luna", 111111111, 509.75, "inactive");

insert into Transactions (accountID,transactionType,transactionAmount)
values(1, "deposit", 650.98),
(3, "withdraw", 899.87),
(3, "deposit", 350.00);

delimiter #
create procedure accountTransactions(in accountIDD int, out success boolean)
begin
    declare exit handler for sqlexception
        begin
            set success = false;
            rollback;
        end;

    start transaction;

    select * from Transactions
    where accountIDD = accountID;

    set success = true;
    commit;
```



```sql
    set success = true;
    commit;
end #
delimiter ;

delimiter #
create procedure deposit(in accountIDD int, in amount int, out success boolean)
begin
    declare exit handler for sqlexception
        begin
            set success = false;
            rollback;
        end;

    start transaction;

    if exists (select * from accounts where accountId = accountIDD and account_statu
    values(accountIDD,"deposit",amount);

    update accounts
    set balance = balance + amount where accountId = accountIDD and account_status

    set success = true;
    end if;
```



```sql
    set success = true;
    end if;
    commit;
end #
delimiter ;

delimiter #
create procedure withdraw(in accountIDD int, in amount int, out success boolean)
begin
    declare exit handler for sqlexception
        begin
            set success = false;
            rollback;
        end;

    start transaction;

    if exists (select * from accounts where accountId = accountIDD and account_statu
    values(accountIDD,"withdraw",amount);

    update accounts
    set balance = balance - amount where accountId = accountIDD and account_status

    set success = true;
```



```
        set success = true;
        end if;
        commit;
    end #
    delimiter ;

    delimiter #
    create procedure deleteAccount(in accountIDD int, out success boolean)
    begin
        declare exit handler for sqlexception
            begin
                set success = false;
                rollback;
            end;

        start transaction;

        update accounts
        set account_status = "inactive" where accountId = accountIDD;

        set success = true;
        commit;
    end #
    delimiter ;
```

The Python Part:

The addAccount function takes in the owner name, the owner ssn, the balance of the account

and the status of the account. The cursor object is used to send queries to the database.

cursor.execute() takes in the query, and the values for the new account. self.connection.commit() commits the addition to the database, then both the cursor and the connection are closed.

The deposit function takes into account id we wish to deposit and the amount of money to be deposited. It is the same process as for the addAccount function. This time as the query, the procedure deposit is called.

The withdraw function works in the same way as the deposit function.

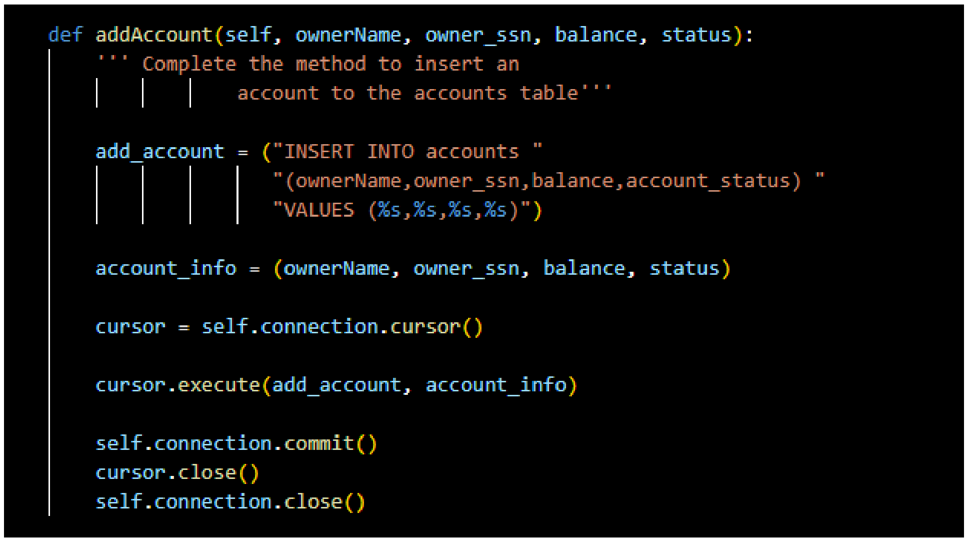
The account Transactions takes in an accountId. First, we check if we are connected to the database,

if we are, then we execute the query that selects all transactions which have the same accountId that is passed in. The fetch all function returns a list of all the values that match the accountId and then returns it.

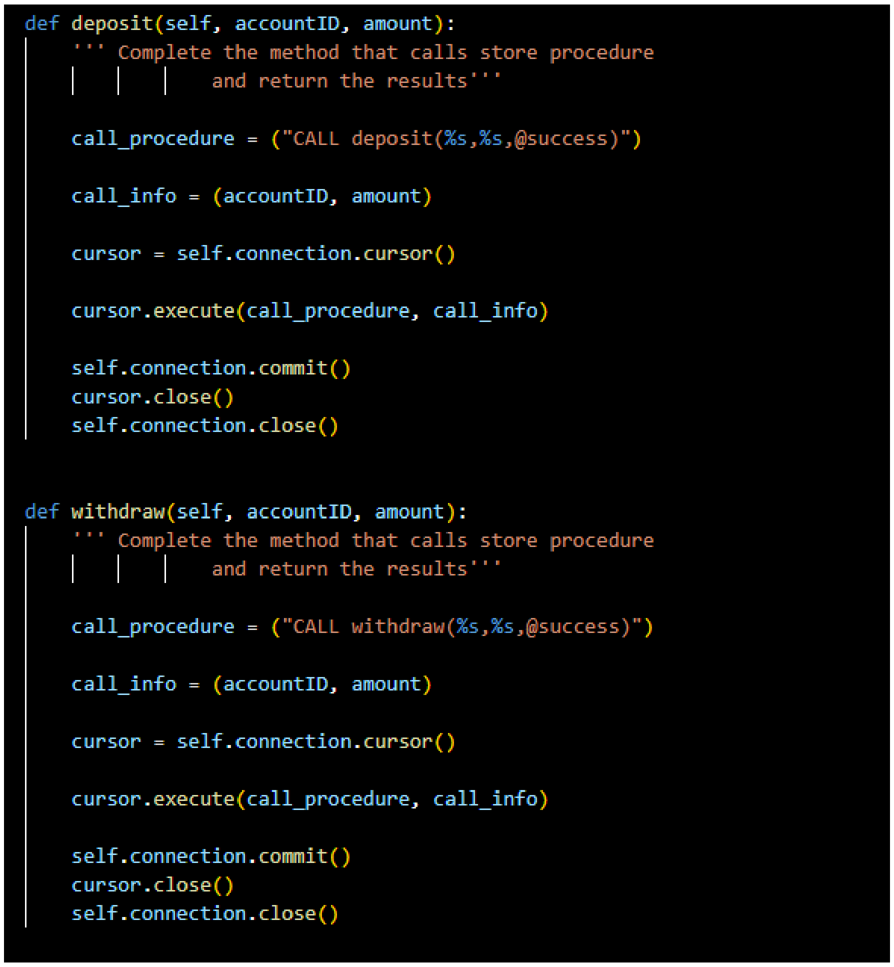
The delete Account function takes in the accountId of the account we wish to deactivate. The query is a call to the delete Account procedure which updates the status of the account to be equals to "inactive".

The getAllTransactions functions first checks if we are connected to the database, if we are, then it selects all values from the transactions table and returns them as a list.

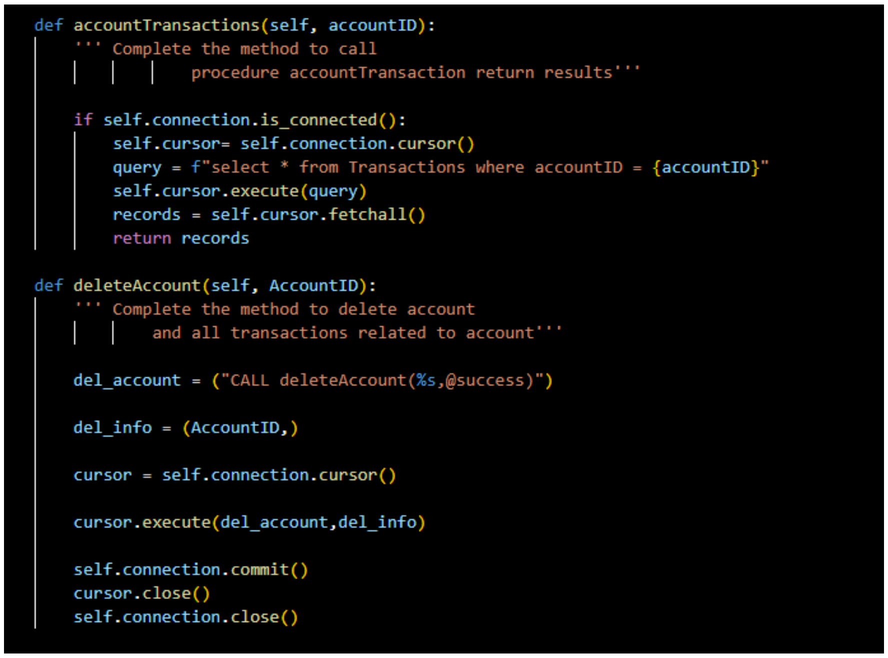
There are few screenshots of the work below:



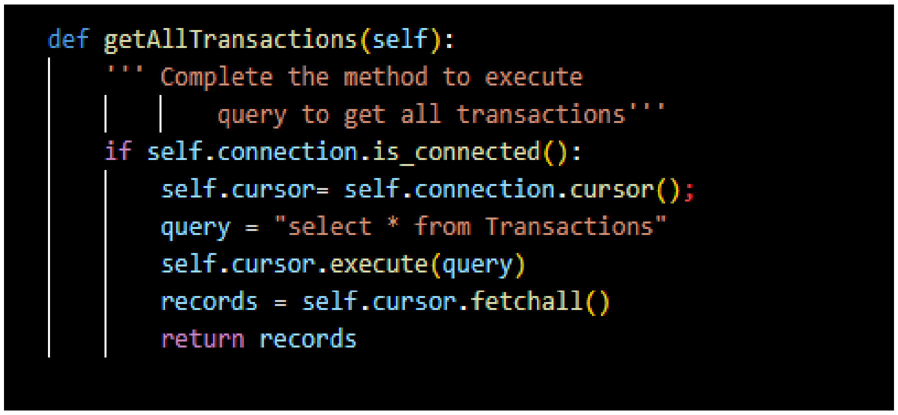
```
def addAccount(self, ownerName, owner_ssn, balance, status):
    ''' Complete the method to insert an
                account to the accounts table'''

    add_account = ("INSERT INTO accounts "
                   "(ownerName,owner_ssn,balance,account_status) "
                   "VALUES (%s,%s,%s,%s)")

    account_info = (ownerName, owner_ssn, balance, status)

    cursor = self.connection.cursor()

    cursor.execute(add_account, account_info)

    self.connection.commit()
    cursor.close()
    self.connection.close()
```



```python
def deposit(self, accountID, amount):
    ''' Complete the method that calls store procedure
                and return the results'''

    call_procedure = ("CALL deposit(%s,%s,@success)")

    call_info = (accountID, amount)

    cursor = self.connection.cursor()

    cursor.execute(call_procedure, call_info)

    self.connection.commit()
    cursor.close()
    self.connection.close()


def withdraw(self, accountID, amount):
    ''' Complete the method that calls store procedure
                and return the results'''

    call_procedure = ("CALL withdraw(%s,%s,@success)")

    call_info = (accountID, amount)

    cursor = self.connection.cursor()

    cursor.execute(call_procedure, call_info)

    self.connection.commit()
    cursor.close()
    self.connection.close()
```

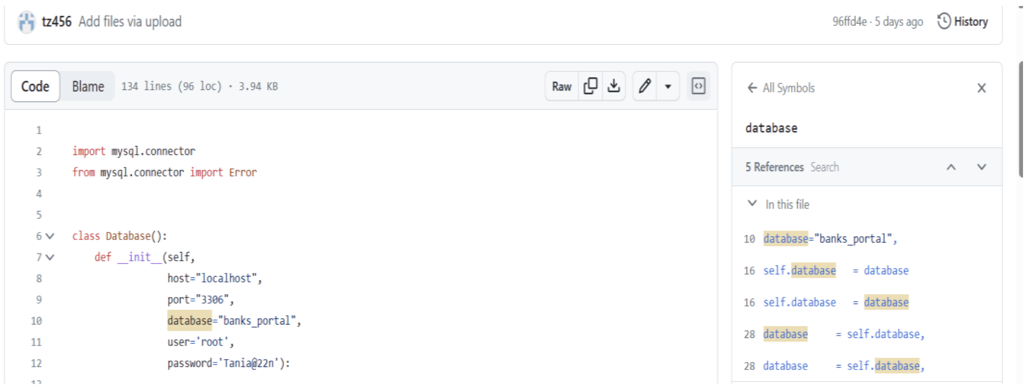


```python
def accountTransactions(self, accountID):
    ''' Complete the method to call
                procedure accountTransaction return results'''

    if self.connection.is_connected():
        self.cursor= self.connection.cursor()
        query = f"select * from Transactions where accountID = {accountID}"
        self.cursor.execute(query)
        records = self.cursor.fetchall()
        return records

def deleteAccount(self, AccountID):
    ''' Complete the method to delete account
            and all transactions related to account'''

    del_account = ("CALL deleteAccount(%s,@success)")

    del_info = (AccountID,)

    cursor = self.connection.cursor()

    cursor.execute(del_account,del_info)

    self.connection.commit()
    cursor.close()
    self.connection.close()
```



```python
def getAllTransactions(self):
    ''' Complete the method to execute
        query to get all transactions'''
    if self.connection.is_connected():
        self.cursor= self.connection.cursor();
        query = "select * from Transactions"
        self.cursor.execute(query)
        records = self.cursor.fetchall()
        return records
```

Portal Database: Change the default parameter values to match my MySQL server



tz456 Add files via upload 96ffd4e · 5 days ago History

Code Blame 134 lines (96 loc) · 3.94 KB Raw

```python

import mysql.connector
from mysql.connector import Error


class Database():
    def __init__(self,
                host="localhost",
                port="3306",
                database="banks_portal",
                user='root',
                password='Tania@22n'):
```

All Symbols

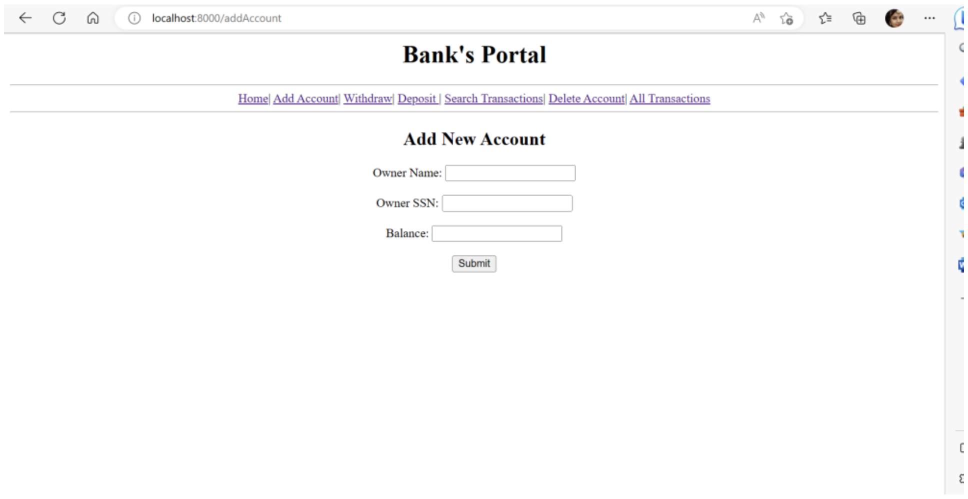
database

5 References Search

In this file

```
10 database="banks_portal",
16 self.database   = database
16 self.database   = database
28 database      = self.database,
28 database      = self.database,
```

Bank Portal: After running the portal server. Py on Command Prompt, it was starting httpd on port 8000 (localhost:8000/). Then, I went to the web browser to see the account I added that was working perfectly. There are few screenshots below from the work:


localhost:8000/addAccount

Bank's Portal

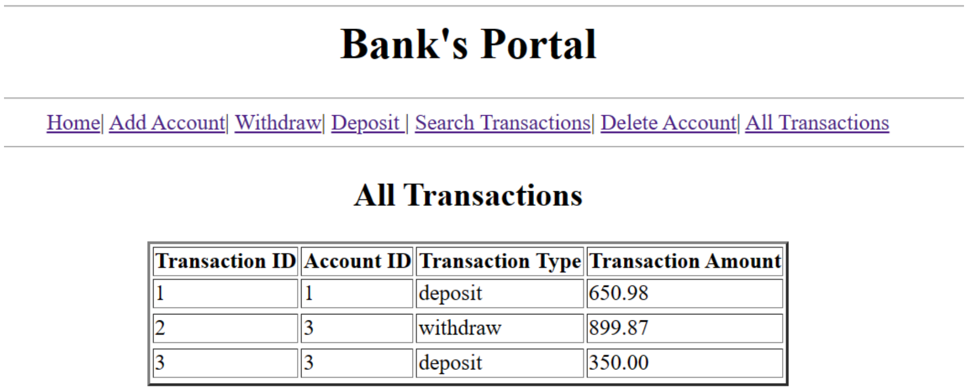
Home| Add Account| Withdraw| Deposit | Search Transactions| Delete Account| All Transactions

Add New Account

Owner Name:

Owner SSN:

Balance:


Bank's Portal

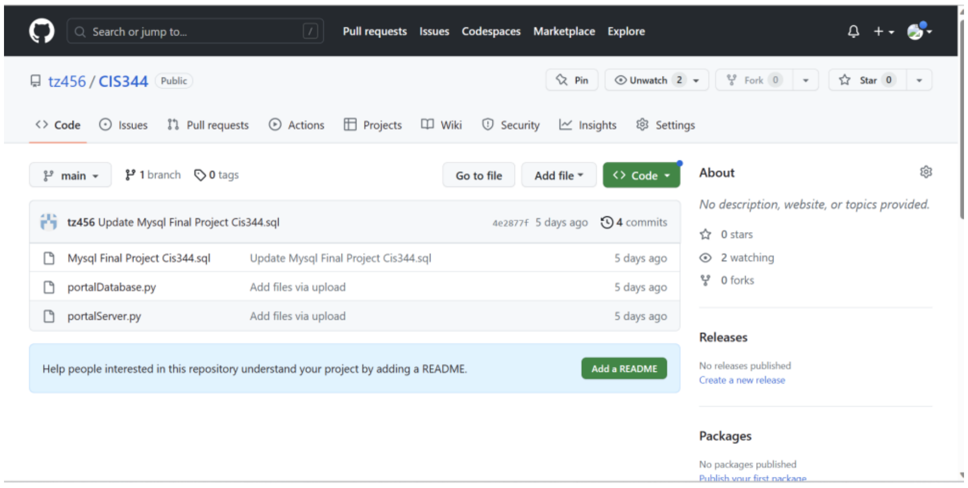
Home| Add Account| Withdraw| Deposit | Search Transactions| Delete Account| All Transactions

All Transactions

Transaction ID	Account ID	Transaction Type	Transaction Amount
1	1	deposit	650.98
2	3	withdraw	899.87
3	3	deposit	350.00

Link For GitHub in a public repository:

https://github.com/tz456/CIS344.git



tz456 Update Mysql Final Project Cis344.sql 4e2877f 5 days ago 4 commits

Mysql Final Project Cis344.sql	Update Mysql Final Project Cis344.sql	5 days ago
portalDatabase.py	Add files via upload	5 days ago
portalServer.py	Add files via upload	5 days ago

Help people interested in this repository understand your project by adding a README.

About

No description, website, or topics provided.

0 stars

2 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published