

SEMESTER PROJECT REPORT

COMPUTER PROGRAMMING

(CSC-113)



DEPARTMENT OF COMPUTER SCIENCES **BAHRIA UNIVERSITY, ISLAMABAD CAMPUS**

CLASS: BS(CS)-1A

SESSION: FALL 2022

Submitted To: Sir Burhan Abbasi

Submitted By:

Saad Ahmad (01-134222-130)

Sohaib Ahmed (01-134222-142)

Muhammad Talha Umer (01-134222-104)

Project: Bank Management System

STS Bank Limited

Introduction:

This report outlines the design and development of a bank management system coded in C++, Qt(C++) and SQL. The management is comprised of three main sections; Customer, Employee and Admin.

The Customer section or you can say interface is completely Graphical-based and it allows the customer to add funds, withdraw cash, transfer funds from one bank account to another, print a bank statement and change pin and passwords.

The Employee section is console based and it allows the employees to mark their attendance for the ongoing month.

The Admin section is also console based and it allows the administrator to add new employees, check attendance of employees, calculate salary of employees, display all records of the employees and last but not the least, display all records of the customers.

Functional Requirements:

Welcome/ Login Screen (GUI)

Allows the customer to sign-in to their account or signup and make a new account.

Customer (GUI/SQL)

Funds Transfer

The customer can transfer funds from one account to another.

Cash Withdrawal

The customer can withdraw cash from the bank.

Add Funds

The customer can add funds to their account.

Bank Statement

This allows the customer to print the bank statement.

Change Pin

The pin is used for authentication and it can be changed by the respective customers.

Change Password

This allows the customer to change their account password which is used for logging in to the bank.

Welcome/ Login Screen (CLI/SQL)

Allows the employee or the admin to sign-in to the BMS.

Employee (CLI)

Attendance

Allows the employee to mark their attendance for the ongoing month.

Admin (CLI)

Add New Employees

The administrator can add new employees.

Employee Attendance

This allows the admin user to check the attendance of the employees for the ongoing month.

Salary Calculation

The admin user can calculate the Salary of the Employees by just entering their IDs and the salary will be calculated based on the grades which were given to the employees when they were added.

Employee Records

All the records of employees can be printed by the admin user.

Customer Records

The admin can print all the records of the customers which are fetched from SQLite database.

Code: (CLI) (Employee and Admin)

Header.h (Header File)

```
#include <iostream>
#include <string>
#include <fstream>
#include <iomanip>
#include <cstdlib>
#include <time.h>
#include <conio.h>
#include "Sql.h"
#include <sqlite3.h> //Used for SQL
using namespace std;
int pword();
void dashsep();
void frontpage();
void adminoptions();
char checkea;
int x;
int pword() { //This function generates a random password for
the employee
    srand((unsigned)time(NULL));
    for (int i = 0; i < 11; i++)
    {

        x = 1000000 + (rand() % +9999999);
    }
    return 0;
}
void cts()
{
    system("CLS");
```

```

dashsep();
frontpage();
dashsep();
}
void dashsep()
{
    cout << "\n";
    cout << setw(3);
    for (int i = 0; i < 207; i++)
    {
        cout << "-";
    }
    cout << endl;
}
struct account
{
    string username, password;
};
void frontpage()
{
    cout << endl << setw(110) << "STS Bank unLimited" << endl;
}
void login()
{
    account admin;
    account employee;
    string username;
    string password;
    string cau, cap, ceu, cep;
    string txt = ".txt", add;
    int b = 0;
    cout << "Press E for Employee Login" << endl;
    cout << "Press A for Admin Login" << endl;
    cin >> checkea;
    cts();

    if (checkea == 'e' || checkea == 'E')
    {
        for (int i1 = 0; i1 != 1;)
        {
            ifstream in;
            in.open("empdetails.txt");

            cout << "Username: " << endl;
            getline(cin >> ws, employee.username);
            cout << "Password: " << endl;
            getline(cin >> ws, employee.password);

            while (!in.eof())
            {
                getline(in, ceu);
                getline(in, cep);

                if (employee.username == ceu && employee.password ==
cep)
                {
                    cout << "Logged in successfully" << endl;
                    i1++;
                    bool attendance = 0;
                    for (;;)
                    {

```

```

Present) " << endl;

        cout << "1. Mark attendance (Enter 1 for
        cin >> attendance;
        if (attendance == 0)
        {
            cts();
            continue;
        }
        else
        {
            break;
        }
    }
    if (attendance == true && employee.username ==
ceu)
    {
        add = ceu + txt; // makes the file name
        like 1001.txt
        string abc;
        int f;
        ifstream in;
        in.open(add);
        in >> abc;
        f = abc.length();
        in.close();
        if (f >= 21)
        {
            ofstream out;
            out.open(add, ios::trunc); // If
the attendance exceeds 21 days it resets the file back to zero
            out.close();
        }
        ofstream out;
        out.open(add, ios::app);
        out << "p";
        out.close();
        cout << "Attendance marked succesfully" <<
endl;

        system("pause");
        cts();
        b++;
        break;
    }
}
if (b == 0)
{
    cout << "Invalid username or password" << endl;
}
}
cts();
login();
}
if (checkea == 'a' || checkea == 'A')
{
    for (int i1 = 0; i1 != 1;)
    {
        ifstream in;
        in.open("credentials.txt");
        cout << "Username: " << endl;
        getline(cin >> ws, admin.username);
        cout << "Password: " << endl;
        char ch;

```

```

        string password;
        //used to mask the password
        while ((ch = _getch()) != 13) { // 13 is the ASCII code for
the "Enter" key
            if (ch == 8) { // 8 is the
ASCII code for the "Backspace" key
                if (!password.empty()) {
                    cout << "\b \b";
                    password.pop_back();
                }
            }
            else {
                password += ch;
                putchar('*');
            }
        }
        employee.password = password;
        cout << endl;
        while (!in.eof())
        {
            getline(in, cau);
            getline(in, cap);

            if (admin.username == cau && employee.password == cap)
            {
                cout << "Logged in successfully as an admin" <<
endl << endl;

                cts();
                il++;
                adminoptions();
            }
            else
            {
                cts();
                cout << "Invalid username or password" << endl;
                break;
            }
        }
    }
}

void adminoptions()
{
    cout << "1. Add a New Employee" << endl;
    cout << "2. Check Attendance of an Employee" << endl;
    cout << "3. Calculate Salary of an Employee" << endl;
    cout << "4. Display all records of the Employees" << endl;
    cout << "5. Display all records of the Customers" << endl;
    cout << "0. Go Back" << endl;
    int i, a = 0;
    int empid = 1000, cempid = 1000;
    string empname, empadd, empgrade;
    string cempname, cempadd, cempgrade;
    int agrade = 85000, bgrade = 65000, cgrade = 50000, dgrade = 35000;
    int b = 0;
    cin >> i;
    if (i == 1) // Lets the user add a new employee
    {
        cts();
        ofstream out;
        out.open("emp.txt", ios::app);
        out.close();
        ifstream in;

```

```

in.open("emp.txt");
in.sync();
for (int i4 = 0; i4 == 0;)
{
    a = 0;
    cout << "Name: " << endl;
    getline(cin >> ws, empname);
    cout << "Address: " << endl;
    getline(cin >> ws, empadd);
    cout << "Grade (A-D): " << endl;
    for (;;)
    {
        getline(cin >> ws, empgrade);
        if (!(empgrade == "a" || empgrade == "A" || empgrade ==
"b" || empgrade == "B" || empgrade == "c" || empgrade == "C" || empgrade == "d"
|| empgrade == "D"))
        {
            cout << "Grade must be between A and D" << endl;
        }
        else
        {
            break;
        }
    }
    while (!in.eof())
    {
        in >> cempid;
        getline(in, cempname);
        getline(in, cempadd);
        getline(in, cempgrade);
        if (empname == cempname && empadd == cempadd)
        {
            cts();
            cout << empname << " is already an employee!" <<
endl;

            in.seekg(0, ios::beg);
            a++;
            break;
        }
    }
    if (a == 0)
    {
        in.seekg(0, ios::beg);
        break;
    }
}
while (!in.eof())
{
    in.close();
    if (a == 0)
    {
        empid = cempid;
        empid++;
        ofstream out;
        out.open("emp.txt", ios::app);
        out << empid;
        out << empname << endl << empadd << endl << empgrade <<
endl;

        out.close();
        out.open("empdetails.txt", ios::app);
        out << empid << endl;
        pword();
        out << x << endl;
    }
}

```

```

        out.close();
        cts();
        cout << empname << " added as an employee!" << endl <<
endl;
        adminoptions();
    }
}
else if (i == 2) // Checks the attendance of the employee
{
    account employee;
    string checka;
    string txt = ".txt", add;
    string a;
    int num;
    cts();
    cout << "Enter ID to check attendance" << endl;
    cin >> checka;
    add = checka + txt;
    ifstream inn;
    inn.open("empdetails.txt");

    while (!inn.eof())
    {
        getline(inn, employee.username);
        getline(inn, employee.password);
        if (checka == employee.username)
        {
            ifstream in;
            in.open(add);
            in >> a;
            in.close();
            num = a.length();
            if (num == 1)
            {
                cout << "This employee was present " << num << "
day in this month" << endl;
            }
            else
            {
                cout << "This employee was present " << num << "
days in this month" << endl;
            }
            break;
        }
    }
    system("pause");
    cts();
    adminoptions();
}

else if (i == 3) // Calculates the salary of the employees
{
    cts();
    ifstream in;
    in.open("emp.txt");
    in.sync();
    for (int i5 = 0; i5 == 0;)
    {
        cout << "Enter employee id: " << endl;
        cin >> empid;
        while (!in.eof())
        {

```



```

        in >> cempid;
        getline(in, cempname);
        getline(in, cempadd);
        getline(in, cempgrade);
        if (empid == cempid)
        {
            i5++;
            b++;
            break;
        }
    }
    break;
}
in.close();
if (b == 0)
{
    cout << "The id you entered is incorrect or doesn't exists"
<< endl;
}
else if (cempgrade == "A" || cempgrade == "a")
{
    cout << "The salary of " << empid << " is Rs." << agrade;
}
else if (cempgrade == "B" || cempgrade == "b")
{
    cout << "The salary of " << empid << " is Rs." << bgrade;
}
else if (cempgrade == "C" || cempgrade == "c")
{
    cout << "The salary of " << empid << " is Rs." << cgrade;
}
else if (cempgrade == "D" || cempgrade == "d")
{
    cout << "The salary of " << empid << " is Rs." << dgrade;
}

cout << endl;
system("pause");
cts();
adminoptions();
}

else if (i == 4) // Prints all the records of the employees
{
    ifstream in;
    in.open("emp.txt", ios::beg);
    in.sync();
    cts();
    while (!in.eof())
    {
        in >> cempid;
        getline(in, cempname);
        getline(in, cempadd);
        getline(in, cempgrade);
        cout << "Name: " << cempname << endl;
        cout << "Address: " << cempadd << endl;
        cout << "Grade: " << cempgrade << endl;
        cout << endl;
    }
    in.close();
    system("pause");
    cts();
    adminoptions();
}

```

```
    }
    if (i == 5) { //Prints all the records of the customers
        selectdata("C:/Users/Saad/Desktop/Database/customer.db"); // The
address of the sql file
        system("pause");
        cts();
        adminoptions();
    }
    else if (i == 0)
    {
        cts();
        login();
    }
}
```

Sql.h (Header File)

```
#include <sqlite3.h>
#include <iostream>
#include <string>
using namespace std;
static int selectdata(const char* s);
static int callback(void* NotUsed, int argc, char** argv, char** azColName);
static int selectdata(const char* s) {
    sqlite3* DB;
    int exit = sqlite3_open(s, &DB);
    string sql = "SELECT Accountnum,Name,Username,Gender,DOB,Balance FROM
record";
    sqlite3_exec(DB, sql.c_str(), callback, NULL, NULL);
    return 0;
}
static int callback(void* NotUsed, int argc, char** argv, char** azColName) {
    for (int i = 0; i < argc; i++) {
        cout << azColName[i] << ": " << argv[i] << endl;
    }
    cout << endl;
    return 0;
}
```

Source.cpp

```
#include "Header.h"
int main()
{
    cts();
    login();
    return 0;
}
```

Code: Customer (GUI) (Qt (C++)) (SQL)

BMS.pro

QT += core gui sql

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++17

You can make your code fail to compile if it uses deprecated APIs.

In order to do so, uncomment the following line.

#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

accountstatement.cpp \

changepin.cpp \

changepword.cpp \

deposit.cpp \

fundstransfer.cpp \

main.cpp \

mainpage.cpp \

options.cpp \

withdrawl.cpp

HEADERS += \

accountstatement.h \

changepin.h \

changepword.h \

deposit.h \

fundstransfer.h \
mainpage.h \
options.h \
withdrawl.h

FORMS += \
 accountstatement.ui \
 changepin.ui \
 changepword.ui \
 deposit.ui \
 fundstransfer.ui \
 mainpage.ui \
 options.ui \
 withdrawl.ui

Default rules for deployment.
qnx: target.path = /tmp/`\${TARGET}/bin
else: unix:!android: target.path = /opt/`\${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

mainpage.h

```
#ifndef MAINPAGE_H
#define MAINPAGE_H

#include <QMainWindow>
#include <QtSql>
#include <QMessageBox>
#include <QDebug>
#include "options.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainPage; }
```

QT_END_NAMESPACE

class MainPage : public QMainWindow

{

Q_OBJECT

public:

QSqlDatabase mydb;

void connClose(){

mydb.close();

mydb.removeDatabase(QSqlDatabase::defaultConnection);

}

bool connOpen()

{

mydb = QSqlDatabase::addDatabase("QSQLITE");

mydb.setDatabaseName("C:/Users/Saad/Desktop/Database/customer.db");

if(!mydb.open()){

qDebug() << "NOT OPEN";

return false;

}

else{

qDebug() << "OPEN";

return true;

}

}

public:

MainPage(QWidget *parent = nullptr);

~MainPage();

private slots:

void on_pushButton_clicked();

```
void on_pushButton_2_clicked();

void on_username_login_editingFinished();

void on_password_login_editingFinished();

void on_name_reg_editingFinished();

void on_username_reg_editingFinished();

void on_password_reg_editingFinished();

void on_dateEdit_dateChanged(const QDate &date);

void on_pin_reg_editingFinished();

private:
    Ui::MainPage *ui;

};

#endif // MAINPAGE_H
```

options.h

```
#ifndef OPTIONS_H
#define OPTIONS_H
#include <QDialog>
#include "withdrawl.h"
#include "deposit.h"
#include "fundstransfer.h"
#include "changepin.h"
#include "changepword.h"
```

```
#include "accountstatement.h"

namespace Ui {
class Options;
}

class Options : public QDialog
{
    Q_OBJECT

public:
    explicit Options(QWidget *parent = nullptr);
    ~Options();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

    void on_pushButton_7_clicked();

    void on_pushButton_4_clicked();

    void on_pushButton_5_clicked();

    void on_pushButton_8_clicked();

    void on_pushButton_9_clicked();
```



```
private:

    Ui::Options *ui;

};

#endif // OPTIONS_H
```

withdrawl.h

```
#ifndef WITHDRAWL_H
#define WITHDRAWL_H

#include <QDialog>

extern QString username;

namespace Ui {
class withdrawl;
}

class withdrawl : public QDialog
{
    Q_OBJECT

public:
    explicit withdrawl(QWidget *parent = nullptr);
    ~withdrawl();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_lineEdit_editingFinished();
}
```

```
void on_pin_editingFinished();
```

```
private:
```

```
    Ui::withdrawl *ui;
```

```
};
```

```
#endif // WITHDRAWL_H
```

deposit.h

```
#ifndef DEPOSIT_H
```

```
#define DEPOSIT_H
```

```
#include <QDialog>
```

```
extern QString username;
```

```
namespace Ui {
```

```
class deposit;
```

```
}
```

```
class deposit : public QDialog
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    explicit deposit(QWidget *parent = nullptr);
```

```
    ~deposit();
```

```
private slots:
```

```
    void on_pushButton_clicked();
```

```
    void on_pushButton_2_clicked();
```

```
void on_lineEdit_editingFinished();
```

```
void on_pin_editingFinished();
```

```
private:
```

```
    Ui::deposit *ui;
```

```
};
```

```
#endif // DEPOSIT_H
```

fundstrasnfer.h

```
#ifndef FUNDSTRANSFER_H
```

```
#define FUNDSTRANSFER_H
```

```
#include <QDialog>
```

```
extern QString username;
```

```
namespace Ui {
```

```
class Fundstransfer;
```

```
}
```

```
class Fundstransfer : public QDialog
```

```
{
```

```
    Q_OBJECT
```

```
public:
```

```
    explicit Fundstransfer(QWidget *parent = nullptr);
```

```
    ~Fundstransfer();
```

```
private slots:
```

```
void on_pushButton_clicked();

void on_pushButton_2_clicked();

void on_accountnum_editingFinished();

void on_amount_editingFinished();

void on_lineEdit_editingFinished();
```

```
private:
```

```
    Ui::Fundstransfer *ui;
};
```

```
#endif // FUNDSTRANSFER_H
```

changekeyword.h

```
#ifndef CHANGEKEYWORD_H
#define CHANGEKEYWORD_H
```

```
#include <QDialog>
```

```
namespace Ui {
class changekeyword;
}
```

```
class changekeyword : public QDialog
{
    Q_OBJECT
```

```

public:
    explicit changepword(QWidget *parent = nullptr);
    ~changepword();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_lineEdit_2_editingFinished();

    void on_lineEdit_editingFinished();

private:
    Ui::changepword *ui;
};

#endif // CHANGEWORD_H

```

changepin.h

```

#ifndef CHANGEPIN_H
#define CHANGEPIN_H

#include <QDialog>

namespace Ui {
class changepin;
}

class changepin : public QDialog

```

```

{
    Q_OBJECT

public:
    explicit changepin(QWidget *parent = nullptr);
    ~changepin();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_lineEdit_editingFinished();

    void on_lineEdit_2_editingFinished();

private:
    Ui::changepin *ui;
};

#endif // CHANGEPIN_H

```

accountstatement.h

```

#ifndef ACCOUNTSTATEMENT_H
#define ACCOUNTSTATEMENT_H

#include <QDialog>

namespace Ui {
class Accountstatement;

```

```

}

class Accountstatement : public QDialog
{
    Q_OBJECT

public:
    explicit Accountstatement(QWidget *parent = nullptr);
    ~Accountstatement();

private slots:
    void on_pushButton_clicked();

private:
    Ui::Accountstatement *ui;
};

#endif // ACCOUNTSTATEMENT_H

```

mainpage.cpp

```

#include "mainpage.h"
#include "ui_mainpage.h"
#include <QIntValidator>
QString username;
MainPage::MainPage(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainPage)
{
    ui->setupUi(this);

```

```

        ui->pushButton->setDisabled(true);
        ui->pushButton_2->setDisabled(true);

    }

```

```

MainPage::~MainPage()
{
    delete ui;
}

```

```

void MainPage::on_pushButton_clicked()
{
    if(ui->uname_login->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }

    QString password;
    username = ui->uname_login->text();
    password = ui->pword_login->text();
    connOpen();
    if(!mydb.isOpen()){
        qDebug() << "Failed to open";
        return;
    }

```

```

    QSqlQuery qry;

    qry.prepare("SELECT * FROM record WHERE Username='"+ username+"' AND
Password='"+password+"'");

    if(qry.exec()){
        int count=0;

```



```

while(qry.next()){
    count++;
}
if(count==1){
    QMessageBox::information(this,"Success","Login Success");
    connClose();
    this->hide();
    Options options;
    options.setModal(true);
    options.exec();
}
else{
    QMessageBox::critical(this,"Not Success","Login Failed");
}
}
}

```

```

void MainPage::on_pushButton_2_clicked()
{
    QString Name,Username>Password,Gender,DOB,PIN;
    if(ui->male->isChecked()){
        Gender = "Male";
    }
    if(ui->female->isChecked()){
        Gender = "Female";
    }
    Name= ui->name_reg->text();
    Username = ui->uname_reg->text();
    Password = ui->pword_reg->text();
}

```

```

DOB=ui->dateEdit->text();
PIN=ui->pin_reg->text();
connOpen();
if(!mydb.isOpen()){
    qDebug() << "Failed to open";
    return;
}
QSqlQuery query;
query.prepare("INSERT INTO record(Name,Username>Password,Gender,DOB,PIN)
VALUES('"+Name+"','"+Username+"','"+Password+"','"+Gender+"','"+DOB+"','"+PIN+"')");
if(query.exec()){
    QMessageBox::information(this,"Success","User Registered Successfully");
    connClose();
}
else{
    QMessageBox::critical(this,"Failed","Username already taken");
}
}

```

```

void MainPage::on_uname_login_editingFinished()
{
    if(ui->uname_login->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

```
void MainPage::on_pword_login_editingFinished()
{
    if(ui->pword_login->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}
```

```
void MainPage::on_name_reg_editingFinished()
{
    if(ui->name_reg->text().isEmpty()){
        ui->pushButton_2->setDisabled(true);
    }else{
        ui->pushButton_2->setDisabled(false);
    }
}
```

```
void MainPage::on_uname_reg_editingFinished()
{
    if(ui->uname_reg->text().isEmpty()){
        ui->pushButton_2->setDisabled(true);
    }else{
        ui->pushButton_2->setDisabled(false);
    }
}
```

```
void MainPage::on_pword_reg_editingFinished()
```

```

{
    if(ui->pword_reg->text().isEmpty()){
        ui->pushButton_2->setDisabled(true);
    }else{
        ui->pushButton_2->setDisabled(false);
    }
}

void MainPage::on_dateEdit_dateChanged(const QDate &date)
{
    if(ui->dateEdit->text().isEmpty()){
        ui->pushButton_2->setDisabled(true);
    }else{
        ui->pushButton_2->setDisabled(false);
    }
}

void MainPage::on_pin_reg_editingFinished()
{
    if(ui->pin_reg->text().isEmpty()){
        ui->pushButton_2->setDisabled(true);
    }else{
        ui->pushButton_2->setDisabled(false);
    }
}

```

options.cpp

```

#include "options.h"
#include "ui_options.h"
#include "mainpage.h"

```

```
#include <QMessageBox>

Options::Options(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Options)
{
    ui->setupUi(this);
}

Options::~Options()
{
    delete ui;
}

void Options::on_pushButton_clicked()
{
    this->hide();
    withdrawl obj;
    obj.setModal(true);
    obj.exec();
}

void Options::on_pushButton_2_clicked()
{
    this->hide();
    deposit obj1;
    obj1.setModal(true);
    obj1.exec();
}
```

```
void Options::on_pushButton_3_clicked()
```

```
{  
    this->hide();  
    Fundstransfer obj;  
    obj.setModal(true);  
    obj.exec();  
}
```

```
void Options::on_pushButton_7_clicked()
```

```
{  
    MainPage obj;  
    obj.connOpen();  
    QSqlQuery qry;  
    QString balance;  
    qry.prepare("SELECT Balance FROM record Where Username='"+username+"'");  
    qry.exec();  
    qry.first();  
    balance = qry.value(0).toString();  
    QMessageBox::information(this,"Balance Check","Your balance is Rs."+balance);  
    obj.connClose();  
}
```

```
void Options::on_pushButton_4_clicked()
```

```
{  
    this->hide();  
    changepin obj;  
    obj.setModal(true);  
}
```

```
    obj.exec();  
}
```

```
void Options::on_pushButton_5_clicked()  
{  
    this->hide();  
    changepword obj;  
    obj.setModal(true);  
    obj.exec();  
}
```

```
void Options::on_pushButton_8_clicked()  
{  
    this ->hide();  
    Accountstatement obj;  
    obj.setModal(true);  
    obj.exec();  
}
```

```
void Options::on_pushButton_9_clicked()  
{  
    this->close();  
}
```

withdrawl.cpp

```
#include "withdrawl.h"  
#include "ui_withdrawl.h"
```

```
#include "mainpage.h"

#include <QTime>

withdrawl::withdrawl(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::withdrawl)
{
    ui->setupUi(this);
    ui->pushButton->setDisabled(true);
}
```

```
withdrawl::~~withdrawl()
{
    delete ui;
}
```

```
void withdrawl::on_pushButton_clicked()
{
    QString current = ui->pin->text();
    MainPage obj;
    obj.connOpen();
    QSqlQuery qrya;
    qrya.prepare("SELECT PIN FROM record where Username = '"+username+"'");
    qrya.exec();
    qrya.first();
    QString check = qrya.value(0).toString();
    obj.connClose();
    if(current == check){
        int a = ui->lineEdit->text().toInt();
        int b;
        QString balance;
        obj.connOpen();
```



```

QSqlQuery qry;

qry.prepare("SELECT * FROM record WHERE Username='"+ username+"'");

if(qry.exec()){

    int count=0;

    while(qry.next()){

        count++;

    }

    if(count==1){

        qry.first();

        balance = qry.value(6).toString();

        obj.connClose();

    }

}

if(balance.toInt()>=a){

    b = balance.toInt()-a;

    QString str;

    QString time = QDateTime::currentDateTime().toString();

    str.setNum(b);

    obj.connOpen();

    QSqlQuery query;

    query.prepare("UPDATE record set Balance='"+str+"' WHERE Username='"+ username+"'");

    if(query.exec()){

        QMessageBox::information(this,"Success","Funds withdrawn successfully");

    }

    else{

        QMessageBox::critical(this,"Failed","Failed");

    }

    obj.connClose();

    obj.connOpen();

    QSqlQuery qr1;

```

```

        qr1.prepare("INSERT INTO statement(Username,Amount,Mode,Date,Status)
VALUES(:Username,:Amount,:Mode,:Date,:Status)");

        qr1.bindValue(":Username",username);
        qr1.bindValue(":Amount",a);
        qr1.bindValue(":Mode","Cash Withdrawl");
        qr1.bindValue(":Status","Debited");
        qr1.bindValue(":Date",time);
        qr1.exec();
        obj.connClose();
    }
    else{
        QMessageBox::critical(this,"Failed","Insufficient Funds");
    }
    this->hide();
    Options obj1;
    obj1.setModal(true);
    obj1.exec();
}
else{
    QMessageBox::critical(this,"Failed","Incorrect PIN");
}

}

```

```

void withdrawl::on_pushButton_2_clicked()
{
    this->hide();
    Options obj;
    obj.setModal(true);

```

```

        obj.exec();
    }

void withdrawl::on_lineEdit_editingFinished()
{
    if(ui->lineEdit->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

```

void withdrawl::on_pin_editingFinished()
{
    if(ui->pin->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

desposit.cpp

```

#include "deposit.h"
#include "ui_deposit.h"
#include "mainpage.h"

deposit::deposit(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::deposit)

```

```

{
    ui->setupUi(this);
    ui->pushButton->setDisabled(true);
}

```

```

deposit::~deposit()

```

```

{
    delete ui;
}

```

```

void deposit::on_pushButton_clicked()

```

```

{
    QString current = ui->pin->text();
    MainPage obj;
    obj.connOpen();
    QSqlQuery qrya;
    qrya.prepare("SELECT PIN FROM record where Username = '"+username+"'");
    qrya.exec();
    qrya.first();
    QString check = qrya.value(0).toString();
    obj.connClose();
    if(current == check){
        int a = ui->lineEdit->text().toInt();
        int b;

        QString balance;

        QString time = QDateTime::currentDateTime().toString();
        obj.connOpen();
        QSqlQuery qry;
        qry.prepare("SELECT * FROM record WHERE Username='"+ username+"' ");
        if(qry.exec()){

```

```

        int count=0;
        while(qry.next()){
            count++;
        }
        if(count==1){
            qry.first();
            balance = qry.value(6).toString();

            obj.connClose();
        }

    }

    b = balance.toInt()+a;
    QString str;
    str.setNum(b);
    obj.connOpen();
    QSqlQuery query;
    query.prepare("UPDATE record set Balance='"+str+"' WHERE Username='"+ username+"'");
    if(query.exec()){
        QMessageBox::information(this,"Success","Funds deposited successfully");
    }
    else{
        QMessageBox::critical(this,"Failed","Failed");
    }
    obj.connOpen();
    QSqlQuery qr1;
    qr1.prepare("INSERT INTO statement(Username,Amount,Mode,Date,Status)
VALUES(:Username,:Amount,:Mode,:Date,:Status)");
    qr1.bindValue(":Username",username);
    qr1.bindValue(":Amount",a);
    qr1.bindValue(":Mode","Cash Deposit");

```

```
qr1.bindValue(":Status","Credited");
qr1.bindValue(":Date",time);
qr1.exec();
obj.connClose();
this->hide();
Options obj1;
obj1.setModal(true);
obj1.exec();
}
else{
    QMessageBox::critical(this,"Failed","Incorrect PIN");
}

}
```

```
void deposit::on_pushButton_2_clicked()
{
    this->hide();
    Options obj1;
    obj1.setModal(true);
    obj1.exec();
}
```

```
void deposit::on_lineEdit_editingFinished()
{
    if(ui->lineEdit->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}
```

```

    }
}

void deposit::on_pin_editingFinished()
{
    if(ui->pin->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

fundstransfer.cpp

```

#include "fundstransfer.h"
#include "ui_fundstransfer.h"
#include "mainpage.h"

Fundstransfer::Fundstransfer(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Fundstransfer)
{
    ui->setupUi(this);

    ui->pushButton->setDisabled(true);

}

Fundstransfer::~Fundstransfer()
{
    delete ui;
}

```

```
}
```

```
void Fundstransfer::on_pushButton_clicked()
```

```
{
```

```
    QString current = ui->lineEdit->text();
```

```
    MainPage obj;
```

```
    obj.connOpen();
```

```
    QSqlQuery qrya;
```

```
    qrya.prepare("SELECT PIN FROM record where Username = '"+username+"'");
```

```
    qrya.exec();
```

```
    qrya.first();
```

```
    QString check = qrya.value(0).toString();
```

```
    obj.connClose();
```

```
    if(current == check){
```

```
        int a = ui->amount->text().toInt();
```

```
        int b;
```

```
        QString accountnum = ui->accountnum->text();
```

```
        QString str;
```

```
        QString balance;
```

```
        QString time = QDateTime::currentDateTime().toString();
```

```
    obj.connOpen();
```

```
    QSqlQuery qry1;
```

```
    qry1.prepare("SELECT * FROM record WHERE Username='"+username+"' ");
```

```
    if(qry1.exec()){
```

```
        int count=0;
```

```
        while(qry1.next()){
```

```
            count++;
```

```
        }
```

```
        if(count==1){
```

```
            qry1.first();
```



```

        balance = qry1.value(6).toString();

        obj.connClose();
    }

}

b = balance.toInt()-a;
str.setNum(b);
if(balance.toInt() >= a){
    obj.connOpen();
    QSqlQuery query1;
    query1.prepare("UPDATE record set Balance='"+str+"' WHERE Username='"+username+"'");
    if(query1.exec()){
        QMessageBox::information(this,"Success","Funds transferred successfully");
    }
    obj.connClose();

    obj.connOpen();
    QSqlQuery qry;
    qry.prepare("SELECT * FROM record WHERE Accountnum='"+accountnum+"' ");
    if(qry.exec()){
        int count=0;
        while(qry.next()){
            count++;
        }
        if(count==1){
            qry.first();
            balance = qry.value(6).toString();

            obj.connClose();
        }
    }
}

```

```

    }
    b = balance.toInt()+a;
    str.setNum(b);
    obj.connOpen();
    QSqlQuery query;
    query.prepare("UPDATE record set Balance='"+str+"' WHERE Accountnum='"+accountnum+"'");
    query.exec();
    obj.connClose();
    obj.connOpen();
    QSqlQuery qr1;
    qr1.prepare("INSERT INTO statement(Username,Amount,Mode,Date,Status,Receiver)
VALUES(:Username,:Amount,:Mode,:Date,:Status,:Receiver)");
    qr1.bindValue(":Username",username);
    qr1.bindValue(":Amount",a);
    qr1.bindValue(":Mode","Funds Transfer");
    qr1.bindValue(":Status","Debited");
    qr1.bindValue(":Date",time);
    qr1.bindValue(":Receiver",accountnum);
    qr1.exec();
    obj.connClose();
}
else{
    QMessageBox::critical(this,"Failed","Insufficient Funds");
}
this->hide();
Options obj1;
obj1.setModal(true);
obj1.exec();
}
else{
    QMessageBox::critical(this,"Invalid","Incorrect PIN");
}

```

```
}  
}
```

```
void Fundstransfer::on_pushButton_2_clicked()
```

```
{  
    this->hide();  
    Options obj;  
    obj.setModal(true);  
    obj.exec();  
}
```

```
void Fundstransfer::on_accountnum_editingFinished()
```

```
{  
    if(ui->accountnum->text().isEmpty()){  
        ui->pushButton->setDisabled(true);  
    }else{  
        ui->pushButton->setDisabled(false);  
    }  
}
```

```
void Fundstransfer::on_amount_editingFinished()
```

```
{  
    if(ui->amount->text().isEmpty()){  
        ui->pushButton->setDisabled(true);  
    }else{  
        ui->pushButton->setDisabled(false);  
    }  
}
```

```

void Fundstransfer::on_lineEdit_editingFinished()
{
    if(ui->lineEdit->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

changepword.cpp

```

#include "changepword.h"
#include "ui_changepword.h"
#include "mainpage.h"
changepword::changepword(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::changepword)
{
    ui->setupUi(this);
    ui->pushButton->setDisabled(true);

}
changepword::~changepword()
{
    delete ui;
}

void changepword::on_pushButton_clicked()
{

```

```

QString current = ui->lineEdit->text();
QString pword = ui->lineEdit_2->text();

MainPage obj;

obj.connOpen();

QSqlQuery qry;

qry.prepare("SELECT Password FROM record where Username = '"+username+"'");

qry.exec();

qry.first();

QString check = qry.value(0).toString();

obj.connClose();

if(current == check){

    obj.connOpen();

    QSqlQuery qry1;

    qry1.prepare("UPDATE record set Password= '"+pword+"' WHERE Username = '"+username+"'");

    qry1.exec();

    obj.connClose();

    QMessageBox::information(this,"Success","Password changed sucessfully");

}

else{

    QMessageBox::critical(this,"Failed", "Incorrect Password");

}

this->hide();

Options obj1;

obj1.setModal(true);

obj1.exec();

}

```

```

void changepword::on_pushButton_2_clicked()

{

    this->hide();

```

```

Options obj1;
obj1.setModal(true);
obj1.exec();
}

void changepword::on_lineEdit_2_editingFinished()
{
    if(ui->lineEdit_2->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

```

void changepword::on_lineEdit_editingFinished()
{
    if(ui->lineEdit->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

changepin.cpp

```

#include "changepin.h"
#include "ui_changepin.h"
#include "mainpage.h"
changepin::changepin(QWidget *parent) :

```

```

QDialog(parent),
ui(new Ui::changePin)
{
    ui->setupUi(this);
    ui->pushButton->setDisabled(true);
}

```

```

changePin::~~changePin()
{
    delete ui;
}

```

```

void changePin::on_pushButton_clicked()
{
    QString current = ui->lineEdit->text();
    MainPage obj;
    obj.connOpen();
    QSqlQuery qry;
    qry.prepare("SELECT PIN FROM record where Username = '"+username+"'");
    qry.exec();
    qry.first();
    QString check = qry.value(0).toString();
    obj.connClose();
    if(current == check){
        QString newpin = ui->lineEdit_2->text();
        obj.connOpen();
        QSqlQuery qry1;
        qry1.prepare("UPDATE record set PIN = '"+newpin+"' WHERE Username = '"+username+"'");
        qry1.exec();
        obj.connClose();
    }
}

```

```
        QMessageBox::information(this,"Success","PIN changed sucessfully");
    }
    else{
        QMessageBox::critical(this,"Failed","Authentication Failed:Wrong PIN entered");
    }
    this->hide();
    Options obj1;
    obj1.setModal(true);
    obj1.exec();
}
```

```
void changepin::on_pushButton_2_clicked()
{
    this->hide();
    Options obj1;
    obj1.setModal(true);
    obj1.exec();
}
```

```
void changepin::on_lineEdit_editingFinished()
{
    if(ui->lineEdit->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}
```



```

void changepin::on_lineEdit_2_editingFinished()
{
    if(ui->lineEdit_2->text().isEmpty()){
        ui->pushButton->setDisabled(true);
    }else{
        ui->pushButton->setDisabled(false);
    }
}

```

accountstatement.cpp

```

#include "accountstatement.h"
#include "ui_accountstatement.h"
#include "mainpage.h"
Accountstatement::Accountstatement(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Accountstatement)
{
    ui->setupUi(this);
    MainPage obj;
    QSqlQueryModel *modal = new QSqlQueryModel();
    obj.connOpen();
    QSqlQuery*qry = new QSqlQuery(obj.mydb);
    qry->prepare("SELECT * FROM statement WHERE Username='"+username+"' ");
    qry->exec();
    modal->setQuery(*qry);
    ui->tableView->setModel(modal);
    obj.connClose();
}

Accountstatement::~Accountstatement()

```

```
{  
    delete ui;  
}  
  
void Accountstatement::on_pushButton_clicked()  
{  
    this->hide();  
    Options obj1;  
    obj1.setModal(true);  
    obj1.exec();  
}
```

main.cpp

```
#include "mainpage.h"  
  
#include <QApplication>  
  
int main(int argc, char *argv[])  
{  
    QApplication a(argc, argv);  
    MainPage w;  
    w.show();  
    return a.exec();  
}
```

Output (Screenshots)

MainPage

WELCOME TO STS BANK LIMITED

SIGN IN

Enter Username

Enter Password

Sign-in

SIGN UP

Enter Name

Enter Username

Enter Password

☐ Male

☐ Female

1/1/2000

Enter PIN

Sign-Up

MainPage

WELCOME TO STS BANK LIMITED

SIGN IN

Enter Username

Enter Password

Sign-in

Success

i

User Registered Successfully

OK

SIGN UP

Hammad Ahmad Tajwar

hammad1234

●●●●●●●●

☒ Male

☐ Female

7/5/1998

●●●●

Sign-Up

MainPage

WELCOME TO STS BANK LIMITED

SIGN IN

hammad1234

Sign-in

SIGN UP

Enter Name

Enter User

Enter Pass

Success

Login Success

OK

Male

Female

1/1/2000

Enter PIN

Sign-Up

Dialog

Cash Withdrawl

Cash Deposit

Funds Transfer

Balance Enquiry

Change PIN

Change Password

Transactions History

LOG OUT

Dialog

Failed

Incorrect PIN

OK

Enter Amount:

5000

Enter PIN:

●●●●

ConfirmBack

Dialog

Success

i

Funds deposited successfully

OK

ConfirmBack

Dialog

Cash Withdrawl

Cash Deposit

Funds Transfer

Balance Enquiry

Change PIN

Change Password

Transactions History

LOG OUT

Balance Check

i

Your balance is Rs.5000

OK

Dialog

Enter Account Number

1000008

Enter Amount:

2500

Enter PIN:

●●●●

Confirm

Back

Success

i

Funds transferred successfully

OK

Dialog

Success

i

Funds withdrawn successfully

OK

Enter Amount

1000

Enter PIN:

●●●●

Confirm

Back

Dialog

Enter Current Password:

●●●●●●●●●●

Enter New Password:

●●●●●●●●●●

Confirm

Back

Dialog

Enter Current Pin:

●●●●

Enter New Pin:

●●●●

Confirm

Back

Success

i

PIN changed sucessfully

OK

Dialog

	Username	Amount	Mode	Date	Status	Receiver
1	hammad1234	5000	Cash Deposit	Sun Jan 8 22:55:07 2023	Credited	NONE
2	hammad1234	1000	Cash Withdrawl	Sun Jan 8 22:55:40 2023	Debited	NONE
3	hammad1234	2500	Funds Transfer	Sun Jan 8 22:56:09 2023	Debited	1000008

Back


```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
Press E for Employee Login
Press A for Admin Login
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
Press E for Employee Login
Press A for Admin Login
A
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
Username:
hello
Password:
*****
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
Invalid username or password
Username:
```

```
C:\Users\Saad\source\repos\sql\64\Debug\sql.exe

-----
STS Bank unlimited
-----

Username:
Admin
Password:
*****
```

```
C:\Users\Saad\source\repos\sql\64\Debug\sql.exe

-----
STS Bank unlimited
-----

1. Add a New Employee
2. Check Attendance of an Employee
3. Calculate Salary of an Employee
4. Display all records of the Employees
5. Display all records of the Customers
0. Go Back
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----

Name:
Haris Asghar
Address:
House no 1 Street no 2 Islamabad
Grade (A-D):
█
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----

1. Add a New Employee
2. Check Attendance of an Employee
3. Calculate Salary of an Employee
4. Display all records of the Employees
5. Display all records of the Customers
0. Go Back
2
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
Enter ID to check attendance
1003
This employee was present 2 days in this month
Press any key to continue . . .
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
1. Add a New Employee
2. Check Attendance of an Employee
3. Calculate Salary of an Employee
4. Display all records of the Employees
5. Display all records of the Customers
0. Go Back
_
```

```
C:\Users\Saad\source\repos\sql\sql64\Debug\sql.exe

-----
STS Bank unlimited
-----
Enter employee id:
1003
The salary of 1003 is Rs.50000
Press any key to continue . . .
```

```
C:\Users\Saad\source\repos\sql\sql64\Debug\sql.exe

-----
STS Bank unlimited
-----
1. Add a New Employee
2. Check Attendance of an Employee
3. Calculate Salary of an Employee
4. Display all records of the Employees
5. Display all records of the Customers
0. Go Back
4
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----

Name: Saad ahmad
Address: asdf
Grade: a

Name: zxc
Address: zxcv
Grade: b

Name: Haris Asghar
Address: House no 1 Street no 2 Islamabad
Grade: C

Press any key to continue . . .
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----

1. Add a New Employee
2. Check Attendance of an Employee
3. Calculate Salary of an Employee
4. Display all records of the Employees
5. Display all records of the Customers
0. Go Back
9
```

```
Select C:\Users\Saad\source\repos\sql\sql.exe
Name: Shuja Uddin
Username: shuja456
Gender: Male
DOB: 16/10/2001
Balance: 2500.0

Accountnum: 1000007
Name: testuser
Username: testuser123
Gender: Male
DOB: 10/12/2004
Balance: 0.0

Accountnum: 1000008
Name: Omer Javed
Username: omer123
Gender: Male
DOB: 1/1/2003
Balance: 9991.0

Accountnum: 1000009
Name:
Username:
Gender:
DOB: 1/1/2000
Balance: 0.0

Accountnum: 1000010
Name: testing
Username: test50
Gender: Male
DOB: 1/1/2004
Balance: 0.0

Accountnum: 1000011
Name: Sohaib Ahmed
Username: sohaib9009
Gender: Male
DOB: 19/2/2001
Balance: 6500.0

Accountnum: 1000012
Name: Hammad Ahmad Tajwar
Username: hammad1234
Gender: Male
DOB: 7/5/1998
Balance: 1500.0

Press any key to continue . . .

C:\Users\Saad\source\repos\sql\sql.exe

-----
STS Bank unlimited
-----

Press E for Employee Login
Press A for Admin Login
E
```



```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
Username:
1000
Password:
asdasd
Invalid username or password
Username:
```

```
C:\Users\Saad\source\repos\sql\vt64\Debug\sql.exe

-----
STS Bank unlimited
-----
Username:
1003
Password:
1031148_
```

```
C:\Users\Saad\source\repos\sql\64\Debug\sql.exe

-----
                        STS Bank unlimited
-----

Username:
1003
Password:
1031148
Logged in successfully
1. Mark attendance (Enter 1 for Present)
1
Attendance marked successfully
Press any key to continue . . .
```