**PIMPRI CHINCHWAD EDUCATION TRUST'S**

**PIMPRI CHINCHWAD COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER ENGINEERING**



**SUBJECT: ADVANCED DATA STRUCTURES LAB**

**Under the Guidance of : Kapil Tajane Sir & Madhura madam**

**Group Members:**

**1]Gaurav Arvind Borse.(SECOC308)**

**2]Manish Sakharam Choudhary.(SECOC310)**

**3]Bhavit Pramod Dagade.(SECOC311)**

**4]Tanishq Manoj Dhussa.(SECOC314)**

**5]Gopal Kabra.(SECOC326)**

# AIM

To create a program that can perform bank operations.

# OBJECTIVE

1]To create application program.

2]To create a GUI application.

# PROBLEM

Create a program that can perform certain operation that are be performed in bank such as Create account, Display account balance, Deposit money in account and Withdraw money from account.

# SOLUTION

* As bank does not contain limit to number of accounts so we will have to use concept of dynamic memory allocation.
* For an account holder the main part is his account number and pin so we will use private access specifier for data members in class.
* In an bank the account number defers by an one number than previous account number but the pin number cannot defer by one account privacy issue so we will use an srand() function to generate an different random pin number.

# ALGORITHM

1]Create a structure named as member.

2]Structure member contains various attributes related to bank account holder.

3]Create a pointer of member (user defined data type) in class named as Bank.

4]Create a header of type member in class.

5]Create various methods (member functions) to operate on data member.

6]Use a linked list concept for dynamic memory allocation.

# GUI based code

#include "bankoperation.h"

#include "ui\_bankoperation.h"

#include <cstdlib>

#include <iostream>

#include <cmath>

#include <string>

#include <QString>

#include <ctime>

#include "QMessageBox"

#include "QPixmap"

#include <QHash>

using namespace std;

long long int AccountNumber;

int accountPinNumber;

string accountName;

string accountType;

double accountBalance;

int flag=0;

int flag1=0;

struct Member

{

string name;

long long int accountNumber;

string accountType;

double balance;

int pin;

Member \* next;

};

class Bank

{

private:

QHash <int,Member \* > myHash;

public:

void **setDetails**(string str1 , string str2);

void **create**(string str1 , string str2);

void **display**(long long int accNumber, int accPinNumber);

void **deposit**(long long int accNum, double amount);

void **withdraw**(long long int accNumber, int accPinNumber , double amount);

static int c;

};

int Bank::c;

Bank acc;

void Bank::**setDetails**(string str1 , string str2) //function to accept details

{

Member \* newMember;

newMember->name = str1;

newMember->accountType = str2;

newMember->accountNumber=10000+c;

srand(time(0));

newMember->pin=1000+rand()%8999;

AccountNumber = newMember->accountNumber;

accountPinNumber = newMember->pin;

accountName=newMember->name;

accountType=newMember->accountType;

newMember->balance=0;

newMember->next=NULL;

c++;

}

void Bank::**create**(string str1 , string str2)

{

Member \* newMember=new Member;

setDetails(str1 , str2);

myHash.insert(newMember->accountNumber,newMember);

}

void Bank::**display**(long long accNumber ,int accPinNumber) //function to display details

{

flag=0;

Member \* aMember = NULL;

aMember = myHash.value(accNumber);

if(aMember){

if(aMember->pin == accPinNumber){

accountName = aMember->name;

accountBalance = aMember->balance;

flag=1;

}

}

}

void Bank::**deposit**(long long int accNumber, double amount) //function to deposit an money

{

flag=0;

Member \* aMember = NULL;

aMember = myHash.value(accNumber);

if(aMember){

aMember->balance = aMember->balance + amount;

flag=1;

}

}

void Bank::**withdraw**(long long int accNumber, int accPinNumber , double amount) //function to withdraw an money

{

flag=0;

flag1=0;

Member \* aMember = NULL;

aMember = myHash.value(accNumber);

if(aMember){

if(aMember->accountNumber==accNumber && aMember->pin==accPinNumber)

{

flag=1;

{

if(amount > aMember->balance )

{

flag1=1;

}

else

aMember->balance = aMember->balance - amount;

}

}

}

}

BankOperation::**BankOperation**(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::BankOperation)

{

ui->setupUi(this);

QPixmap pix (":/resource/img/a.jpg");

int w = ui->label\_pic->width();

int h = ui->label\_pic->height();

ui->label\_pic->setPixmap(pix.scaled(w,h,Qt::KeepAspectRatio));

}

BankOperation::~***BankOperation***()

{

delete ui;

}

void BankOperation::**on\_pushButton\_clicked**() // Account Creation

{

QString name = ui->lineEdit->text();

string accName = name.toStdString();

QString type = ui->lineEdit\_2->text();

string accType = type.toStdString();

acc.Bank::create(accName , accType);

long long int accNum = AccountNumber;

int accPin = accountPinNumber;

string accname = accountName;

string acctype = accountType;

QString AccNum;

AccNum.setNum(accNum);

QString AccPin;

AccPin.setNum(accPin);

QString Accname = QString::fromStdString(accname);

QString Acctype = QString::fromStdString(acctype);

ui->label\_25->setText(Accname);

ui->label\_26->setText(Acctype);

ui->label\_7->setText(AccNum);

ui->label\_8->setText(AccPin);

ui->lineEdit->clear();

ui->lineEdit\_2->clear();

}

void BankOperation::**on\_pushButton\_2\_clicked**() // Display Button

{

bool ok;

QString accnum = ui->lineEdit\_3->text();

long long int accNum = accnum.toInt(&ok,10);

QString accpin = ui->lineEdit\_4->text();

int accPin = accpin.toInt(&ok,10);

acc.Bank::display(accNum,accPin);

string accname = accountName;

double accbalance = accountBalance;

QString AccName = QString::fromStdString(accname);

QString AccBalance;

AccBalance.setNum(accbalance);

if(flag==0)

{

QMessageBox::StandardButton reply = QMessageBox::warning(this,"Warning","PLEASE CHECK ACCOUNT NUMBER AND PIN",QMessageBox::Yes);

if(reply==QMessageBox::Yes)

{

AccName.clear();

AccBalance.clear();

ui->label\_14->setText(AccName);

ui->label\_15->setText(AccBalance);

ui->lineEdit\_3->clear();

ui->lineEdit\_4->clear();

}

}

else

{

ui->label\_14->setText(AccName);

ui->label\_15->setText(AccBalance);

}

accountName.clear();

accountBalance=0;

ui->lineEdit\_3->clear();

ui->lineEdit\_4->clear();

}

void BankOperation::**on\_pushButton\_3\_clicked**() //Deposit Button

{

bool ok;

QString accnum = ui->lineEdit\_5->text();

QString amount = ui->lineEdit\_6->text();

long long int accNum = accnum.toInt(&ok,10);

double Amount = amount.toInt(&ok,10);

acc.Bank::deposit(accNum,Amount);

if(flag==0)

{

QMessageBox::StandardButton reply = QMessageBox::warning(this,"Warning","PLEASE CHECK ACCOUNT NUMBER",QMessageBox::Yes);

}

ui->lineEdit\_5->clear();

ui->lineEdit\_6->clear();

}

void BankOperation::**on\_pushButton\_4\_clicked**() //Withdraw Button

{

bool ok;

QString accnum = ui->lineEdit\_7->text();

QString accpin = ui->lineEdit\_8->text();

QString withdrawamount = ui->lineEdit\_9->text();

long long int accNum = accnum.toInt(&ok,10);

int accPin = accpin.toInt(&ok,10);

double withdrawAmount = withdrawamount.toInt(&ok,10);

acc.Bank::withdraw(accNum , accPin , withdrawAmount);

if(flag==0)

{

QMessageBox::StandardButton reply = QMessageBox::warning(this,"Warning","PLEASE CHECK ACCOUNT NUMBER AND PIN",QMessageBox::Yes);

if(reply==QMessageBox::Yes)

{

ui->lineEdit\_7->clear();

ui->lineEdit\_8->clear();

ui->lineEdit\_9->clear();

}

}

if(flag1==1)

{

QMessageBox::StandardButton reply = QMessageBox::warning(this,"Warning","INSUFFICIENT BALANCE!!!",QMessageBox::Yes);

if(reply==QMessageBox::Yes)

{

ui->lineEdit\_7->clear();

ui->lineEdit\_8->clear();

ui->lineEdit\_9->clear();

}

}

ui->lineEdit\_7->clear();

ui->lineEdit\_8->clear();

ui->lineEdit\_9->clear();

if(flag==1&&flag1==0)

{

ui->label\_27->setText("Amount Successfully Withdrawn!!!");

}

else

ui->label\_27->setText("Transaction failed!!!");

}

void BankOperation::**on\_pushButton\_5\_clicked**()

{

ui->label\_25->clear();

ui->label\_26->clear();

ui->label\_7->clear();

ui->label\_8->clear();

}

void BankOperation::**on\_pushButton\_6\_clicked**()

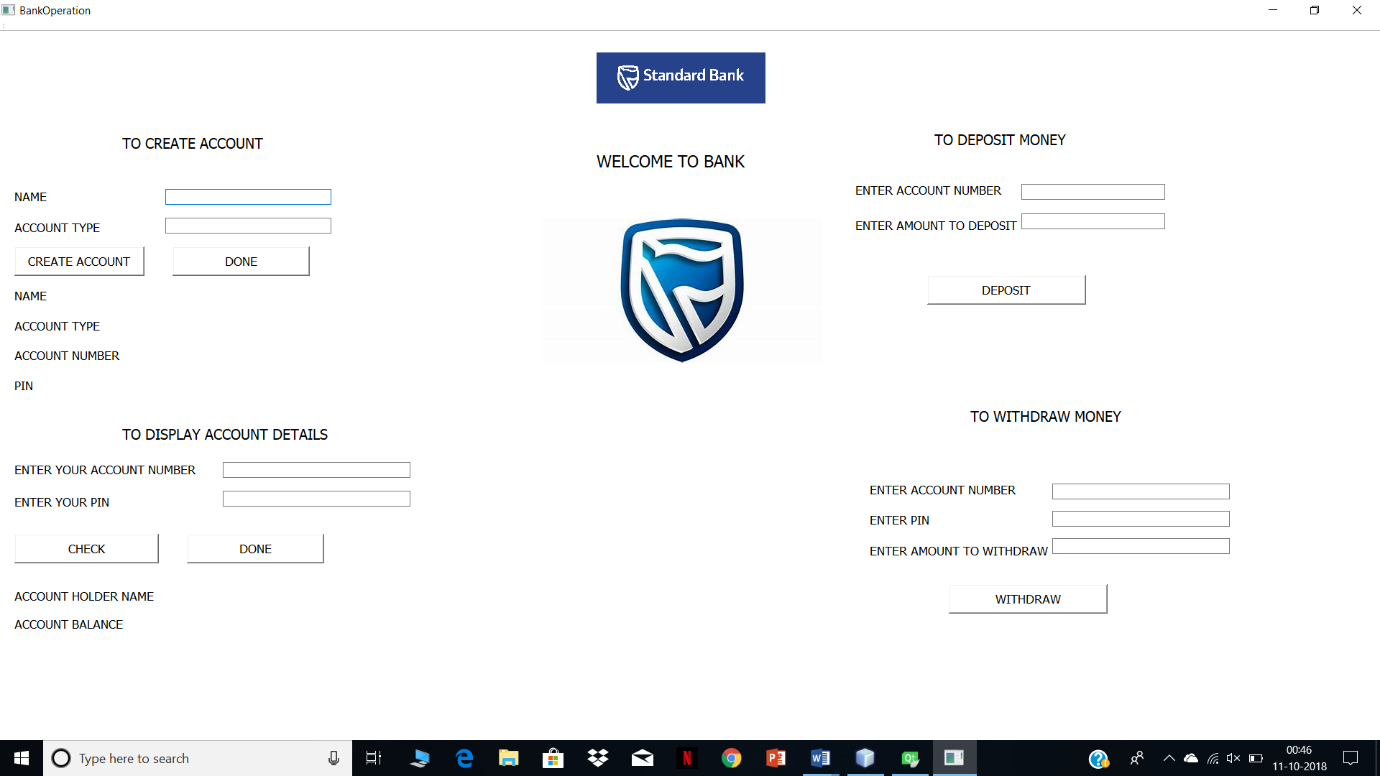
{

ui->label\_14->clear();

ui->label\_15->clear();

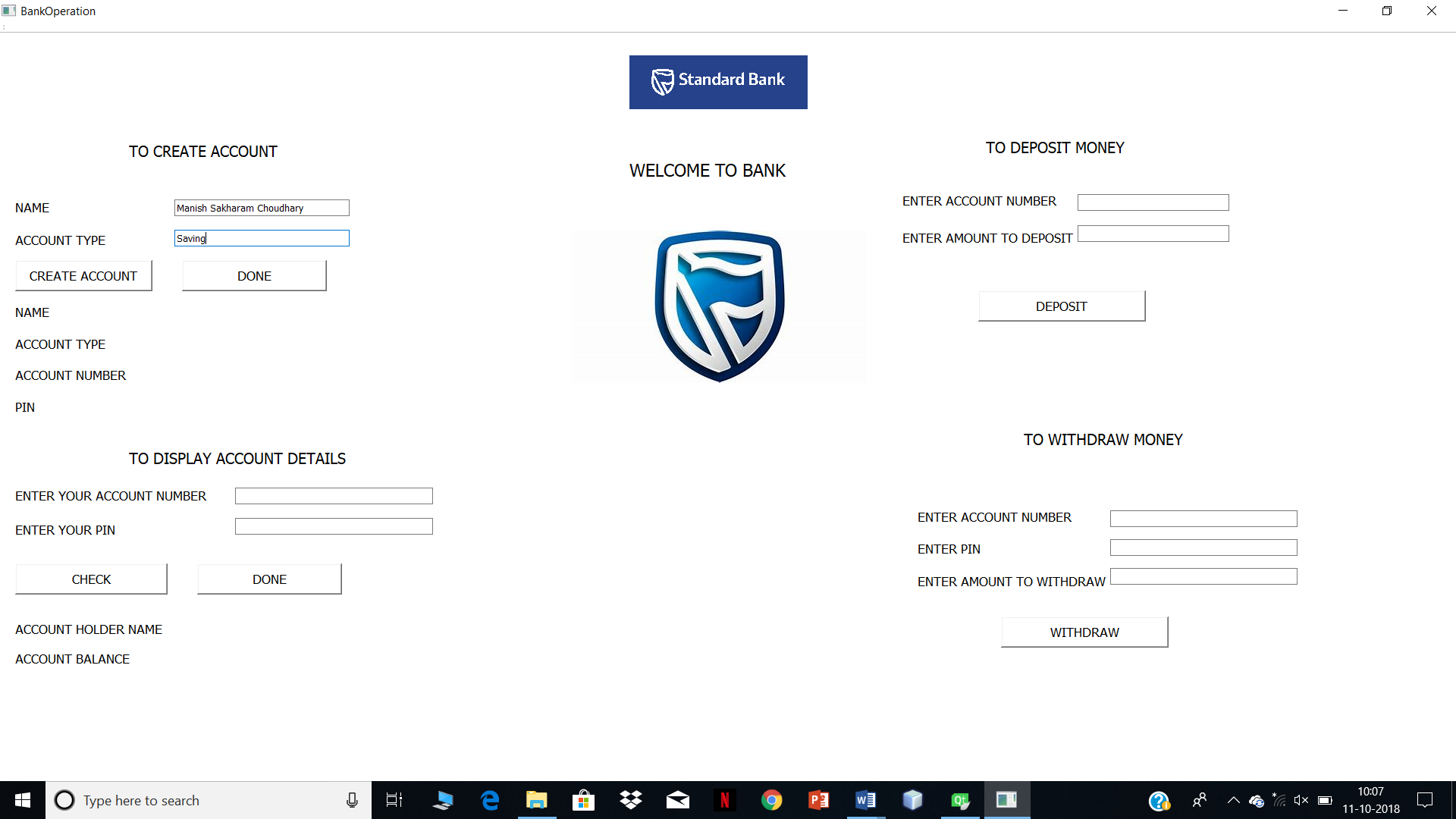
}

OUTPUT

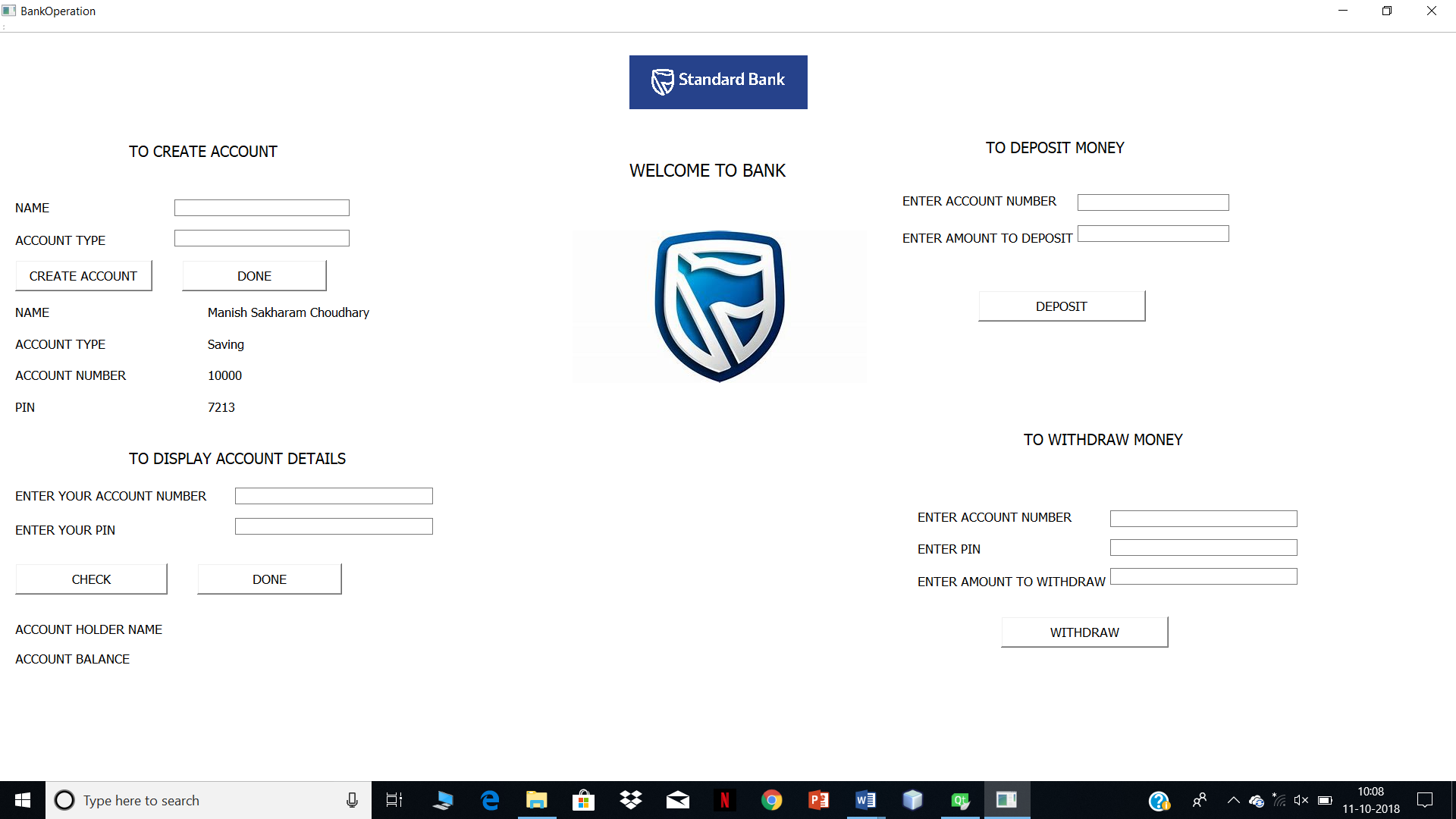


Creating new account

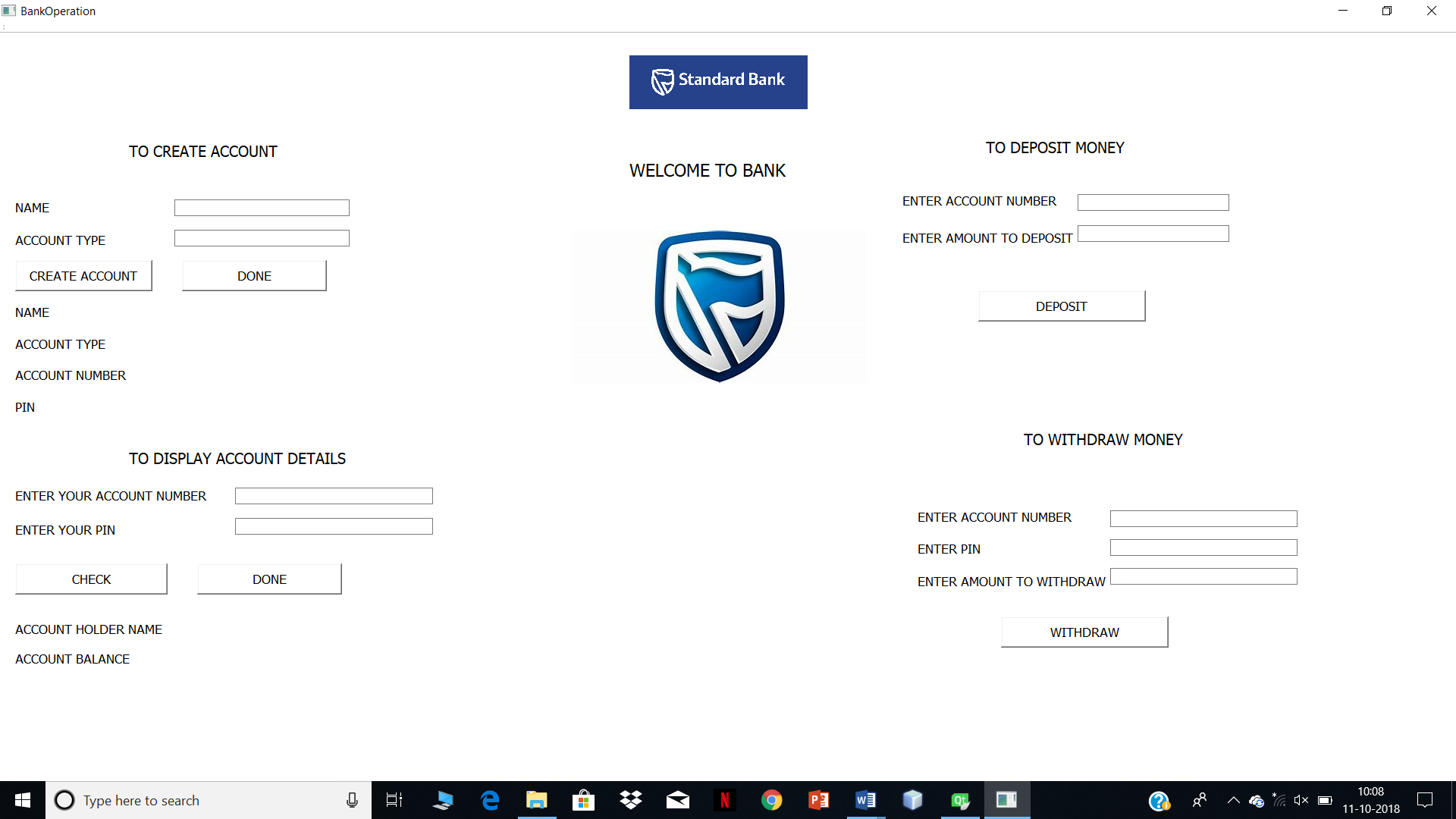
User entering details



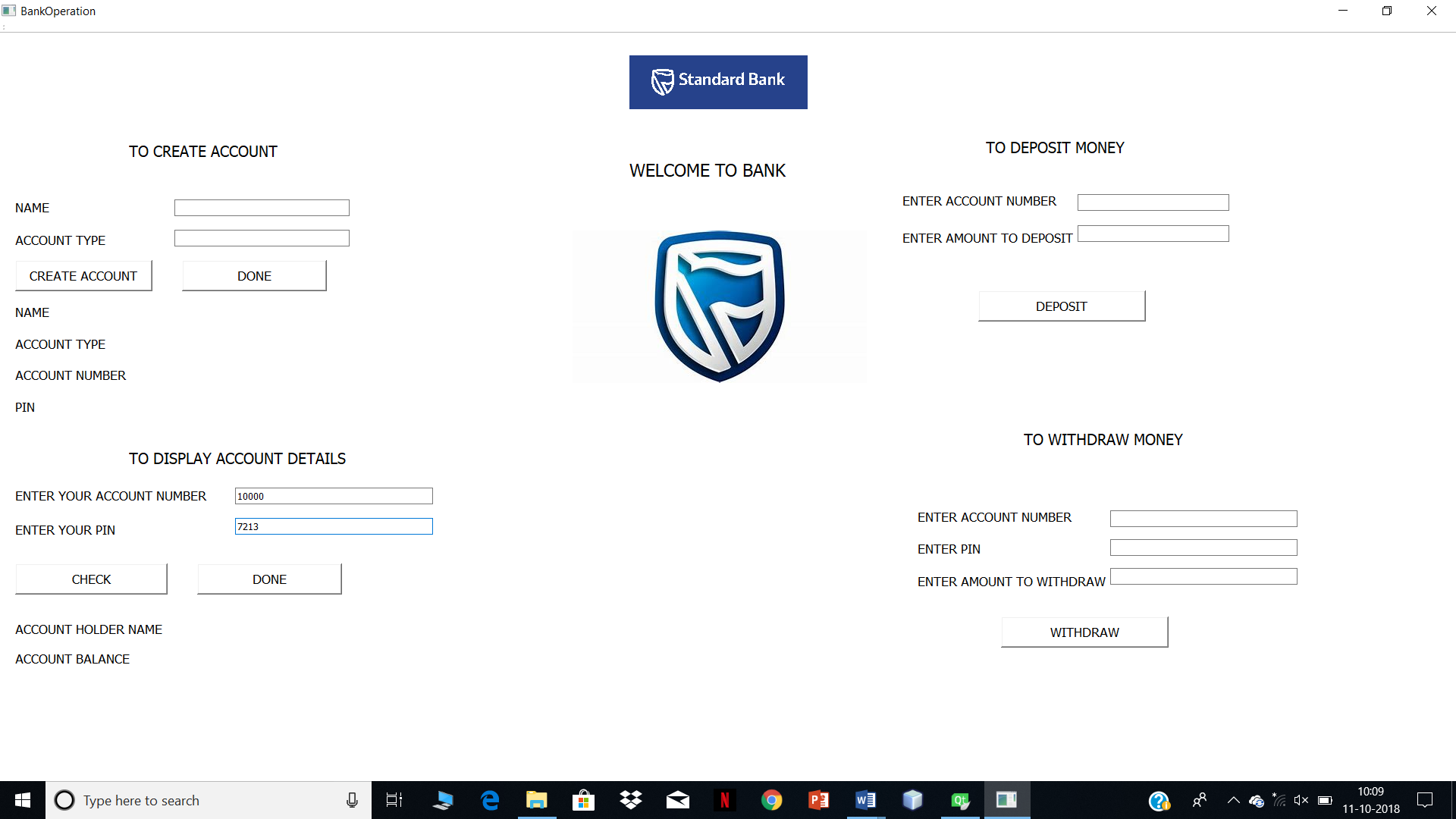
On clicking on CREATE ACCOUNT button



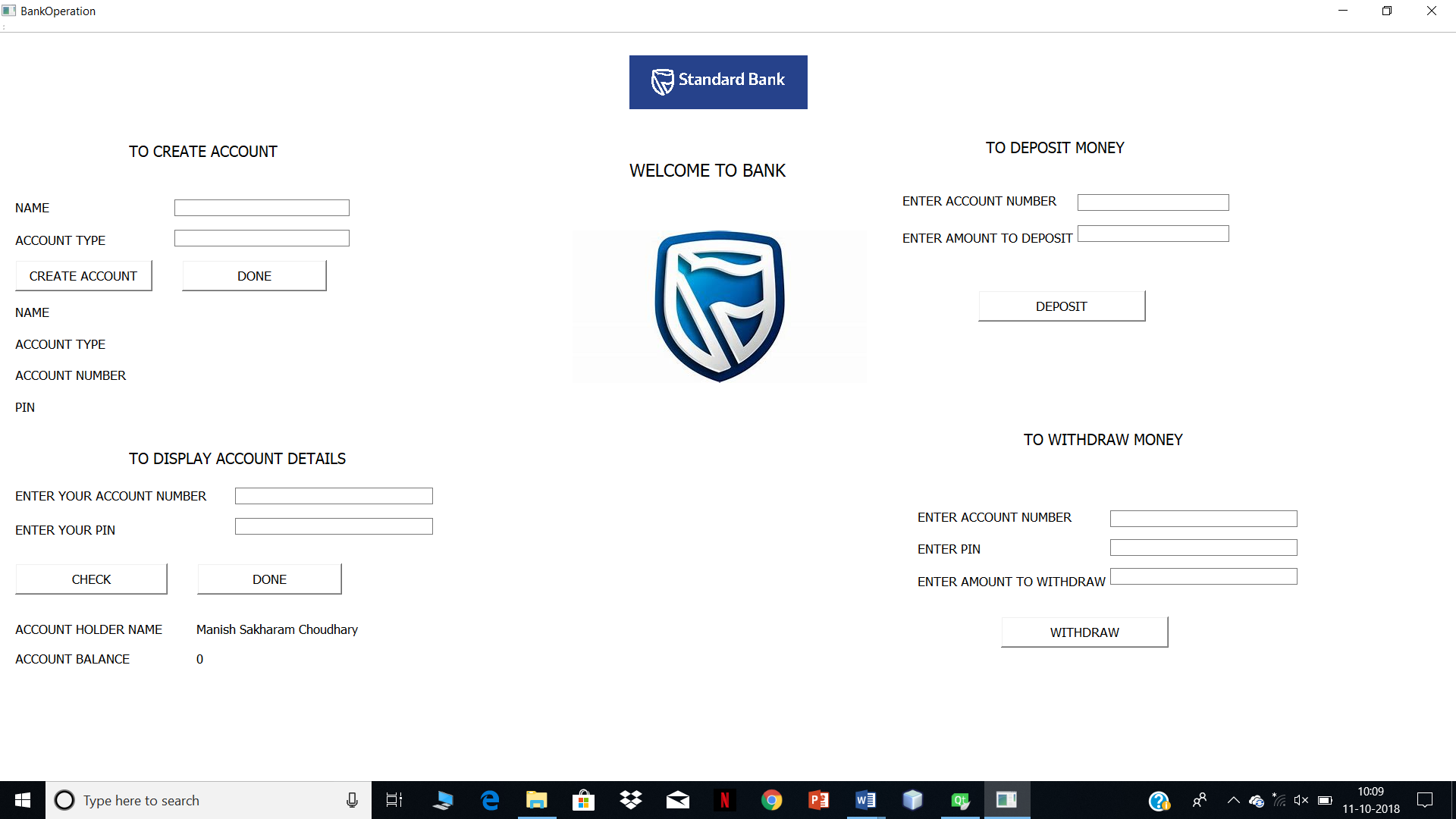
On clicking on DONE button



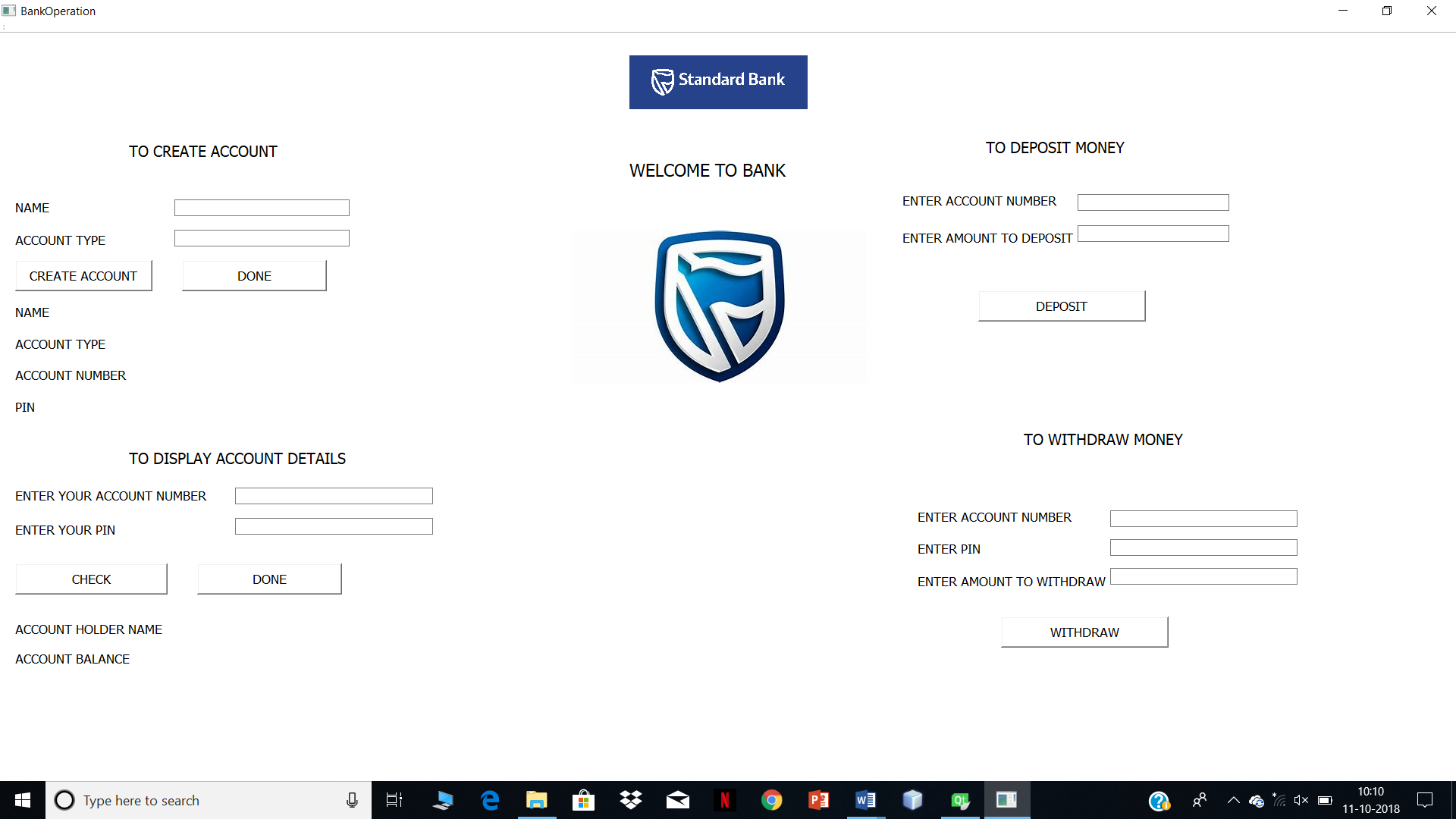
To display account balance



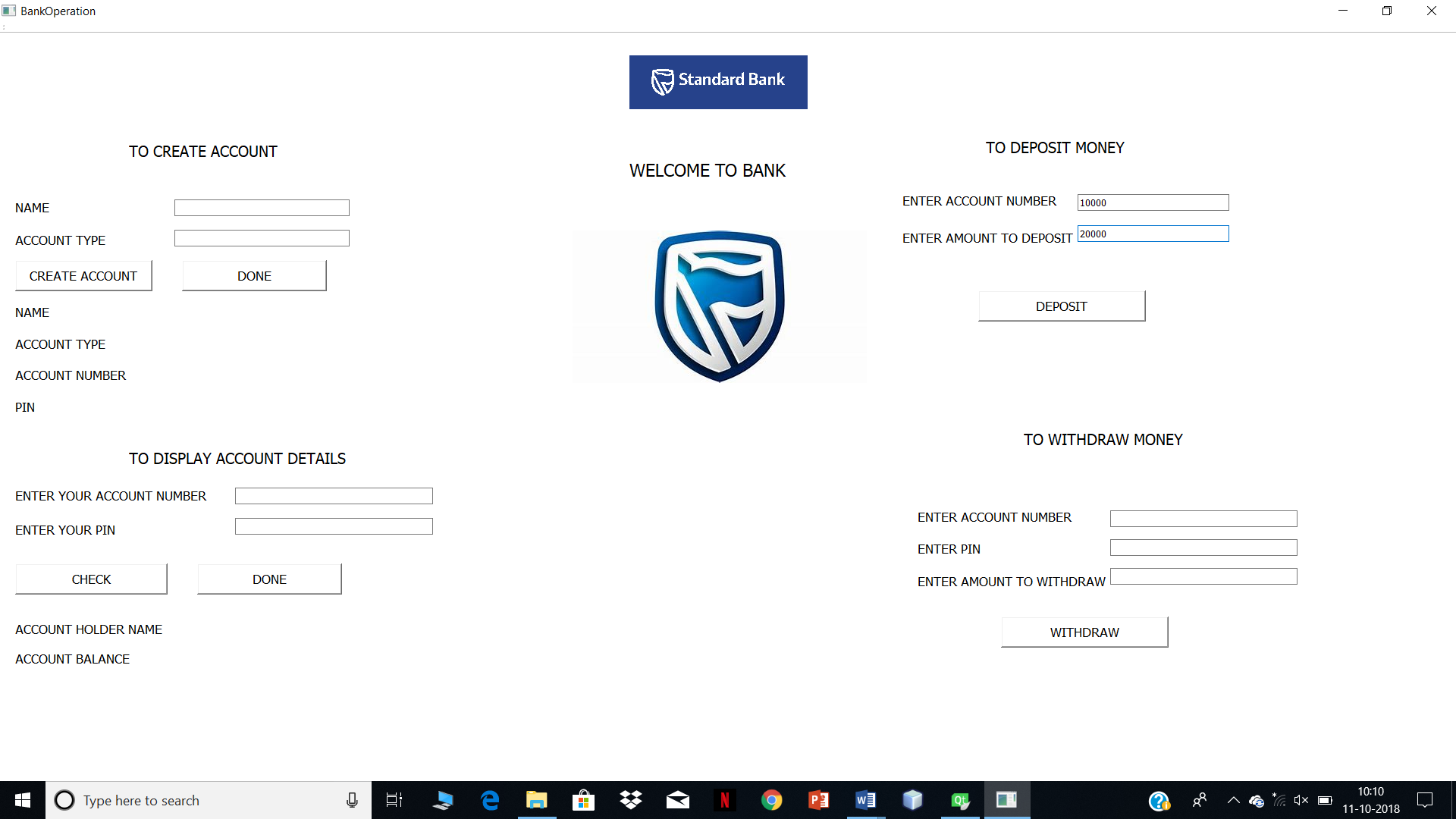
On clicking on CHECK button



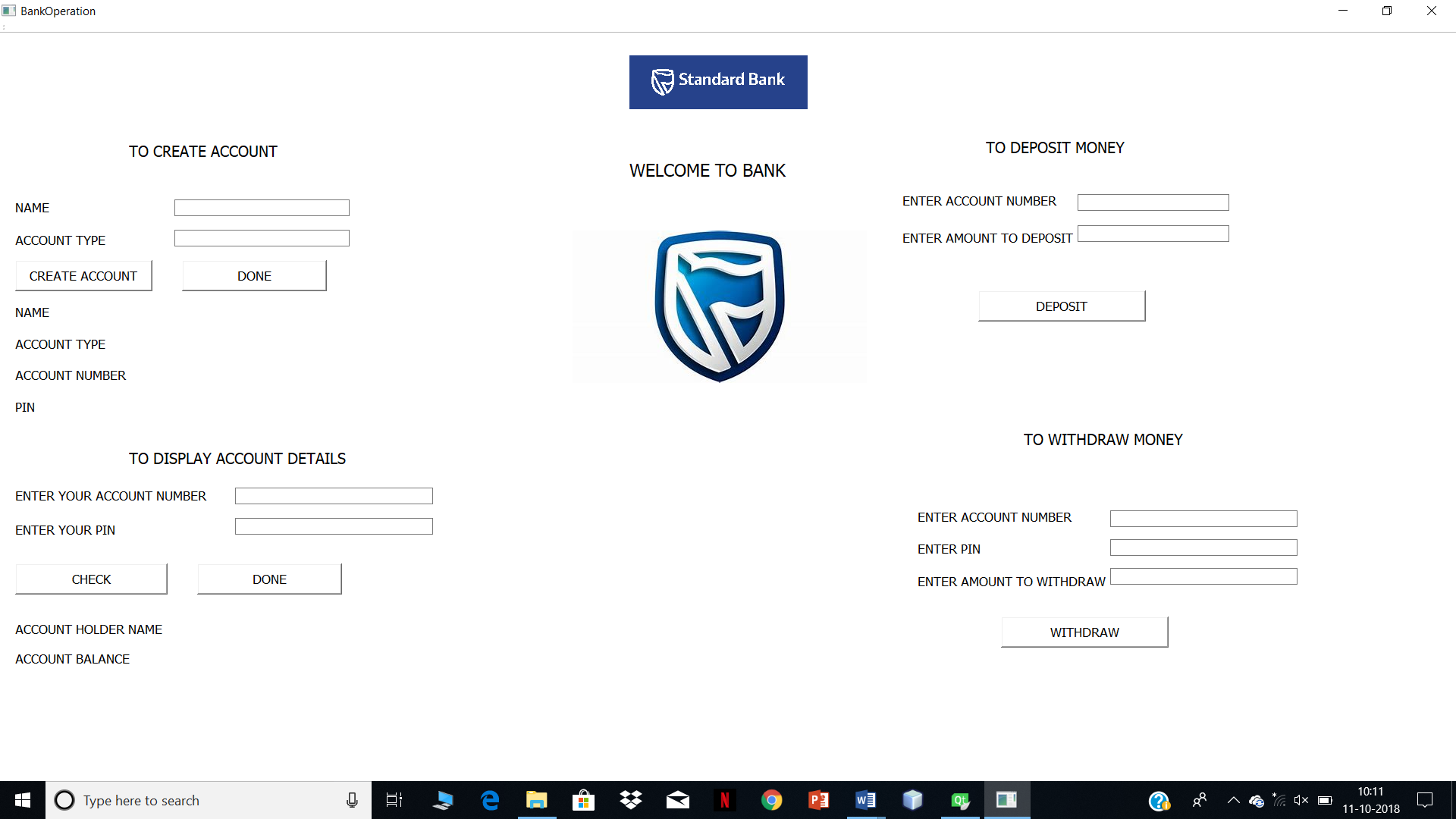
On clicking on DONE button



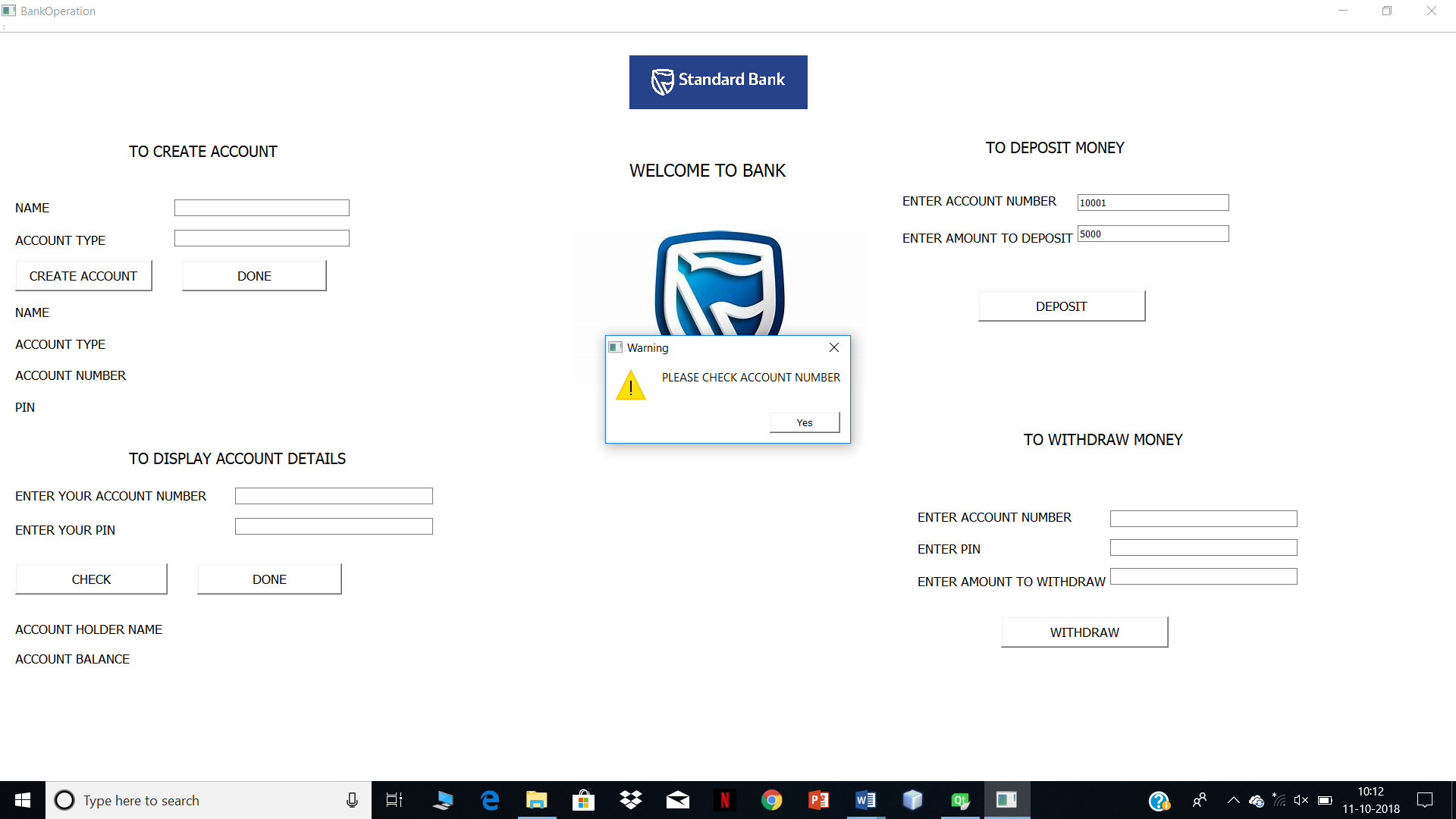
To deposit money in account



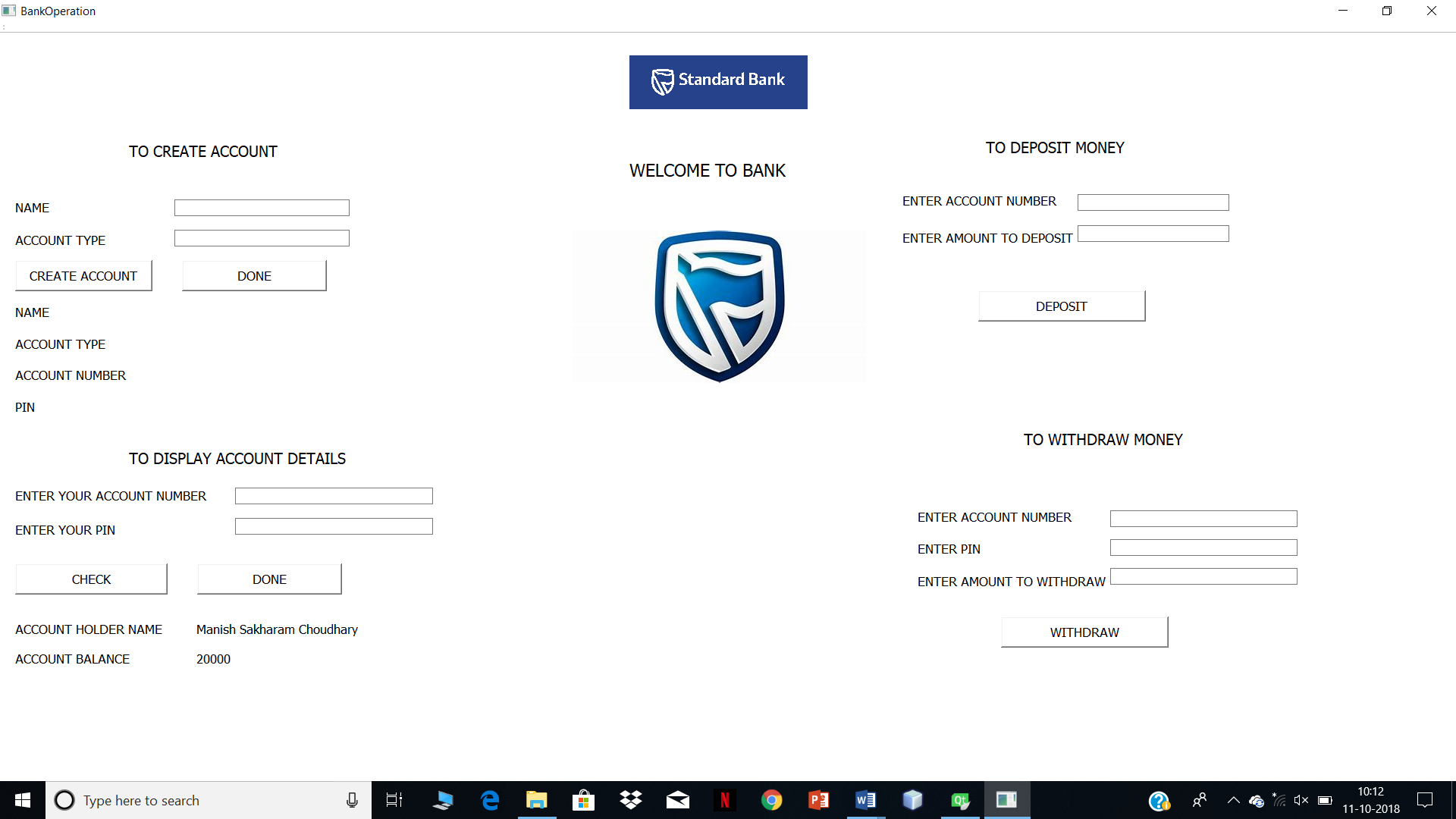
On clicking on DEPOSIT button



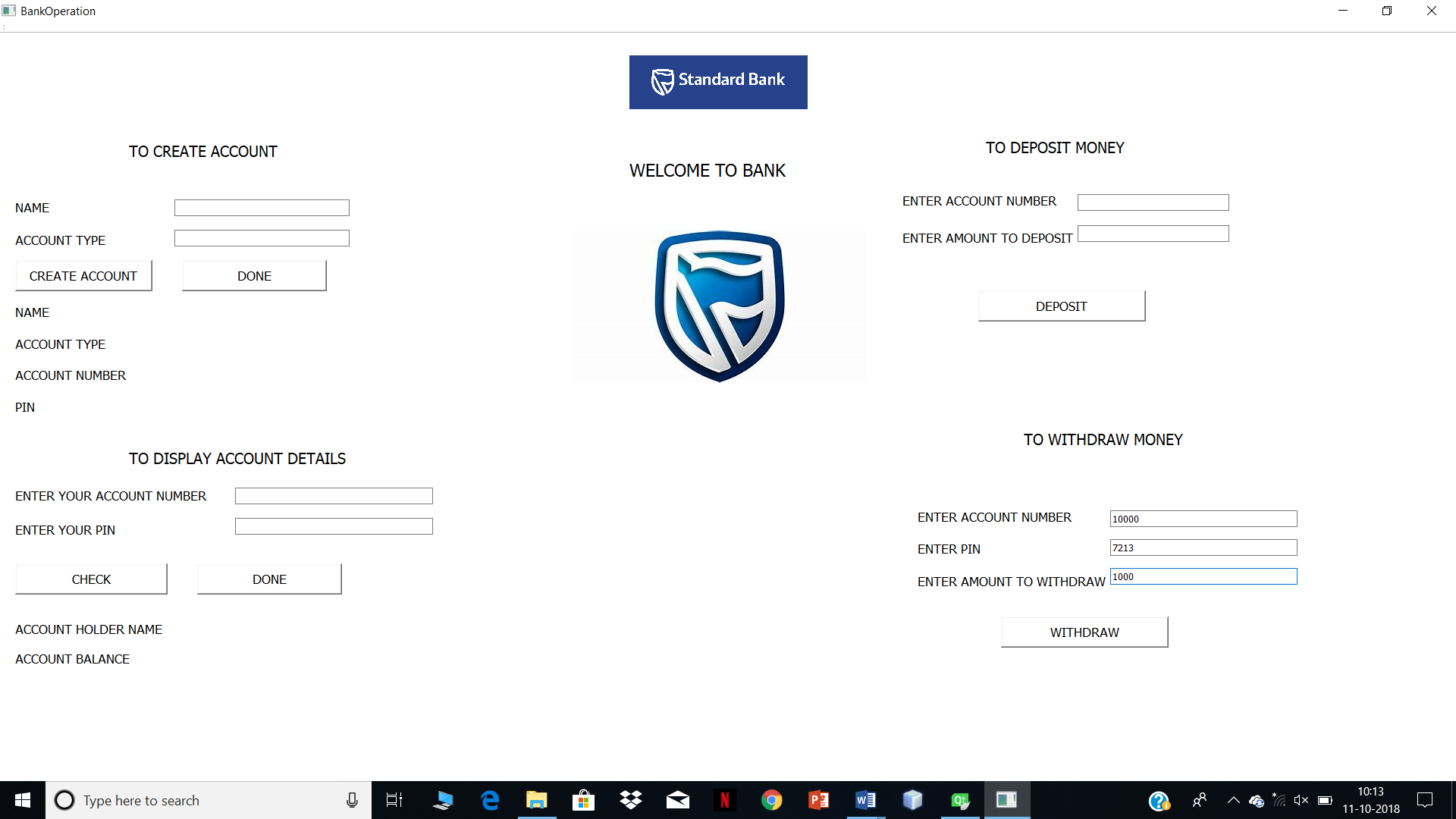
When an user enters invalid account number



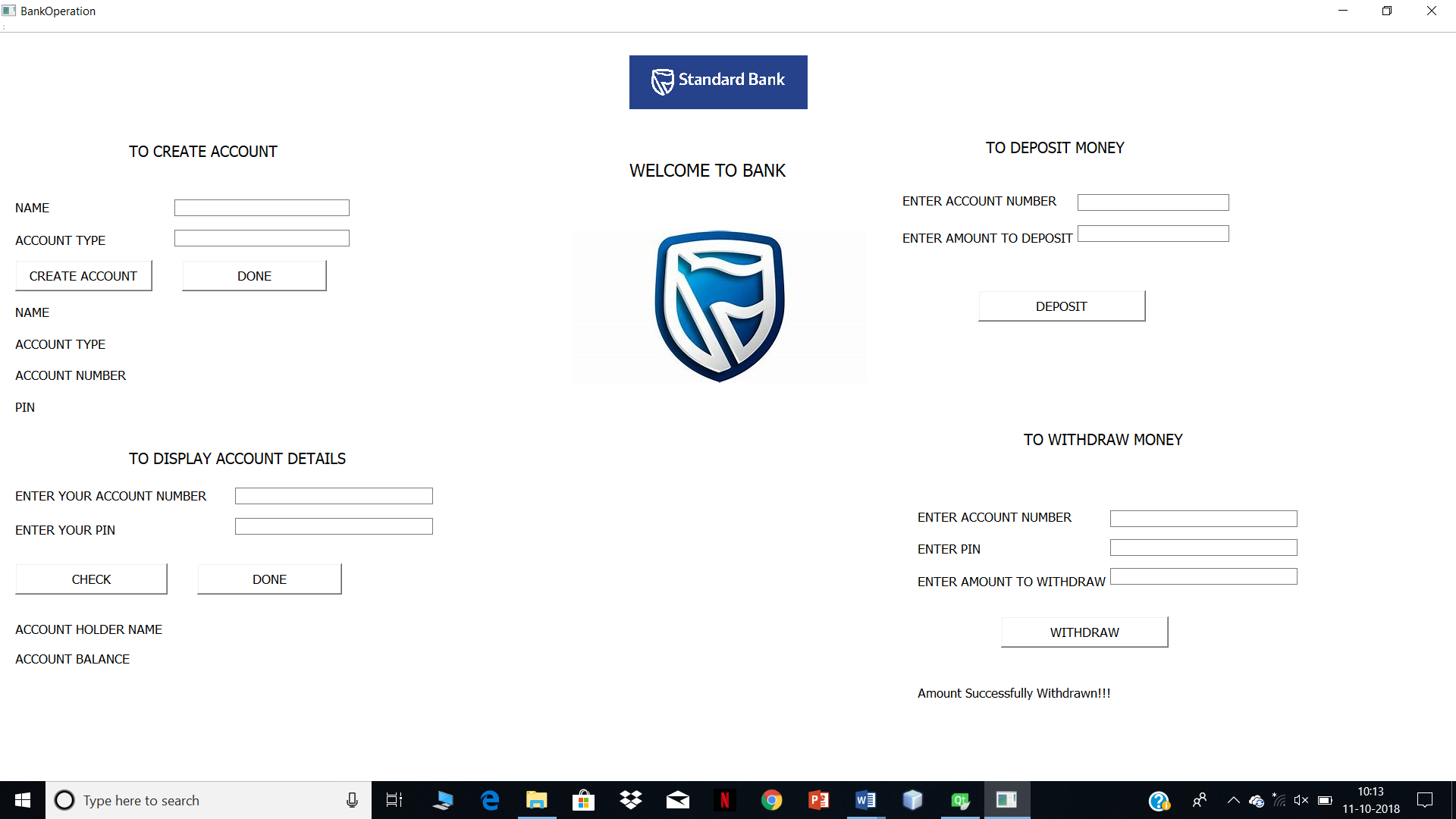
Checking Account Balance



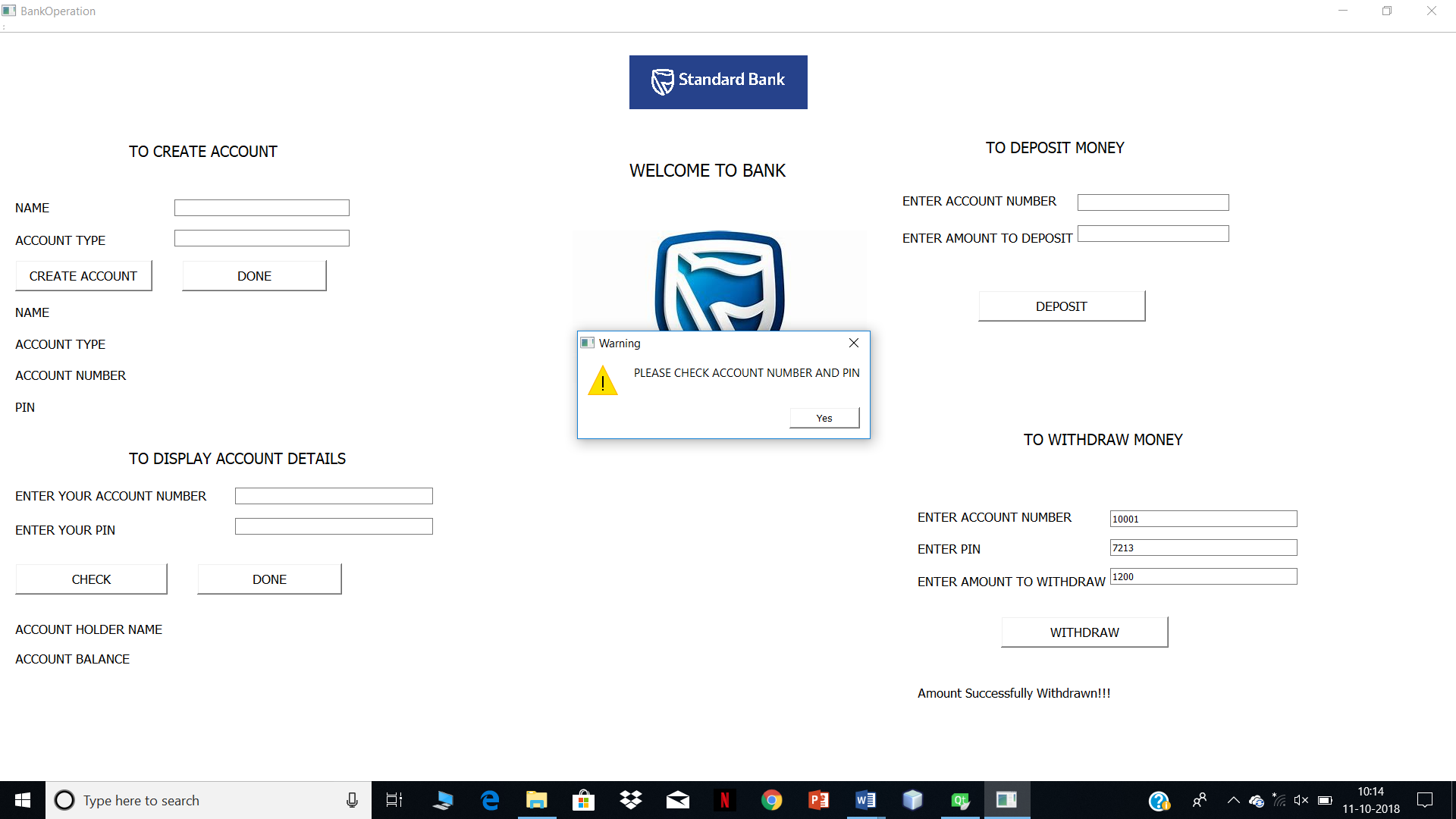
User withdraw money from account



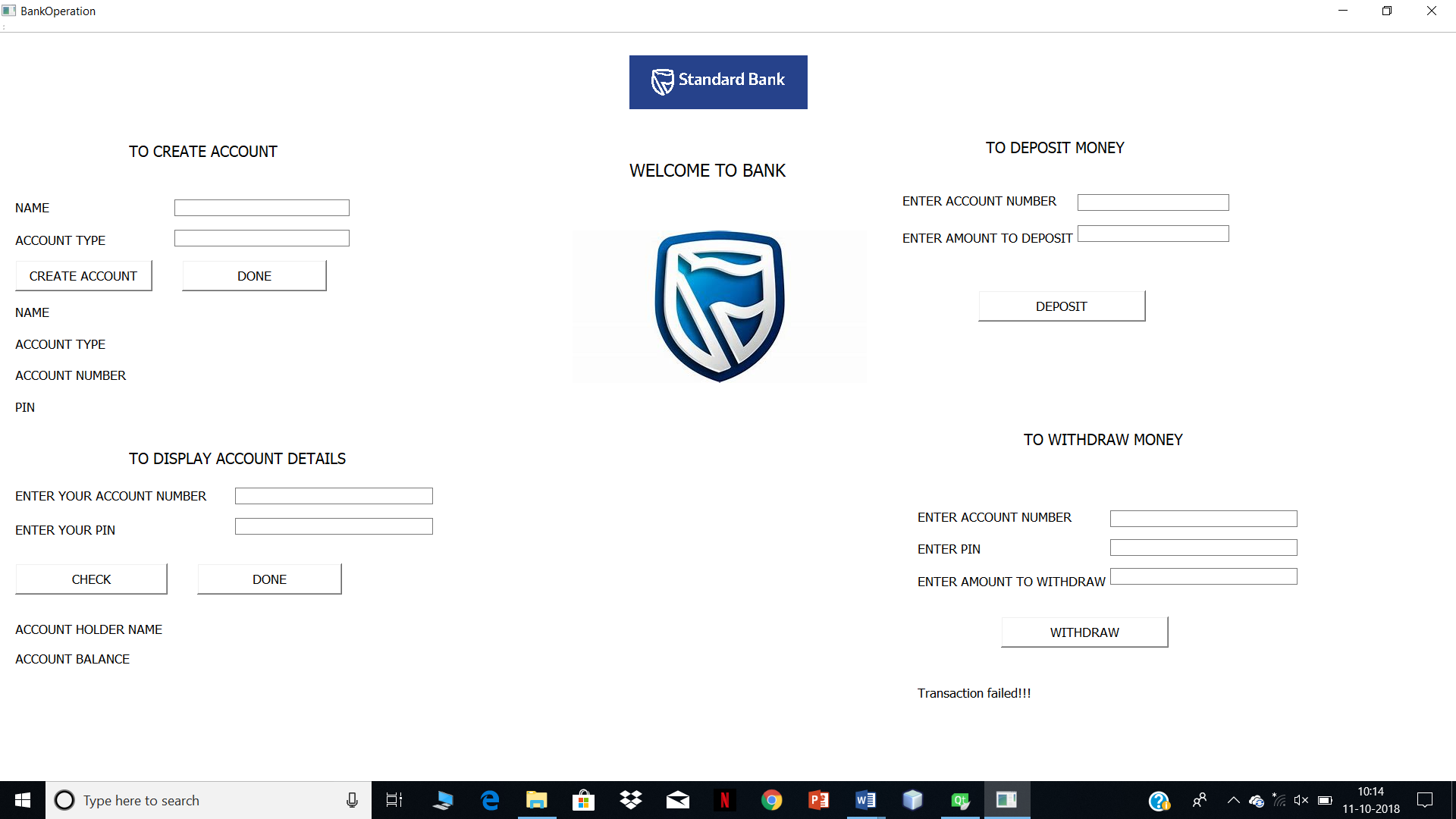
On clicking on WITHDRAW button



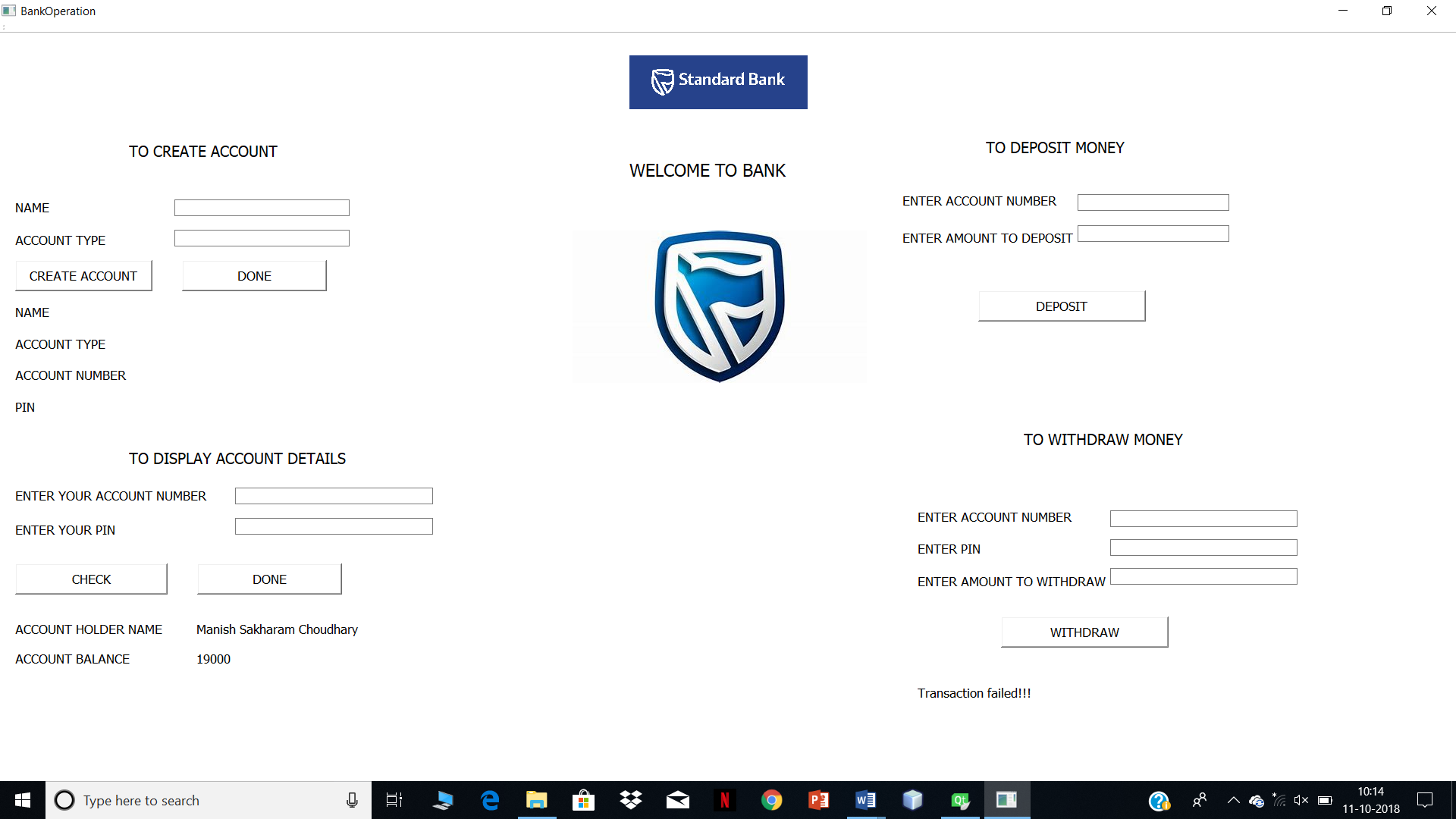
When user enters invalid account details



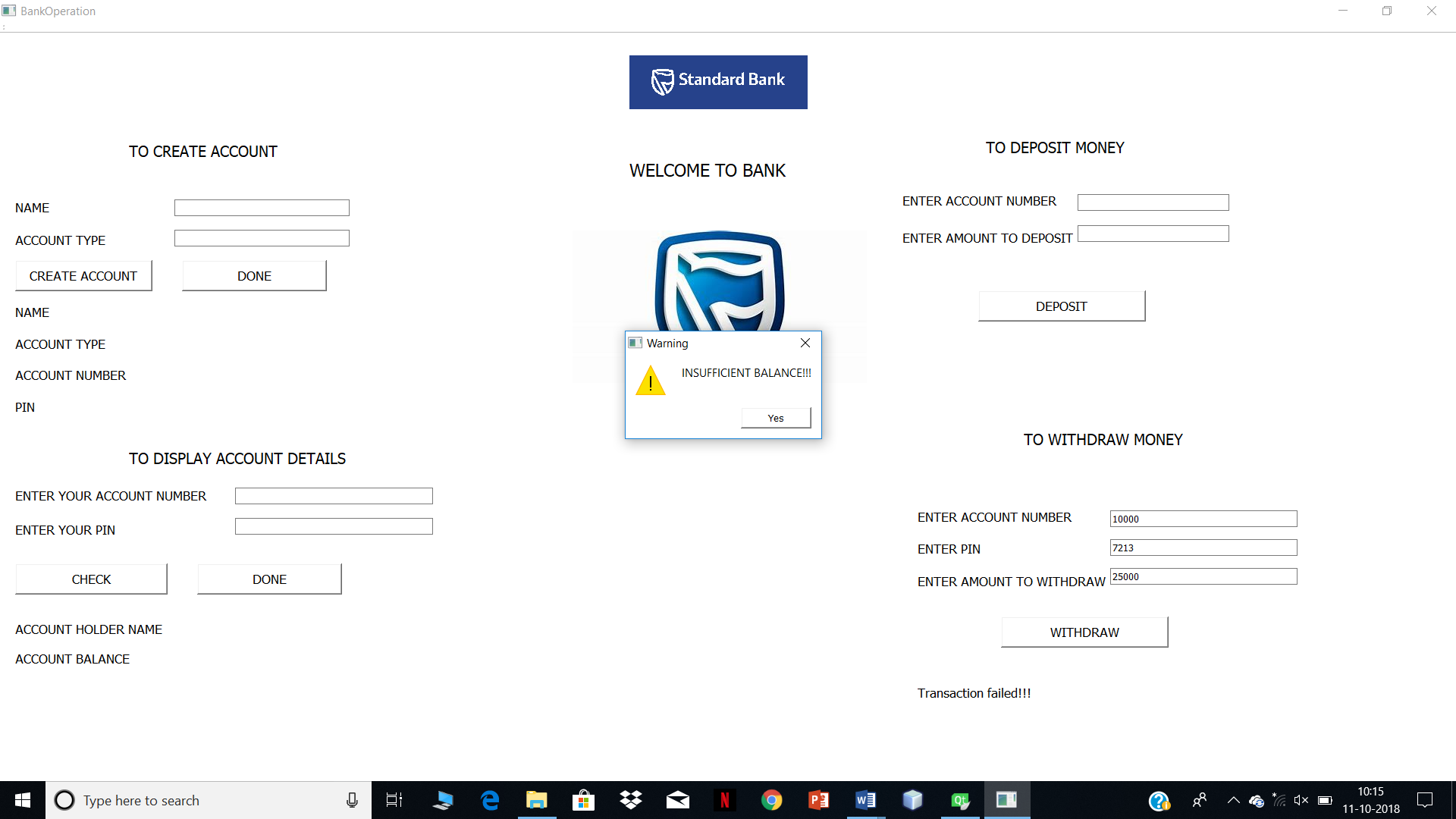
Transaction failed error message



Checking account balance



When user try to withdraw excess money than is account balance



# SOFTWARE USED

1]QT CREATOR

2]NET BEANS IDE.

# CONCLUSION

Thus we Successfully implemented Program in QT CREATOR software and used ADS concept in it for performing different BANKING SERVICES.