Console Application – ATM

Program.cs

using atm.DAL;

using Microsoft.Extensions.Configuration;

namespace atm

{

class Program

{

private static IConfiguration \_iconfiguration;

static void Main(string[] args)

{

char ch = 'n';

GetAppSettingsFile();

Console.WriteLine("WELCOME TO INDIAN BANK ATM SERVICE\n");

do

{

PrintOptions();

int opt = Convert.ToInt32(Console.ReadLine());

switch (opt)

{

case 1: CheckBalance(); break;

case 2: WithdrawMoney(); break;

case 3: DepositMoney(); break;

case 4: AmountTransfer(); break;

case 5: ExitOperation(); break;

default: Console.WriteLine("No Such Option Available.."); break;

}

Console.WriteLine("Do You Want to Continue Banking..");

ch = Convert.ToChar(Console.ReadLine());

}

while (ch == 'y');

Console.WriteLine("\n\nTHANKS FOR USING YES ATM SERVICE");

}

static void GetAppSettingsFile()

{

var builder = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true);

\_iconfiguration = builder.Build();

}

static void PrintOptions()

{

Console.WriteLine("1. Check Balance \n");

Console.WriteLine("2. Withdraw Money \n");

Console.WriteLine("3. Deposit Money \n");

Console.WriteLine("4. Transfer to another account \n");

Console.WriteLine("5. Cancel\n");

Console.WriteLine("ENTER YOUR CHOICE : ");

}

static void CheckBalance()

{

Console.WriteLine("\nEnter Account Number \n");

long accno = Convert.ToInt64(Console.ReadLine());

Console.WriteLine("Enter CardPin \n");

int pin = Convert.ToInt32(Console.ReadLine());

var BalanceDAL = new CountryDAL(\_iconfiguration);

var UserBalancelist = BalanceDAL.GetBalance(accno, pin);

if (UserBalancelist.Count == 0)

{

Console.WriteLine("Enter Correct Credentials");

return;

}

Console.WriteLine("Your Balance is:" + UserBalancelist[0].TotalBalance);

}

static void WithdrawMoney()

{

Console.WriteLine("Enter Account Number \n");

long accno = Convert.ToInt64(Console.ReadLine());

Console.WriteLine("Enter CardPin \n");

int pin = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Amount to Withdraw \n");

int amt = Convert.ToInt32(Console.ReadLine());

var BalanceDAL = new CountryDAL(\_iconfiguration);

var UserBalancelist = BalanceDAL.GetBalance(accno, pin);

if (UserBalancelist.Count == 0)

{

Console.WriteLine("Enter Correct Credentials");

return;

}

else if (UserBalancelist[0].TotalBalance < amt)

{

Console.WriteLine($"Cannot Withdraw Rs {amt} your account have Rs {UserBalancelist[0].TotalBalance}");

return;

}

BalanceDAL.SetWithdraw(accno, amt);

Console.WriteLine("Your Balance is Updated:");

Console.WriteLine("Your Balance is:" + (UserBalancelist[0].TotalBalance - amt));

}

static void DepositMoney()

{

Console.WriteLine("Enter Account Number \n");

long accno = Convert.ToInt64(Console.ReadLine());

Console.WriteLine("Enter CardPin \n");

int pin = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Amount to Deposit \n");

int amt = Convert.ToInt32(Console.ReadLine());

var BalanceDAL = new CountryDAL(\_iconfiguration);

var UserBalancelist = BalanceDAL.GetBalance(accno, pin);

if (UserBalancelist.Count == 0)

{

Console.WriteLine("Enter Correct Credentials");

return;

}

BalanceDAL.SetDeposit(accno, amt);

Console.WriteLine("Your Balance is Updated:");

Console.WriteLine("Your Balance is:" + (UserBalancelist[0].TotalBalance + amt));

}

static void AmountTransfer()

{

Console.WriteLine("Enter Your Account Number \n");

long accno = Convert.ToInt64(Console.ReadLine());

Console.WriteLine("Enter Receipent Account Number \n");

long Toaccno = Convert.ToInt64(Console.ReadLine());

Console.WriteLine("Enter Receipent Branch Name \n");

string branch = Console.ReadLine();

Console.WriteLine("Enter Amount to Transfer \n");

int amt = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Your Pin \n");

int pin = Convert.ToInt32(Console.ReadLine());

var BalanceDAL = new CountryDAL(\_iconfiguration);

var UserBalancelist = BalanceDAL.GetBalance(accno, pin);

if (UserBalancelist.Count == 0)

{

Console.WriteLine("Enter Correct Credentials");

return;

}

else if (UserBalancelist[0].TotalBalance < amt)

{

Console.WriteLine($"Cannot Transfer Rs {amt} your account have Rs {UserBalancelist[0].TotalBalance}");

return;

}

BalanceDAL.SetTransaction(accno, Toaccno, branch, amt);

Console.WriteLine($"Rs {amt} is Successfully Transferred to {Toaccno} Receipent. ");

BalanceDAL.SetWithdraw(accno, amt);

BalanceDAL.SetDeposit(Toaccno, amt);

Console.WriteLine("Your Balance is:" + (UserBalancelist[0].TotalBalance - amt));

}

static void ExitOperation()

{

return;

}

}

}

Model.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

namespace atm.Model

{

public class UserModel

{

public int Id { get; set; }

public long AccountNumber { get; set; }

public string FullName { get; set; }

public string BranchName { get; set; }

public float TotalBalance { get; set; }

public int CardPin { get; set; }

}

}

Dal.cs

using atm.Model;

using Microsoft.Extensions.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace atm.DAL

{

public class CountryDAL

{

private string \_connectionString;

public CountryDAL(IConfiguration iconfiguration)

{

\_connectionString = iconfiguration.GetConnectionString("Default");

}

public List<UserModel> GetBalance(long Accno, int Pin)

{

var UserModellist = new List<UserModel>();

try

{

using (SqlConnection con = new SqlConnection(\_connectionString))

{

SqlCommand cmd = new SqlCommand("CheckBalance", con);

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter

{

ParameterName = "@AccountNumber",

SqlDbType = SqlDbType.BigInt,

Value = Accno,

Direction = ParameterDirection.Input,

};

SqlParameter param2 = new SqlParameter

{

ParameterName = "@CardPin",

SqlDbType = SqlDbType.Int,

Value = Pin,

Direction = ParameterDirection.Input,

};

cmd.Parameters.Add(param1);

cmd.Parameters.Add(param2);

con.Open();

SqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

UserModellist.Add(new UserModel

{

TotalBalance = Convert.ToInt32(rdr[0]),

});

}

}

}

catch (Exception ex)

{

throw ex;

}

return UserModellist;

}

public void SetDeposit(long Accno, float amt)

{

try

{

using (SqlConnection con = new SqlConnection(\_connectionString))

{

SqlCommand cmd = new SqlCommand("DepositMoney", con);

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter

{

ParameterName = "@AccountNumber",

SqlDbType = SqlDbType.BigInt,

Value = Accno,

Direction = ParameterDirection.Input,

};

SqlParameter param2 = new SqlParameter

{

ParameterName = "@Amount",

SqlDbType = SqlDbType.Float,

Value = amt,

Direction = ParameterDirection.Input,

};

cmd.Parameters.Add(param1);

cmd.Parameters.Add(param2);

con.Open();

cmd.ExecuteReader();

}

}

catch (Exception ex)

{

throw ex;

}

}

public void SetWithdraw(long Accno, float amt)

{

try

{

using (SqlConnection con = new SqlConnection(\_connectionString))

{

SqlCommand cmd = new SqlCommand("WithdrawMoney", con);

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter

{

ParameterName = "@AccountNumber",

SqlDbType = SqlDbType.BigInt,

Value = Accno,

Direction = ParameterDirection.Input,

};

SqlParameter param2 = new SqlParameter

{

ParameterName = "@Amount",

SqlDbType = SqlDbType.Float,

Value = amt,

Direction = ParameterDirection.Input,

};

cmd.Parameters.Add(param1);

cmd.Parameters.Add(param2);

con.Open();

cmd.ExecuteReader();

}

}

catch (Exception ex)

{

throw ex;

}

}

public void SetTransaction(long Accno, long ToAccno, string branch, float amt)

{

try

{

using (SqlConnection con = new SqlConnection(\_connectionString))

{

SqlCommand cmd = new SqlCommand("NewTransactions", con);

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter param1 = new SqlParameter

{

ParameterName = "@AccountNumber",

SqlDbType = SqlDbType.BigInt,

Value = Accno,

Direction = ParameterDirection.Input,

};

SqlParameter param2 = new SqlParameter

{

ParameterName = "@TransferAccount",

SqlDbType = SqlDbType.BigInt,

Value = ToAccno,

Direction = ParameterDirection.Input,

};

SqlParameter param3 = new SqlParameter

{

ParameterName = "@BranchName",

SqlDbType = SqlDbType.VarChar,

Value = branch,

Direction = ParameterDirection.Input,

};

SqlParameter param4 = new SqlParameter

{

ParameterName = "@AmountToTransfer",

SqlDbType = SqlDbType.Float,

Value = amt,

Direction = ParameterDirection.Input,

};

cmd.Parameters.Add(param1);

cmd.Parameters.Add(param2);

cmd.Parameters.Add(param3);

cmd.Parameters.Add(param4);

con.Open();

cmd.ExecuteReader();

}

}

catch (Exception ex)

{

throw ex;

}

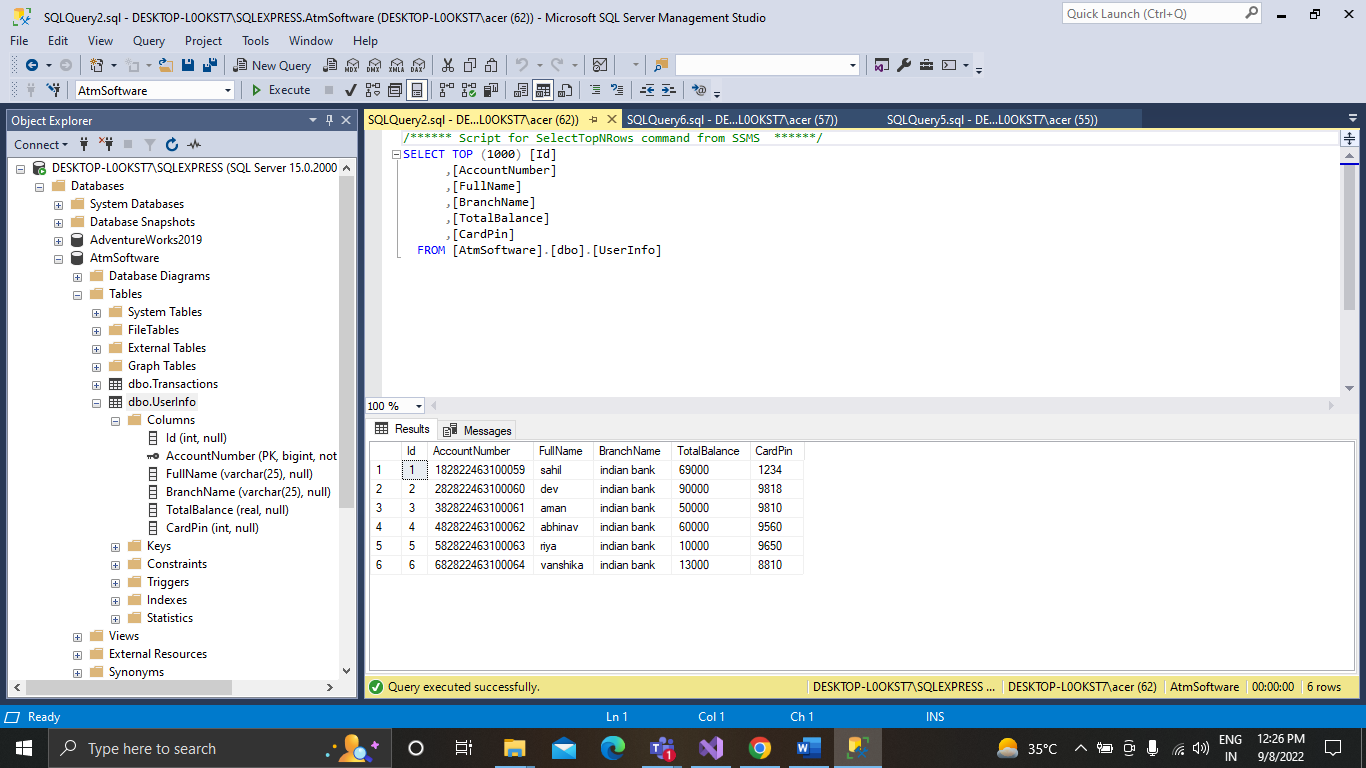
}

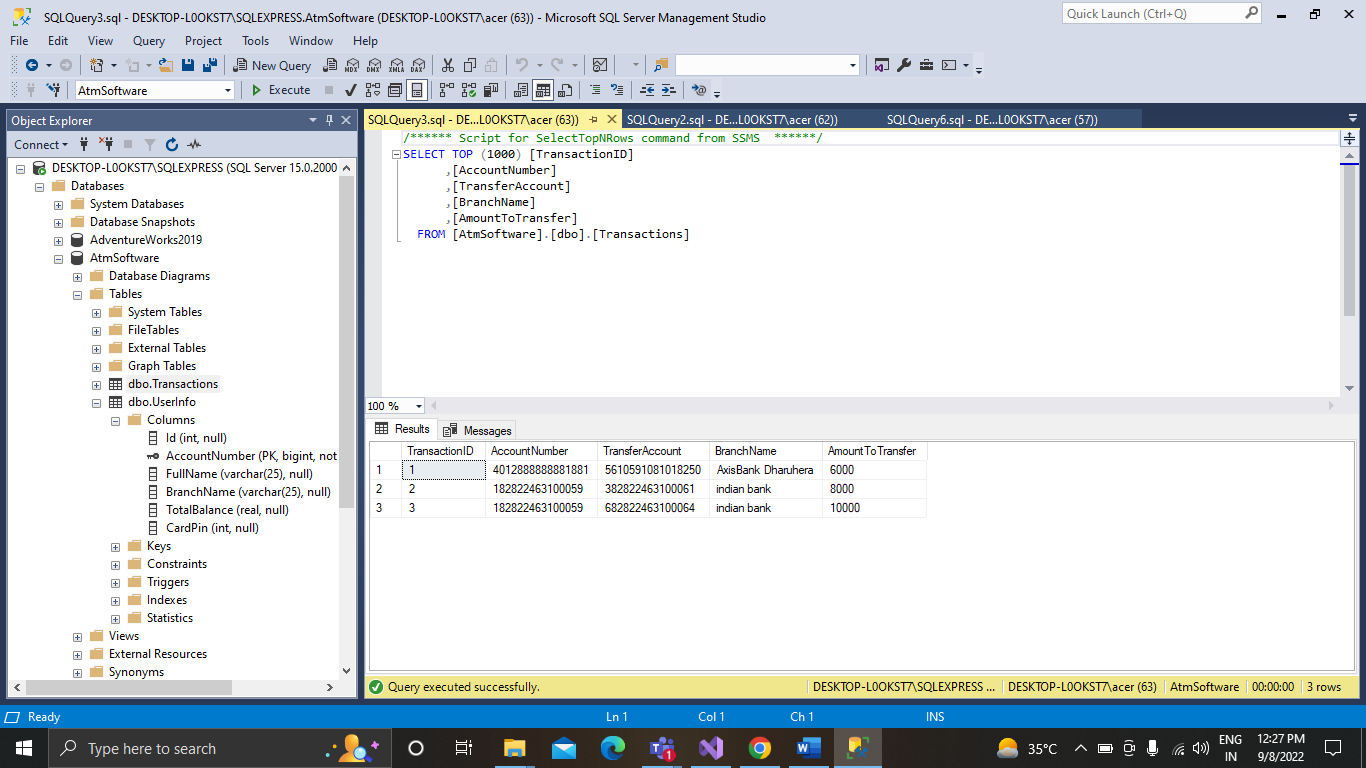
}

}

SQL

Tables-





Procedures-

CREATE PROCEDURE CheckBalance

(@AccountNumber BigInt, @CardPin int)

AS Begin

Select TotalBalance FROM UserInfo WHERE AccountNumber = @AccountNumber AND CardPin = @CardPin

END

EXECUTE CheckBalance @AccountNumber = 182822463100059, @CardPin = 1234

CREATE PROCEDURE DepositMoney(@AccountNumber BigInt, @Amount float)

AS Begin

Update UserInfo SET TotalBalance = TotalBalance + @Amount

WHERE AccountNumber = @AccountNumber

END

CREATE PROCEDURE WithdrawMoney(@AccountNumber BigInt, @Amount float)

AS Begin

Update UserInfo SET TotalBalance = TotalBalance - @Amount

WHERE AccountNumber = @AccountNumber

END

INSERT INTO Transactions(AccountNumber,TransferAccount,BranchName,AmountToTransfer)

VALUES (4012888888881881, 5610591081018250, ‘AxisBank Dharuhera’, 6000)

CREATE PROCEDURE NewTransactions

(@AccountNumber BigInt,

@TransferAccount BigInt,

@BranchName varchar(25),

@AmountToTransfer float

)

AS Begin

INSERT INTO Transactions(AccountNumber,TransferAccount,BranchName,AmountToTransfer)

VALUES (@AccountNumber,@TransferAccount,@BranchName,@AmountToTransfer);

END

EXECUTE NewTransactions @AccountNumber=182822463100059,@TransferAccount=382822463100061,@BranchName=’indian bank’,@AmountToTransfer=8000

OUTPUT

